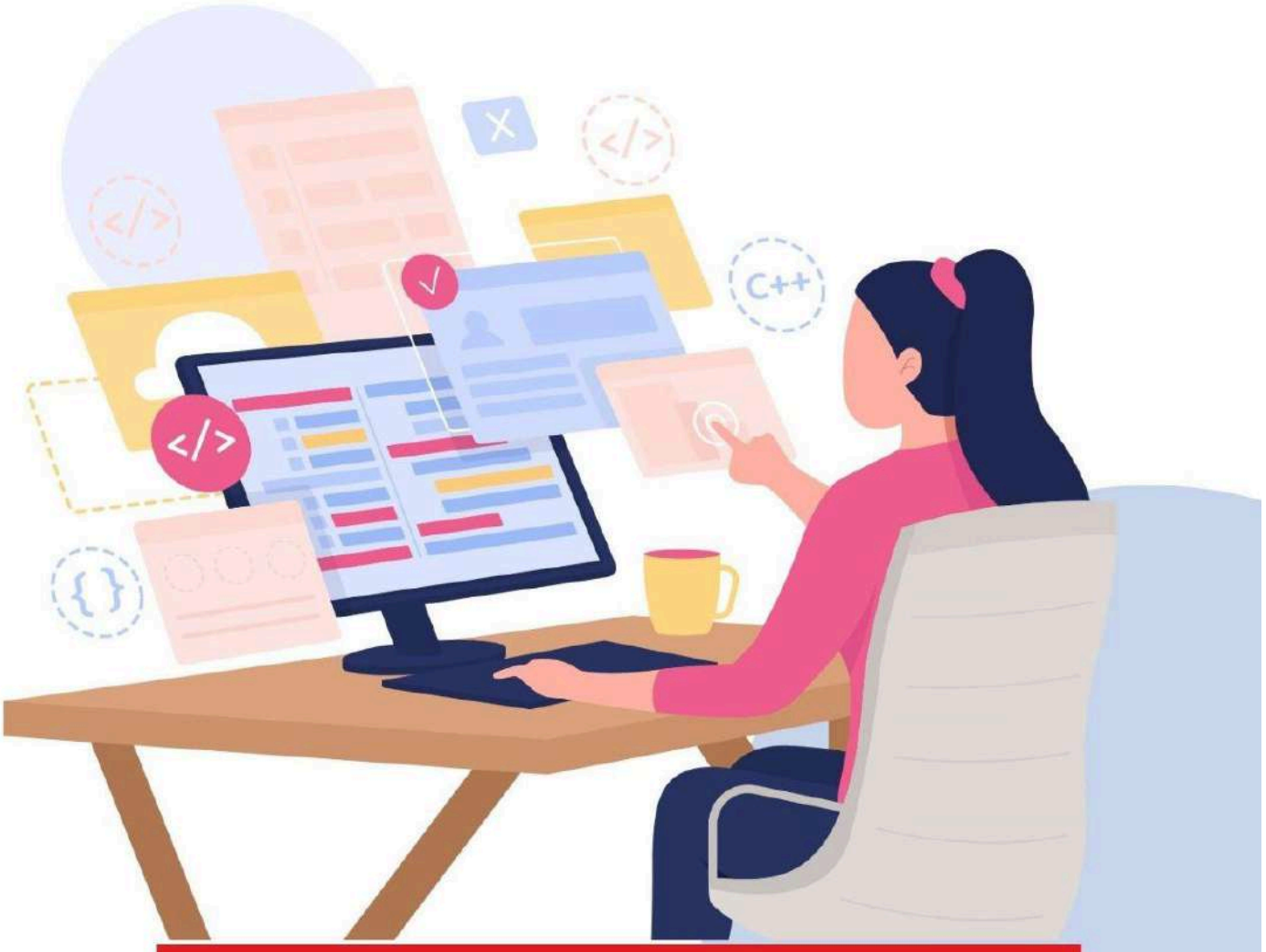
Identitas Digital Indonesia

# ARTIFICIAL INTELLEGENCE

## Level Up Coding!
## Bikin Game, Aplikasi, dan AI Sendiri!

**UNTUK SMP KELAS VIII SEMESTER 1**




Institut Teknologi
Tangerang Selatan
2025

# Buku AI SMP Kelas 8 Semester 1

## Level Up Coding!
## Bikin Game, Aplikasi, dan AI Sendiri!

**Jumlah Pertemuan:** 14-16 (Dapat disesuaikan)

Onno W. Purbo

**Institut Teknologi Tangerang Selatan (ITTS)**
**2025**

# Daftar Isi

# Lisensi & Catatan Karya



Desain Cover: Irwan Siswanto

# Disclaimer

Materi pembelajaran ini dibuat dengan **dana swadaya masyarakat Indonesia** dan **kontribusi sukarela dari para dosen di Institut Teknologi Tangerang Selatan (ITTS)**. Karya ini didistribusikan secara bebas untuk mendukung pendidikan digital dan kecerdasan buatan di kalangan pelajar Indonesia.

Kami **memohon maaf** jika terdapat kekurangan dalam isi maupun penyajian materi ini. Kritik dan saran sangat kami harapkan untuk terus menyempurnakan karya serupa di masa depan.

# Kata Pengantar

Buku ini disusun sebagai bentuk partisipasi sederhana dari sivitas akademika Institut Teknologi Tangerang Selatan (ITTS) dalam mendukung kemajuan pendidikan di Indonesia. Di tengah pesatnya perkembangan teknologi, kami terdorong untuk menghadirkan materi pembelajaran yang relevan dan dapat diakses secara luas oleh masyarakat, khususnya generasi muda.

Dengan semangat berbagi dan memberdayakan, kami berharap buku ini dapat menjadi langkah awal yang bermakna dalam memperkenalkan konsep dasar kecerdasan buatan (AI) kepada pelajar tingkat SMP dan SMA. Materi disusun secara sistematis dan aplikatif, mencakup antara lain:

- Dasar-dasar *Computational Thinking*
- Pembuatan aplikasi Android menggunakan App Inventor 2
- Pemrograman visual dengan Scratch
- Pengenalan AI melalui Google Teachable Machine
- Eksplorasi AI generatif seperti ChatGPT, Gemini, dan Grok
- Pembelajaran AI berbasis data dengan Orange Data Mining, disertai contoh yang sederhana

Penyusunan dan penerbitan buku ini sepenuhnya dibiayai secara mandiri berkat dukungan berbagai pihak yang memiliki kepedulian terhadap kemajuan ilmu pengetahuan di tanah air. Untuk itu, kami mengucapkan terima kasih yang tulus atas segala bantuan dan dorongan yang telah diberikan.

Kami menyadari bahwa karya ini masih jauh dari sempurna. Karena itu, segala bentuk kritik, saran, maupun masukan yang membangun sangat kami hargai dan nantikan. Silakan sampaikan melalui kontak resmi ITTS atau langsung kepada tim penulis. Pembaruan dan perbaikan akan terus kami lakukan demi meningkatkan kualitas isi dan manfaat buku ini.

Semoga buku ini dapat menjadi pijakan awal bagi para pelajar dalam memahami dunia kecerdasan buatan, serta turut menginspirasi langkah kecil menuju masa depan Indonesia yang lebih cerah.

Dengan rendah hati, kami memohon doa dan dukungan dari para pembaca agar upaya sederhana ini dapat membawa manfaat yang luas dan menjadi amal jariyah bagi semua penulis yang terlibat.

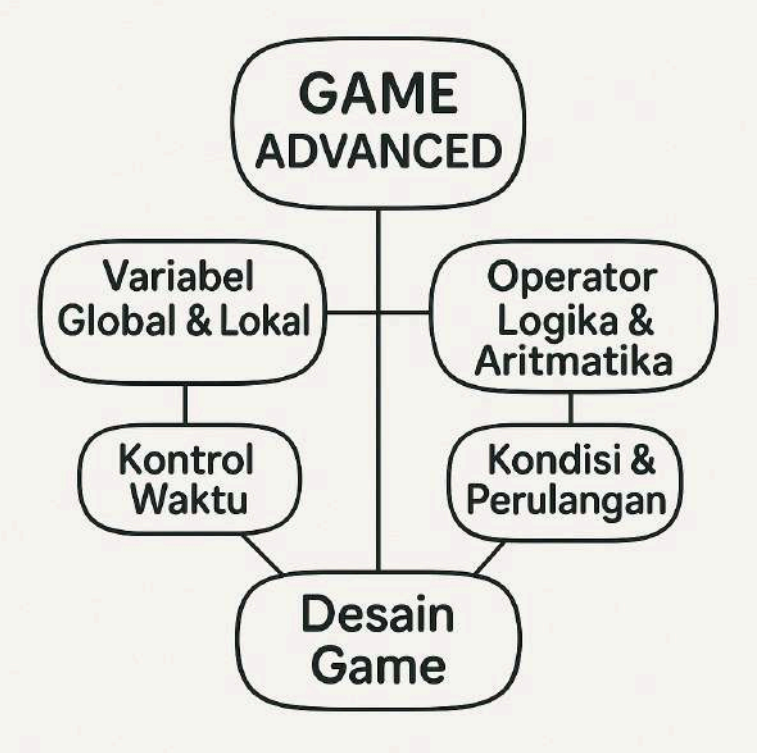
Tangerang Selatan, Agustus 2025
**Penulis**

# BAB 1: Pengembangan Game Scratch Lanjutan (Level, Skor, Timer)

## Tujuan Pembelajaran

Peserta didik mampu mengembangkan game Scratch dengan fitur level yang dinamis, sistem skor yang akurat, dan timer yang fungsional, serta memahami konsep di balik setiap fitur.

## Peta Konsep

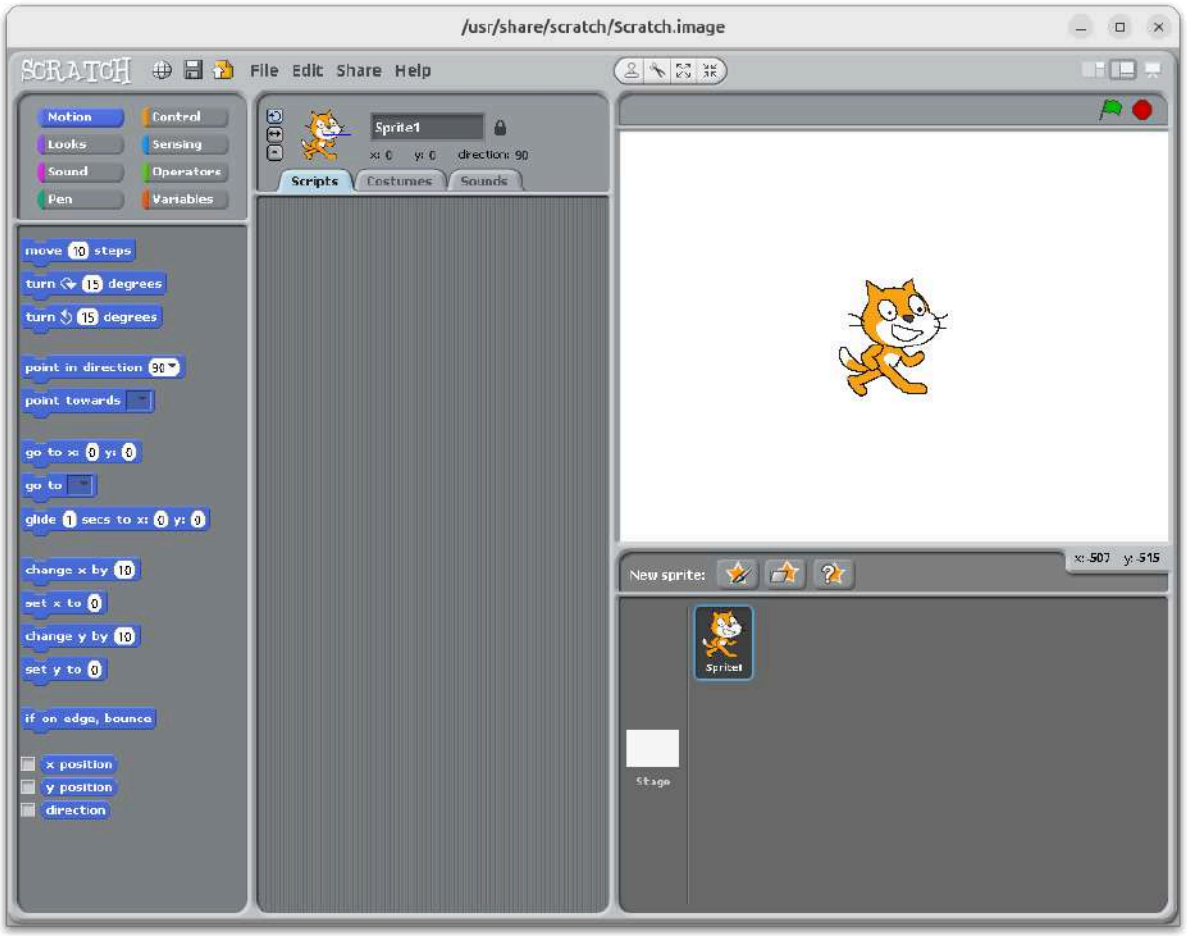



## Apersepsi

Bagaimana game-game seru yang sering kalian mainkan, seperti "Subway Surfers" atau "Candy Crush", bisa punya level yang berbeda-beda tingkat kesulitannya? Bagaimana mereka menghitung skor yang kita dapatkan, atau menampilkan waktu hitung mundur yang menegangkan? Mari kita bongkar rahasia di baliknya dan terapkan di Scratch!

# Penjelasan Konsep (Teori)

## Membangun Game yang Lebih Canggih dengan Scratch




Pernahkah kalian bertanya-tanya bagaimana game favorit seperti *Subway Surfers*, *Candy Crush*, atau *Among Us* bisa memiliki fitur-fitur keren seperti **naiknya level**, **skor yang terus bertambah**, dan **timer yang bikin deg-degan**? Rahasianya terletak pada **pemrograman logika dan penggunaan variabel yang cerdas**. Di bagian ini, kita akan membongkar semua konsep penting itu agar kalian bisa menciptakan game Scratch yang *lebih seru, lebih menantang, dan lebih hidup*!

## Apa Itu Variabel?

Dalam **Scratch**, **variabel adalah seperti kantong penyimpan informasi**. Kita bisa menyimpan angka, kata, atau nilai tertentu, lalu mengambil atau mengubahnya kapan saja saat game berjalan. Variabel sangat penting untuk menciptakan game yang interaktif dan responsif.

## Variabel Skor: Menghitung Poin Pemain

Skor adalah bagian penting dari game—tanpa skor, kita tidak tahu siapa yang menang!

**Fungsi:**

- Menyimpan poin pemain saat melakukan sesuatu yang benar.
- Menambahkan atau mengurangi skor sesuai aksi pemain.

**Contoh Penerapan:**

- Menambah skor saat pemain menangkap koin:
  ```
  change [skor] by (10)
  ```
- Mengatur skor awal saat game dimulai:
  ```
  set [skor] to (0)
  ```


## Variabel Level: Meningkatkan Tantangan

Game seru selalu punya tantangan yang bertambah seiring waktu. Di sinilah **variabel level** berperan!

**Fungsi:**

- Menyimpan tingkat kesulitan atau tahap dalam game.
- Digunakan untuk mengganti kecepatan objek, jenis rintangan, atau tampilan game.

**Contoh Penerapan:**

Naik level setelah skor mencapai angka tertentu:

```
if <(skor) > (100)> then
  change [level] by (1)
end
```


## Variabel Timer: Mengatur Waktu

Timer membuat permainan jadi lebih seru karena ada tekanan waktu! Bisa berupa **hitung mundur** atau **hitung naik**.

**Fungsi:**

- Mengukur waktu bermain.

- Menentukan kapan game berakhir atau berpindah ke level baru.

**Contoh Penerapan:**

Mengurangi waktu setiap detik:

```
repeat until <(sisa waktu) = 0>
  wait (1) seconds
  change [sisa waktu] by (-1)
end
```


## Blok Variabel yang Wajib Dikuasai

Agar variabel bisa bekerja dalam game, kita perlu memanfaatkan blok-blok berikut:

- `set [variabel] to [nilai]`: Mengatur nilai awal.
- `change [variabel] by [nilai]`: Menambahkan atau mengurangi nilai.


Menampilkan atau menyembunyikan variabel di panggung agar terlihat atau tersembunyi sesuai kebutuhan.

## Operator Matematika dan Logika: Otak dari Game!

Agar game bisa “berpikir”, kita menggunakan **operator** untuk membuat perhitungan dan keputusan otomatis.

**Operator Aritmatika:**
Digunakan untuk menghitung:

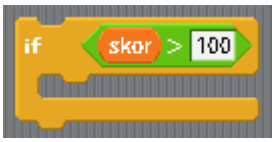
- `+` Tambah
- `-` Kurang
- `*` Kali
- `/` Bagi

Contoh: menghitung total skor, kecepatan objek, atau waktu.

**Operator Perbandingan:**
Digunakan untuk membuat syarat:

- `<`, `>`, `=`, `<=`, `>=`
  Contoh:

```
if <(skor) > (100)> then
  // naik level
end
```

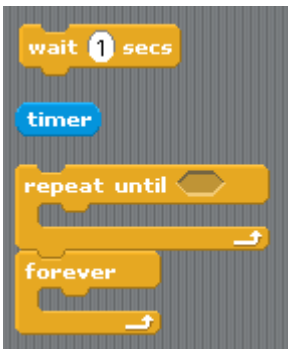



**Operator Logika:**
Digunakan untuk menggabungkan kondisi:

- `and`, `or`, `not`
  Contoh:

```
if <<(level) = 2> and <(sisa waktu) > 0>> then
  // lanjut main
end
```


## Kontrol Waktu dan Perulangan




Waktu dan pengulangan sangat penting dalam game:

- `wait [detik]`: Memberi jeda sebelum aksi selanjutnya.
- `timer`: Blok bawaan Scratch yang bisa menghitung waktu sejak game dimulai.
- `repeat until`: Ulangi aksi hingga suatu kondisi terjadi.
- `forever`: Ulangi aksi terus-menerus selama game berjalan.

## Kesimpulan Mini: Yuk Mulai Menerapkan!

Dengan menguasai variabel, operator, dan kontrol waktu, kalian sudah memiliki **bekal super penting untuk membuat game yang lebih hidup**. Kalian bisa mengatur kapan pemain naik level, menambah skor otomatis, dan memberi batas waktu agar game semakin menantang.

Jadi, siap membuat game kalian jadi luar biasa? Yuk kita lanjut ke bagian praktik!

# Contoh Kasus atau Ilustrasi

## Kasus 1: Hujan Meteor: Misi Selamatkan Kota

**Ilustrasi:**
Bayangkan kamu adalah pilot pesawat luar angkasa yang menjaga bumi dari serangan meteor.

- Setiap meteor yang berhasil kamu hindari: **+1 skor**.
- Tapi jika meteor menabrak pesawatmu: **Health berkurang 10 poin**.
- Kamu punya **100 Health poin**, dan jika habis, game berakhir.
- Target: Bertahan hidup selama **1 menit** untuk menang!

**Tantangan:**
Bisakah kamu membuat logika agar pesawat bisa selamat dengan skor tertinggi?

## Kasus 2: Tangkap Bintang, Hindari Batu

**Ilustrasi:**
Seekor kucing lucu sedang bermain di bawah langit malam.

- Tangkap **bintang**: **+5 poin**
- Tangkap **batu (salah sasaran!)**: **-2 poin**
- Waktu bermain: **30 detik**
- Jika skor mencapai **50 poin** → Naik ke **Level 2**, di mana bintang jatuh lebih cepat!

**Tantangan:**
Bisakah kamu membuat sistem level dan skor yang menyesuaikan kecepatan bintang?

## Kasus 3: Ninja Semangka

**Ilustrasi:**
Kamu adalah ninja yang harus membelah semangka yang dilontarkan ke udara.

- Jika kamu memotong **semangka**: **+3 poin**
- Tapi hati-hati, kadang ada **bom**! Kalau kamu kena bom, **Health -15**
- Kamu punya **3 nyawa** (setiap nyawa = 30 Health)
- Jika nyawa habis, game selesai!

**Tantangan:**
Bisakah kamu merancang sistem nyawa dan skor dalam aksi ninja ini?

## Kasus 4: Lari dari Hantu

**Ilustrasi:**
Seorang anak tersesat di rumah berhantu. Dia harus lari dari hantu sambil mengumpulkan kunci untuk keluar.

- Ambil **kunci**: **+1 skor**
- Terkena **hantu**: **Health -20 poin**
- Jika berhasil dapat **5 kunci** sebelum Health habis → Menang!

**Tantangan:**
Bagaimana kamu membuat sistem skor dan kesehatan yang bisa membuat game ini menegangkan?

### Kasus 5: Balapan Roket Mars

**Ilustrasi:**
Kamu mengendarai roket di permukaan Mars, harus menghindari batu besar dan mengumpulkan bahan bakar.

- Ambil **bahan bakar**: **+10 poin**
- Tabrak **batu**: **Health -10 poin**
- Kamu hanya punya **60 detik** untuk mencapai skor **100 poin** agar roket bisa lepas landas.

**Tantangan:**
Bisakah kamu membuat timer, skor, dan sistem health untuk simulasi roket?

### Kasus 6: Penjelajah Hutan Misterius

**Ilustrasi:**
Karakter utama menjelajahi hutan dan harus memilih jalur yang aman.

- Pilih jalur benar: **+5 poin**
- Pilih jalur jebakan: **Health -15 poin**
- Setiap 3 pilihan benar → Naik level (hutan makin gelap dan menantang!)
- Game selesai jika Health = 0 atau sampai Level 5.

**Tantangan:**
Bagaimana kamu mengatur level yang semakin menantang dan skor yang adil?

## Contoh Soal dan Pembahasan

### Soal 1: Membuat Timer Mundur Seru!

**Tantangan:**
Buatlah sebuah program di Scratch yang menghitung mundur dari 60 detik. Ketika waktu habis, sprite harus menyampaikan pesan “Waktu Habis!” selama 2 detik.

**Pembahasan:**
Langkah-langkah logis:

1. Buat variabel bernama **Waktu Sisa**.
2. Atur nilainya menjadi 60 saat bendera hijau diklik.
3. Buat pengulangan untuk menurunkan nilai setiap detik.
4. Setelah waktu habis, tampilkan pesan.

```
when green flag clicked
set [Waktu Sisa v] to (60)
repeat until <(Waktu Sisa) = (0)>
  wait (1) seconds
  change [Waktu Sisa v] by (-1)
end
say [Waktu Habis!] for (2) seconds
```

## Soal 2: Skor 100? Naik Level, Yuk!

**Tantangan:**
Saat pemain mencapai skor 100, bagaimana cara membuat sprite naik level, mengatur ulang skor, dan mengucapkan “Level Up!”?

**Pembahasan:**

1. Buat dua variabel: **Skor** dan **Level**.
2. Gunakan struktur percabangan untuk memeriksa apakah skor lebih dari 99.
3. Naikkan level, atur ulang skor, dan beri ucapan semangat.

```
if <(Skor) > (99)> then
  change [Level v] by (1)
  set [Skor v] to (0)
  say [Level Up!] for (2) seconds
end
```

## Soal 3: Ayo Ubah Timer Jadi Lebih Fleksibel!

**Tantangan:**
Bagaimana caranya agar waktu mundur bisa diatur sendiri oleh pemain sebelum permainan dimulai?

**Pembahasan:**

1. Buat variabel **Waktu Sisa** dan **Waktu Awal**.
2. Tambahkan kotak input (ask) untuk pemain memasukkan durasi waktu.
3. Gunakan jawaban untuk mengatur durasi timer.

```
ask [Berapa detik waktu permainan?] and wait
set [Waktu Sisa v] to (answer)
repeat until <(Waktu Sisa) = (0)>
```

```
  wait (1) seconds
  change [Waktu Sisa v] by (-1)
end
say [Waktu Habis!] for (2) seconds
```

## Soal 4: Uji Refleks! Hitung Mundur & Klik Sprite

**Tantangan:**
Buat permainan dimana pemain harus mengklik sprite sebelum timer 10 detik habis. Jika berhasil, tampilkan "Selamat!", jika tidak, tampilkan "Terlambat!".

**Pembahasan:**

1. Buat variabel **Waktu Sisa** dan **Berhasil**.
2. Tambahkan skrip klik sprite untuk mengubah status keberhasilan.
3. Gunakan timer mundur dan periksa status di akhir waktu.

```
when green flag clicked
set [Waktu Sisa v] to (10)
set [Berhasil v] to [Tidak]
repeat until <(Waktu Sisa) = (0)>
  wait (1) seconds
  change [Waktu Sisa v] by (-1)
end
if <(Berhasil) = [Ya]> then
  say [Selamat!] for (2) seconds
else
  say [Terlambat!] for (2) seconds
end

when this sprite clicked
set [Berhasil v] to [Ya]
```

## Soal 5: Level Tak Terbatas? Bisa Dong!

**Tantangan:**
Modifikasi sistem level agar setiap kelipatan 100 skor akan menambah level, tanpa batas!

**Pembahasan:**
Gunakan logika modular dan pengulangan agar sprite terus naik level setiap skor mencapai kelipatan 100.

```
when green flag clicked
set [Skor v] to (0)
set [Level v] to (1)
forever
  if <(Skor mod 100) = 0 and (Skor) ≠ 0> then
    change [Level v] by (1)
```

```
    change [Skor v] by (-100)
    say [Level Up!] for (2) seconds
  end
end
```

## Fakta Menarik / Fun Facts

Tahukah kalian? Banyak game terkenal seperti "Pac-Man" atau "Space Invaders" dimulai dari ide sederhana seperti ini, menggunakan logika skor dan waktu untuk membuat permainan semakin menantang! Konsep "lives" (nyawa) dalam game juga adalah bentuk variabel yang mengurangi jumlah kesempatan pemain.

## Tips Belajar atau Tips Cepat Menghafal

- **Pikirkan "Variabel itu penting untuk data":** Setiap informasi yang berubah dalam game (skor, level, waktu, nyawa) butuh variabel.
- **Visualisasikan:** Bayangkan game kalian sedang berjalan, lalu pikirkan informasi apa saja yang perlu dihitung atau disimpan.
- **Coba-coba:** Jangan takut mencoba mengubah angka atau kondisi. Lihat apa yang terjadi!

- **Mind Mapping: Variabel Game di Scratch**


**Cerita Seru**:
Bayangkan game seperti ujian lari. Pelari punya waktu (Timer), harus mengumpulkan poin (Skor), dan setiap kali menyelesaikan tantangan, naik ke rintangan yang lebih sulit (Level). Itulah logika game!

**Tips Ringkas:**

- **Skor** → Tambah nilai
- **Level** → Tambah kesulitan
- **Timer** → Bikin deg-degan 😄
- **Uji coba terus** → Belajar dari kesalahan

# Aktivitas Siswa

## Aktivitas 1: Modifikasi Game Scratch – Tantang Dirimu Menambahkan Skor, Timer, dan Level!

**Deskripsi Aktivitas:**
Pilih game Scratch buatanmu atau dari gurumu, lalu ubah jadi lebih seru dengan menambahkan **fitur skor**, **waktu permainan**, dan **level tantangan**. Bayangkan kamu jadi desainer game profesional!

**Langkah Saran:**

- Buka editor Scratch dan pilih game yang akan kamu modifikasi.
- Tambahkan **variabel "skor"** yang bertambah saat karakter mengambil objek.
- Tambahkan **timer** 30 detik menggunakan blok "ulang sampai".
- Buat **level** baru jika skor mencapai jumlah tertentu – tingkatkan kecepatan atau tambahkan rintangan.

*Tantangan:* Coba buat Level 3 jika skor sangat tinggi. Bisa nggak?

## Aktivitas 2: Eksplorasi Game Keren di Komunitas Scratch

**Deskripsi Aktivitas:**
Jelajahi dunia game buatan anak-anak seumurmu dari seluruh dunia lewat situs Scratch. Cari inspirasi dari game yang punya **fitur skor, timer, dan level**!

**Langkah Saran:**

- Kunjungi scratch.mit.edu.
- Gunakan fitur pencarian untuk temukan game yang menarik.
- Klik tombol **“See Inside”** untuk membongkar cara kerja game itu.
- Perhatikan dan pelajari bagaimana mereka menghitung skor, mengatur waktu, dan menambah level.

*Tantangan:* Temukan game dengan sistem level unik yang belum pernah kamu lihat sebelumnya.

## Aktivitas 3: Jadi Reviewer Game – Uji dan Evaluasi Game Temanmu

**Deskripsi Aktivitas:**
Mainkan game temanmu seperti seorang penguji game profesional. Cari tahu apa yang sudah keren dan apa yang bisa ditingkatkan.

**Langkah Saran:**

- Ajak temanmu bertukar game buatan masing-masing.
- Coba mainkan game mereka beberapa kali.
- Diskusikan hal-hal berikut:

    - Bagaimana sistem skornya bekerja?
    - Apakah ada level atau peningkatan tantangan?
    - Apakah timer membuat permainan jadi lebih seru?

*Tantangan:* Beri masukan positif dan satu ide baru yang bisa temanmu tambahkan.

## Aktivitas 4: Tantangan Desain Game – Turnamen Mini Scratch!

**Deskripsi Aktivitas:**
Ayo berlomba membuat game paling seru dengan sistem skor dalam waktu **60 detik** permainan! Kamu bisa menambahkan fitur ekstra agar lebih menantang.

**Langkah Saran:**

- Gunakan waktu permainan 60 detik dengan variabel timer.
- Tambahkan skor dan tampilkan secara real-time di layar.
- Buat level yang bertambah sesuai skor pemain. Mungkin rintangan baru muncul?

*Tantangan:* Tambahkan efek suara atau animasi keren saat naik level!

## Aktivitas 5: Buat Panduan Sendiri – Jelaskan Cara Kerja Game-mu

**Deskripsi Aktivitas:**
Jelaskan kepada teman atau adik kelas bagaimana game buatanmu bekerja. Kamu akan belajar lebih banyak dengan menjelaskan, lho!

**Langkah Saran:**

- Siapkan slide atau kertas penjelasan singkat.
- Gambarkan alur permainannya (misalnya: mulai → tangkap objek → naik level).
- Tunjukkan skrip penting seperti skor, timer, dan level.
- Bisa juga membuat video pendek menjelaskan cara bermain.

*Tantangan:* Simulasikan jadi YouTuber edukasi Scratch!

## Aktivitas 6: Modifikasi Game Lama – Jadikan Lebih Seru!

**Deskripsi Aktivitas:**
Ambil game Scratch buatanmu yang lama – lalu upgrade dengan fitur baru, tantangan ekstra, dan tampilan yang lebih menarik.

**Langkah Saran:**

- Periksa bagian mana dari game yang terasa membosankan atau terlalu mudah.
- Tambahkan sistem seperti:
    - Objek bonus yang muncul hanya sebentar.

    - Musuh yang bergerak cepat saat level naik.
- Ganti tampilan atau suara untuk memberi nuansa segar.

*Tantangan:* Bisa nggak kamu bikin pemain ketagihan main game kamu?

# Rangkuman

**Fitur-fitur seperti skor, level, dan timer sangat penting** untuk membuat game jadi lebih **seru, menantang, dan tidak membosankan**. Dengan fitur ini, game terasa lebih hidup dan bisa membuat pemain betah memainkannya.

Di *Scratch*, kita bisa menggunakan **variabel** untuk menyimpan informasi penting dalam game, seperti **skor pemain**, **level yang sedang dimainkan**, atau **sisa waktu**. Variabel bisa kita atur dengan blok seperti *set* (untuk memberi nilai awal) dan *change* (untuk menambah atau mengurangi nilai).

Untuk membuat game yang lebih menarik dan *dinamis*, kita juga perlu belajar menggunakan **logika *if*** (jika suatu kondisi terjadi), **operator matematika** (seperti tambah dan kurang), dan **blok kontrol** seperti *repeat* (ulang) dan *wait* (tunggu). Mengatur **timer**, menghitung **skor**, dan menaikkan **level saat skor tertentu tercapai** bisa membuka banyak ide kreatif. Dengan banyak latihan, kamu bisa membuat **game buatanmu sendiri yang keren dan interaktif**!

# Latihan Soal

## A. Benar atau Salah (5 Soal)

1. Blok `ask` dan `wait` digunakan untuk menampilkan pertanyaan dan menerima jawaban dari pengguna.
   **Jawaban: Benar**

2. Variabel `answer` harus dibuat sendiri oleh pengguna sebelum digunakan.
   **Jawaban: Salah**

3. Operator `=` digunakan untuk menjumlahkan angka dalam kuis Scratch.
   **Jawaban: Salah**

4. Kita bisa menambahkan skor menggunakan blok `change skor by 1`.
   **Jawaban: Benar**

5. Struktur `if...then...else` hanya bisa digunakan satu kali dalam satu proyek Scratch.
   **Jawaban: Salah**

## B. Pilihan Ganda (10 Soal)

1. Fungsi dari blok `ask [pertanyaan] and wait` adalah...
   a. Memberi nilai ke variabel
   b. Menyimpan skor
   c. Menampilkan pertanyaan dan menunggu jawaban
   d. Mengganti sprite
   **Jawaban: c**

2. Jawaban pengguna akan otomatis tersimpan di variabel...
   a. skor
   b. pertanyaan
   c. input
   d. answer
   **Jawaban: d**

3. Blok `if answer = "Jakarta"` berfungsi untuk...
   a. Menambah skor
   b. Mengecek jawaban pengguna
   c. Menyimpan pertanyaan
   d. Mengubah latar belakang
   **Jawaban: b**

4. Blok `say "Betul!"` digunakan untuk...
   a. Menambah skor
   b. Menampilkan gambar
   c. Memberi umpan balik
   d. Menghapus jawaban
   **Jawaban: c**

5. Untuk menyimpan skor pemain, kita memerlukan...
   a. sprite baru
   b. blok forever
   c. variabel
   d. efek suara
   **Jawaban: c**

6. Jika pengguna menjawab salah, kita bisa menggunakan...
   a. `say "Coba lagi"`
   b. `change skor by 1`
   c. `hide sprite`
   d. `play sound`
   **Jawaban: a**

7. Struktur `if...then...else` berguna untuk...
   a. Mengganti nama sprite

b. Menambahkan pertanyaan
c. Mengecek kondisi dan memberi umpan balik
d. Menjalankan musik
**Jawaban: c**

8. Apa fungsi dari list dalam pembuatan kuis di Scratch?
   a. Memutar musik
   b. Menyimpan pertanyaan dan jawaban
   c. Menyimpan sprite
   d. Menyimpan skor
   **Jawaban: b**

9. Dalam kuis Scratch, variabel `skor` biasanya diubah ketika...
   a. Sprite bergerak
   b. Jawaban benar
   c. Lagu diputar
   d. Background berubah
   **Jawaban: b**

10. Kapan variabel `answer` akan berubah?
    a. Saat skor ditambah
    b. Saat sprite berbicara
    c. Setelah `ask` dan `wait` selesai
    d. Saat pertanyaan selesai
    **Jawaban: c**

## C. Isian Singkat (5 Soal)

Isilah titik-titik di bawah ini dengan jawaban yang tepat.

16. Untuk menambahkan skor setiap kali objek tertangkap, kita bisa menggunakan blok `change skor by ___`.
    **Jawaban: 1**

17. Untuk membuat timer mundur dari 30, kita bisa menggunakan `set waktu to ___`.
    **Jawaban: 30**

18. `if skor = 100 then` digunakan untuk memeriksa apakah nilai skor sudah mencapai ___.
    **Jawaban: 100**

19. Variabel yang digunakan untuk menunjukkan tingkat kesulitan dalam game biasanya dinamakan ___.
    **Jawaban: Level**

20. Blok yang digunakan untuk menunggu dalam waktu tertentu adalah ___.
    **Jawaban: wait**

## D. Soal Eksplorasi (3 Soal Tantangan)

**Soal:**

Buatlah sebuah skenario game dimana pemain harus mengumpulkan buah dalam waktu 20 detik. Jika pemain berhasil mengumpulkan minimal 10 buah sebelum waktu habis, maka level akan meningkat. Jelaskan langkah-langkah pembuatan variabel, logika if, dan penggunaan timer untuk membuat mekanisme ini berjalan di Scratch.

**Soal:**

Bayangkan kamu sedang membuat game “Hindari Bola Api.” Dalam game ini, skor bertambah 5 poin setiap kali bola api berhasil dihindari. Tapi, jika bola api mengenai pemain, skor berkurang 10 poin. Jelaskan bagaimana kamu akan menggunakan blok if, change skor by, dan kondisi perbandingan untuk mengatur sistem skor dan akhir game.

**Soal:**

Desainlah mekanisme level dalam game Scratch yang naik secara otomatis setiap kali pemain mendapatkan kelipatan 20 skor (misalnya: 20, 40, 60). Apa yang akan terjadi saat level naik? Buatlah ide perubahan dalam game, seperti mempercepat objek atau mengurangi waktu, dan jelaskan bagaimana kamu akan memprogramnya di Scratch.

# Tugas Proyek

Kembangkan game Scratch yang orisinil atau modifikasi game yang sudah ada, dengan setidaknya dua dari tiga fitur berikut: sistem level yang meningkat secara dinamis, sistem skor yang melacak poin pemain, dan timer (hitungan mundur atau hitungan maju). Pastikan game memiliki tujuan yang jelas dan umpan balik yang informatif kepada pemain.

## Proyek 1: Game "Balap Kura-Kura Waktu 20 Detik"

**Deskripsi:**
Buatlah game di mana pemain mengendalikan kura-kura yang harus mencapai garis akhir dalam **20 detik**. Jika berhasil, pemain mendapat **10 poin**.

**Langkah-Langkah:**

- Buat karakter kura-kura dan garis finish.
- Tambahkan **timer mundur 20 detik**.
- Jika kura-kura menyentuh garis finish sebelum waktu habis → tambahkan **skor +10** dan tampilkan pesan kemenangan.
- Jika waktu habis sebelum finis → tampilkan pesan “Coba Lagi!”.
- Tambahkan efek suara ketika menang atau gagal.

**Fitur Fokus:** Timer, Skor, Tujuan Jelas.

## Proyek 2: Game “Lari Kura-Kura Bertingkat”

**Deskripsi:**
Rancang game berlevel di mana pemain harus menggerakkan kura-kura ke garis akhir dalam **30 detik**. Jika berhasil, pemain naik ke level berikutnya dan kecepatan musuh bertambah.

**Langkah-Langkah:**

- Desain musuh (misalnya kepiting atau batu bergerak).
- Gunakan **variabel level** untuk meningkatkan kesulitan (misalnya dengan `change [speed] by 1`).
- Tambahkan logika: jika mencapai finish dan waktu belum habis → naik level, musuh bertambah cepat.
- Tambahkan indikator level dan waktu di layar.

**Fitur Fokus:** Level Dinamis, Timer, Musuh Cerdas.

## Proyek 3: Game "Tangkap Apel Berlevel"

**Deskripsi:**
Buat game di mana pemain harus menangkap apel jatuh. Setiap apel yang ditangkap → skor bertambah. Saat skor mencapai 50, pemain naik ke level berikutnya dan apel jatuh lebih cepat.

**Langkah-Langkah:**

- Buat sprite apel yang jatuh acak dari atas.
- Gunakan `change [score] by 1` saat apel ditangkap.
- Saat `score = 50`, naik level dan tambah kecepatan jatuh (`change y by -X` → makin besar).
- Tambahkan animasi dan efek suara agar lebih menarik.

**Fitur Fokus:** Skor, Level, Gerakan Dinamis.

## Proyek 4: Game "Tangkap Buah Dalam Batas Waktu"

**Deskripsi:**
Pemain harus menangkap **10 buah dalam waktu 20 detik** untuk naik level. Gunakan **skor**, **timer**, dan **level** sebagai variabel utama.

**Langkah-Langkah:**

- Tambahkan **timer mundur** dari 20 detik.
- Setiap buah yang ditangkap → skor bertambah 1.
- Jika `skor = 10` sebelum waktu habis → level up.
- Setel ulang skor dan waktu setiap kali naik level.
- Tambahkan efek suara dan gerakan buah yang bervariasi.

**Fitur Fokus:** Timer, Skor, Level Bertingkat.

## Proyek 5: Game Edukasi “Jawab dan Menang”

**Deskripsi:**
Buat game kuis sederhana. Pemain harus menjawab pertanyaan dengan benar. Jawaban benar → skor +10, salah → skor -5. Jika skor mencapai 50 → naik level.

**Langkah-Langkah:**

- Buat daftar pertanyaan dan jawaban.
- Gunakan **blok if-else** untuk mengecek jawaban.
- Tambahkan skor (`change [score] by 10` untuk benar, `-5` untuk salah).
- Jika `score ≥ 50`, naik level dan tambah kesulitan (waktu jawab lebih singkat atau soal lebih sulit).
- Tambahkan suara/visual untuk jawaban benar/salah.

**Fitur Fokus:** Skor, Level, Pembelajaran Aktif.

## Proyek 6: Game “Hindari Musuh!”

**Deskripsi:**
Desain game aksi di mana pemain harus menghindari musuh. Jika menyentuh musuh → skor -5. Jika berhasil menghindari selama 10 detik → skor +2 setiap detik.

**Langkah-Langkah:**

- Tambahkan sprite musuh yang bergerak acak atau mengikuti pemain.
- Buat logika: jika menyentuh musuh → `change [score] by -5`.
- Jika tidak menyentuh musuh selama durasi → `change [score] by 2` setiap beberapa detik.
- Tambahkan umpan balik suara atau visual (misalnya efek berkedip atau suara “Oops”).

**Fitur Fokus:** Skor, Interaksi Musuh, Umpan Balik Visual.

## Catatan Tambahan: Perbedaan Pemrograman

**Apa beda antara `set [variabel] to 0` dan `change [variabel] by -1`?**

- `set [variabel] to 0` digunakan untuk **mengatur ulang nilai variabel**, misalnya saat game dimulai atau saat naik level.
- `change [variabel] by -1` digunakan untuk **mengurangi nilai variabel secara bertahap**, seperti mengurangi waktu (`timer`) setiap detik.

## Proyek Analisis Game Scratch

**Tugas:**

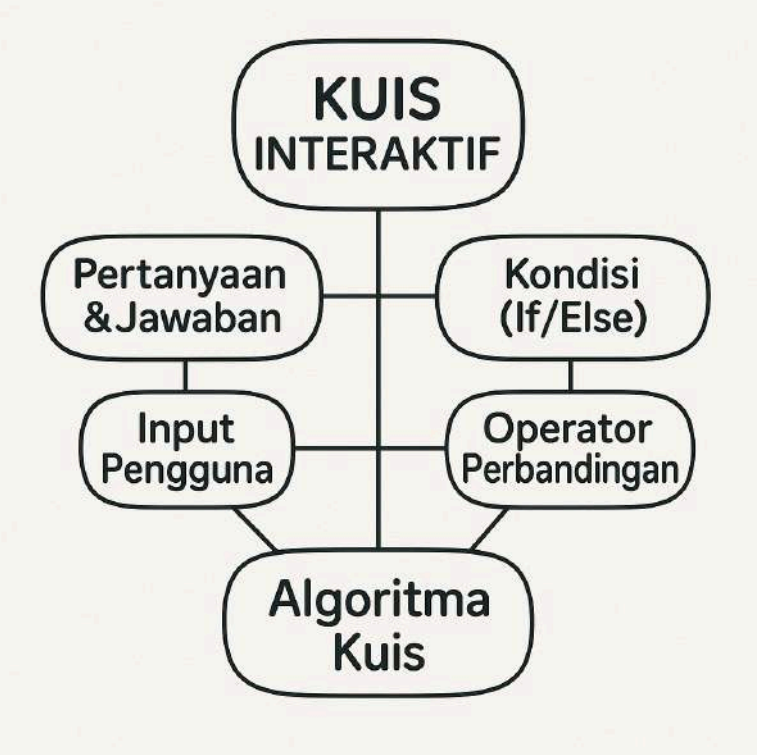
- Pilih 1 game Scratch dari galeri.
- Jelaskan apakah game tersebut memiliki **fitur skor, level, dan timer**.
- Jelaskan logika kode dalam **5–6 kalimat**: Bagaimana skor dihitung? Kapan level bertambah? Bagaimana timer bekerja?

# BAB 2: Membuat Quiz Interaktif dengan Scratch

## Tujuan Pembelajaran

Peserta didik mampu merancang dan membuat aplikasi kuis interaktif yang memiliki pertanyaan, pilihan jawaban, validasi jawaban, dan umpan balik menggunakan Scratch.

## Peta Konsep




## Apersepsi

Kalian pasti sering mengikuti kuis, baik itu di acara TV, aplikasi di HP, atau bahkan saat ulangan di sekolah. Bagaimana ya caranya sebuah aplikasi bisa tahu apakah jawaban kita benar atau salah? Di bab ini, kita akan belajar membuat kuis sendiri di Scratch!

## Penjelasan Konsep (Teori)

### Mengenal Cara Kerja Kuis Interaktif di Scratch

Pernahkah kalian mengikuti kuis di aplikasi game, di televisi, atau saat ulangan di sekolah? Kalian menjawab pertanyaan, lalu layar menunjukkan apakah jawaban kalian benar atau salah. Tapi... bagaimana komputer bisa tahu itu? Apa yang terjadi di balik layar?

Nah, di bagian ini, kita akan belajar teori dasar yang sangat penting sebelum membuat kuis digital kita sendiri menggunakan **Scratch**. Kita akan memahami cara menerima jawaban dari pemain, memeriksa kebenarannya, memberi skor, dan menyusun pertanyaan secara berurutan. Yuk, kita mulai!

## Menerima Jawaban dari Pengguna (Input)

Agar kuis bisa berinteraksi dengan pemain, kita harus bisa *mengajukan pertanyaan* dan *menerima jawaban*. Di Scratch, ini bisa dilakukan dengan mudah menggunakan blok berikut:


- **Blok `ask [pertanyaan] and wait`**
  Blok ini digunakan untuk menampilkan pertanyaan di layar dan *menunggu pemain mengetik jawaban*. Scratch akan berhenti sejenak sampai pemain mengisi jawabannya.

- **Variabel `answer` (bawaan Scratch)**
  Jawaban yang diketik oleh pemain akan otomatis disimpan dalam variabel bernama `answer`. Kita bisa menggunakan variabel ini untuk diperiksa nanti.
  **Catatan penting:** Pastikan *segera menggunakan* nilai dari `answer` setelah blok `ask`, karena jika ada pertanyaan berikutnya, nilainya akan digantikan!

## Memeriksa Apakah Jawaban Benar (Validasi Jawaban)

Setelah kita mendapatkan jawaban dari pemain, selanjutnya kita perlu **memeriksa apakah jawaban itu benar atau salah**. Caranya adalah dengan menggunakan struktur kondisi.

**Operator Perbandingan (=)**
Kita memakai operator ini untuk membandingkan isi variabel `answer` dengan jawaban yang seharusnya.
Contoh:

```
if <(answer) = [Jakarta]> then …
```

- **Blok `if ... then ... else`**
  Blok ini memungkinkan kita menjalankan dua kemungkinan kode: satu jika jawaban benar, satu lagi jika salah.
  - **Bagian `if`**: dijalankan *jika kondisi benar* (misalnya, jawaban benar).
  - **Bagian `else`**: dijalankan *jika kondisi salah* (misalnya, jawaban salah).

- **Blok `say [pesan]`**
  Digunakan untuk memberi *umpan balik langsung* ke pemain.
  Misalnya:

  - `say [Betul! Kamu hebat!]`
  - `say [Oops! Jawabanmu salah. Coba lagi, ya!]`

## Menyusun Kuis dengan Banyak Pertanyaan

Sebuah kuis tidak seru kalau hanya satu pertanyaan, kan? Nah, untuk membuat kuis menjadi lebih seru dan menantang, kita perlu menyusun beberapa pertanyaan secara berurutan. Ini dia caranya:

- **Urutan Pertanyaan**
  Kita menulis beberapa blok `ask`, lalu menggunakan `if/else` setelah masing-masing untuk mengecek jawaban. Setiap pertanyaan bisa ditampilkan satu per satu.

- **Membuat Variabel `Skor Kuis`**
  Agar pemain tahu seberapa banyak jawaban yang benar, kita buat variabel baru bernama `Skor Kuis`.
  - Skor ditambah **1** setiap kali pemain menjawab benar.
  - Gunakan blok `change [Skor Kuis] by (1)` di dalam bagian `if`.


## (Bonus Level) Menggunakan List untuk Pertanyaan dan Jawaban

Bagi kalian yang merasa siap untuk tantangan tambahan, ini dia:
Scratch juga punya **struktur data bernama List** yang bisa digunakan untuk menyimpan *banyak pertanyaan* dan *jawaban* dalam satu tempat.

- **Apa itu List?**
  List seperti "daftar belanja digital" — kita bisa menyimpan banyak data di dalamnya, misalnya:
  - List `Pertanyaan` berisi: "Ibukota Indonesia?", "2+2?", dll.

    - List `Jawaban` berisi: "Jakarta", "4", dll.

- **Manfaat List:**
  Dengan list, kuis kita jadi lebih rapi, fleksibel, dan bisa di-*update* dengan mudah. Ini sangat berguna kalau pertanyaannya banyak.

### Kesimpulan Mini

Dengan memahami konsep-konsep di atas, kalian akan siap membuat kuis interaktif yang tidak hanya seru, tetapi juga pintar! Kalian sekarang tahu cara:

- Mengajukan pertanyaan,
- Menerima jawaban,
- Memeriksa kebenaran jawaban,
- Memberi umpan balik dan skor,
- (Bahkan) menyusun kuis besar dengan list!

Siap jadi pembuat kuis profesional? Yuk kita lanjut ke praktik membuatnya!

## Contoh Kasus atau Ilustrasi

### Kasus 1: Kuis Sejarah Tokoh Bangsa

**Ilustrasi:**
Bayangkan kamu jadi host kuis TV nasional. Layar menyala, musik dramatis terdengar, dan kamu tanya,
"Siapakah presiden pertama Indonesia?"
Temanmu di rumah menjawab lewat mikrofon: "Soekarno".
Kamu cek jawabannya dan berkata: "Betul! Kamu hebat!" lalu nilai bertambah 1 poin.
Kalau salah? Kamu tetap ramah dan memberi tahu jawaban yang benar.

**Tujuan:**
Membuat program Scratch yang bisa:

- Menampilkan pertanyaan
- Menerima jawaban
- Memeriksa apakah benar atau salah
- Menambahkan skor jika benar

### Kasus 2: Detektif Sejarah

**Ilustrasi:**
Kamu adalah detektif yang sedang mengungkap rahasia kemerdekaan Indonesia.

NPC (tokoh virtual) di layar bertanya,
**"Kapan Indonesia merdeka?"**
Kamu harus menjawab dengan tepat: **"17 Agustus 1945"**.
Kalau benar, layar menunjukkan bendera berkibar dan suara “Hebat!” muncul.
Kalau salah, tokoh itu bilang: “Kurang tepat. Jawabannya 17 Agustus 1945.”

**Tujuan:**
Melatih ketepatan jawaban dalam format teks, dan membuat efek visual jika jawaban benar atau salah.

## Kasus 3: Kuis Berantai – 3 Pertanyaan Berurutan

**Ilustrasi:**
Seperti kuis "Siapa Cepat, Dia Dapat". Kamu harus menjawab 3 pertanyaan sejarah secara berurutan:

1. Siapa presiden pertama Indonesia?
2. Kapan Indonesia merdeka?
3. Dimana proklamasi dibacakan?

Setiap jawaban benar membuat skor bertambah. Jika salah, program tetap menunjukkan jawaban yang benar.

**Tujuan:**
Membuat sistem kuis berurutan dengan banyak pertanyaan, penilaian, dan komentar otomatis.

## Kasus 4: Tantangan Level Sejarah

**Ilustrasi:**
Kamu membuat game Scratch bertingkat: setiap jawaban benar akan membuka "level" baru.

- Level 1: Presiden Pertama
- Level 2: Tanggal Kemerdekaan
- Level 3: Isi Teks Proklamasi

Jika salah, pemain mengulang level itu sampai benar!

**Tujuan:**
Menggunakan percabangan logika (`if`, `else`) untuk membuat kuis berbasis level.

## Kasus 5: Kuis Waktu Nyata – Siapa Cepat Dia Dapat!

**Ilustrasi:**
Bayangkan kamu bermain kuis dengan timer!
Pertanyaan muncul: **"Apa nama lagu kebangsaan Indonesia?"**
Pemain hanya punya 10 detik untuk menjawab.
Jika jawaban "Indonesia Raya" muncul tepat waktu, skor bertambah dan sprite

berkata: “Jawaban cepat dan tepat!”
Jika tidak, waktu habis dan sprite berkata: “Sayang, waktumu habis!”

**Tujuan:**
Menggunakan konsep waktu (`timer`) untuk membuat kuis jadi lebih menantang.

### Kasus 6: Tantangan Duel Sejarah

**Ilustrasi:**
Dua pemain, satu komputer.
Pemain 1 dapat pertanyaan: “Siapa wakil presiden pertama Indonesia?”
Pemain 2 menjawab pertanyaan lain: “Apa isi teks proklamasi?”
Skor masing-masing ditampilkan. Di akhir, Scratch mengumumkan siapa yang menang.

**Tujuan:**
Menggunakan dua variabel skor dan membandingkan hasil akhir.

Semua kasus ini bisa dibuat menggunakan blok:

- `ask ... and wait`
- `if ... then ... else`
- `join`, `=` (operator)
- `set skor`, `change skor by`
- `say ... for ... seconds`

## Contoh Soal dan Pembahasan

### Soal 1: Siapa Menjawab, Siapa Menilai!

**Pertanyaan:**
Kamu ingin membuat program Scratch yang menampilkan pertanyaan: “Ibukota Indonesia?” dan memeriksa apakah pemain menjawab “Jakarta”.
Blok apa saja yang perlu kamu gunakan dan bagaimana urutannya?

**Pembahasan:**
Agar program bisa bertanya dan memeriksa jawaban:

1. Gunakan `ask [Ibukota Indonesia?] and wait` – ini akan menampilkan pertanyaan dan menunggu jawaban.
2. Gunakan `if <(answer) = [Jakarta]> then` untuk memeriksa apakah jawaban benar.
3. Di dalam blok `if`, tambahkan `say [Betul! Kamu hebat!]`.
4. Tambahkan bagian `else` dengan `say [Oops! Jawabanmu salah. Coba lagi, ya!]`.

Ini menunjukkan kemampuan Scratch memahami input dari pemain dan memberi tanggapan sesuai.

## Soal 2: Skor-ku, Nilai-ku

**Pertanyaan:**
Bagaimana caranya agar skor pemain bertambah 1 setiap kali mereka menjawab pertanyaan dengan benar?

**Pembahasan:**
Langkah-langkah yang dibutuhkan:

1. Buat variabel baru dengan nama `Skor Kuis`.
2. Tambahkan blok `change [Skor Kuis v] by (1)` di dalam bagian `if` (jawaban benar).
3. Jangan letakkan blok ini di bagian `else`, agar tidak menambah skor jika jawaban salah.

Dengan begitu, setiap kali pemain menjawab dengan benar, skor mereka akan naik secara otomatis!

## Soal 3: Berderet Pertanyaan, Seru Banget!

**Pertanyaan:**
Kamu ingin membuat kuis dengan 3 pertanyaan. Bagaimana cara menyusunnya agar ditampilkan satu per satu dan mengecek setiap jawabannya?

**Pembahasan:**
Gunakan urutan berikut:

1. `ask [Pertanyaan 1] and wait`, lalu `if (answer = ...)`
2. Tambahkan umpan balik dan skor seperti sebelumnya.
3. Ulangi untuk Pertanyaan 2 dan 3 dengan blok yang sama, disusun secara berurutan.

Contoh:

```
ask [Ibukota Indonesia?] and wait
if <(answer) = [Jakarta]> then
  change [Skor Kuis] by (1)
  say [Hebat!]
else
  say [Salah!]

ask [2+2?] and wait
if <(answer) = [4]> then
  change [Skor Kuis] by (1)
  say [Bagus!]
```

```
else
  say [Coba lagi!]
```

## Soal 4: Misteri Jawaban dalam List

**Pertanyaan:**
Apa keuntungan menggunakan **List** untuk menyimpan pertanyaan dan jawaban dalam kuis interaktif? Bagaimana cara sederhana menggunakannya?

**Pembahasan:**
Keuntungan:

- Kuis lebih rapi dan bisa memiliki banyak soal.
- Kita bisa memakai perulangan (loop) untuk menanyakan semua pertanyaan.
  Cara sederhana:
  1. Buat dua list: `List Pertanyaan`, `List Jawaban`.
  2. Tambahkan pertanyaan dan jawaban sesuai urutan.
  3. Gunakan perulangan `repeat (length of List Pertanyaan)` untuk:
     - Menampilkan pertanyaan ke-i: `item (i) of List Pertanyaan`
     - Mengecek apakah `answer = item (i) of List Jawaban`

Ini seperti membuat “bank soal” otomatis!

## Soal 5: Tantangan Rahasia Skor

**Pertanyaan:**
Dalam suatu kuis, kamu menemukan kode:

```
if <(answer) = [Merah]> then
  change [Skor Kuis] by (1)
  say [Jawabanmu benar!]
else
  say [Jawaban salah! Skor tidak berubah]
```

Jika pemain menjawab “merah” (dengan huruf kecil semua), apakah skornya bertambah? Mengapa?

**Pembahasan:**
**Tidak!** Karena Scratch membandingkan teks secara **sensitif huruf besar-kecil**.

`Merah` ≠ `merah`.

Untuk menghindari ini, kamu bisa ubah jawaban dan `answer` menjadi huruf kecil semua dengan fungsi seperti `lowercase` (fitur tambahan). Atau, beri tahu pemain untuk menjawab dengan huruf besar sesuai contoh.

## Fakta Menarik / Fun Facts

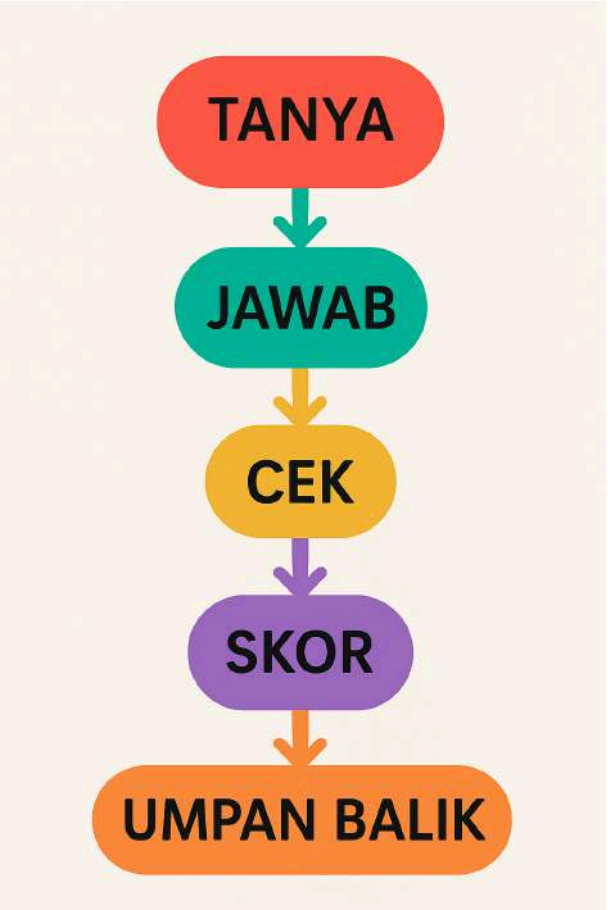
Kuis interaktif sudah ada sejak zaman dahulu dalam bentuk permainan papan atau teka-teki. Dengan komputer, kuis bisa menjadi lebih dinamis dan memberikan umpan balik instan, seperti yang kalian alami di aplikasi edukasi!

## Tips Belajar atau Tips Cepat Menghafal

### Tips:

- **Ingat "tanya dan periksa jawaban":** Setiap kali kalian mau membuat kuis, ingat dua langkah utama ini.
- Ingat urutannya: **ask** → **cek** → **nilai** → **komentar**
- **Flowchart:** Gambarlah alur kuis kalian di kertas (pertanyaan -> cek jawaban -> umpan balik) sebelum mulai membuat kode.

### Mind Mapping: Cara Kerja Kuis Scratch




**Cerita Seru:**

Bayangkan kamu membuat game kuis sejarah Indonesia. Setiap jawaban benar akan membuat karakter utama (sprite) melompat senang dan naik level. Kalau salah? Sprite kecewa dan bilang, “Yuk belajar lagi!”

## Aktivitas Siswa

### Aktivitas 1: Kuis Mata Pelajaran Favorit

**Deskripsi:**
Kamu akan membuat kuis interaktif tentang pelajaran favoritmu (contohnya: IPA, IPS, Matematika, Bahasa Indonesia). Kuis ini berisi **minimal 3 pertanyaan**. Setiap pertanyaan harus memberikan **umpan balik** (jawaban benar atau salah), dan ada **sistem skor**.

**Langkah-langkah:**

1. Buka **Scratch** dan buat proyek baru.
2. Buat atau pilih karakter (sprite) yang akan menyampaikan pertanyaan.
3. Gunakan **“say”** atau **“ask”** untuk menampilkan pertanyaan.

4. Gunakan blok **"if...then...else"** untuk mengecek apakah jawaban benar atau salah.
5. Tambahkan **variabel “skor”** untuk menghitung poin.
6. Tambahkan 3 soal, dan setiap kali soal dijawab benar, skor bertambah.
7. Setelah soal terakhir, tampilkan skor akhir.

## Aktivitas 2: Kuis Tebak Gambar / Suara

**Deskripsi:**
Kamu membuat kuis seru di Scratch di mana pemain harus menebak isi gambar atau suara. Gunakan **fitur “costume” (untuk gambar)** atau **“sound” (untuk suara)**.

**Langkah-langkah:**

1. Buat proyek baru di Scratch.
2. Tambahkan beberapa **gambar** (costume) atau **suara** (di tab "Sounds").
3. Buat sprite yang menampilkan gambar atau memutar suara.
4. Tampilkan pertanyaan seperti: “Apa ini?” atau “Suara siapa ini?”
5. Buat pilihan jawaban, misalnya A, B, dan C (gunakan sprite tombol).
6. Saat tombol diklik, tampilkan apakah jawabannya benar atau salah.
7. Tambahkan skor dan umpan balik seperti “Benar!” atau “Coba lagi!”

## Aktivitas 3: Kuis Klik Jawaban (Tombol A, B, C)

**Deskripsi:**
Buat kuis sederhana di mana pemain harus memilih jawaban dengan mengklik tombol A, B, atau C. Setelah diklik, akan muncul informasi apakah jawabannya benar atau salah.

**Langkah-langkah:**

1. Buat sprite untuk tombol **A**, **B**, dan **C**.
2. Tampilkan pertanyaan dengan karakter utama.
3. Saat salah satu tombol diklik:
    - Gunakan **“when this sprite clicked”**.
    - Cek apakah pilihan itu benar.
    - Tampilkan pesan “Benar” atau “Salah”.
4. Bisa juga tambahkan **skor** jika ingin.

## Aktivitas 4: Tantangan Timer (Waktu Terbatas)

**Deskripsi:**
Tambahkan waktu untuk menjawab setiap soal. Kalau waktu habis, pertanyaan langsung diganti atau dianggap salah.

**Langkah-langkah:**

1. Buat **variabel waktu** (misal: “timer”).
2. Saat soal dimulai, atur timer jadi 10 (detik).

3. Gunakan blok **"repeat until"** untuk mengurangi waktu.
4. Jika waktu = 0 dan belum dijawab, tampilkan “Waktu habis!”
5. Bisa tambahkan efek suara atau animasi kalau waktu habis.

### Aktivitas 5: Main Bersama Teman

**Deskripsi:**
Setelah membuat kuismu, **tukar kuis dengan teman**. Mainkan kuis buatan temanmu dan kasih nilai dari jawabannya!

**Langkah-langkah:**

1. Simpan dan beri nama proyek kuismu.
2. Tukar proyek dengan teman lewat file atau Scratch Studio.
3. Mainkan kuis buatan temanmu.
4. Catat jawaban teman, beri skor dari jawaban yang benar.
5. Kasih komentar: “Seru!”, “Gambarnya keren!”, atau “Soalnya menantang!”

## Rangkuman

Dengan bantuan **blok** `ask and wait`, kita bisa **menerima jawaban dari pengguna** secara langsung di Scratch. Jawaban tersebut otomatis tersimpan di dalam variabel *answer*. Setelah itu, kita dapat **memeriksa kebenaran jawaban** menggunakan **operator perbandingan** dan **struktur** `if...then...else` untuk memberikan **umpan balik yang sesuai**—apakah jawaban benar atau salah.

Jika jawaban benar, kita bisa **menambah skor** dengan blok `change skor by 1`, dan menampilkan komentar positif menggunakan **blok** `say`. Sebaliknya, jika jawaban salah, kita bisa memberikan pesan motivasi atau petunjuk.

Dengan alur ini, kita bisa dengan mudah membangun **kuis interaktif** yang **menarik dan mendidik**. Scratch juga memungkinkan kita menambahkan elemen tambahan seperti **gambar, suara**, bahkan **batas waktu**, agar kuis menjadi lebih seru dan menantang. Kuis ini tidak hanya menyenangkan, tetapi juga merupakan **alat yang efektif untuk belajar dan menguji pemahaman** secara langsung.

## Latihan Soal

### A. Benar atau Salah (5 Soal)

1. Blok `ask ... and wait` digunakan untuk memberikan umpan balik ke pengguna.
   **Jawaban: Salah**

2. Jawaban pengguna disimpan dalam variabel `answer`.
   **Jawaban: Benar**

3. Blok `if ... then ... else` hanya bisa digunakan untuk satu kondisi.
   **Jawaban: Salah**

4. Skor dapat ditambahkan dengan blok `change skor by 1`.
   **Jawaban: Benar**

5. Scratch tidak mendukung penggunaan daftar (list).
   **Jawaban: Salah**

## B. Pilihan Ganda (10 Soal)

1. Apa yang dilakukan blok `ask "Siapa nama kamu?" and wait`?
   a. Memberi skor
   b. Menyimpan data
   c. Menampilkan pertanyaan dan menunggu jawaban
   d. Memberi umpan balik
   **Jawaban: c**

2. Jawaban dari pengguna secara otomatis masuk ke variabel...
   a. skor
   b. nama
   c. result
   d. answer
   **Jawaban: d**

3. Untuk mengecek apakah jawaban pengguna benar, kita menggunakan...
   a. Blok suara
   b. List
   c. Operator =
   d. Variabel baru
   **Jawaban: c**

4. Struktur `if ... then ... else` digunakan untuk...
   a. Mengganti sprite
   b. Menambah skor
   c. Memberi respon benar atau salah
   d. Membuat pertanyaan baru
   **Jawaban: c**

5. Untuk menyimpan skor dalam kuis, kita menggunakan...
   a. List
   b. Blok suara
   c. Variabel
   d. Sprite

**Jawaban: c**

6. Apa fungsi dari blok `say "Betul!"`?
   a. Mengubah skor
   b. Menjawab pertanyaan
   c. Memberikan jawaban
   d. Memberi umpan balik visual
   **Jawaban: d**

7. Apa yang dilakukan blok `if answer = "Surabaya"`?
   a. Memberi skor
   b. Mengecek jawaban
   c. Menyimpan nama kota
   d. Menampilkan pertanyaan
   **Jawaban: b**

8. Mengapa list digunakan dalam kuis lanjutan?
   a. Untuk menggambar sprite
   b. Untuk menyimpan pertanyaan dan jawaban
   c. Untuk mencetak nilai
   d. Untuk menyimpan warna
   **Jawaban: b**

9. Bagaimana kita memberikan tanggapan saat jawaban salah?
   a. Dengan `say "Salah"`
   b. Dengan `delete`
   c. Dengan `wait`
   d. Dengan `next costume`
   **Jawaban: a**

10. Bagaimana cara mengatur banyak pertanyaan dalam kuis?
    a. Gunakan satu sprite
    b. Gunakan banyak variabel
    c. Gunakan beberapa `ask` dan `if`
    d. Gunakan banyak background
    **Jawaban: c**

## C. Isian Singkat (5 Soal)

1. Blok Scratch untuk menampilkan pertanyaan dan menunggu jawaban adalah _____.
   **Jawaban: ask and wait**

2. Variabel yang menyimpan jawaban pengguna adalah _____.
   **Jawaban: answer**

3. Untuk membandingkan jawaban pengguna dengan jawaban benar, digunakan operator _____.
   **Jawaban: =**

4. Untuk menambah nilai skor ketika jawaban benar, digunakan blok _____.
   **Jawaban: change skor by 1**

5. Untuk menyimpan banyak pertanyaan, kita bisa menggunakan _____.
   **Jawaban: list**

### D. Soal Eksplorasi (3 Soal)

**Soal:**

Jelaskan dengan tiga kalimat bagaimana kuis interaktif di Scratch bekerja dari awal hingga akhir. Sertakan bagaimana pertanyaan diberikan, bagaimana jawaban dicek, dan bagaimana umpan balik diberikan.

**Soal:**

Bagaimana kamu menangani perbedaan penulisan huruf besar dan kecil dalam jawaban pengguna (misalnya: "soekarno", "SOEKARNO", "Soekarno") agar dianggap benar semua? Tuliskan pendekatan logikanya.

**Soal:**

Buatlah satu pertanyaan kuis tentang "Hewan Langka" menggunakan format teks Scratch seperti berikut:

```
ask "Apa nama hewan langka yang hanya ada di Papua?" and wait
if answer = "Cendrawasih" then say "Betul!" else say "Salah, jawabannya Cendrawasih."
```

Tuliskan blok tersebut lengkap seperti contoh di atas.

## Tugas Proyek

### Proyek 1: Kuis Sejarah Indonesia: Jelajah Waktu

**Deskripsi Proyek:**
Buatlah kuis interaktif bertema sejarah Indonesia dengan minimal 5 pertanyaan pilihan ganda. Sertakan sistem skor otomatis, umpan balik benar/salah, dan desain visual yang menarik agar pengguna semangat menjawab.

**Langkah-langkah Saran:**

- Tentukan topik sejarah (misal: Proklamasi, Kerajaan Hindu-Buddha, Perjuangan Kemerdekaan).
- Rancang pertanyaan dan jawaban pilihan ganda di Scratch.

- Gunakan **“if-else” blocks** untuk memeriksa jawaban.
- Tambahkan variabel skor dan efek suara atau animasi untuk umpan balik.
- Hias latar dengan tema sejarah agar terasa seperti “mesin waktu digital”.

## Proyek 2: Kuis Benar atau Salah: Fakta Seru

**Deskripsi Proyek:**
Kembangkan kuis dengan minimal 5 pernyataan yang harus dijawab dengan "Benar" atau "Salah". Tampilkan hasilnya secara langsung dan hitung skor akhir pemain.

**Langkah-langkah Saran:**

- Buat daftar 5 fakta unik atau mengejutkan seputar pengetahuan umum atau pelajaran sekolah.
- Tampilkan pernyataan satu per satu dengan dua tombol pilihan: "Benar" dan "Salah".
- Gunakan variabel untuk menghitung jumlah jawaban benar.
- Tambahkan efek suara lucu saat jawaban salah, dan suara kemenangan jika benar!

## Proyek 3: Kuis Kilat: Tantangan 5 Detik!

**Deskripsi Proyek:**
Buat kuis dengan batas waktu 5 detik per soal. Pemain harus menjawab cepat sebelum waktunya habis. Cocok untuk melatih refleks dan pengetahuan secara bersamaan!

**Langkah-langkah Saran:**

- Tambahkan variabel waktu menggunakan **timer** di Scratch.
- Setiap pertanyaan dimulai dengan menghitung mundur dari 5.
- Gunakan **“wait” dan “reset timer”** untuk mengatur batas waktu.
- Jika waktu habis, tampilkan “Waktu Habis!” dan lanjut ke soal berikutnya.

## Proyek 4: Jawaban Ganda: Kuis Fleksibel

**Deskripsi Proyek:**
Buat pertanyaan dengan dua atau lebih jawaban yang benar. Misalnya: “Hewan peliharaan favorit?” bisa dijawab “kucing” atau “Kucing”.

**Langkah-langkah Saran:**

- Gunakan **“or” operator** di block **“if”** untuk menerima lebih dari satu jawaban.
- Buat satu pertanyaan terbuka dan uji berbagai variasi jawaban.
- Tampilkan umpan balik “Jawaban Kamu Termasuk Benar!” bila jawaban cocok dengan opsi.
- Cocok untuk pertanyaan dengan banyak kemungkinan jawaban.

## Proyek 5: Kuis Tentang Teman: Seru-Seruan Bareng!

**Deskripsi Proyek:**
Rancang kuis tentang teman sekelasmu! Tanyakan hal-hal menyenangkan seperti hobi, makanan favorit, atau warna kesukaannya. Tambahkan komentar lucu dan efek animasi!

**Langkah-langkah Saran:**

- Minta izin temanmu terlebih dahulu untuk menggunakan data mereka.
- Susun pertanyaan berdasarkan fakta ringan dan menyenangkan.
- Buat efek suara atau teks lucu jika jawaban salah, misalnya “Ups, itu makanan favorit dia yang ke-2!”
- Cocok untuk acara kelas atau hiburan saat istirahat.

## Proyek 6: Kuis Mini Festival: Gabungkan Semua!

**Deskripsi Proyek (Opsional Bonus):**
Gabungkan semua jenis kuis di atas menjadi satu proyek besar seperti “Mini Festival Kuis”. Pemain bisa memilih jenis kuis yang ingin dimainkan dari menu awal.

**Langkah-langkah Saran:**

- Buat menu utama dengan tombol untuk memilih jenis kuis.
- Gunakan **broadcast** untuk berpindah antar bagian kuis.
- Tambahkan skor akhir total untuk keseluruhan sesi.
- Cocok sebagai proyek akhir semester atau pameran karya digital.

Setiap proyek ini bisa menjadi tantangan yang seru dan edukatif! Jangan takut mencoba ide-ide unik, dan ingat: **coding itu seperti petualangan – selalu ada hal baru untuk ditemukan!**

# BAB 3: Latihan Computational Thinking dalam Game

## Tujuan Pembelajaran

Peserta didik mampu mengidentifikasi dan secara sadar menerapkan empat pilar utama Computational Thinking (Decomposition, Pattern Recognition, Abstraction, dan Algorithm Design) dalam proses desain, pengembangan, dan perbaikan game.

## Peta Konsep


## Apersepsi

Pernahkah kalian merasa bingung saat membuat game yang kompleks? Atau kesulitan mencari kesalahan (bug) di program kalian? Ternyata, saat kita membuat game, otak kita secara otomatis sudah menggunakan cara berpikir seperti seorang ilmuwan komputer. Cara berpikir inilah yang disebut Computational Thinking (CT)! Yuk, kita pelajari lebih dalam!

# Penjelasan Konsep (Teori)

Saat kamu membuat sebuah game, baik di Scratch maupun platform lainnya, sebenarnya kamu sedang melakukan sesuatu yang sangat mirip dengan pekerjaan seorang ilmuwan komputer! Tanpa kamu sadari, kamu sedang memakai sebuah cara berpikir khusus yang disebut **Computational Thinking**, atau disingkat **CT**. Nah, di bagian ini, kita akan membongkar apa itu CT, bagaimana cara kerjanya, dan mengapa penting banget untuk dipahami jika kamu ingin menjadi *game creator* yang handal!

## Apa Itu Computational Thinking?

**Computational Thinking (CT)** adalah sebuah metode berpikir logis dan sistematis yang digunakan untuk menyelesaikan masalah—khususnya masalah yang bisa dipecahkan dengan bantuan komputer.

CT bukan hanya soal menulis kode, tapi lebih ke *bagaimana kita memecah dan memahami masalah dengan cara yang bisa dimengerti komputer*. Ini adalah *cara berpikir seorang programmer sejati*!

## Empat Pilar Utama Computational Thinking

CT terdiri dari **empat pilar utama**. Masing-masing pilar punya fungsi penting dalam proses membuat, mengembangkan, dan memperbaiki game. Yuk, kita pelajari satu per satu:

### 1. Decomposition (Dekomposisi)

> *“Pecah masalah besar jadi potongan kecil.”*

**Apa artinya?**
Dekomposisi adalah proses memecah sebuah masalah besar menjadi bagian-bagian kecil yang lebih mudah ditangani.

**Dalam dunia game:**
Bayangkan kamu sedang membuat game petualangan. Tentu tidak dikerjakan sekaligus, kan? Kamu bisa memecahnya menjadi beberapa komponen:

- Karakter pemain (sprite utama)
- Musuh (sprite musuh)
- Sistem skor
- Level game
- Efek suara dan musik
- Tampilan latar belakang

Setiap bagian bisa dibuat dan diuji secara terpisah. Ini membuat proses pembuatan game lebih terorganisir dan tidak membingungkan.

### 2. Pattern Recognition (Pengenalan Pola)

> *“Cari tahu pola atau kesamaan dari berbagai hal.”*

**Apa artinya?**
Pengenalan pola adalah kemampuan untuk menemukan kesamaan dalam data, perilaku, atau struktur.

**Dalam dunia game:**
Contoh: Semua musuh dalam game kamu mungkin bergerak naik-turun dengan kecepatan yang sama. Semua koin punya aturan "jika menyentuh pemain → hilang + tambah skor".

Nah, kalau kamu mengenali pola ini, kamu bisa menulis satu bagian kode saja, lalu gunakan ulang untuk objek yang serupa. Efisien, kan?

### 3. Abstraction (Abstraksi)

> *“Fokus pada hal penting, abaikan yang tidak perlu.”*

**Apa artinya?**
Abstraksi adalah kemampuan untuk menyaring informasi penting dan mengabaikan detail yang tidak relevan.

**Dalam dunia game:**
Misalnya, kamu tidak perlu mengatur setiap piksel karakter. Yang penting adalah:

- Karakter bisa bergerak ke kiri dan kanan
- Karakter bisa melompat
- Karakter bisa mengambil item atau menyerang musuh

Di Scratch, kamu bisa membuat *blok kustom* untuk mengatur fungsi tertentu. Ini juga bentuk dari abstraksi: menyederhanakan agar tidak perlu menulis ulang kode yang sama.

### 4. Algorithm Design (Perancangan Algoritma)

> *“Buat langkah-langkah solusi yang jelas dan teratur.”*

**Apa artinya?**
Ini adalah proses merancang urutan langkah yang harus dilakukan untuk menyelesaikan sebuah tugas atau masalah.

**Dalam dunia game:**
Contoh algoritma:

- Cara karakter bergerak saat menekan tombol panah
- Langkah-langkah ketika skor bertambah
- Urutan ketika pemain naik level

Dengan algoritma yang baik, game kamu akan berjalan mulus dan sesuai rencana.

## Debugging: Detektif dalam Dunia Coding!

Kadang-kadang, meskipun kamu sudah menulis kode dengan benar, *tiba-tiba ada bug!* (alias kesalahan dalam program). Tenang, itu wajar. Bahkan programmer profesional juga sering mengalaminya.

**Kenapa bisa terjadi bug?**

- Salah ketik (*typo*)
- Logika yang tidak tepat
- Blok kode yang tidak diurutkan dengan benar

**Teknik Debugging Sederhana di Scratch:**

- **Cek blok satu per satu**: Jalankan bagian-bagian kecil kode untuk melihat di mana masalahnya.
- **Gunakan blok say**: Menampilkan nilai variabel agar kita tahu apa yang terjadi di dalam game.
- **Gunakan mode "Step-by-step"**: Fitur di Scratch yang membantu menjalankan kode *blok demi blok*, jadi kita bisa melihat prosesnya secara perlahan.
- **Isolasi masalah**: Nonaktifkan bagian kode tertentu sementara waktu untuk mencari penyebab error.

## Optimasi: Biar Game Kamu Lebih Keren dan Lancar!

Setelah game kamu berjalan, apakah selesai? Belum tentu! Kamu masih bisa **mengoptimalkan** kode supaya:

- Lebih ringkas dan rapi
- Tidak ada bagian yang berulang-ulang
- Game berjalan tanpa *lag*

**Tips optimasi:**

- Gunakan **broadcast** untuk mengatur komunikasi antar sprite
- Buat **blok kustom** untuk fungsi yang dipakai berulang
- Periksa dan buang kode yang tidak terpakai

## Kesimpulan

Computational Thinking adalah *alat berpikir super penting* bagi siapa pun yang ingin membuat game (atau bahkan jadi ilmuwan komputer!). Dengan memahami:

- **Decomposition** (memecah)
- **Pattern Recognition** (mengenali pola)
- **Abstraction** (menyaring yang penting)
- **Algorithm Design** (membuat langkah solusi)

...kamu akan semakin pintar, terampil, dan siap menghadapi tantangan dalam dunia pengembangan game!

Kalau kamu sudah siap, yuk kita lanjut ke latihan-latihannya dan buktikan kalau kamu bisa berpikir seperti ilmuwan komputer sejati!

# Contoh Kasus atau Ilustrasi

## Kasus 1: Flappy Bird Versi Kamu!

Bayangkan kamu ingin membuat game sederhana seperti **Flappy Bird**. Di dalam game itu, kamu mengendalikan seekor burung yang harus terbang melewati pipa-pipa tanpa menabrak. Gampang? Belum tentu!

Mari kita lihat game ini dari sudut pandang **Computational Thinking**:

- **Decomposition**: Kita pecah dulu, ya! Ada burung, ada pipa, ada sistem skor, ada tombol mulai, dan ada layar game over.
- **Pattern Recognition**: Semua pipa bergerak dari kanan ke kiri. Pola gerak mereka sama. Kita bisa pakai satu kode yang sama untuk semua pipa.
- **Abstraction**: Kita nggak usah mikirin bulu burung atau bentuk pipanya. Fokus aja ke logika: saat burung menyentuh pipa, game over!
- **Algorithm Design**: Buat urutan langkah-langkah. Misalnya: jika tombol ditekan, burung naik. Kalau tidak, burung turun. Kalau burung kena pipa, game selesai.

Seru kan? Pikiranmu seperti ilmuwan komputer!

## Kasus 2: Ilustrasi Debugging dalam Scratch

Pernahkah kamu membuat game di Scratch, tapi tiba-tiba skornya tidak bertambah, atau karakternya malah tembus dinding? Tenang, itu hal yang wajar terjadi saat coding. Justru di sinilah kita belajar keterampilan penting dalam dunia pemrograman: *debugging*.

Debugging adalah proses mencari dan memperbaiki kesalahan (atau *bug*) dalam program kita. Misalnya, kamu membuat permainan dimana pemain harus mengumpulkan bintang agar skornya naik. Tapi saat dicoba, bintang berhasil dikumpulkan, tapi skor tetap nol. Nah, itu tandanya ada *bug*.

Untuk memperbaikinya, kita bisa mengikuti langkah-langkah debugging secara sistematis:

1. **Pahami masalahnya**: Apa yang tidak berjalan seperti seharusnya? Misalnya, skor tidak bertambah saat bintang diambil.

2. **Periksa kode yang terkait**: Lihat bagian kode yang seharusnya menambahkan skor. Apakah benar perintah "ubah skor" sudah ditulis? Apakah posisinya tepat?
3. **Uji coba bagian kecil**: Coba jalankan kode itu secara terpisah atau tambahkan blok "say" untuk menampilkan nilai skor, agar kamu tahu apakah perintah itu dijalankan.
4. **Perbaiki dan uji kembali**: Setelah menemukan kesalahannya, perbaiki. Misalnya, tambahkan blok "ubah skor +1" ketika karakter menyentuh bintang. Kemudian coba jalankan lagi.
5. **Ulangi jika perlu**: Kadang ada lebih dari satu bug, jadi ulangi proses ini sampai program berjalan sesuai harapan.

Dengan belajar debugging, kamu tidak hanya memperbaiki kesalahan, tapi juga melatih cara berpikir logis dan sabar. Ini adalah bagian penting dari computational thinking, dan sangat berguna kalau kamu ingin jadi programmer hebat di masa depan!

# Contoh Soal dan Pembahasan

## Soal 1: Game Adventure Terlalu Besar? Pecah Saja!

Bayangkan kamu sedang membuat game petualangan dengan 3 area: hutan, sungai, dan gunung. Pemain bisa melompat, mengambil koin, dan melawan musuh.

**Pertanyaan:**
Bagaimana kamu bisa menggunakan *decomposition* untuk menyederhanakan proses pembuatan game ini?

**Pembahasan:**
Gunakan *decomposition* dengan memecah game menjadi beberapa bagian kecil seperti:

- **Sprite pemain**: Gerakan, animasi, dan logika interaksi.
- **Area hutan**: Musuh hutan, latar belakang hutan, item yang ditemukan di hutan.
- **Area sungai dan gunung**: Didesain terpisah, dengan item dan musuh berbeda.
- **Sistem koin dan skor**: Satu bagian tersendiri agar mudah dikelola.

Dengan ini, setiap bagian bisa kamu kerjakan terpisah, lalu digabungkan kembali. Proyek jadi lebih rapi dan tidak bikin pusing!

## Soal 2: Pola di Balik Pipa!

Dalam game seperti *Flappy Bird*, pipa selalu bergerak dari kanan ke kiri dan burung harus menghindar.

**Pertanyaan:**
Pilar CT mana yang kamu gunakan saat menyadari semua pipa punya pola gerak yang sama? Dan apa manfaat dari pengamatan ini?

**Pembahasan:**
Ini adalah contoh *Pattern Recognition* (Pengenalan Pola).
Dengan mengenali bahwa semua pipa memiliki pola gerak yang seragam, kamu bisa membuat **satu skrip gerak**, lalu digunakan berulang untuk semua pipa.
Manfaatnya? Menghemat waktu, membuat kode lebih singkat, dan meminimalkan bug karena tidak perlu menulis skrip berulang-ulang.

## Soal 3: Fokus ke yang Penting!

Saat membuat sprite karakter pemain, kamu tidak membuat detail seperti rambut atau sepatu. Kamu hanya membuat karakter bisa loncat, bergerak ke kanan-kiri, dan mengambil item.

**Pertanyaan:**
Prinsip CT apa yang kamu terapkan? Mengapa ini penting dalam pengembangan game?

**Pembahasan:**
Kamu sedang menerapkan prinsip *Abstraction* (Abstraksi).
Ini penting karena:

- Fokus hanya pada fitur penting untuk gameplay.
- Mengabaikan detail visual yang tidak mempengaruhi cara main.

Dengan begitu, kamu bisa lebih cepat menyelesaikan game dan menghindari kehabisan waktu untuk hal-hal kecil.

## Soal 4: Langkah-langkah untuk Level Up

Kamu ingin pemain naik level setelah mengumpulkan 5 bintang.

**Pertanyaan:**
Buatlah contoh *algorithm design* (perancangan algoritma) sederhana untuk fitur ini!

**Pembahasan:**
Berikut contoh langkah algoritmik:

1. Cek jumlah bintang yang dikumpulkan.
2. Jika jumlahnya = 5 → tampilkan efek "Naik Level!"
3. Ganti latar ke level baru.
4. Reset jumlah bintang jadi 0.
5. Tambahkan musuh/level tantangan baru.

Ini menunjukkan kamu sudah bisa membuat urutan langkah sistematis yang bisa diubah menjadi kode!

### Soal 5: Skor Tidak Bertambah? Saatnya Jadi Detektif Debugging!

Kamu membuat game menangkap bintang, tapi setelah pemain menyentuh bintang, skor tetap 0.

**Pertanyaan:**
Langkah-langkah debugging apa saja yang bisa kamu lakukan untuk mencari dan memperbaiki masalah ini?

**Pembahasan:**
Langkah debugging:

1. **Cek kode bintang**: Apakah ada blok “ubah skor +1” saat menyentuh pemain?
2. **Gunakan blok "say"**: Tampilkan nilai variabel skor untuk melihat apakah nilainya berubah.
3. **Isolasi skrip**: Jalankan hanya bagian yang mengatur skor.
4. **Periksa urutan blok**: Bisa jadi urutan blok logikanya salah.
5. **Perbaiki & coba lagi**: Setelah tahu penyebabnya, ubah bloknya dan jalankan ulang.

Dengan cara ini, kamu berpikir seperti detektif digital—dan semakin jago jadi game developer!

## Fakta Menarik / Fun Facts

- Istilah "debugging" (mencari dan memperbaiki kesalahan pada program) pertama kali muncul karena seorang ilmuwan komputer bernama Grace Hopper menemukan ngengat (bug) sungguhan di dalam komputer yang menyebabkan masalah!
- Banyak desainer game profesional menggunakan lembar kerja dan diagram untuk merancang game mereka, persis seperti menerapkan Computational Thinking.

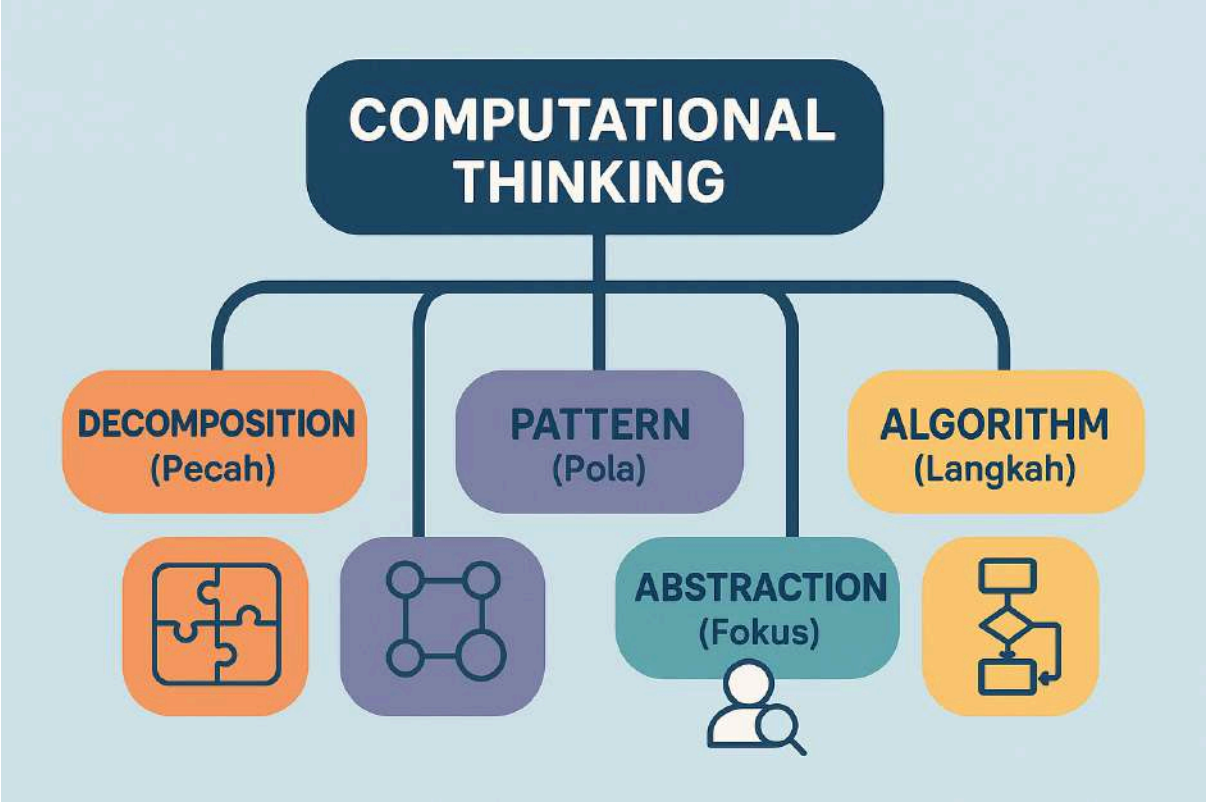
## Tips Belajar atau Tips Cepat Menghafal

### Tips Cepat: Ingat "4D"

- **D**ekomposisi
- Deteksi **D**etail pola (Pattern Recognition)
- **D**esain Algoritma (Algorithm Design)
- Deteksi **D**ebug!

Abstraksi sedikit tersembunyi dari 4D, tapi penting.

### Mind Mapping Sederhana:




### Cerita Singkat:

Bayangkan kamu chef. Kamu harus masak nasi goreng, telur, dan membuat minuman. Kamu tidak bisa kerjakan semua sekaligus, kan? Kamu *memecah tugas* jadi bagian-bagian: nasi goreng dulu, lalu telur, lalu minuman. Kamu melihat *pola* resep. Kamu *mengabaikan hal kecil* (warna sendok, misalnya). Dan kamu ikuti *langkah-langkah resep*. Itulah **CT versi dapur!**

## Aktivitas Siswa

### Aktivitas 1: Identifikasi Computational Thinking (CT) dalam Game Teman

**Tujuan Aktivitas:**

Memahami dan mengenali bagaimana temanmu menggunakan **4 konsep utama Computational Thinking (CT)** dalam game yang mereka buat di Scratch, lalu memberikan saran untuk memperbaiki atau meningkatkan game tersebut.

**Langkah-langkah Aktivitas:**

- **Tukar Proyek Game Scratch**

    - Buka proyek game buatanmu di Scratch.
    - Temanmu juga membuka proyek gamenya.
    - Tukarkan proyek kalian, lalu masing-masing mencoba game dari temannya.

- **Mainkan dan Amati Game Temanmu**

    - Mainkan game temanmu dari awal sampai akhir.
    - Perhatikan bagaimana game itu berjalan: seperti cara karakter bergerak, aturan main, skor, level, dan sebagainya.

- **Analisis Game Temanmu Menggunakan Konsep CT**

    Berikut ini panduan untuk mengenali masing-masing konsep CT:

    - **Decomposition (Pemecahan Masalah):**
      Apakah game ini dibagi menjadi bagian-bagian kecil? Misalnya: satu bagian untuk gerakan karakter, satu bagian untuk skor, satu bagian untuk musuh, dll.
      **Contoh**: Kode untuk skor dibuat terpisah dari kode gerakan karakter.

    - **Pattern Recognition (Pengenalan Pola):**
      Apakah ada bagian dari game yang berulang? Misalnya: musuh muncul dengan cara yang sama, gerakan karakter yang sama setiap kali menekan tombol, dsb.
      **Contoh**: Semua musuh muncul setiap 5 detik dan bergerak ke arah yang sama.

    - **Abstraction (Penyederhanaan):**
      Apakah ada bagian yang disederhanakan agar game tidak terlalu rumit? Biasanya, hanya bagian penting saja yang ditampilkan.
      **Contoh**: Karakter tidak punya banyak detail, hanya fokus pada fungsinya (misalnya: bisa lompat dan menembak).

    - **Algorithm Design (Perancangan Langkah-langkah):**
      Apakah game memiliki urutan langkah yang jelas untuk menjalankan sesuatu? Ini biasanya terlihat dalam skrip perintah (kode) di dalam game.
      **Contoh**: Ketika bendera hijau diklik → karakter mulai bergerak → jika menyentuh musuh → skor berkurang.

- **Diskusi dan Beri Saran**

    - Diskusikan temuanmu dengan temanmu.
    - Beri saran untuk:

        - **Debugging (Memperbaiki kesalahan):** Apakah ada bagian yang tidak bekerja dengan baik?
        - **Optimasi (Peningkatan):** Apakah ada bagian yang bisa dibuat lebih efisien atau lebih menarik?

**Contoh Saran:**

- "Karakter kadang tersangkut di tembok, mungkin bisa diperbaiki dengan menambahkan batas gerakan."
- "Musuh muncul terlalu cepat, bisa ditambahkan jeda waktu agar lebih adil."
- "Skornya tidak muncul, coba periksa variabel dan tampilkan di layar."

**Tujuan Akhir:**

Dengan kegiatan ini, kamu belajar mengenali cara berpikir komputasional dan juga saling membantu untuk membuat game yang lebih baik.

## Aktivitas 2: Debugging Challenge – Jadi Detektif Kode!

**Tujuan Aktivitas:Tujuan Aktivitas:**

Melatih kemampuanmu dalam mencari dan memperbaiki kesalahan (bug) di proyek Scratch. Kamu akan belajar berpikir seperti **detektif kode** untuk menyelidiki kenapa game tidak berjalan dengan baik.

**Deskripsi Aktivitas:**

Guru akan memberikan proyek Scratch yang **sengaja dibuat ada kesalahannya (bug)**. Tugasmu adalah:

- **Mencoba game tersebut**, lalu mencari tahu bagian mana yang tidak bekerja seperti seharusnya.
- **Memeriksa kode di balik game** untuk mencari penyebab bug-nya.
- **Memperbaiki bug** tersebut.
- **Menjelaskan langkah-langkah** yang kamu lakukan selama proses debugging.

**Langkah-langkah Aktivitas:**

- **Mainkan Game yang Diberikan**
    - Jalankan game dari awal.
    - Perhatikan: apakah ada yang aneh?
        - Karakter tidak bergerak?
        - Skor tidak bertambah?
        - Suara tidak keluar?
        - Game tidak selesai?

- **Catat Bug yang Kamu Temukan**

    - Tulis semua masalah yang kamu lihat.
    - Contoh catatan:
      "Saat menyentuh musuh, karakter tidak mati. Seharusnya game over."

- **Periksa Kode di Dalam Scratch**
    - Klik sprite atau bagian yang bermasalah.
    - Buka blok-blok kode dan baca satu per satu.
    - Coba pahami: Apakah ada blok yang salah? Atau mungkin kurang blok?

- **Perbaiki Bug-nya**
    - Ubah atau tambahkan blok kode agar sesuai dengan tujuan game.
    - Tes lagi apakah bug-nya sudah hilang.

- **Jelaskan Proses Debugging Kamu**
  Seperti seorang detektif, ceritakan:
    - **Apa bug-nya?**
    - **Kenapa itu terjadi?**
    - **Bagaimana kamu menemukannya?**
    - **Apa yang kamu lakukan untuk memperbaikinya?**

- **Contoh Penjelasan:**
    - "Bug-nya adalah skor tidak bertambah saat menangkap koin. Setelah saya periksa, ternyata blok 'ubah skor' belum ditambahkan di bagian saat karakter menyentuh koin. Saya tambahkan blok itu, lalu skor jadi bertambah."

**Boleh Dikerjakan Secara Individu atau Kelompok**

Kamu bisa mengerjakannya sendiri, atau berdiskusi bersama teman satu kelompok agar bisa saling bantu mencari solusi.

**Tujuan Akhir:**

Setelah kegiatan ini, kamu akan jadi lebih terampil dalam:

- Mencari kesalahan dalam kode (debugging)
- Berpikir logis dan teliti
- Memperbaiki masalah seperti programmer sungguhan!

# Rangkuman

**Computational Thinking** merupakan **keterampilan berpikir penting** yang sangat berguna dalam membuat game maupun menyelesaikan berbagai masalah. Dengan menerapkan empat komponen utama—*Decomposition*, *Pattern Recognition*, *Abstraction*, dan *Algorithm*

*Design*—kita dapat merancang game secara **lebih terstruktur dan efisien**. Tak kalah penting, proses *debugging* menjadi langkah esensial untuk memastikan bahwa program berjalan sesuai harapan.

Melalui **Computational Thinking (CT)**, kita belajar menyelesaikan masalah **layaknya ilmuwan komputer**. Dengan *Decomposition*, kita **memecah masalah besar** menjadi bagian-bagian kecil yang lebih mudah ditangani. *Pattern Recognition* membantu kita **mengidentifikasi pola**, sehingga dapat menghemat waktu dan usaha. *Abstraction* memungkinkan kita **memusatkan perhatian pada hal-hal penting**, tanpa terganggu oleh detail yang tidak relevan. Sementara *Algorithm Design* memberi kita cara untuk **menyusun langkah-langkah sistematis** dalam menyelesaikan tugas.

Proses *debugging* sangat krusial untuk memastikan bahwa program **berfungsi dengan benar** dan **mencapai tujuan** yang diinginkan. Menariknya, prinsip-prinsip CT ini dapat digunakan **tidak hanya dalam game**, tetapi juga dalam kehidupan sehari-hari.

Dengan menggunakan CT, **membuat game jadi lebih mudah, menyenangkan, dan terarah**. Jadi, saat kamu sedang bermain atau mengembangkan game, sebenarnya kamu sedang **melatih otak seperti seorang ilmuwan**. **Keren, kan?**

# Latihan Soal

## A. Benar atau Salah (5 Soal)

1. Computational Thinking hanya digunakan oleh programmer profesional.
   **Jawaban:** Salah

2. Decomposition membantu kita memecah masalah besar menjadi bagian kecil.
   **Jawaban:** Benar

3. Dalam Pattern Recognition, kita menghindari pengulangan pola dalam kode.
   **Jawaban:** Salah

4. Abstraction berguna untuk fokus pada hal penting dan mengabaikan detail tidak perlu.
   **Jawaban:** Benar

5. Debugging bisa dilakukan dengan cara mencoba-coba acak tanpa strategi.
   **Jawaban:** Salah

## B. Pilihan Ganda (10 Soal)

6. Apa yang dimaksud dengan Decomposition?
   A. Menghindari pola yang berulang
   B. Memecah masalah besar menjadi bagian kecil
   C. Menyusun urutan langkah-langkah

D. Menghapus bagian kode yang tidak penting
**Jawaban:** B

7. Contoh Pattern Recognition dalam game adalah:
   A. Membuat sprite baru
   B. Memecah skor menjadi dua variabel
   C. Menyadari semua musuh bergerak ke kiri
   D. Membuat tampilan game lebih menarik
   **Jawaban:** C

8. Dalam membuat algoritma, kita harus:
   A. Menghindari pola
   B. Menyusun langkah-langkah logis dan teratur
   C. Menggambar sprite secara manual
   D. Menghapus semua blok yang tidak berguna
   **Jawaban:** B

9. Apa fungsi dari blok **say** dalam debugging di Scratch?
   A. Menampilkan skor akhir
   B. Menggambar sprite
   C. Menunjukkan nilai variabel atau pesan
   D. Mengatur suara latar belakang
   **Jawaban:** C

10. Contoh Abstraction dalam game adalah:
    A. Membuat karakter sangat detail hingga ke rambut
    B. Fokus pada interaksi karakter, bukan tampilannya
    C. Menulis ulang semua kode
    D. Menggunakan banyak efek suara
    **Jawaban:** B

11. Ketika kamu menggunakan mode "step-by-step" di Scratch, kamu sedang:
    A. Mempercepat gerakan karakter
    B. Membuat animasi
    C. Men-debug kode satu per satu
    D. Mengatur level musuh
    **Jawaban:** C

12. Jika kamu mendesain gerakan karakter utama, kamu sedang menerapkan:
    A. Decomposition
    B. Pattern Recognition
    C. Abstraction
    D. Algorithm Design
    **Jawaban:** D

13. Apa keuntungan dari menggunakan blok kustom di Scratch?
    A. Menambah bug
    B. Membuat kode lebih panjang

C. Menyederhanakan dan mengurangi duplikasi
D. Menghapus karakter
**Jawaban:** C

14. Game Flappy Bird bisa dianalisis menggunakan CT. Apa bentuk Decomposition-nya?
A. Menyusun peta level
B. Burung, pipa, deteksi tabrakan, skor, layar akhir
C. Gerakan musuh
D. Membuat efek suara
**Jawaban:** B

15. Teknik debugging yang baik adalah:
A. Menghapus semua skrip
B. Menggunakan blok say dan periksa kode satu per satu
C. Menambahkan musuh sebanyak mungkin
D. Mengubah warna latar belakang
**Jawaban:** B

## C. Isian Singkat (5 Soal)

16. Menyusun langkah-langkah logis dalam game termasuk dalam pilar _____.
**Jawaban:** Algorithm Design

17. Mengabaikan detail kecil dan hanya fokus pada hal penting disebut _____.
**Jawaban:** Abstraction

18. Jika karakter musuh selalu muncul dari kanan dan bergerak ke kiri, kita melihat adanya _____.
**Jawaban:** pola (Pattern Recognition)

19. Saat kita memisahkan sistem skor dari sistem level, kita sedang melakukan _____.
**Jawaban:** Decomposition

20. Menampilkan nilai variabel dalam sprite untuk melacak bug menggunakan blok _____.
**Jawaban:** say

## D. Soal Eksplorasi (3 Soal Tantangan)

**Soal:**

Jelaskan bagaimana kamu akan menggunakan Pattern Recognition saat membuat game balapan mobil. Fokuskan jawabanmu pada bagaimana kamu bisa menggunakan pola-pola yang berulang untuk menyederhanakan pengkodean lintasan, mobil musuh, atau item tambahan di jalan. Tuliskan jawabanmu dalam satu paragraf.

**Soal:**

Kamu diminta membuat game platformer sederhana (seperti Mario) di Scratch. Tuliskan bagaimana kamu akan menerapkan empat pilar Computational Thinking (Decomposition, Pattern Recognition, Abstraction, Algorithm Design) dalam proses perancangannya. Gunakan satu paragraf pendek untuk menjelaskan penerapan masing-masing pilar.

**Soal:**

Bayangkan kamu menemukan bug di game yang kamu buat: karakter pemain tidak bisa melompat ketika menyentuh tanah. Tuliskan langkah-langkah debugging yang akan kamu lakukan secara sistematis untuk mencari tahu dan memperbaiki masalah ini. Ceritakan seolah kamu sedang menulis laporan kecil kepada gurumu.

# Tugas Proyek

## Proyek 1: Misi: Evaluasi & Perbaiki Game Scratch

**Tujuan Proyek**: Mengasah kemampuan berpikir kritis dan kreatif dalam memperbaiki serta mengoptimalkan game buatan sendiri atau dari guru.

**Langkah-langkah:**

- **Mainkan gamenya** dan cari *minimal dua bug* (kesalahan seperti karakter tidak bergerak, skor tidak bertambah, dll).
- **Perbaiki bug** tersebut dengan menganalisis logika dan blok kode yang digunakan.
- **Optimalkan kode**: Gunakan blok kustom untuk menghindari pengulangan kode atau blok yang terlalu panjang.
- **Tuliskan laporan perbaikan** yang menjelaskan bagaimana kamu menerapkan 4 pilar Computational Thinking:
    - **Decomposition**: Memecah masalah ke dalam bagian-bagian kecil.
    - **Pattern Recognition**: Menemukan pola dari bug atau bagian yang sering terjadi.
    - **Abstraction**: Fokus hanya pada informasi penting (misalnya hanya perhatikan blok yang terkait gerakan).
    - **Algorithm Design**: Menyusun ulang langkah-langkah logis untuk memperbaiki game.

## Proyek 2: Menemukan Abstraksi di Dunia Nyata

**Tujuan Proyek**: Melatih kepekaan terhadap konsep abstraksi dalam kehidupan sehari-hari.

**Langkah-langkah:**

- Pikirkan satu kegiatan sehari-hari (contoh: naik bus, membuat mie instan, atau membuka HP).

- Temukan bagian-bagian yang merupakan bentuk **abstrak** (misalnya, menekan tombol “Power” tanpa memikirkan sistem listrik di dalamnya).
- Jelaskan **kenapa itu disebut abstraksi**, yaitu proses menyederhanakan hal rumit menjadi mudah digunakan tanpa harus tahu cara kerjanya secara mendetail.

## Proyek 3: Tantangan Labirin (Maze Challenge!)

**Tujuan Proyek**: Menggunakan pola dan strategi pintar dalam merancang game labirin.

**Langkah-langkah:**

- Buat game labirin di Scratch dengan **musuh yang bergerak otomatis** dan **jalan yang berliku-liku**.
- Gunakan **Pattern Recognition** untuk:
    - Menemukan pola pergerakan musuh (misalnya maju-mundur, zig-zag, atau mengikuti jalur tetap).
    - Mendesain dinding labirin dengan pola tertentu (misalnya simetri atau pengulangan bentuk).
- Tulis penjelasan bagaimana pola-pola tersebut membantu **menghemat waktu dalam coding** dan membuat game lebih rapi serta menantang.

## Proyek 4: Jadi Detektif Bug!

**Tujuan Proyek**: Melatih kemampuan debugging (mencari dan memperbaiki kesalahan logika).

**Langkah-langkah:**

- Bayangkan kamu membuat game lari-larian, tapi karakter kadang **tidak bisa melompat**.
- Buat laporan seperti seorang **detektif bug**, misalnya:
    - Apa masalahnya?
    - Kapan masalah terjadi?
    - Blok mana yang kemungkinan menyebabkan masalah?
    - Apa tes yang dilakukan?
    - Apa perbaikannya?
- Laporan bisa ditulis seperti jurnal, misalnya: *"Saat karakter menyentuh tanah miring, loncat tidak berfungsi. Saya periksa blok ‘if touching ground’ dan menemukan deteksi tanah tidak tepat.”*

## Proyek 5: Analisis Game Favorit dengan CT

**Tujuan Proyek**: Melatih cara berpikir komputasional lewat game populer.

**Langkah-langkah:**

- Pilih satu game favoritmu (Roblox, Minecraft, Mobile Legends, dll).

- Analisis bagaimana 4 pilar Computational Thinking muncul di game itu, contohnya:

  > “Dalam game Minecraft, saya menggunakan *Decomposition* untuk memecah proyek rumah ke dalam bagian atap, dinding, dan pintu. Saya melihat *Pattern Recognition* saat membuat desain jendela yang berulang. *Abstraction* saya gunakan saat hanya fokus pada blok yang saya butuhkan. Terakhir, *Algorithm Design* saya gunakan untuk mengatur urutan pembangunan dari bawah ke atas.”

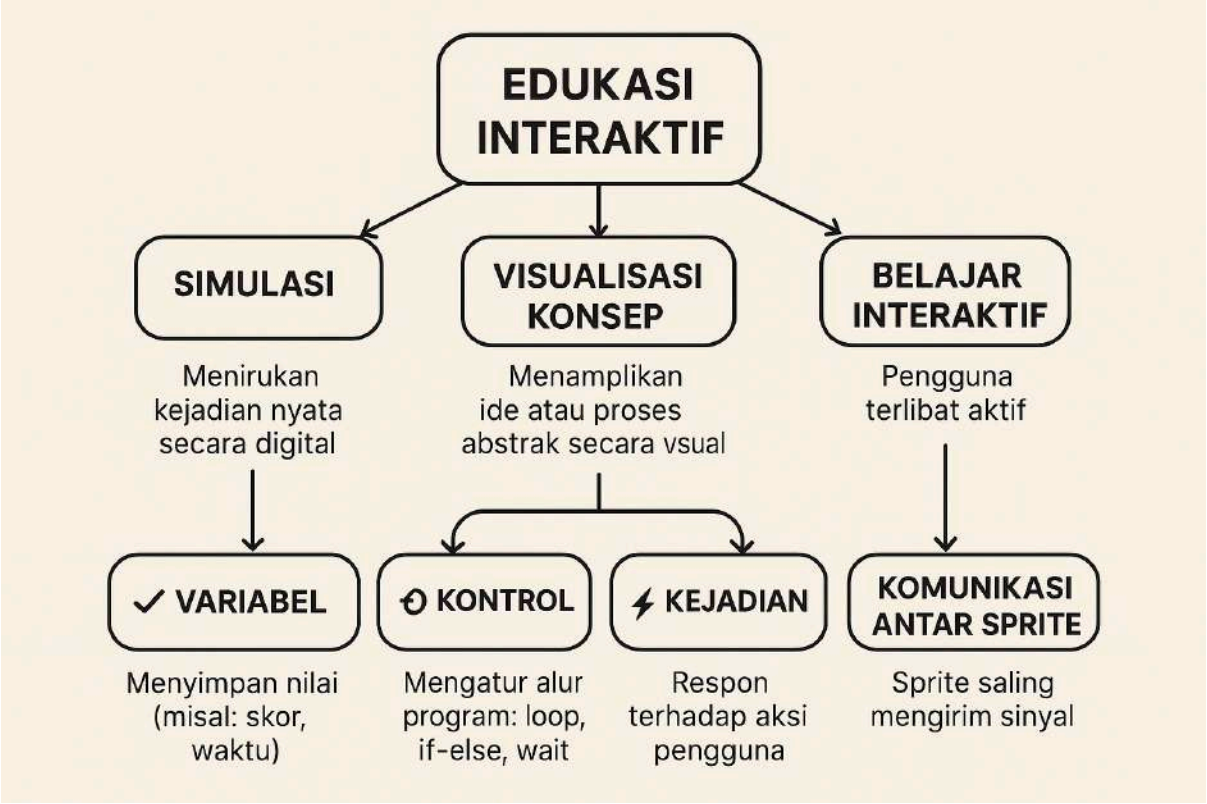
- Tuliskan dalam satu paragraf pendek, padat, tapi menggambarkan semua pilar CT.

# BAB 4: Simulasi Edukasi dengan Scratch (Sains atau Bahasa)

## Tujuan Pembelajaran

Peserta didik mampu merancang dan membuat simulasi edukasi interaktif sederhana menggunakan Scratch untuk menjelaskan konsep dasar sains atau bahasa, serta memahami bagaimana visualisasi dapat membantu proses belajar.

## Peta Konsep




## Apersepsi

Bagaimana ya cara memahami siklus air, atau pergerakan planet, tanpa harus pergi ke luar angkasa? Atau bagaimana cara belajar kosa kata bahasa Inggris dengan cara yang lebih seru daripada sekadar menghafal? Ternyata, kita bisa membuat "dunia kecil" di komputer yang bisa menjelaskan konsep-konsep itu! Ini namanya simulasi edukasi.

# Penjelasan Konsep (Teori)

## Apa itu Simulasi Edukasi, dan Mengapa Penting?

Bayangkan kamu bisa menjelaskan bagaimana air berubah menjadi hujan, atau bagaimana planet-planet mengelilingi matahari—*tanpa harus pergi ke luar angkasa!* Atau kamu ingin belajar bahasa Inggris, tapi bukan dengan cara membosankan seperti menghafal daftar kata? Nah, jawabannya adalah: **simulasi edukasi**!

Simulasi edukasi adalah cara membuat "dunia kecil" di komputer, tempat berbagai konsep sains dan bahasa bisa divisualisasikan, dijelaskan, dan bahkan dimainkan! Dengan bantuan Scratch, kita bisa membuat simulasi yang **interaktif, seru, dan membantu kita memahami pelajaran dengan lebih mudah.**

### A. Peran Simulasi dalam Dunia Pendidikan

Simulasi edukasi memiliki banyak manfaat dalam pembelajaran. Berikut ini beberapa peran pentingnya:

1. **Visualisasi Data dan Proses**
   Dengan Scratch, kamu bisa membuat grafik, animasi, atau ilustrasi yang menjelaskan proses rumit.
   Contoh:
   - Grafik pertumbuhan tanaman berdasarkan suhu.
   - Animasi siklus hidup kupu-kupu dari telur → ulat → kepompong → kupu-kupu dewasa.

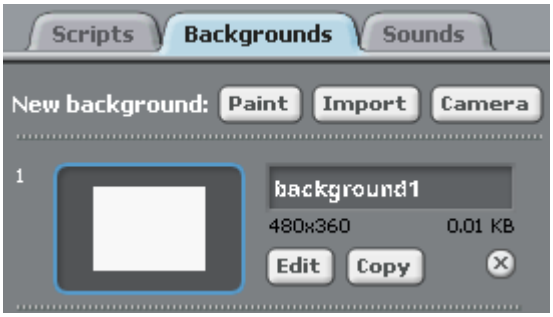
2. **Belajar yang Interaktif**
   Simulasi memungkinkan kamu mengubah nilai, menekan tombol, atau menggerakkan karakter untuk melihat apa yang terjadi.
   Misalnya:
   - Mengubah suhu untuk melihat kapan air berubah menjadi es atau uap.
   - Memilih kata yang benar dalam kuis bahasa Inggris.

3. **Menyederhanakan Konsep yang Rumit**
   Proses yang besar bisa dipecah menjadi bagian kecil yang mudah dipahami.
   Animasi dan interaksi membuat pelajaran terasa seperti bermain sambil belajar!

### B. Teknik Dasar Membuat Simulasi Edukasi di Scratch

Agar simulasi kita berjalan dengan baik, ada beberapa teknik Scratch yang perlu dipahami dan digunakan:

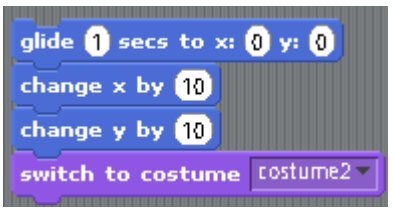
1. **Sprite dan Latar Belakang (Backdrop)**
   - *Sprite* adalah karakter atau objek di Scratch (misal: matahari, planet, huruf).
   - *Backdrop* adalah latar atau tempat kejadian simulasi.
   - Gunakan *beberapa latar belakang* untuk menggambarkan perubahan tempat atau tahap proses.




2. **Variabel untuk Menyimpan Data**
   - Gunakan variabel untuk menyimpan informasi yang bisa berubah.
   - Contoh: suhu, kecepatan, skor jawaban, atau waktu.
   - Variabel sangat penting untuk menciptakan simulasi yang dinamis.

3. **Broadcast dan When I Receive**
   - Fitur ini memungkinkan sprite “berkomunikasi”.
   - Contoh: saat sprite “matahari” terbit, ia bisa memberi tahu sprite “tanaman” agar mulai tumbuh.
   - Ini seperti memberi perintah dari satu karakter ke yang lain.

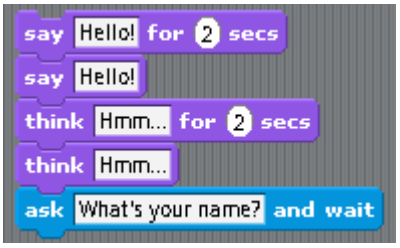

4. **Gerakan dan Perubahan Visual**
   Gunakan blok seperti:
   - `glide` (meluncur ke posisi)
   - `change x/y` (bergerak ke kanan/kiri atau atas/bawah)
   - `change costume` (mengubah tampilan sprite)




   Blok-blok ini membuat simulasi lebih hidup dan realistis.

5. **Interaksi dengan Pengguna**
   Gunakan blok:
   - `say`, `think` → menampilkan teks/komentar dari sprite.

- `ask and wait` → memberikan pertanyaan kepada pengguna dan menunggu jawaban.




Ini membuat simulasi terasa seperti dialog dua arah—bukan hanya tontonan!

### Kesimpulan Mini

Dengan Scratch, kita tidak hanya membuat animasi lucu, tapi juga membangun *alat belajar interaktif*. Kita bisa menjelaskan pelajaran sains atau bahasa dengan cara yang menarik, sederhana, dan interaktif. Belajar pun jadi lebih seru!

Siap mencoba membuat simulasi pertamamu? Yuk kita lanjut ke bagian berikutnya!

## Contoh Kasus atau Ilustrasi

### Kasus 1: Petualangan di Dunia Siklus Air

**Kasus:** Bayangkan kamu seorang penjelajah yang masuk ke dalam dunia cuaca!
**Ilustrasi:**

- Kamu melihat **Matahari bersinar terang**. Saat kamu menekan tombol "Mulai", matahari mulai **menghangatkan laut**, dan air mulai **berubah warna menjadi lebih terang**.
- Perlahan, **awan terbentuk** dan mulai **naik ke langit**, membawa uap air.
- Tidak lama kemudian, **awan menjadi berat** dan tetesan air mulai jatuh ke bumi dalam bentuk **hujan**.
- Kamu menyaksikan secara langsung bagaimana air **mengalir kembali ke laut**, dan **siklus dimulai lagi**!

**Tantangan:** Dapatkah kamu mengenali tahapan-tahapan ini dan menjelaskan mengapa air tidak pernah habis?

### Kasus 2: Misteri Kata-Kata Ajaib (Bahasa Inggris)

**Kasus:** Kamu masuk ke ruang kelas rahasia tempat guru misterius mengucapkan kata-kata asing.
**Ilustrasi:**

- Seorang **guru animasi** muncul dan berkata, “**Apple!**”

- Tiba-tiba, muncul **gambar apel berwarna merah cerah**.
- Di layar, kamu harus **mengetik kata yang kamu dengar**. Jika kamu mengetik "Apple" dengan benar, layar akan bersinar dan terdengar suara "**Good job!**"
- Tapi jika kamu salah, gambar akan memudar dan kata akan diulang dengan pelafalan yang jelas.

**Tantangan:** Bisakah kamu menaklukkan semua kata dan menjadi ahli kosa kata?

## Kasus 3: Gravitasi vs Apel: Siapa Menang?

**Kasus:** Apakah kamu pernah berpikir mengapa apel selalu jatuh ke bawah, bukan ke atas?
**Ilustrasi:**

- Di layar, kamu melihat **sebuah tangan memegang apel**.
- Saat kamu klik tangan itu, tangan **melepaskan apel**, dan apel **jatuh ke tanah dengan cepat!**
- Setelah itu, muncul penjelasan singkat: "**Itulah gaya gravitasi – gaya yang menarik benda ke arah bumi.**"
- Kamu bahkan bisa **ulang percobaan** dan lihat jika apel bisa "ditahan" oleh benda lain (misalnya payung, tangan kedua, dll).

**Tantangan:** Apakah kamu bisa membuat simulasi lain dengan benda yang berbeda?

## Kasus 4: Misi Rahasia: Perkenalkan Dirimu (Bahasa Inggris)

**Kasus:** Kamu bertemu karakter dari berbagai negara. Mereka ingin tahu siapa kamu!
**Ilustrasi:**

- Kamu klik pada sprite seorang anak perempuan, dan dia berkata, "**My name is Anna. I am 12 years old.**"
- Kamu lalu **dipandu** untuk membuat sprite baru, dan karakter itu akan memperkenalkan dirimu dengan suara atau teks.
- Simulasi ini akan memperlihatkan **percakapan perkenalan yang sederhana namun menyenangkan**.

**Tantangan:** Bisakah kamu membuat perkenalan diri yang unik dan ekspresif?

## Kasus 5: Cuaca Ekstrem: Tantangan dari Alam

**Kasus:** Dunia sedang dilanda perubahan cuaca ekstrim. Bisakah kamu menjelaskan penyebabnya?
**Ilustrasi:**

- Kamu menyaksikan urutan animasi: **Matahari sangat panas**, membuat **air cepat menguap**, awan gelap muncul, dan **hujan deras disertai petir**.
- Lalu muncul teks: "**Ini disebut hujan badai. Perubahan cuaca bisa terjadi karena suhu dan kelembaban yang tinggi.**"

**Tantangan:** Bisakah kamu menghubungkan ini dengan siklus air dan menjelaskan bagaimana badai terjadi?

### Kasus 6: Teka-Teki Gaya Tak Terlihat (Gaya Fisika)

**Kasus:** Kamu menemukan benda-benda yang bergerak sendiri. Kenapa bisa begitu?
**Ilustrasi:**

- Kamu melihat sebuah **bola di atas meja miring**. Saat dilepas, bola **bergerak sendiri** ke bawah.
- Lalu muncul teks: "**Gaya gravitasi dan kemiringan permukaan membuat bola bergerak.**"
- Kamu bisa **ubah kemiringan meja** untuk melihat apakah gerak bola berubah!

**Tantangan:** Bisakah kamu menemukan sudut terbaik agar bola bergerak paling cepat?

## Contoh Soal dan Pembahasan

**Soal:**

Rancang sebuah simulasi sederhana untuk menjelaskan konsep gravitasi di Scratch. Apa saja sprite yang kamu butuhkan dan bagaimana alurnya?

**Pembahasan:**

- **Sprite:** Apel, Tanah/Permukaan, Tangan (opsional, untuk melepaskan apel).
- **Alur:**
  - Ketika bendera hijau diklik, Apel berada di posisi atas.
  - Ketika tombol "jatuhkan" diklik atau setelah beberapa detik, Apel mulai bergerak ke bawah (`change y by -10` dalam perulangan `repeat until touching [Tanah]`).
  - Apel bisa memantul atau berhenti saat menyentuh tanah.

**Soal:**

Bagaimana cara membuat alat bantu belajar kosa kata di Scratch yang menampilkan gambar dan suara, lalu meminta pengguna mengulang kata tersebut?

**Pembahasan:**

- **Sprite:** Guru/Pemandu, Gambar Objek (misal: "Kucing"), Tombol "Play Sound".
- Ketika tombol "Play Sound" diklik, `start sound "kucing"`.
- `ask "Ulangi kata yang kamu dengar:" and wait`.

    - `if answer = "kucing" then say "Bagus!" else say "Coba lagi."`.

## Fakta Menarik / Fun Facts

- Banyak alat belajar interaktif yang kalian temukan di internet, seperti simulasi fisika atau kimia, adalah bentuk aplikasi yang dibuat dengan logika dasar seperti yang kita pelajari di sini!
- Simulasi bisa membantu kita "bereksperimen" dengan hal-hal yang terlalu besar, terlalu kecil, terlalu cepat, atau terlalu berbahaya di dunia nyata.

## Tips Belajar atau Tips Cepat Menghafal

- **Pikirkan "bagaimana cara menjelaskan sesuatu dengan gambar bergerak":** Jika kalian harus menjelaskan suatu konsep kepada teman, bagaimana kalian akan menampilkannya secara visual? Itulah dasar simulasi.

- **Sederhanakan:** Jangan coba membuat simulasi yang terlalu rumit di awal. Mulai dari konsep paling dasar.

- **Tips Cepat Menghafal:**
    - Ubah informasi menjadi gambar bergerak.
    - Ceritakan kembali dengan suaramu sendiri.
    - Gunakan blok-blok warna di Scratch agar lebih mudah dipahami.

### Mind Mapping Simulasi Scratch:


### Cerita Mudah Diingat:

Bayangkan kamu jadi ilmuwan mini yang masuk ke dalam komputer. Kamu membuat dunia kecil: hujan turun, tanaman tumbuh, planet berputar. Nah, itulah dunia

simulasi! Dengan Scratch, kamu bisa menciptakan semuanya itu hanya dengan klik dan coding sederhana.

# Aktivitas Siswa

## Aktivitas 1: Simulasi Fotosintesis Sederhana

**Deskripsi:**

Kita akan membuat simulasi bagaimana tumbuhan membuat makanannya sendiri. Tanaman akan mengambil **air ($H_2O$)** dari akar, **karbon dioksida ($CO_2$)** dari udara, dan **cahaya matahari** dari matahari. Setelah itu, tanaman akan menghasilkan **oksigen ($O_2$)** dan **makanan**.

**Langkah-langkah Pseudocode:**

1. Tambahkan sprite tanaman, matahari, awan $CO_2$, dan air di layar.
2. Saat tombol "Mulai" diklik:
   - Awan $CO_2$ bergerak ke arah daun tanaman.
   - Tetesan air naik dari akar ke daun.
   - Cahaya matahari turun ke tanaman.
3. Setelah semua sampai di tanaman:
   - Ganti kostum tanaman menjadi lebih cerah/segar.
   - Muncul sprite baru: oksigen keluar dari daun dan makanan muncul di dalam tanaman.
4. Tampilkan teks "Fotosintesis selesai!" di layar.

## Aktivitas 2: Perkenalan Diri dalam Bahasa Inggris (Interaktif)

**Deskripsi:**

Membuat sprite yang bisa memperkenalkan diri dalam bahasa Inggris. Pengguna bisa mengklik bagian tertentu (misalnya tombol "Name", "Age", "Hobby") untuk mendengar pengucapan dan melihat artinya dalam Bahasa Indonesia.

**Langkah-langkah Pseudocode:**

1. Tambahkan sprite karakter siswa dan tiga tombol: "Name", "Age", "Hobby".
2. Saat tombol "Name" diklik:
    - Suara berkata "My name is John."
    - Tampilkan terjemahan: "Namaku John."
3. Saat tombol "Age" diklik:
    - Suara berkata "I am 13 years old."
    - Tampilkan terjemahan: "Aku berumur 13 tahun."
4. Saat tombol "Hobby" diklik:
    - Suara berkata "My hobby is playing football."
    - Tampilkan terjemahan: "Hobiku bermain sepak bola."

## Aktivitas 3: Kuis Interaktif tentang Fotosintesis

**Deskripsi:**

Setelah siswa melihat simulasi fotosintesis, mereka bisa mencoba kuis sederhana untuk menguji pemahaman mereka. Jawaban bisa dipilih lewat tombol, dan langsung ada umpan balik benar/salah.

**Langkah-langkah Pseudocode:**

1. Tampilkan pertanyaan pertama: "Apa yang dibutuhkan tanaman untuk fotosintesis?"
    - Tampilkan pilihan: A. Oksigen, B. Air, C. Nitrogen
2. Saat siswa klik "B. Air":
    - Tampilkan teks: "Benar! Tanaman butuh air."
3. Kalau jawab salah:
    - Tampilkan teks: "Salah. Coba lagi!"
4. Lanjut ke pertanyaan berikutnya setelah jawaban benar.

## Aktivitas 4: Percakapan Bahasa Inggris Sederhana (2 Orang)

**Deskripsi:**

Membuat simulasi dua sprite yang berbicara dalam bahasa Inggris, seperti sedang saling menyapa. Misalnya: "Hello!" – "Hi!" – "How are you?" – "I’m fine."

**Langkah-langkah Pseudocode:**

1. Tambahkan dua sprite siswa, sebut saja Anna dan Tom.
2. Saat tombol "Mulai Percakapan" diklik:
    - Sprite Anna berkata: "Hello!"
    - Sprite Tom membalas: "Hi!"
    - Anna: "How are you?"
    - Tom: "I'm fine, thank you."
3. Tambahkan teks terjemahan setiap dialog di bawah sprite.
4. Setelah selesai, tampilkan tombol "Ulangi" untuk memutar ulang percakapan.

# Rangkuman

**Scratch** adalah alat yang **sangat berguna untuk membuat simulasi pembelajaran**. Dengan Scratch, kita bisa menggabungkan **gambar karakter (*sprite*)**, **latar belakang**, **variabel**, dan **blok perintah interaktif** untuk menciptakan **pembelajaran yang menarik dan mudah dipahami**.

Simulasi yang dibuat di Scratch bisa membantu kita **belajar sains atau bahasa** dengan cara yang lebih **seru dan menyenangkan**, karena menggunakan **animasi dan interaksi**. Misalnya, kita bisa membuat **dunia kecil** seperti **planet yang berputar**, **tanaman yang tumbuh**, atau bahkan **guru virtual** yang mengajarkan **kosa kata baru**.

Dengan Scratch, siswa bisa membuat **proyek kreatif** yang tidak hanya keren, tapi juga **berguna untuk belajar**. Belajar pun jadi lebih **aktif, visual, dan menyenangkan**!

# Latihan Soal

## A. Benar atau Salah (5 Soal)

1. AI adalah singkatan dari Artificial Intelligence.
   **Jawaban:** Benar

2. Robot adalah satu-satunya bentuk AI.
   **Jawaban:** Salah

3. AI bisa digunakan untuk mengenali wajah seseorang.
   **Jawaban:** Benar

4. AI tidak bisa digunakan dalam dunia pendidikan.
   **Jawaban:** Salah

5. Chatbot seperti asisten virtual adalah contoh dari AI.
   **Jawaban:** Benar

## B. Pilihan Ganda (10 Soal)

1. Apa kepanjangan dari AI?
   a. Auto Information
   b. Automatic Intelligence
   c. Artificial Intelligence
   d. Artificial Interaction
   **Jawaban:** c. Artificial Intelligence

2. AI digunakan dalam aplikasi berikut, kecuali...
   a. Pengenalan suara
   b. Bermain catur
   c. Membuat teh
   d. Menerjemahkan bahasa
   **Jawaban:** c. Membuat teh

3. Contoh AI yang sering digunakan di ponsel adalah...
   a. Kamera
   b. Google Assistant
   c. Senter
   d. Galeri foto
   **Jawaban:** b. Google Assistant

4. AI dapat membantu manusia dengan cara...
   a. Menggantikan semua pekerjaan manusia
   b. Memberikan saran dan informasi otomatis
   c. Menyuruh manusia bekerja lebih keras
   d. Mematikan semua komputer
   **Jawaban:** b. Memberikan saran dan informasi otomatis

5. Teknologi AI bisa ditemukan di...
   a. Buku tulis
   b. Lemari pakaian
   c. Game komputer
   d. Pensil
   **Jawaban:** c. Game komputer

6. AI dapat belajar dari data yang diberikan, ini disebut...
   a. Coding
   b. Hardware
   c. Machine Learning
   d. Manual Control
   **Jawaban:** c. Machine Learning

7. Apa peran AI dalam pendidikan?
   a. Menghapus tugas
   b. Membantu belajar secara personal
   c. Mengganti guru
   d. Menurunkan nilai
   **Jawaban:** b. Membantu belajar secara personal

8. Contoh AI dalam kendaraan adalah...
   a. Rem biasa
   b. Setir kayu
   c. Sistem parkir otomatis
   d. Pintu manual

**Jawaban:** c. Sistem parkir otomatis

9. ChatGPT adalah contoh dari...
   a. Mesin ketik otomatis
   b. Aplikasi kalkulator
   c. AI berbasis teks
   d. Editor gambar
   **Jawaban:** c. AI berbasis teks

10. Apa tantangan dari penggunaan AI?
    a. Tidak pernah rusak
    b. Selalu benar
    c. Privasi dan etika
    d. Membuat semua orang pintar
    **Jawaban:** c. Privasi dan etika

## C. Isian Singkat (5 Soal)

1. AI adalah teknologi yang membuat komputer bisa berpikir seperti _____.
   **Jawaban:** manusia

2. Program yang bisa menjawab pertanyaan dan berbicara disebut _____.
   **Jawaban:** chatbot

3. AI bisa belajar dari data melalui proses yang disebut _____.
   **Jawaban:** machine learning

4. Salah satu contoh AI dalam kehidupan sehari-hari adalah _____.
   **Jawaban:** Google Assistant

5. Dalam pendidikan, AI bisa membantu siswa dengan memberikan _____ belajar.
   **Jawaban:** rekomendasi

## D. Soal Eksplorasi (3 Soal Paragraf)

1. **Bayangkan kamu membuat simulasi edukasi tentang AI untuk anak-anak.**
   Apa saja yang akan kamu tampilkan di dalamnya? Jelaskan sprite yang kamu gunakan dan apa saja interaksinya agar siswa bisa belajar tentang AI dengan cara menyenangkan.

2. **Coba jelaskan bagaimana AI bisa digunakan untuk membantu guru dalam mengajar.**
   Tuliskan ide kamu dengan contoh sederhana yang bisa dibuat menggunakan Scratch sebagai simulasi.

3. **Menurut kamu, apakah semua hal bisa digantikan oleh AI di masa depan?**
   Buat penjelasan dengan pendapat pribadi kamu, lalu berikan satu contoh hal yang sebaiknya tetap dilakukan oleh manusia, bukan AI.

# Tugas Proyek

## Proyek 1: Simulasi Cuaca Interaktif

**Deskripsi Proyek:**
Buatlah simulasi cuaca yang menampilkan **cuaca cerah, mendung, dan hujan**. Gunakan sprite seperti **awan, matahari, dan tetes hujan**, lalu tambahkan **tombol untuk mengganti kondisi cuaca**.

**Langkah-langkah:**

- Buat latar belakang (background) untuk tiap jenis cuaca.
- Gunakan sprite matahari, awan, dan hujan.
- Tambahkan tombol seperti "Cerah", "Mendung", dan "Hujan" menggunakan sprite atau teks.
- Gunakan blok kode **"when this sprite clicked"** untuk mengganti latar dan menampilkan sprite sesuai kondisi.
- Tambahkan suara efek (seperti hujan) agar lebih hidup.

## Proyek 2: Perkenalan Bahasa Inggris dengan Karakter

**Deskripsi Proyek:**
Buat sprite yang bisa **memperkenalkan diri dalam Bahasa Inggris**, lalu minta pengguna **mengetikkan balasan**. Proyek ini melatih pemahaman dan interaksi bahasa.

**Langkah-langkah:**

- Buat sprite dengan karakter yang menarik (misalnya, robot atau kartun).
- Gunakan blok **"say"** untuk memperkenalkan diri, seperti “Hello! My name is Max. What is your name?”
- Tambahkan **blok input pengguna** (ask–answer) agar siswa bisa membalas.
- Tambahkan suara karakter dan ekspresi wajah agar tampilan makin menarik.
- Buat sprite merespon balasan pengguna, misalnya, “Nice to meet you, [nama].”

## Proyek 3: Simulasi Perubahan Wujud Air

**Deskripsi Proyek:**
Buat animasi sederhana yang memperlihatkan **air berubah menjadi uap karena panas**, lalu **menjadi awan**, dan **turun sebagai hujan**. Proyek ini cocok untuk memperkuat pemahaman konsep sains.

**Langkah-langkah:**

- Siapkan sprite air, uap, awan, dan tetesan hujan.
- Gunakan animasi bergerak ke atas (untuk uap), lalu munculkan awan.
- Setelah beberapa detik, munculkan hujan turun dari awan kembali ke tanah.

- Gunakan blok **"broadcast"** untuk berpindah antar-tahap perubahan.
- Tambahkan penjelasan singkat dengan blok **"say"** untuk menjelaskan setiap tahap perubahan.

## Proyek 4: Game Interaktif Kosa Kata Bahasa Inggris

**Deskripsi Proyek:**
Tampilkan sebuah **gambar objek** (seperti "apple", "dog", atau "car"), lalu pengguna harus **mengetikkan nama objek tersebut dalam Bahasa Inggris**. Jika benar, skor bertambah.

**Langkah-langkah:**

- Siapkan beberapa gambar sprite (buah, hewan, benda).
- Gunakan blok **"ask"** untuk meminta pengguna mengetik jawaban.
- Gunakan kondisi **"if answer = [kata benar]"** untuk memeriksa jawaban.
- Tambahkan sistem skor dengan variabel.
- Buat tantangan waktu atau level kesulitan agar lebih menantang.

## Proyek 5: Simulasi “Tebak Planet”

**Deskripsi Proyek:**
Tampilkan gambar **planet-planet di tata surya**, lalu berikan **fakta singkat saat planet diklik**. Setelah itu, munculkan **kuis tentang planet**, misalnya: “Planet mana yang paling besar?”

**Langkah-langkah:**

- Siapkan sprite gambar untuk planet (Merkurius hingga Neptunus).
- Tambahkan blok **"when sprite clicked"** untuk menampilkan fakta unik setiap planet.
- Setelah seluruh planet diperkenalkan, munculkan kuis pilihan ganda.
- Buat sistem umpan balik: "Jawaban kamu benar!" atau "Coba lagi ya."
- Tambahkan efek suara dan tampilan latar belakang luar angkasa.

## Proyek 6: Petualangan Karakter di Dunia Matematika

**Deskripsi Proyek:**
Buat karakter utama yang harus menyelesaikan **soal matematika sederhana** untuk bisa melewati tantangan atau rintangan.

**Langkah-langkah:**

- Buat sprite karakter yang bisa berjalan ke kanan atau kiri.
- Tambahkan rintangan dengan soal, misalnya "5 + 3 = ?".
- Gunakan blok **"ask"** dan kondisi **"if answer = 8"** untuk membuka jalan.
- Buat level-level berbeda agar siswa tertantang menyelesaikannya.
- Tambahkan animasi, efek suara, dan poin skor.

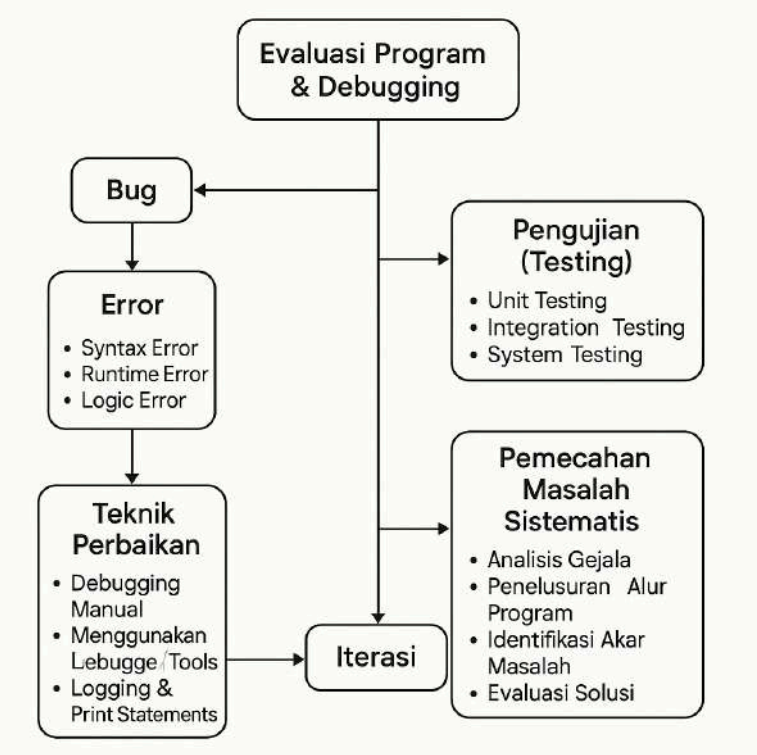
Setiap proyek bisa dikembangkan lebih lanjut sesuai kreativitas siswa. Guru disarankan memberikan tantangan tambahan seperti: menambahkan suara, waktu terbatas, atau leaderboard skor. Proyek-proyek ini tidak hanya menyenangkan tetapi juga memperkuat kemampuan berpikir logis, kreatif, dan literasi digital siswa SMP.

# BAB 5: Evaluasi Proyek & Debugging

## Tujuan Pembelajaran

Peserta didik mampu secara sistematis mengevaluasi proyek Scratch mereka sendiri dan proyek teman, mengidentifikasi kesalahan (bug), dan menerapkan teknik debugging yang efektif untuk mencari serta memperbaiki kesalahan tersebut.

## Peta Konsep




## Apersepsi

Pernahkah kalian membuat program atau game, lalu saat dicoba, kok tidak berjalan sesuai harapan? Atau ada bagian yang aneh? Jangan khawatir, itu sangat normal! Semua programmer, bahkan yang paling jago sekalipun, pasti menemukan "bug" dalam program mereka. Di bab ini, kita akan belajar menjadi detektif digital dan mencari serta memperbaiki bug itu!

## Penjelasan Konsep (Teori)

### Menjadi Detektif Digital: Evaluasi Proyek & Debugging

Pernahkah kalian membuat game di Scratch, tapi tiba-tiba sprite-nya tidak bergerak? Atau malah bergerak terus-terusan tanpa berhenti? Hmm... itu bukan karena kalian salah total, tapi karena program kalian sedang *kena bug*! Nah, mari kita belajar bagaimana cara

mengevaluasi proyek kita sendiri dan menjadi “detektif digital” untuk menemukan serta memperbaiki kesalahan-kesalahan tersebut. Yuk, simak penjelasan konsep-konsep penting berikut ini!

## Evaluasi Proyek: Mengapa Ini Penting?

**Evaluasi proyek** adalah proses untuk memeriksa kembali apakah program yang kita buat **sudah sesuai tujuan** dan **berfungsi dengan baik**. Evaluasi penting dilakukan agar program:

- Tidak mengalami error saat dijalankan.
- Memberikan pengalaman yang menyenangkan bagi pengguna.
- Memenuhi semua tujuan yang sudah direncanakan dari awal.

**Ibaratnya seperti mencicipi masakan sebelum disajikan** – kita harus pastikan semuanya oke!

## Testing: Uji Coba yang Cerdas

Testing adalah bagian dari evaluasi yang dilakukan dengan mencoba program secara langsung untuk mengetahui apakah ada bagian yang tidak berfungsi.

Beberapa jenis *testing* yang bisa kalian lakukan:

### Unit Testing

- Menguji bagian kecil dari program, seperti:
  “Apakah sprite berjalan ke kanan saat tombol ditekan?”

### Integration Testing

- Menguji apakah **semua bagian program bisa bekerja sama dengan baik**, seperti interaksi antar sprite atau antar skrip.

### User Acceptance Testing

- Mengajak teman atau guru mencoba program, lalu **memberikan masukan**. Siapa tahu mereka menemukan bug yang tidak kita sadari!

## Bug & Debugging: Musuh dan Jurus Andalan Programmer

Mari kita kenalan dengan dua istilah penting:

- **Bug**: Ini adalah **kesalahan dalam program** yang membuat program berjalan tidak seperti yang diharapkan.
- **Debugging**: Proses untuk **mencari, menemukan, dan memperbaiki bug**. Ini seperti menjadi detektif yang harus menyelidiki kasus misterius dalam program!

Semua programmer, bahkan yang profesional, pasti pernah berurusan dengan bug. Jadi, jangan takut kalau kalian menemukannya. Yang penting tahu cara menghadapinya!

### Teknik Debugging Efektif di Scratch

Agar proses debugging lebih mudah dan menyenangkan, berikut beberapa **teknik cerdas** yang bisa kalian gunakan:

#### Metode Observasi

- Jalankan program, lalu **amati dengan seksama**.
    - Apakah sprite bergerak aneh?
    - Apakah variabel menunjukkan nilai yang tidak sesuai harapan?

#### Metode Trace (Melacak Jejak Program)

- Gunakan blok `say` atau `say join [variabel] [pesan]` untuk menampilkan informasi saat program berjalan.
  Ini seperti **meninggalkan jejak roti** agar kita bisa melihat alur yang sedang dilalui program.
- Coba fitur **"Step-by-step"** di Scratch: jalankan program **blok demi blok** untuk melihat urutan kejadian dan perubahan nilai variabel.

#### Metode Isolasi Masalah

- **Nonaktifkan (disable)** bagian-bagian tertentu dari skrip agar bisa menemukan bagian mana yang bermasalah.
- **Sederhanakan** bagian kode yang rumit menjadi bentuk yang lebih simpel dulu.
- Periksa **kondisi `if` dan perulangan (`repeat` atau `forever`)**. Salah satu penyebab bug paling umum adalah kondisi yang salah atau perulangan yang tidak berhenti!

### Tips Tambahan

> Debugging itu seperti bermain teka-teki! Semakin kalian sering latihan, semakin jago kalian mengatasi masalah dalam program!

Dengan memahami konsep evaluasi dan debugging ini, kalian akan jadi programmer yang **teliti, cerdas, dan tidak mudah panik saat program error**. Selamat menjadi detektif digital di dunia Scratch, dan teruslah berlatih menyelesaikan “kasus-kasus misterius” dalam proyek kalian!

## Contoh Kasus atau Ilustrasi

#### Kasus 1: Program Tidak Berjalan Saat Tombol Diklik (App Inventor)

**Contoh Kasus:**

Kamu membuat aplikasi kalkulator. Ketika tombol "**Tambah**" di klik, tidak ada yang terjadi.

**Penyebab Umum:**

- Event "**Ketika tombol diklik**" belum dipasang.
- Blok perintah di dalam event belum lengkap.
- Salah memilih komponen (contoh: menaruh perintah di tombol yang salah).

**Strategi Debugging:**

- Periksa apakah kamu sudah memakai blok "when Button.Click".
- Tambahkan blok "**Show Alert"** atau "**Speak text**" untuk mengecek apakah event dijalankan.
- Cek ulang nama komponen (tombol) dan logika di dalamnya.

## Kasus 2: Karakter Tidak Bergerak Saat Ditekan Panah (Scratch)

**Contoh Kasus:**

Dalam game Scratch, kamu ingin sprite (karakter) bergerak ke kanan saat panah kanan ditekan. Tapi sprite tidak bergerak.

**Penyebab Umum:**

- Salah mengetik nama tombol (bukan “**panah kanan”).**
- Blok gerakan belum dimasukkan.
- Script tidak dijalankan dari awal.

**Strategi Debugging:**

- Gunakan blok "**ketika tombol panah kanan ditekan**", pastikan benar tulisannya.
- Tambahkan blok "**ubah x sebanyak 10**" untuk membuat sprite bergerak ke kanan.
- Jalankan ulang program dengan klik bendera hijau.

## Kasus 3: Nilai Skor Tidak Bertambah Saat Menyentuh Objek (Scratch)

**Contoh Kasus:**

Dalam game menangkap buah, skor seharusnya bertambah 1 setiap kali menyentuh buah. Tapi skor tetap 0.

**Penyebab Umum:**

- Blok **"jika menyentuh [buah]"** belum dipasang.
- Variabel skor tidak diubah nilainya.
- Objek yang disentuh salah.

**Strategi Debugging:**

- Pastikan blok **"ubah skor sebanyak 1"** ada dalam **"jika menyentuh".**
- Gunakan blok "**tampilkan variabel skor"** di layar untuk melihat nilainya.
- Tambahkan suara atau efek saat menyentuh objek untuk memastikan program berjalan.

## Kasus 4: Tampilan Aplikasi Tidak Sesuai di Layar HP (App Inventor)

**Contoh Kasus:**

Tampilan aplikasi terlihat rapi di komputer, tapi saat dijalankan di HP, komponen tumpang tindih atau tidak terlihat.

**Penyebab Umum:**

- Ukuran layar HP berbeda.
- Komponen tidak diatur agar menyesuaikan ukuran layar.

**Strategi Debugging:**

- Gunakan pengaturan **"Fill parent"** agar komponen menyesuaikan lebar/tinggi layar.
- Uji tampilan langsung dari HP menggunakan App Inventor Companion.
- Gunakan Label dan Button dengan ukuran dan margin yang konsisten.

### Kasus 5: Tips Praktis Saat Debugging

1. **Baca Blok Satu per Satu:** Coba pahami alur logika program dengan membaca blok dari atas ke bawah.
2. **Uji Coba Bertahap:** Jalankan program bagian per bagian, jangan langsung semua sekaligus.
3. **Gunakan "Tampilkan Variabel" atau "Alert":** Untuk melihat apakah nilai dan aksi sudah sesuai.
4. **Tanya Teman atau Guru:** Terkadang, orang lain bisa melihat kesalahan yang kita lewatkan.
5. **Jangan Panik!** Semua programmer pasti pernah melakukan kesalahan. Yang penting, sabar dan coba perbaiki satu per satu.

## Contoh Soal dan Pembahasan

### Aplikasi Kalkulator Tidak Menampilkan Hasil

**Soal:**

Kamu membuat aplikasi kalkulator di App Inventor. Setelah mengetik dua angka dan menekan tombol "Tambah", tidak ada hasil yang muncul di layar.

**Pertanyaan:**

Apa yang bisa kamu lakukan untuk menemukan dan memperbaiki masalah ini?

**Pembahasan:**

Langkah debugging:

1. Periksa apakah tombol **“Tambah”** benar-benar terhubung ke blok perintah.
2. Cek apakah hasil penjumlahan disimpan ke dalam label yang tampil di layar.
3. Gunakan "**Notifikasi"** atau blok "**Show Alert"** untuk melihat apakah perintah di dalam tombol berjalan.

**Solusi:**

Mungkin karena kamu lupa mengatur LabelHasil.Text ke angka1 + angka2.

### Scratch: Karakter Tidak Bergerak Saat Ditekan Tombol Panah

**Soal:**

Di Scratch, kamu membuat sprite yang seharusnya bergerak ke kanan saat tombol panah kanan ditekan. Tapi sprite tidak bergerak sama sekali.

**Pertanyaan:**

Bagaimana kamu bisa mencari tahu apa yang salah?

**Pembahasan:**

Langkah debugging:

1. Pastikan kamu menggunakan blok **"when right arrow key pressed"**.
2. Periksa apakah di bawah blok itu ada blok **"change x by 10"** (untuk bergerak ke kanan).
3. Jalankan program dan klik **"See inside"** untuk melihat apakah kode aktif.
4. Cek apakah sprite yang dimaksud adalah sprite yang sedang diprogram.

**Solusinya:**

Mungkin kamu salah menulis arah atau belum menambahkan blok **“change x by 10”**.

## App Inventor: Gambar Tidak Muncul di Aplikasi

**Soal:**

Kamu ingin menampilkan gambar di layar awal aplikasi. Tapi saat dijalankan, gambar tidak muncul.

**Pertanyaan:**

Apa saja yang bisa kamu periksa untuk memperbaiki masalah ini?

**Pembahasan:**

Langkah debugging:

1. Pastikan kamu sudah mengunggah file gambar ke **bagian Media.**
2. Cek apakah di **komponen Image,** bagian Picture sudah diatur ke nama file gambar yang benar (misal: kucing.jpg).
3. Pastikan tidak ada spasi atau huruf besar/kecil yang salah di nama file.

**Solusinya:**

Mungkin file gambar tidak terunggah, atau nama file diatur salah.

## Scratch: Suara Tidak Terdengar Saat Sprite Bicara

**Soal:**

Kamu menambahkan blok suara di Scratch agar sprite mengucapkan "Halo!". Tapi saat dijalankan, tidak ada suara yang terdengar.

**Pertanyaan:**

Apa yang harus kamu periksa?

**Pembahasan:**

Langkah debugging:

1. Periksa apakah kamu sudah menambahkan blok "play sound Halo until done".
2. Buka tab “Sounds” dan cek apakah suara "Halo" benar-benar ada.

3. Pastikan volume tidak disetel ke 0 atau sprite tidak dalam kondisi mute.
4. Coba jalankan suara dari tab “Sounds” untuk mengetes apakah file bisa diputar.

**Solusinya:**

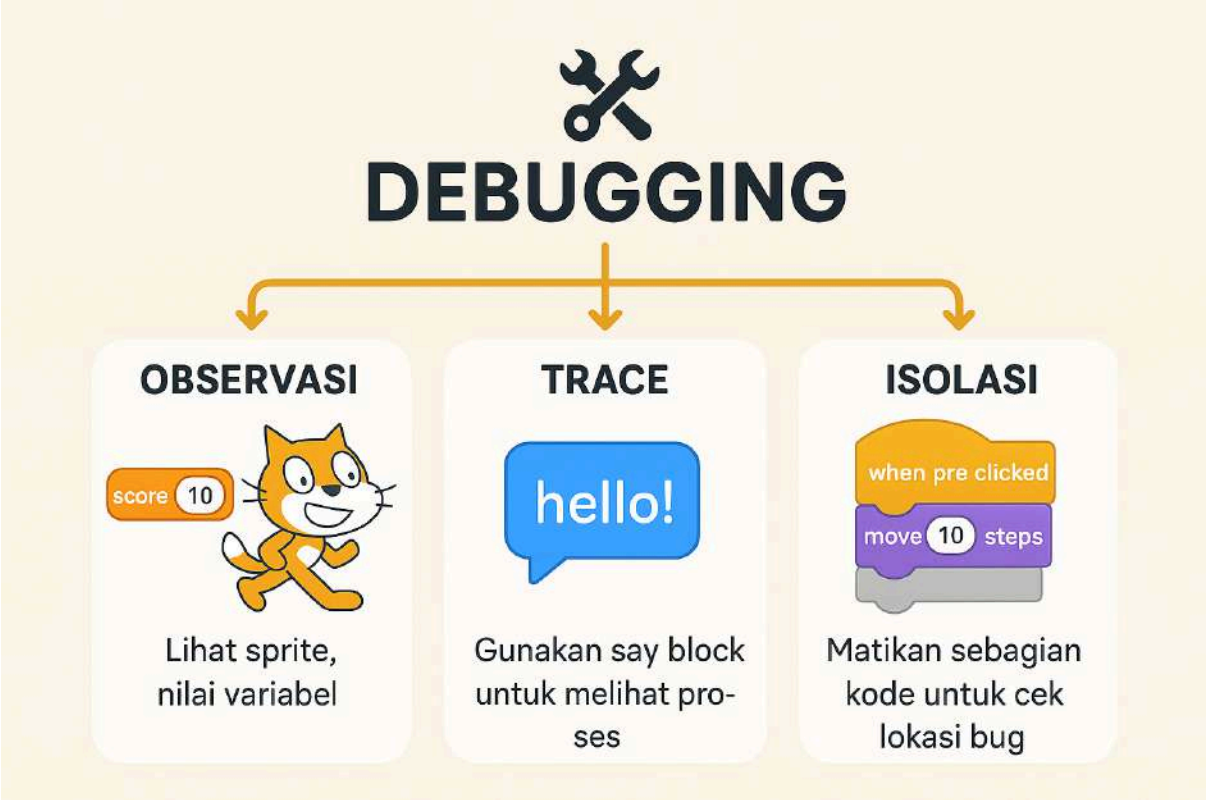
Mungkin suara belum diunggah, atau kamu memakai nama suara yang salah di blok.

## Fakta Menarik / Fun Facts

- Bug pertama yang tercatat dalam sejarah komputer (yang melibatkan ngengat) terjadi pada tahun 1947!
- Debugging bisa memakan waktu lebih banyak daripada menulis kode itu sendiri. Ini adalah bagian normal dari pengembangan perangkat lunak.

## Tips Belajar atau Tips Cepat Menghafal

### Mind Mapping: Teknik Debugging




**Tips Emas:**

- **“Cari tahu penyebabnya, lalu perbaiki":** Jangan langsung mengubah kode secara acak. Pahami dulu di mana masalahnya.
- Gunakan blok say seperti kamu memberi petunjuk pada detektif.
- **Sabar dan logis:** Debugging butuh kesabaran dan pemikiran yang terstruktur. Latih logika dengan menjelaskan kode ke teman seolah-olah mereka belum pernah lihat Scratch.

### Cerita Mudah Diingat:

"Suatu hari Grace Hopper menemukan ngengat / kutu (bug) di dalam komputer besar (mainframe). Itulah bug pertama di dunia komputer!"

# Aktivitas Siswa

## Aktivitas 1: Saling Menguji Proyek Teman (Peer Testing)

**Deskripsi Singkat:**

Kamu dan temanmu akan bertukar proyek yang sudah dibuat di App Inventor. Tugasmu adalah mencoba aplikasi temanmu, menemukan kesalahan (bug), lalu memberi saran perbaikan.

**Langkah-langkah:**

1. Buka proyek App Inventor buatanmu.
2. Tukar file proyek dengan temanmu.
3. Coba jalankan aplikasi temanmu dan amati dengan teliti.
4. Catat jika ada kesalahan. Misalnya:
    - “Tombol tidak berfungsi saat diklik.”
    - “Suara tidak muncul setelah menekan tombol.”
5. Tulis saran atau perbaikan untuk temanmu dengan sopan.
6. Kembalikan proyek beserta komentarmu ke temanmu.

**Contoh Pseudocode untuk Uji Coba:**

```
Saat Button1.Diklik
  Jika Label1.Teks == ""
    Tampilkan "Label belum diisi"
  Selesai
Selesai
```

## Aktivitas 2: Tantangan Debug It! (Versi 1)

**Deskripsi Singkat:**

Kamu akan diberikan proyek App Inventor yang berisi bug tersembunyi. Tugasmu adalah mencari dan memperbaikinya.

**Langkah-langkah:**

1. Terima file proyek dari guru.
2. Jalankan aplikasi, perhatikan apakah ada yang tidak berjalan seperti semestinya.
3. Cari tahu bagian mana dari kode blok yang salah.
4. Perbaiki bug-nya.
5. Tulis catatan: bug apa yang kamu temukan, dan bagaimana kamu memperbaikinya.

**Contoh Bug dan Solusi:**

Masalah: Saat tombol diklik, skor tidak bertambah.

Pseudocode Sebelumnya:

```
Saat Button1.Diklik
    Skor ← Skor
```

Perbaikan:

```
Saat Button1.Diklik
    Skor ← Skor + 1
Selesai
```

## Aktivitas 3: Tantangan Debug It! (Versi 2 dengan Waktu)

**Deskripsi Singkat:**

Sama seperti aktivitas sebelumnya, tapi kamu diberi batas waktu. Tujuannya agar kamu bisa berpikir cepat dan teliti.

**Langkah-langkah:**

1. Terima proyek dengan bug dari guru.
2. Kamu punya waktu 20–30 menit untuk menyelesaikannya.
3. Temukan semua bug secepat dan setepat mungkin.
4. Jelaskan: apa bug-nya dan bagaimana kamu memperbaikinya.

**Contoh Pseudocode Penjelasan:**

Bug: Gambar karakter tidak bergerak saat tombol diklik.

Perbaikan:

```
Saat ButtonGerak.Diklik
    GambarX ← GambarX + 10
Selesai
```

## Aktivitas 4: Presentasi Hasil Debug

**Deskripsi Singkat:**

Setelah kamu selesai memperbaiki bug, kamu akan mempresentasikan hasil kerjamu ke teman-teman.

**Langkah-langkah:**

1. Siapkan penjelasan singkat: bug apa yang kamu temukan dan bagaimana memperbaikinya.
2. Tampilkan kode blok yang kamu ubah di App Inventor.
3. Jelaskan dengan bahasa yang mudah dimengerti.
4. Terima pertanyaan dan masukan dari teman.

**Contoh Presentasi Singkat:**

> "Awalnya, tombol 'Mulai' tidak merespon karena tidak ada blok perintah di dalamnya. Saya tambahkan blok `Navigasi ke Layar2` agar saat diklik, aplikasi masuk ke layar permainan."

# Rangkuman

**Evaluasi** dan **debugging** adalah bagian penting dalam membuat program komputer. Saat membuat program, sering kali muncul *bug* — yaitu **kesalahan dalam kode**. Ini **hal yang wajar** dan dialami hampir semua programmer.

Agar program bisa berjalan dengan baik, kita perlu belajar cara **mendeteksi dan memperbaiki bug**. Teknik yang bisa digunakan antara lain:

- **Observasi**, yaitu memperhatikan hasil program dengan cermat.
- **Melacak alur program** menggunakan blok *say*, untuk melihat bagian mana yang bekerja dan mana yang tidak.
- **Mengisolasi masalah**, artinya memeriksa satu bagian kecil kode untuk mencari sumber kesalahan.

Dengan cara ini, kita bisa menjadi seperti **“detektif kode”** yang hebat!

Melakukan debugging memang tidak selalu mudah, tapi ini adalah **keterampilan penting**. Dengan menjadi **teliti**, **logis**, dan **sabar**, kita bisa membuat program yang **lancar**, **menyenangkan**, dan **bebas masalah**.

# Latihan Soal

## A. Benar atau Salah (5 Soal)

1. Bug adalah fitur tambahan dalam sebuah program.
   **Jawaban: Salah**

2. Debugging bisa membantu memperbaiki kesalahan dalam program.
   **Jawaban: Benar**

3. Testing hanya perlu dilakukan setelah program selesai dibuat.
   **Jawaban: Salah**

4. Menggunakan blok `say` bisa membantu melacak nilai variabel saat program berjalan.
   **Jawaban: Benar**

5. Semua bug dapat diperbaiki hanya dengan menambahkan sprite baru.
   **Jawaban: Salah**

## B. Pilihan Ganda (10 Soal)

6. Apa itu debugging?
   A. Menambahkan musik ke program
   B. Membuat tampilan lebih menarik
   C. Mencari dan memperbaiki kesalahan dalam kode
   D. Menghapus semua skrip
   **Jawaban: C**

7. Mengapa evaluasi proyek itu penting?
   A. Agar proyek terlihat besar
   B. Untuk mengetahui siapa yang menyalin
   C. Untuk memastikan program berjalan sesuai tujuan
   D. Untuk membuat warna sprite jadi cerah
   **Jawaban: C**

8. Teknik trace biasanya dilakukan dengan cara...
   A. Menghapus variabel
   B. Menambahkan sprite
   C. Menggunakan blok say untuk melihat alur program
   D. Menambahkan musik latar
   **Jawaban: C**

9. Testing jenis apa yang dilakukan oleh orang lain untuk mencoba program kita?
   A. Unit testing
   B. Self testing
   C. Integration testing
   D. User acceptance testing
   **Jawaban: D**

10. Jika sprite tidak bergerak saat di klik, apa kemungkinan masalahnya?
    A. Blok warna
    B. Blok event tidak digunakan
    C. Musik tidak dinyalakan
    D. Ukuran sprite terlalu kecil

**Jawaban: B**

11. Langkah awal yang baik untuk debugging adalah...
    A. Menutup laptop
    B. Menghapus semua skrip
    C. Mengamati jalannya program
    D. Menambahkan efek suara
    **Jawaban: C**

12. Jika skor tidak bertambah saat koin disentuh, apa yang perlu diperiksa?
    A. Warna sprite
    B. Ukuran panggung
    C. Blok change skor by
    D. Gambar latar
    **Jawaban: C**

13. Apa fungsi dari mode step-by-step?
    A. Menampilkan tampilan layar
    B. Menjalankan program satu blok demi satu
    C. Mengganti sprite secara otomatis
    D. Mewarnai latar belakang
    **Jawaban: B**

14. Metode debugging yang menonaktifkan sebagian blok kode disebut...
    A. Trace
    B. Isolasi
    C. Integrasi
    D. Modifikasi
    **Jawaban: B**

15. Bug dalam sebuah program adalah...
    A. Bagian dari animasi
    B. Salah satu alat dalam Scratch
    C. Kesalahan yang menyebabkan program tidak berjalan benar
    D. Jenis suara dalam game
    **Jawaban: C**

## C. Isian Singkat (5 Soal)

16. Proses memperbaiki kesalahan dalam program disebut _____.
    **Jawaban: debugging**

17. Bug pertama kali ditemukan oleh Grace Hopper dalam bentuk _____.
    **Jawaban: ngengat**

18. Teknik debugging dengan melihat langsung jalannya program disebut _____.
    **Jawaban: observasi**

19. Blok yang digunakan untuk menampilkan nilai variabel adalah _____.
    **Jawaban: say**

20. Jika ingin menjalankan program satu per satu, gunakan fitur _____.
    **Jawaban: step-by-step**

## D. Soal Eksplorasi

**Soal:**

Kamu sedang menguji proyek Scratch milik temanmu. Saat karakter mengenai musuh, seharusnya muncul “Game Over”, tapi tidak terjadi apa-apa. Jelaskan langkah-langkah yang akan kamu lakukan untuk menemukan dan memperbaiki bug ini menggunakan teknik debugging yang sudah kamu pelajari. Tuliskan urutan metodenya dan apa yang akan kamu cek dari skrip program tersebut.

**Jawaban:** Langkah pertama adalah menggunakan metode observasi dengan menjalankan program dan melihat apakah karakter benar-benar bersentuhan dengan musuh. Lalu saya gunakan blok `say` untuk menampilkan status kondisi "touching musuh?". Jika tidak muncul, saya periksa apakah kondisi if ditulis dengan benar dan apakah nama sprite musuh sesuai. Jika masih belum berhasil, saya coba nonaktifkan beberapa skrip lain untuk mengetahui apakah ada yang mengganggu. Setelah ketemu masalahnya, saya perbaiki logika kondisinya agar `Game Over` muncul saat terkena musuh.

**Soal:**

Dalam sebuah proyek Scratch, temanmu mengatakan bahwa game-nya sering tiba-tiba berhenti sendiri saat dijalankan. Jelaskan cara kamu membantu mencari bug tersebut. Apa saja teknik debugging yang kamu gunakan, dan bagaimana kamu menentukan bagian mana dari skrip yang menyebabkan masalah?

**Jawaban:** Saya akan mulai dengan metode observasi untuk mengetahui kapan program berhenti. Lalu, saya gunakan mode step-by-step untuk menjalankan blok satu per satu. Di saat program berhenti, saya periksa blok terakhir yang dijalankan. Saya juga bisa menggunakan blok `say` untuk menampilkan nilai variabel yang sedang berubah. Jika ditemukan perulangan tak berakhir atau kondisi salah, saya akan perbaiki dan mencatat perubahan apa yang saya lakukan.

**Soal:**

Kamu membuat game “Tangkap Koin”, namun pemain tidak mendapatkan poin meskipun koin sudah disentuh. Gunakan teknik debugging yang kamu ketahui untuk menyelesaikan masalah ini. Jelaskan kemungkinan penyebab dan perbaikan yang kamu lakukan.

**Jawaban:** Pertama saya amati apakah sprite pemain benar-benar menyentuh koin. Lalu saya gunakan blok `say join "Skor: " skor` untuk melihat apakah nilai skor berubah. Jika tidak, saya periksa apakah kondisi "touching koin" ditulis dengan benar. Saya juga memastikan bahwa blok `change skor by 1` berada di tempat

yang tepat dan tidak terlewat. Setelah itu saya coba jalankan ulang program untuk memastikan bug telah diperbaiki.

**Soal:**

Kamu menguji proyek temanmu dan menemukan bahwa sprite “pemain” terkadang tidak bereaksi saat menyentuh sprite “musuh”. Jelaskan langkah-langkah debugging apa yang akan kamu lakukan untuk mencari penyebab masalah ini. Sertakan juga teknik-teknik yang kamu gunakan seperti observasi, trace, atau isolasi.

**Soal:**

Dalam sebuah game Scratch, skor bertambah dua kali setiap kali koin disentuh. Jelaskan bagaimana kamu bisa mengetahui apakah ini bug atau fitur yang memang diinginkan. Apa yang akan kamu tanyakan kepada pembuat proyek untuk memastikannya?

**Soal:**

Kamu membuat program edukasi di Scratch yang harus menampilkan “Benar!” saat jawaban pengguna sesuai. Namun, tidak ada reaksi ketika pengguna memilih jawaban yang benar. Jelaskan cara kamu akan mencari tahu letak kesalahannya dan bagaimana kamu memperbaikinya dengan menggunakan metode debugging yang telah dipelajari.

# Tugas Proyek

## Proyek 1: Detektif Bug di Game Scratch

**Deskripsi:**
Guru telah memberikan kamu sebuah game yang dibuat di Scratch. Tugasmu adalah menjadi "detektif bug"!

**Langkah-langkah:**

1. Jalankan dan mainkan game dengan teliti.
2. Identifikasi minimal **tiga bug** atau kesalahan yang terjadi saat game berjalan.
3. Untuk setiap bug, catat:
    - Apa yang salah?
    - Kapan bug tersebut muncul?
    - Bagaimana kamu memperbaikinya?
4. Tambahkan blok **“say”** ke karakter di Scratch untuk mencatat proses debugging-mu, seolah-olah karakter sedang melaporkan temuannya.

## Proyek 2: Game Misteri Berisi Bug

**Deskripsi:**
Buat game sederhana (misalnya: “Tangkap Bintang” atau “Hindari Alien”) yang kamu ciptakan sendiri. Tapi… tambahkan **satu bug tersembunyi** secara sengaja!

**Langkah-langkah**

1. Rancang dan buat game dengan alur permainan yang menarik.
2. Sisipkan satu bug buatan tanpa memberitahu temanmu.
3. Minta temanmu memainkan dan mendebug game tersebut.
4. Diskusikan bersama:
    - Apakah mereka berhasil menemukannya?
    - Bagaimana cara mereka memperbaikinya?

## Proyek 3: Catatan Petualangan Debugging

**Deskripsi:**
Setiap proses menemukan bug adalah petualangan yang seru. Catat semua prosesmu saat memperbaiki bug dalam proyek game Scratch.

**Langkah-langkah:**

1. Pilih satu proyek Scratch yang kamu kerjakan.
2. Saat menemukan bug, buat log berisi:
    - Langkah-langkah yang kamu coba untuk mencari dan memperbaikinya.
    - Teknik atau strategi apa yang paling membantumu (misalnya: membaca ulang blok kode, menambahkan suara sebagai alarm, mencoba per bagian).
    - Apa pelajaran terpenting dari pengalaman tersebut?

## Proyek 4: Tiga Musuh Utama di Dunia Scratch

**Deskripsi:**
Dalam dunia coding Scratch, bug bisa datang dari mana saja! Buatlah daftar tiga penyebab umum bug yang sering kamu atau temanmu alami.

**Langkah-langkah:**

1. Diskusikan bersama teman atau refleksikan dari pengalamanmu.
2. Buat daftar minimal 3 penyebab bug, misalnya:
    - Salah ketik nama variabel.
    - Blok event tidak dipasang dengan benar.
    - Logika if/else yang tidak tepat.
3. Tambahkan contoh sederhana dan saran cara menghindarinya.

## Proyek 5: Mind Map “Jalan Ninja Debugging”

**Deskripsi:**
Visualisasikan proses debugging ke dalam sebuah **mind map** kreatif yang bisa kamu tempel di buku catatan atau papan belajar.

**Langkah-langkah:**

1. Gunakan kertas warna atau aplikasi digital untuk membuat mind map.
2. Masukkan langkah-langkah utama dalam debugging, seperti:
    - Menemukan masalah.
    - Menganalisis kemungkinan penyebab.
    - Mencoba perbaikan.
    - Menguji ulang.
    - Mencatat hasilnya.
3. Tambahkan ikon, warna, dan gambar supaya mudah diingat dan menyenangkan dilihat.

## Proyek 6: Turnamen Debugging Kelas

**Deskripsi:**
Buat tantangan debugging bersama teman-teman sekelas!

**Langkah-langkah:**

1. Setiap siswa membuat proyek Scratch berisi satu bug.
2. Saling bertukar proyek dan mencoba menemukan serta memperbaiki bug teman.
3. Beri poin untuk kecepatan dan ketepatan debugging.
4. Akhiri dengan refleksi bersama: teknik apa yang paling sering berhasil?

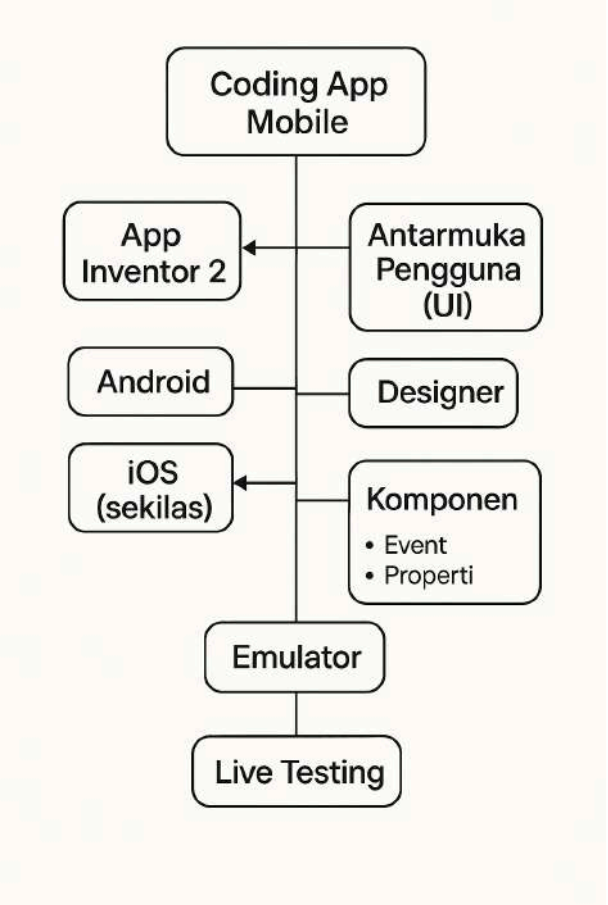
Proyek-proyek ini dirancang untuk membangun **ketelitian, kreativitas, dan kerja sama** dalam belajar coding. Selamat mencoba, calon programmer andal! 💻✨

# BAB 6: Persiapan ke Mobile App dengan App Inventor

## Tujuan Pembelajaran

Peserta didik memahami konsep dasar pengembangan aplikasi mobile, mengenal antarmuka dan fungsionalitas dasar App Inventor sebagai platform pengembangan visual, dan dapat membuat aplikasi mobile sederhana.

## Peta Konsep




## Apersepsi

Setiap hari, kalian pasti menggunakan berbagai aplikasi di smartphone kalian, seperti WhatsApp, TikTok, atau game Mobile Legends. Pernahkah terbayang bagaimana aplikasi-aplikasi itu dibuat? Apakah harus jago coding yang rumit? Ternyata tidak selalu! Ada cara yang lebih mudah, mirip seperti Scratch, untuk membuat aplikasi di HP. Namanya App Inventor!

# Penjelasan Konsep (Teori)

Di bab ini, kita akan menjelajahi dunia pengembangan aplikasi mobile dengan cara yang sederhana namun seru! Siapa bilang membuat aplikasi itu harus jago coding? Dengan bantuan platform visual bernama **App Inventor**, kalian bisa merancang aplikasi Android sendiri, bahkan tanpa menulis kode rumit. Yuk, kita pahami dulu konsep dasarnya!

## Apa Itu Aplikasi Mobile?

Sebelum mulai membuat aplikasi, penting untuk tahu dulu **apa itu aplikasi mobile**.

- **Aplikasi mobile** adalah program atau perangkat lunak yang dibuat khusus agar bisa berjalan di perangkat seperti **smartphone** atau **tablet**.
- Contoh aplikasi mobile yang sering kalian gunakan sehari-hari: **WhatsApp**, **TikTok**, **Instagram**, **Mobile Legends**, dan masih banyak lagi.

### Perbedaan Aplikasi Mobile dan Aplikasi Web

| Aplikasi Mobile | Aplikasi Web |
|---|---|
| Diunduh dan diinstal di HP | Dibuka lewat browser (seperti Chrome) |
| Bisa digunakan **tanpa internet** (tergantung jenis aplikasinya) | Biasanya **butuh koneksi internet** |
| Didesain untuk perangkat mobile | Didesain untuk semua perangkat, tapi lebih cocok untuk layar besar |

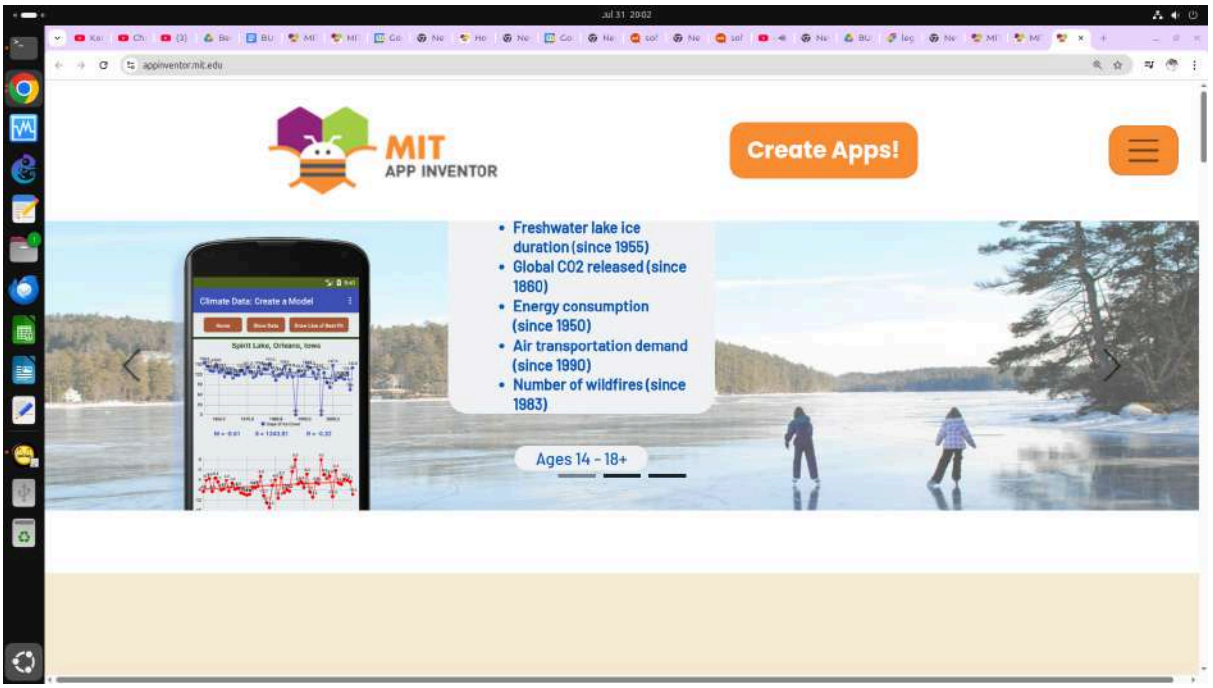
## Sistem Operasi Mobile: Android vs iOS

Aplikasi mobile berjalan di atas sistem operasi (OS) yang berbeda. Dua OS mobile yang paling populer adalah:

- **Android** – dikembangkan oleh Google, digunakan oleh banyak merek seperti Samsung, Oppo, Xiaomi, dll.
- **iOS** – dikembangkan oleh Apple, hanya digunakan di iPhone dan iPad.

*Catatan: Dalam pembelajaran ini, kita akan fokus pada Android karena App Inventor hanya bisa membuat aplikasi Android.*

## Mengenal App Inventor: Bikin Aplikasi Tanpa Ribet!




### Apa Itu App Inventor?

- **App Inventor** adalah sebuah platform visual buatan **MIT (Massachusetts Institute of Technology)** yang memungkinkan siapapun (termasuk kalian!) untuk membuat aplikasi Android **tanpa menulis kode**.
- Alih-alih mengetik program, kalian cukup **menyusun blok-blok seperti puzzle**, mirip dengan Scratch!

### Kenapa Cocok untuk Pemula?

- Tidak perlu pusing dengan aturan penulisan kode (sintaksis).
- Antarmukanya mudah dipahami, cocok untuk kalian yang baru belajar.
- Kalian bisa langsung melihat hasil aplikasi yang dibuat.

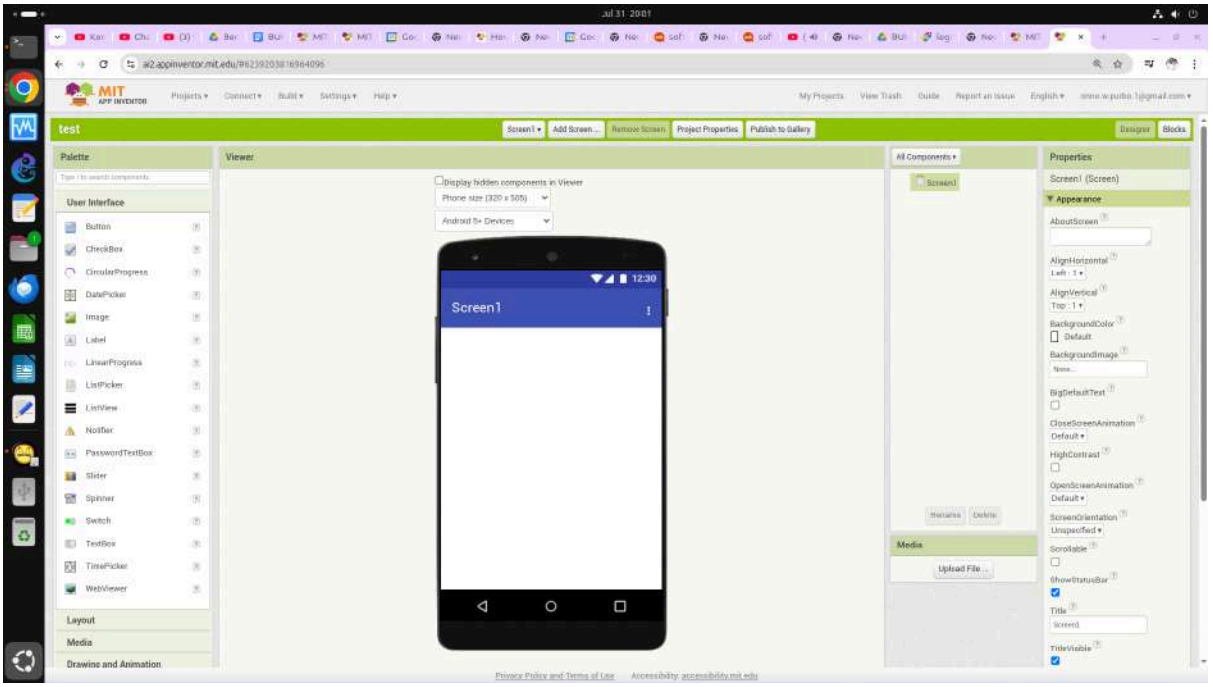
### Apa Saja yang Dibutuhkan?

Untuk mulai membuat aplikasi dengan App Inventor, kalian perlu:

- Akun Google
- Koneksi internet
- Komputer atau laptop
- Akses ke situs: ai2.appinventor.mit.edu

## Mengenal Tampilan App Inventor

App Inventor punya dua tampilan utama yang penting untuk dipahami:




## Designer View (Tampilan Desain)

Di sinilah kalian mendesain tampilan aplikasi, seperti menata tombol, gambar, dan teks.

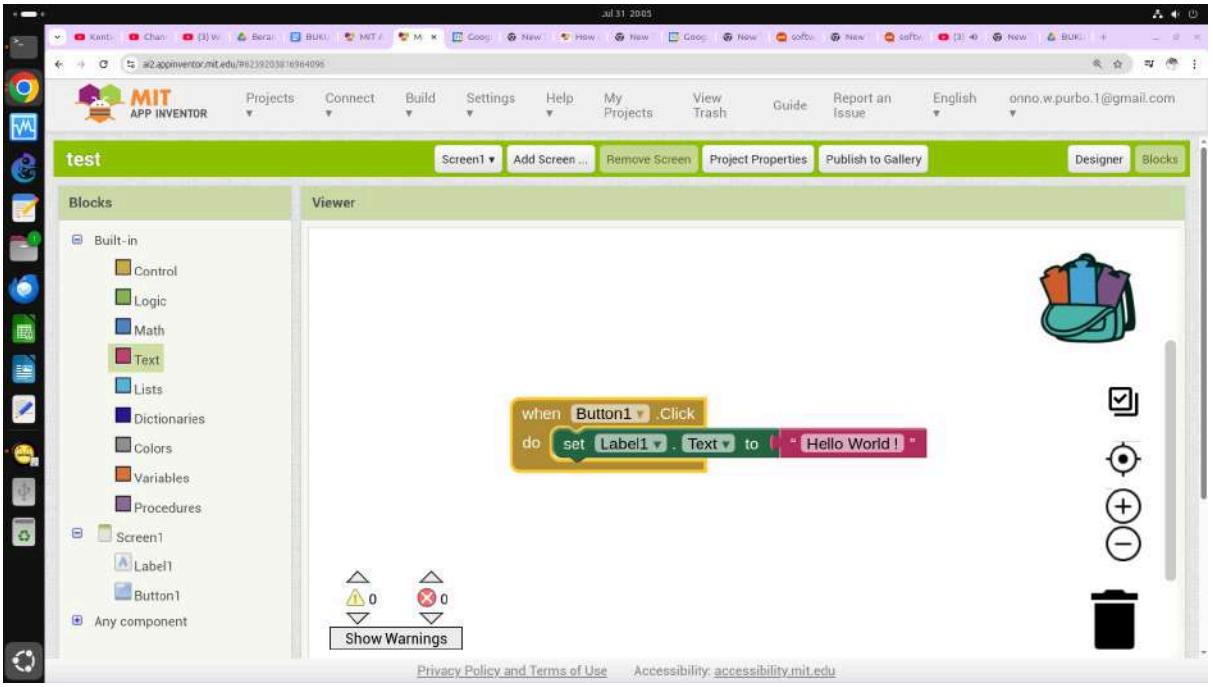
Komponen di Designer View:

- **Palette**: Tempat semua komponen siap pakai (tombol, teks, gambar, dll).
- **Viewer**: Layar tiruan HP tempat kalian menyusun komponen.
- **Components**: Daftar semua komponen yang kalian tambahkan.
- **Properties**: Untuk mengatur tampilan dan perilaku komponen (misalnya warna tombol, tulisan, gambar).

## Blocks Editor (Tampilan Blok Kode)

Di sinilah kalian menambahkan logika atau perintah agar aplikasi bisa "hidup".

- Blok kode disusun seperti potongan puzzle.
- Ada blok untuk:
    - **Kontrol** (contoh: if...then)
    - **Logika** (benar/salah)
    - **Teks dan angka**
    - **Blok khusus untuk tiap komponen (seperti Button atau Label)**




*Penting! Block code App Inventor tetap memakai bahasa Inggris. Contoh:*

```
when Button1.Click do
  set Label1.Text to "Halo Dunia!"
```

## Komponen dan Event Dasar

Saat membuat aplikasi, kalian akan bekerja dengan dua hal utama:

### Komponen (Components)

Komponen adalah bagian-bagian dari aplikasi yang bisa dilihat atau digunakan pengguna.

Beberapa komponen dasar:

- **Button**: Tombol yang bisa diklik
- **Label**: Menampilkan teks
- **TextBox**: Tempat pengguna mengetik teks
- **Image**: Menampilkan gambar
- **SoundPlayer**: Untuk memutar suara

### Event (Kejadian)

Event adalah **aksi atau peristiwa** yang terjadi saat aplikasi digunakan dan bisa memicu blok kode.

Contoh event:

- **Button.Click** – saat tombol diklik
- **Screen.Initialize** – saat aplikasi dibuka pertama kali
- **TextBox.TextChange** – saat teks diubah

*Setiap event bisa dipasangkan dengan blok kode tertentu agar aplikasi bisa merespons tindakan pengguna.*

### Siap Jadi Pembuat Aplikasi?

Setelah memahami teori dasar ini, kalian sudah punya bekal awal untuk memulai membuat aplikasi Android kalian sendiri. Seru, kan? Di bab-bab selanjutnya, kita akan **langsung praktek** membuat aplikasi pertama kalian. Siapkan ide-ide kreatif dan bersiaplah jadi **young app developer**!

## Contoh Kasus atau Ilustrasi

### Kasus 1: "Salam Digital: Halo Dunia!"

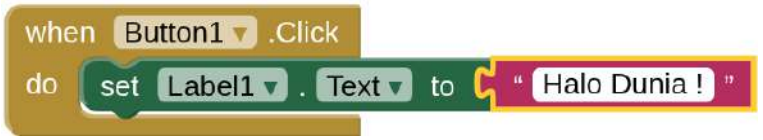
**Deskripsi:**
Bayangkan kamu membuat aplikasi yang menyapa pengguna saat mereka menekan tombol. Ini adalah dasar dari semua aplikasi: **interaksi!**

**Komponen:**

- Label (teks awal: *Halo!*)
- Button (teks: *Klik Saya!*)

**Blok Kode:**

```
when Button1.Click
do set Label1.Text to "Halo Dunia!"
```




**Ilustrasi:**
Tombol ditekan → Label berubah dari "Halo!" menjadi "Halo Dunia!"
Cocok untuk memperkenalkan konsep *event handler* dan *properti komponen*.

### Kasus 2: "Kucing Bersuara"

**Deskripsi:**
Bayangkan ada gambar kucing yang bisa mengeluarkan suara saat disentuh. Seru, bukan?

**Komponen:**

- Image (gambar kucing)
- Sound (mengisi suara meong)

**Blok Kode:**

```
when Image1.Click
do call Sound1.Start
```


**Ilustrasi:**
Saat gambar kucing ditekan → aplikasi pemutar suara “meong”.
Melatih siswa mengenali penggunaan **media dan event** pada gambar.

## Kasus 3: "Hitung Skor Otomatis!"

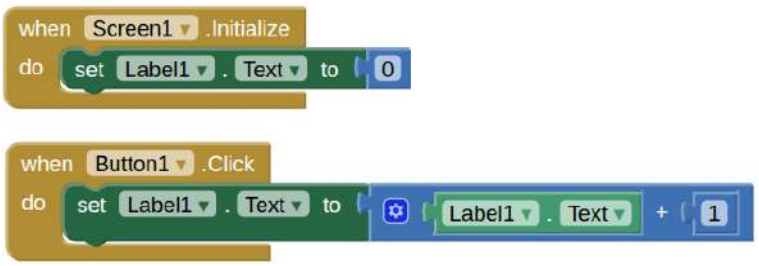
**Deskripsi:**
Kamu membuat aplikasi penghitung skor. Setiap kali tombol diklik, nilainya naik satu.

**Komponen:**

- Button (teks: *Tambah Skor*)
- Label (teks awal: 0)

**Blok Kode:**

```
when Button1.Click
do set Label1.Text to Label1.Text + 1
```




**Ilustrasi:**
Setiap klik → nilai skor bertambah (1, 2, 3, …).
Konsep penting: **variabel**, **aritmatika**, dan **update tampilan**.

## Kasus 4: "Sihir Warna Latar!"

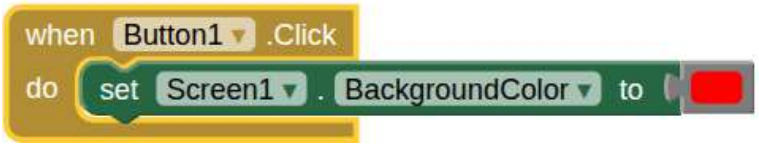
**Deskripsi:**
Buat tombol yang saat ditekan akan mengubah warna latar layar. Ini memberikan efek visual yang menyenangkan.

**Komponen:**

- Button (teks: *Ubah Warna!*)

**Blok Kode:**

```
when Button1.Click
do set Screen1.BackgroundColor to any color
```




**Ilustrasi:**
Setiap tekan tombol → warna latar berubah secara acak.
Mengenalkan konsep **visual feedback** dan **atribut antarmuka**.

## Kasus 5: "Lampu Menyala dan Padam"

**Deskripsi:**
Aplikasi bisa menyalakan atau mematikan gambar lampu. Seperti saklar digital!

**Komponen:**

- Button (teks: *Nyalakan/Matikan*)
- Image (gambar lampu, bisa berubah dari “off” ke “on”)
- Variabel status (boolean: nyala/mati)

**Blok Kode:**

```
when Button1.Click
do if status = true
     set Image1.Picture to "lampu_off.png"
     set status to false
   else
     set Image1.Picture to "lampu_on.png"
     set status to true
```

**Ilustrasi:**
Klik pertama → lampu menyala
Klik kedua → lampu mati
Melatih siswa memahami **kondisi (if-else)** dan **kontrol status aplikasi**.

### Kasus 6: "Kuis Kilat: Tebak Angka!"

**Deskripsi:**
Siswa menekan tombol untuk menampilkan angka acak, lalu menebaknya lewat input.

**Komponen:**

- Button (*Tampilkan Angka*)
- Label (*Tampilkan Angka Acak*)
- TextBox (*Tempat Menjawab*)
- Button (*Cek Jawaban*)
- Label (*Hasil Cek*)

**Blok Kode:**

```
when Button1.Click
do set Label1.Text to random integer from 1 to 10

when Button2.Click
do if TextBox1.Text = Label1.Text
    set Label2.Text to "Benar!"
  else
    set Label2.Text to "Coba lagi!"
```

**Ilustrasi:**
Aplikasi memilih angka → siswa menebak → aplikasi memberi umpan balik.
Memperkenalkan konsep **random**, **perbandingan**, dan **logika interaktif**.

## Contoh Soal dan Pembahasan

**Soal:**
Apa perbedaan antara App Inventor dan Scratch?
**Jawaban:**
Scratch untuk membuat animasi dan game di komputer, sedangkan App Inventor membuat aplikasi nyata yang bisa dijalankan di HP Android.

**Soal:**
Kamu ingin menampilkan teks "Halo, Budi!" saat tombol diklik. Bagaimana susunan komponennya?
**Jawaban:**

- Tambahkan `Label` dan `Button` di Designer View.
- Atur `Button.Text` jadi “Sapa Saya!” dan `Label.Text` jadi “Budi”.
- Di Blocks Editor:
  `when Button1.Click do → set Label1.Text to "Halo, Budi!"`

**Soal:**

Apa event yang digunakan saat tombol diklik?
**Jawaban:**

```
when Button.Click
```

**Soal:**

Jika ingin suara terdengar saat gambar ditekan, blok apa yang digunakan?
**Jawaban:**

```
when Image1.Click do → call SoundPlayer.Start
```

**Soal:**

Jelaskan perbedaan utama antara Scratch dan App Inventor dari segi output dan tujuan.
**Jawaban:**
Scratch lebih fokus pada animasi, game, dan cerita interaktif di komputer/web browser. App Inventor fokus pada pembuatan aplikasi yang bisa diinstal dan dijalankan di perangkat mobile (Android), dengan fitur-fitur yang memanfaatkan sensor HP.

**Soal:**

Jika kamu ingin membuat aplikasi yang menampilkan namamu dan sebuah tombol. Ketika tombol diklik, namamu berubah menjadi "Halo, [Nama Kamu]!". Bagaimana langkah-langkah di Designer dan Blocks Editor?
**Jawaban:**

- **Designer:**
  - Drag `Label` dari Palette ke Viewer. Ubah properti `Text` menjadi namamu (misal: "Budi").
  - Drag `Button` dari Palette ke Viewer. Ubah properti `Text` menjadi "Sapa Saya!".
- **Blocks Editor:**
  - Ambil blok `when Button1.Click do`.
  - Ambil blok `set Label1.Text to`.
  - Ambil blok `Text` dari kategori Text dan ketik "Halo, Budi!".
  - Gabungkan menjadi: `when Button1.Click do` -> `set Label1.Text to "Halo, Budi!"`.

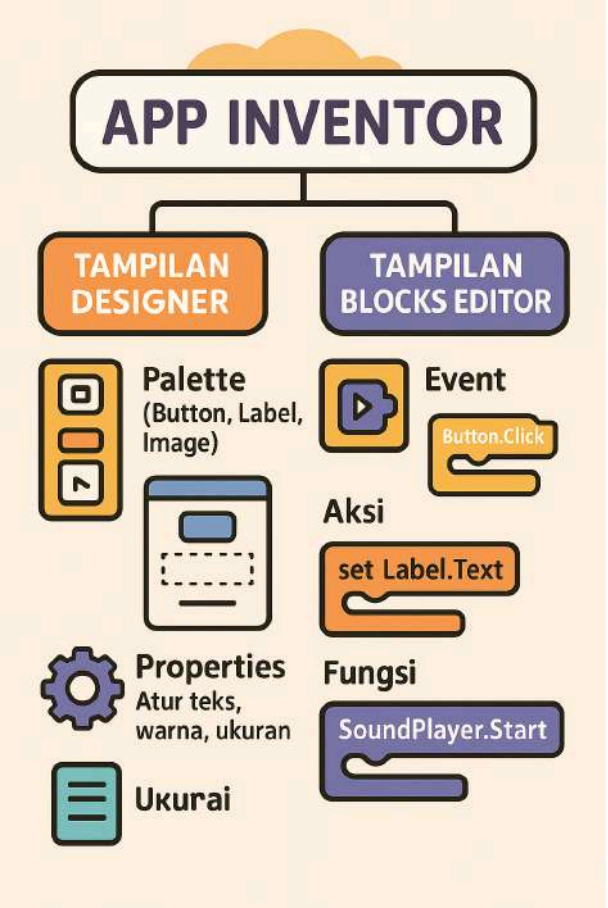
## Fakta Menarik / Fun Facts

- Aplikasi mobile pertama yang populer adalah game sederhana seperti Snake di ponsel Nokia lawas!
- App Inventor dibuat oleh Google sebelum diserahkan ke MIT. Ini adalah salah satu cara termudah untuk mengubah ide aplikasi menjadi kenyataan di ponsel.

## Tips Belajar atau Tips Cepat Menghafal

- **Ingat "App Inventor itu Scratch-nya HP":** Konsep blok-bloknya mirip, tapi hasilnya bisa jalan di ponselmu!
- **Eksplorasi:** Jangan takut mencoba setiap komponen dan blok yang ada. Lihat apa yang bisa mereka lakukan!

### Mind Mapping




### Cerita Seru

Bayangkan kamu membuat aplikasi “Senter Darurat”. Saat kamu tekan tombol, layar jadi putih terang. Seru kan? Ini contoh nyata dari fungsi App Inventor.

## Aktivitas Siswa

### Aktivitas 1: Eksplorasi Situs Web App Inventor

**Tujuan:** Mengenal tampilan dan fitur App Inventor
**Langkah-langkah:**

1. Buka browser dan kunjungi situs: https://ai2.appinventor.mit.edu
2. Login menggunakan akun Google.
3. Klik **“Start new project”,** lalu beri nama (misalnya: BelajarAppInventor)

4. Jelajahi **Designer** (untuk mendesain tampilan aplikasi):
    - Tambahkan tombol, label, gambar, dll.
5. Pindah ke **Blocks Editor** (untuk membuat logika/program):
    - Lihat blok-blok seperti: **when Button.Click do, set Label.Text to,** dll.

**Tujuan kegiatan ini:** Mengenal perbedaan antara tampilan dan logika aplikasi.

## Aktivitas 2: Membuat Aplikasi "Clicker Sederhana"

**Tujuan:** Membuat aplikasi yang bisa menghitung jumlah klik tombol.
**Langkah-langkah:**

1. Buat project baru dengan nama: ClickerSederhana
2. Di **Designer:**
    - Tambahkan **Button** (beri teks: “Klik Saya!”)
    - Tambahkan **Label** (untuk menampilkan angka, ubah teks awal jadi “0”)
3. Di **Blocks:**
    - Tambahkan variabel **skor** = 0
    - Buat logika saat tombol diklik: tambahkan 1 ke skor dan tampilkan hasilnya

**Pseudocode:**

```
skor ← 0
when Button.Click do
  skor ← skor + 1
  Label.Text ← convertToText(skor)
```

## Aktivitas 3: Menambahkan Reset Skor

**Tujuan:** Memberi fitur reset skor agar bisa kembali ke nol.
**Langkah-langkah:**

1. Di **Designer,** tambahkan satu **Button** lagi (ubah teksnya jadi: “Reset”)
2. Di Blocks, buat logika saat tombol reset ditekan:
    - Ubah **skor** jadi 0
    - Perbarui label ke angka 0

**Pseudocode:**

```
when ButtonReset.Click do
  skor ← 0
  Label.Text ← "0"
```

## Aktivitas 4: Menambahkan Efek Suara Saat Klik

**Tujuan:** Tambahkan suara agar aplikasi lebih interaktif
**Langkah-langkah:**

1. Di **Designer:**
    - Tambahkan **Sound** (dari kategori **Media**)
    - Upload **file suara** (misalnya: klik.mp3)
2. Di **Blocks:**
    - Saat tombol diklik, mainkan suara selain menambah skor

**Pseudocode:**

```
when Button.Click do
  skor ← skor + 1
  Label.Text ← convertToText(skor)
  Sound.Play()
```

# Rangkuman

**App Inventor** adalah **platform berbasis web** dari *MIT* yang memungkinkan kamu membuat **aplikasi Android** dengan **mudah**, tanpa perlu menulis kode rumit.

Di App Inventor, kamu akan menggunakan **antarmuka visual** yang mirip dengan *Scratch*, yaitu sistem *drag & drop* untuk menyusun blok-blok kode.

Ada **dua tampilan utama** dalam App Inventor:

- **Designer View**: untuk **merancang tampilan** aplikasi, seperti menambahkan tombol, gambar, dan warna latar.
- **Blocks Editor**: untuk **menyusun logika program** menggunakan blok-blok yang saling disambungkan.

Dengan App Inventor, kamu bisa membuat aplikasi sederhana seperti:

- **Tombol sapa**
- **Suara hewan**
- **Kalkulator uang jajan**

Bahkan ide kecil pun bisa kamu ubah jadi **aplikasi nyata di HP-mu**!

# Latihan Soal

## A. Benar atau Salah (5 Soal)

1. App Inventor memungkinkan membuat aplikasi Android tanpa menulis kode.
   **Jawaban:** Benar

2. Aplikasi mobile hanya bisa dijalankan jika ada koneksi internet.
   **Jawaban:** Salah

3. Viewer adalah tempat untuk menyusun blok-blok logika program.
   **Jawaban:** Salah

4. Palette berisi komponen seperti Button, Label, dan TextBox.
   **Jawaban:** Benar

5. Event `when Button1.Click do` digunakan saat aplikasi pertama kali dibuka.
   **Jawaban:** Salah

## B. Pilihan Ganda (10 Soal)

6. Apa nama platform visual untuk membuat aplikasi Android tanpa coding kompleks?
   A. Python Studio
   B. App Inventor
   C. App Builder Pro
   D. CodeLab
   **Jawaban:** B

7. Di tampilan Designer View, kita bisa melakukan hal berikut, kecuali:
   A. Menyusun tampilan aplikasi
   B. Menambahkan Button dan Label
   C. Mengatur ukuran teks
   D. Menulis kode HTML
   **Jawaban:** D

8. Komponen mana yang digunakan untuk menerima input dari pengguna?
   A. Label
   B. Button
   C. TextBox
   D. Image
   **Jawaban:** C

9. Di mana kamu menyusun blok seperti `set Label1.Text to`?
   A. Properties
   B. Viewer
   C. Blocks Editor
   D. Palette
   **Jawaban:** C

10. Jika kamu ingin suara diputar saat gambar di klik, kamu harus menggunakan:
    A. Label dan Timer
    B. Button dan SoundRecorder
    C. Image dan SoundPlayer
    D. Label dan Canvas
    **Jawaban:** C

11. Untuk mengatur teks pada tombol, kamu harus membuka bagian:
    A. Blocks
    B. Viewer
    C. Components
    D. Properties
    **Jawaban:** D

12. Blok `when Button1.Click do` digunakan untuk:
    A. Menampilkan layar baru
    B. Menangkap aksi pengguna pada tombol
    C. Menampilkan teks secara otomatis
    D. Menjalankan aplikasi secara offline
    **Jawaban:** B

13. App Inventor dikembangkan pertama kali oleh:
    A. Microsoft
    B. MIT
    C. Google
    D. Facebook
    **Jawaban:** C

14. Di mana kamu bisa melihat semua komponen yang sudah digunakan di aplikasi?
    A. Viewer
    B. Palette
    C. Components
    D. Screen
    **Jawaban:** C

15. Situs resmi untuk mengakses App Inventor adalah:
    A. www.ai.appinventor.com
    B. www.mit.edu/app
    C. ai2.appinventor.mit.edu
    D. appinventor.google.com
    **Jawaban:** C

## C. Isian Singkat (5 Soal)

16. Komponen untuk menampilkan teks di aplikasi adalah _____.
    **Jawaban:** Label

17. Untuk memicu aksi ketika tombol ditekan, digunakan event _____.
    **Jawaban:** Button.Click

18. Komponen untuk memutar suara disebut _____.
    **Jawaban:** SoundPlayer

19. Tampilan tempat menyusun logika program menggunakan blok disebut _____.

**Jawaban:** Blocks Editor

20. App Inventor memiliki antarmuka visual yang mirip dengan _____.
**Jawaban:** Scratch

## D. Soal Eksplorasi

**Soal: Desain Aplikasi Pengingat Minum Air**

Buatlah rancangan aplikasi sederhana untuk mengingatkan pengguna agar minum air setiap 2 jam. Di Designer View, kamu bisa menggunakan komponen Label (untuk teks "Minum Air Sekarang!"), Button (untuk menunda alarm), dan Notifier (untuk menampilkan pengingat). Jelaskan bagaimana kamu mengatur event dan blok di Blocks Editor agar aplikasi bisa memberikan notifikasi otomatis atau setelah tombol ditekan. Tunjukkan pula komponen apa yang berperan dalam mengatur waktu atau alarm.

**Soal: Simulasi Aplikasi "Catatan Harian"**

Bayangkan kamu ingin membuat aplikasi sederhana untuk mencatat perasaan harian. Gunakan TextBox untuk menulis catatan, Button untuk menyimpan, dan Label untuk menampilkan catatan terakhir. Jelaskan alur aplikasi mulai dari pengguna membuka aplikasi, mengetik di TextBox, menekan tombol simpan, hingga isi catatan muncul di Label. Bagaimana kamu menggunakan blok set Label.Text to dan dari mana kamu mengambil nilai teks?

**Soal: Aplikasi “Tombol Warna Ajaib”**

Kamu ingin membuat aplikasi seru dengan satu tombol yang ketika ditekan, mengubah warna latar belakang layar secara acak. Di Designer, kamu membutuhkan satu Button dan mengatur properti Screen. Di Blocks Editor, gunakan blok set Screen.BackgroundColor to dan kombinasikan dengan blok warna yang berubah-ubah. Jelaskan logika sederhana agar setiap klik menghasilkan warna baru dan aplikasimu tampak interaktif dan menarik.

**Soal:**

Bayangkan kamu ingin membuat aplikasi sederhana bernama “Kalkulator Uang Jajan” yang membantu menghitung sisa uang setelah membeli barang. Jelaskan rancangan sederhana antarmuka aplikasi tersebut di Designer View, sebutkan komponen apa saja yang kamu butuhkan, dan apa fungsinya masing-masing.

**Soal:**

Kamu diminta membuat aplikasi edukasi suara hewan. Gambar hewan akan muncul, dan saat diklik akan berbunyi. Jelaskan bagaimana kamu menyusun logika blok di Blocks Editor agar saat gambar di klik, suara hewan tertentu diputar. Sebutkan event dan blok yang kamu gunakan.

**Soal:**

Buatlah skenario aplikasi “Papan Skor” untuk mencatat poin dalam permainan. Setiap kali tombol ditekan, skor bertambah 1. Jelaskan bagaimana kamu menyusun

aplikasi ini dari awal: mulai dari komponen yang dipakai, bagaimana tampilannya, dan logika blok yang harus disusun untuk menambah skor setiap kali tombol ditekan.

# Tugas Proyek

## Proyek 1: Desain Aplikasi Sehari-Hari

**Tujuan:** Melatih kreativitas dan pemahaman logika aplikasi.

**Deskripsi:**
Siswa diminta untuk merancang tampilan dan fungsi aplikasi sederhana di atas kertas, seperti:

- Aplikasi **Senter** (menyalakan layar putih atau flashlight)
- Aplikasi **Kalkulator Sederhana** (penjumlahan dan pengurangan),
- Aplikasi **Daftar Belanja** (menambahkan dan menyimpan daftar).

**Langkah-langkah:**

1. Gambar desain antarmuka (UI) di kertas.
2. Tentukan komponen yang dibutuhkan: Label, Button, TextBox, dll.
3. Diskusikan logika blok: contoh, saat tombol ditekan, layar berubah warna (untuk senter), atau saat dua angka di input, hasilnya muncul.
4. Implementasi bisa dilakukan setelah desain disetujui.

## Proyek 2: Aplikasi Suara Hewan

**Tujuan:** Menggabungkan gambar dan suara menjadi interaktif.

**Deskripsi:**
Buat aplikasi edukatif dengan tiga gambar hewan yang bisa mengeluarkan suara saat diklik.

**Langkah-langkah:**

1. Tambahkan tiga komponen **Image** (misalnya gambar kucing, anjing, ayam).
2. Tambahkan tiga **Sound** dan unggah file suara masing-masing hewan.
3. Gunakan blok:
    - *When Image1.Click → call Sound1.Play*
    - Lakukan hal yang sama untuk hewan lainnya.
4. Bonus: Tambahkan teks nama hewan menggunakan Label.

## Proyek 3: Aplikasi Pengingat Minum Air

**Tujuan:** Membuat pengingat otomatis dan manual.

**Deskripsi:**
Aplikasi yang mengingatkan pengguna untuk minum setiap 2 jam atau saat tombol ditekan.

**Langkah-langkah:**

1. Gunakan komponen **Label**, **Button**, dan **Notifier**.
2. Tambahkan blok:
    - *When Button.Click → call Notifier.ShowMessageDialog ("Minum air sekarang!")*
3. Gunakan **Clock** component:
    - Set TimerInterval = 7200000 (2 jam dalam milidetik).
    - *When Clock.Timer → call Notifier.ShowMessageDialog ("Sudah waktunya minum air!")*
4. Tambahkan Label yang menunjukkan waktu pengingat terakhir.

## Proyek 4: Kalkulator Uang Jajan

**Tujuan:** Melatih logika matematika dan input data.

**Deskripsi:**
Aplikasi ini menghitung sisa uang jajan dari jumlah uang yang dimiliki dan pengeluaran.

**Langkah-langkah:**

1. Tambahkan 2 **TextBox** (untuk uang jajan dan pengeluaran).
2. Tambahkan 1 **Button** ("Hitung Sisa").
3. Tambahkan 1 **Label** (untuk hasil).
4. Blok yang digunakan:
    - Ambil nilai dari kedua TextBox, ubah ke angka, lalu kurangi.
    - Tampilkan hasil di Label.

## Proyek 5: Aplikasi Papan Skor

**Tujuan:** Mengelola data angka secara dinamis.

**Deskripsi:**
Buat aplikasi penghitung skor yang bisa ditambah dengan tombol.

**Langkah-langkah:**

1. Tambahkan 1 **Label** (untuk skor, awalnya 0).
2. Tambahkan 1 atau lebih **Button** (misal: "Tambah Skor").
3. Buat variabel global *Skor = 0*.
4. Blok:
    - *When Button.Click → set Skor = Skor + 1 → set Label.Text = Skor*

## Proyek 6: Aplikasi Quiz Sederhana

**Tujuan:** Memahami penggunaan kondisi dan logika IF.

**Deskripsi:**
Aplikasi quiz 1 pertanyaan dengan pilihan jawaban A, B, dan C.

**Langkah-langkah:**

1. Gunakan **Label** untuk pertanyaan, dan **3 Button** untuk jawaban.
2. Gunakan blok IF:
    - *When ButtonA.Click → if (jawaban = benar) then tampilkan "Benar!" else "Salah"*
3. Tambahkan **Notifier** atau Label sebagai feedback.

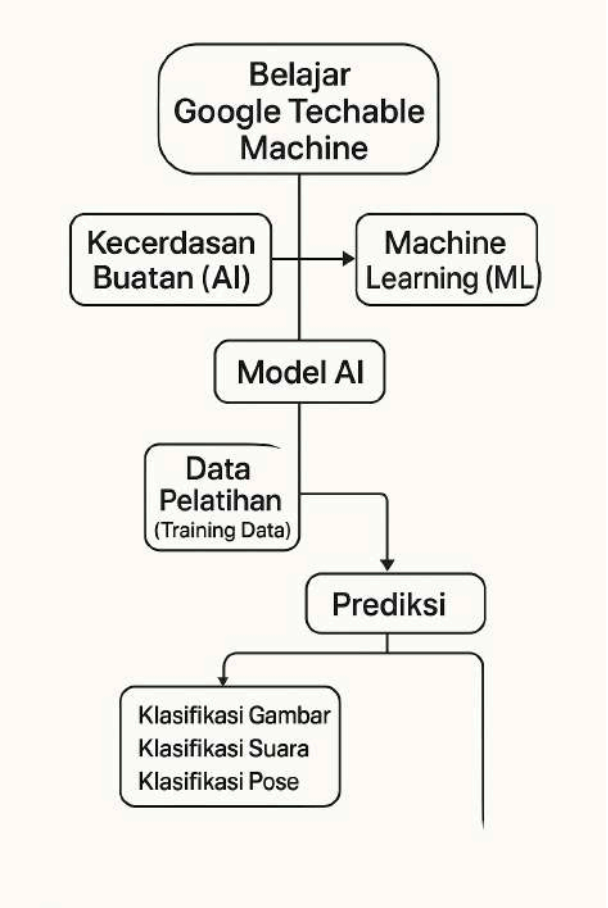
Proyek-proyek ini dapat dikembangkan lebih lanjut oleh siswa dengan menambahkan animasi, suara, atau logika yang lebih kompleks. Cocok untuk pembelajaran kreatif dan eksploratif di tingkat SMP. Jika kamu ingin, saya bisa bantu buatkan blok logikanya langsung juga!

# BAB 7: Belajar AI dengan Google Teachable Machine

## Tujuan Pembelajaran

Peserta didik mampu memahami konsep dasar Kecerdasan Buatan (AI) dan Machine Learning, serta dapat melatih model AI sederhana untuk klasifikasi gambar, suara, atau pose menggunakan Google Teachable Machine tanpa perlu menulis kode.

## Peta Konsep




## Apersepsi

Kalian pernah melihat HP kalian bisa mengenali wajah saat membuka kunci? Atau ketika kalian berbicara ke Google Assistant atau Siri, mereka bisa mengerti apa yang kalian katakan? Bagaimana bisa ya? Itu semua adalah contoh dari "kecerdasan buatan" atau Artificial Intelligence (AI)! Dan yang paling keren, kita bisa membuat AI sederhana kita sendiri tanpa coding, lho!

# Penjelasan Konsep (Teori)

## Apa Itu Kecerdasan Buatan (Artificial Intelligence / AI)?

**Kecerdasan Buatan**, atau yang sering disebut **AI (Artificial Intelligence)**, adalah sebuah teknologi yang memungkinkan **mesin atau komputer** untuk meniru kemampuan otak manusia. Artinya, mesin bisa **belajar**, **berpikir**, **mengenali pola**, **memecahkan masalah**, dan bahkan **memahami bahasa**, hampir seperti manusia!

Bayangkan komputer yang bisa "berpikir" seperti kamu—bukan hanya menjalankan perintah, tapi juga **beradaptasi dan mengambil keputusan sendiri** berdasarkan data atau pengalaman sebelumnya.

### Contoh AI dalam Kehidupan Sehari-hari:

- **Asisten suara** seperti *Siri*, *Google Assistant*, atau *Alexa* yang bisa menjawab pertanyaanmu.
- **Rekomendasi film atau musik** di YouTube, Netflix, atau Spotify yang pas banget dengan seleramu.
- **Filter spam email** yang otomatis memisahkan email penting dan email sampah.
- **Fitur pendeteksi wajah** di kamera ponsel untuk membuka kunci atau menambahkan efek.

## Apa Itu Machine Learning (Pembelajaran Mesin)?

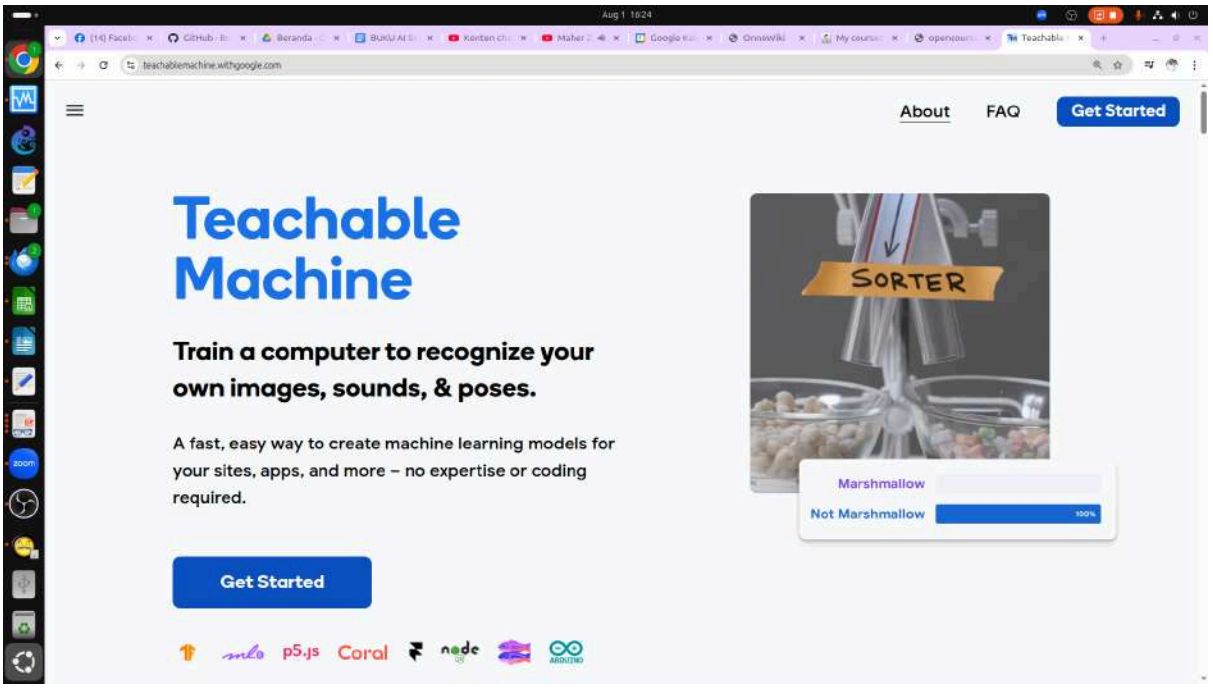
**Machine Learning (ML)** adalah salah satu cabang dari AI. Kalau AI adalah “otak buatan”, maka Machine Learning adalah cara **melatih otak buatan itu supaya makin pintar**.

Bayangkan kamu belajar mengenali jenis buah. Pertama kamu melihat banyak gambar apel, pisang, dan jeruk. Lama-lama kamu tahu mana yang mana, kan? Nah, **komputer juga bisa** belajar seperti itu, tetapi dari data, bukan dari buku atau guru seperti kita.

### Konsep Kunci dalam Machine Learning:

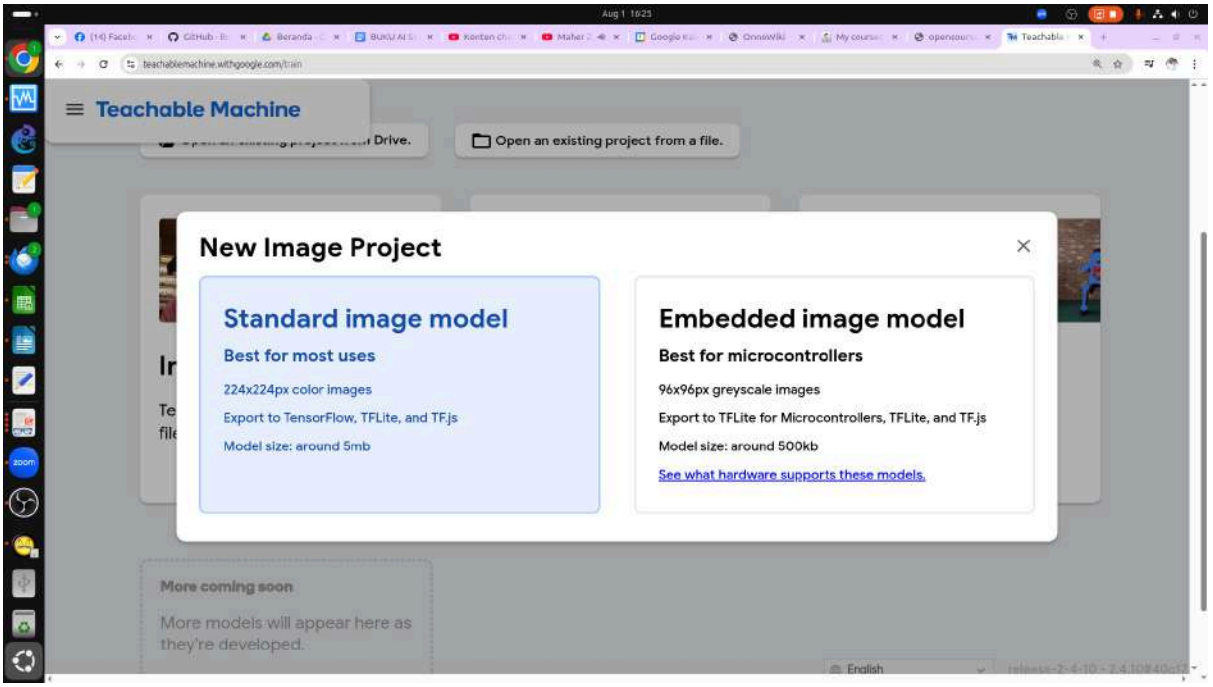
- **Data Pelatihan (Training Data)**
  Ini adalah kumpulan informasi (bisa berupa gambar, suara, atau teks) yang diberikan kepada mesin. Semakin banyak dan relevan datanya, semakin baik mesin belajar.
- **Model AI**
  Ini adalah hasil akhir dari proses belajar. Model ini seperti “otak kecil” yang dibuat komputer setelah belajar dari data. Model ini akan digunakan untuk mengenali hal-hal baru.
- **Prediksi**
  Ketika ada data baru yang belum pernah dilihat sebelumnya, model AI bisa menebak atau mengklasifikasikan data tersebut. Contohnya, jika kamu tunjukkan gambar anjing yang belum pernah dilihat oleh AI, model akan mencoba mengenali: "Ini anjing atau bukan, ya?"

## Google Teachable Machine: Membuat AI Jadi Mudah!




Ingin mencoba membuat model Machine Learning sendiri tanpa perlu menulis kode? **Google Teachable Machine** adalah jawabannya!

Ini adalah **alat berbasis web** yang sangat ramah untuk pemula. Bahkan siswa SMP pun bisa mencobanya! Di sini, kamu bisa membuat model AI hanya dengan **mengunggah gambar, merekam suara, atau menunjukkan gerakan tubuhmu**.




### Jenis Proyek yang Bisa Kamu Buat di Teachable Machine:

1. **Image Project (Proyek Gambar)**
   Melatih AI untuk mengenali gambar. Misalnya:
   - Membedakan antara kucing dan anjing.
   - Mengenali ekspresi wajah: senang, sedih, marah.

2. **Audio Project (Proyek Suara)**
   Melatih AI untuk mengenali jenis suara. Misalnya:
   - Tepuk tangan vs. siulan.
   - Suara “ya” vs. suara “tidak”.
   - Suara hewan seperti anjing atau kucing.

3. **Pose Project (Proyek Pose Tubuh)**
   Melatih AI untuk mengenali gerakan atau posisi tubuh. Misalnya:
   - Berdiri vs. jongkok.
   - Lambaian tangan vs. diam.
   - Gerakan olahraga atau tarian.

### Kesimpulan

Dengan memahami AI dan Machine Learning, kamu sudah selangkah lebih dekat menjadi **penjelajah dunia teknologi masa depan**. Siapa bilang bikin AI harus rumit? Dengan alat seperti Teachable Machine, kamu bisa mulai **eksperimen seru dan menantang** bahkan dari rumah atau sekolah.

**Siap jadi penemu AI masa depan? Yuk, kita mulai petualangan cerdas ini!**

# Contoh Kasus dan Ilustrasi

## Kasus 1: Detektor Suara di Kelas

Bayangkan kamu membuat alat yang bisa mengenali suara "tepuk tangan" dan "diam". Setiap kali ada yang tepuk tangan, AI akan menampilkan teks "Tepuk Tangan Terdeteksi". Saat tidak ada suara, akan muncul "Hening". Ini bisa digunakan saat presentasi!

**Mengapa AI membantu?**

- AI dapat **mengenali pola suara**, seperti "tepuk tangan" dan "hening", yang sulit dibedakan oleh sistem biasa.
- Deteksi otomatis membuat interaksi **lebih interaktif dan real-time**, tanpa harus menekan tombol atau input manual.

**Mengapa Google Teachable Machine cocok?**

- Teachable Machine memiliki **mode "Audio"** yang memungkinkan kamu **melatih model hanya dengan merekam suara**.
- Sangat mudah: cukup rekam contoh “tepuk tangan” dan “hening” → model bisa langsung dijalankan.
- **Tanpa perlu coding**, cocok untuk pemula dan pelajar.

## Kasus 2: Asisten Wajah Ceria

Kamu membuat AI yang mengenali ekspresi wajah temanmu: senang, sedih, dan marah. Saat kamu tersenyum, model akan bilang "Kamu bahagia hari ini!". Seru, kan?

**Mengapa AI membantu?**

- AI bisa **mengenali ekspresi wajah** seperti senang, sedih, dan marah dengan membaca pola pada wajah.
- Ini membuat komunikasi lebih emosional dan interaktif, apalagi untuk **alat bantu sosial atau pendidikan**.

**Mengapa Google Teachable Machine cocok?**

- Mode **"Image" (Gambar)** bisa digunakan untuk melatih AI dari foto wajah dengan ekspresi berbeda.
- Cukup ambil beberapa gambar ekspresi senyum, sedih, marah → model bisa mengenali ekspresi baru secara langsung.

## Kasus 3: Penjaga Pintu Kelas

AI mengenali siapa yang berdiri di depan webcam. Jika itu kamu atau teman sekelas, pintu otomatis terbuka (bisa diprogram nanti!). AI belajar dari banyak foto wajah kalian.

**Mengapa AI membantu?**

- AI dapat **mengenali wajah orang tertentu**, seperti kamu dan temanmu, sehingga hanya orang yang dikenali bisa membuka pintu.
- Ini berguna untuk **keamanan otomatis tanpa kunci fisik**.

**Mengapa Google Teachable Machine cocok?**

- Mode gambar bisa melatih model untuk **mengenali wajah individu** dengan cepat.
- Cukup ambil beberapa foto setiap orang → AI belajar perbedaan antar wajah.
- Sangat sederhana dibandingkan sistem pengenalan wajah profesional.

## Kasus 4: Pelatih Gerakan Olahraga

Gunakan **Pose Project** untuk membedakan gerakan "jongkok", "berdiri", dan "melompat". Cocok untuk senam atau lomba kelas digital.

**Mengapa AI membantu?**

- AI bisa **melacak posisi tubuh** dan mengenali gerakan seperti jongkok, berdiri, dan melompat.
- Membantu pelatih atau guru untuk **memberi umpan balik otomatis** pada murid saat olahraga atau lomba.

**Mengapa Google Teachable Machine cocok?**

- Mode **"Pose"** di Teachable Machine menggunakan kamera untuk membaca **titik-titik pose tubuh secara otomatis**.
- Cukup rekam saat kamu jongkok, berdiri, dan melompat → model bisa mengenalinya.
- Tidak butuh sensor tambahan, hanya kamera biasa.

### Kesimpulan Umum:

**Google Teachable Machine** sangat cocok karena:

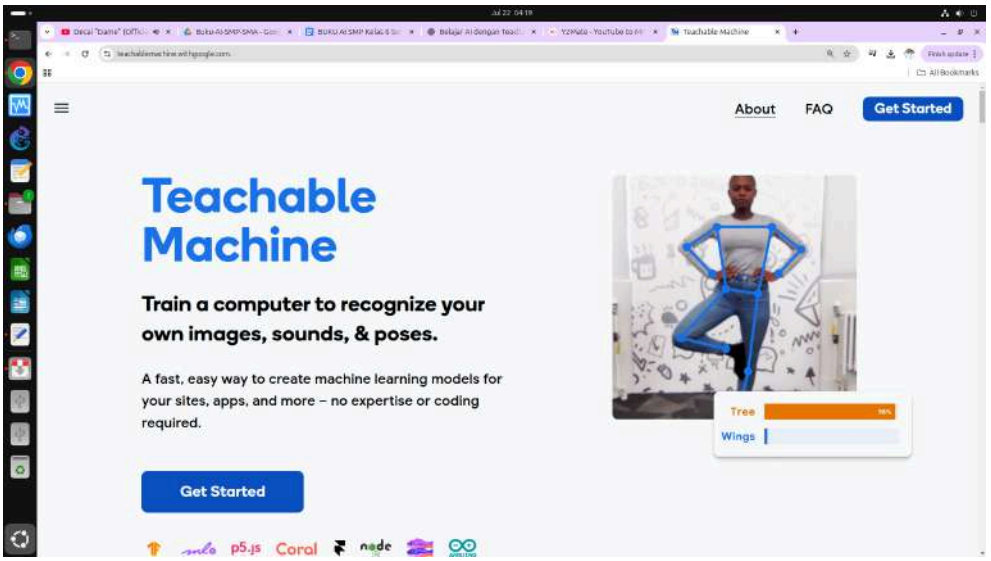
- **Gratis, berbasis web, tanpa coding**.
- Cukup gunakan **kamera atau mikrofon**, dan berikan contoh langsung.
- Proses **latihan model cepat dan visual**.
- Ideal untuk **pelajar, guru, dan pemula** yang ingin memahami AI tanpa perlu latar belakang teknis.

Inilah mengapa tool ini sering jadi pintu masuk untuk mengenal **machine learning dengan cara menyenangkan dan praktis**!

## Ilustrasi Google Teachable Machine

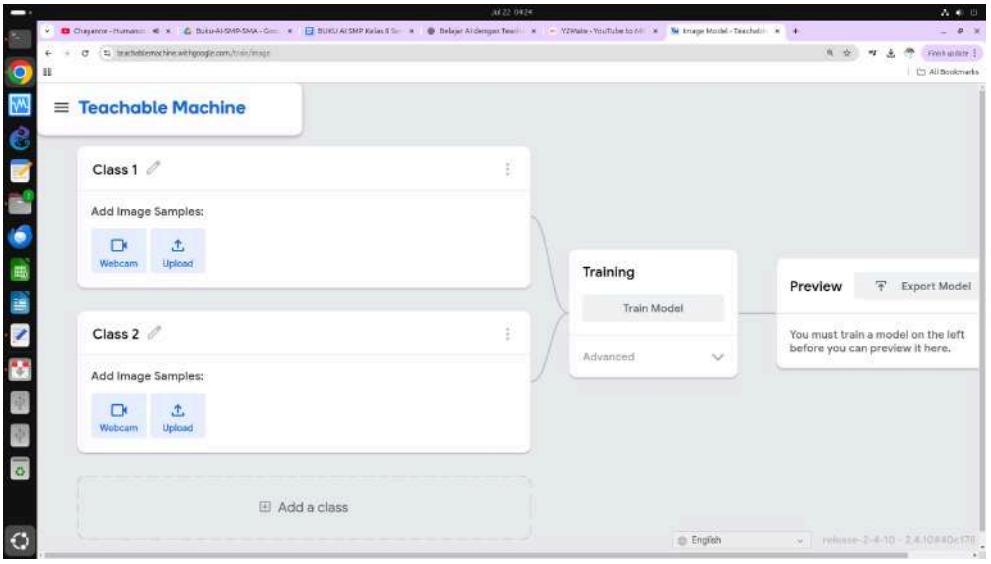
### Membuka Teachable Machine:

- Buka browser dan ketik [teachablemachine.withgoogle.com](teachablemachine.withgoogle.com).



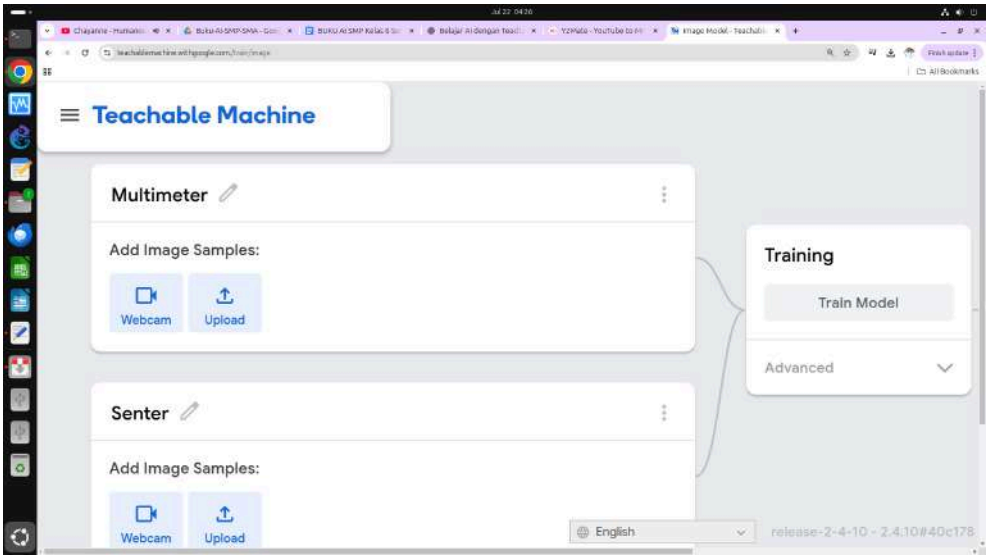
## Memilih Jenis Proyek:

- Klik "Get Started" lalu pilih "Image Project". Pilih "Standard image model".



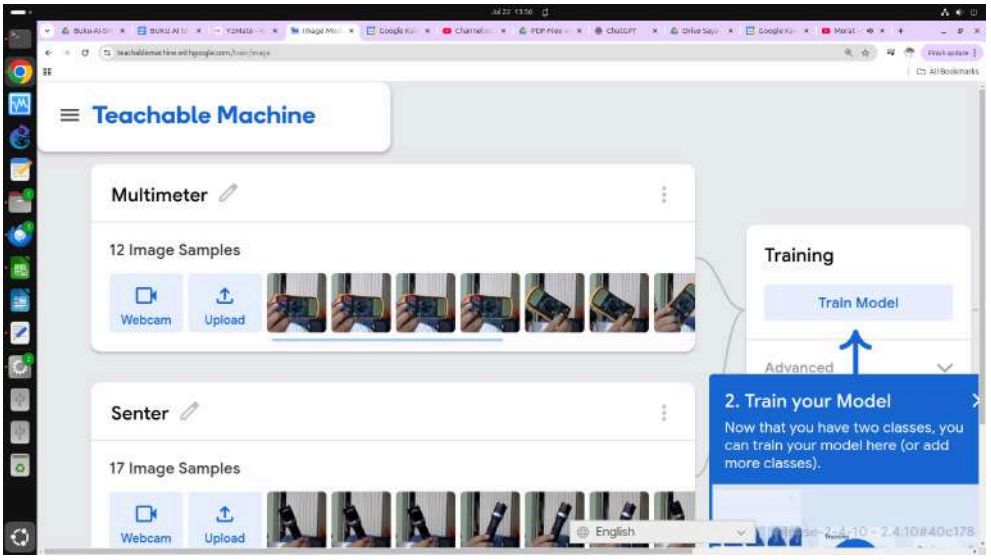

## Menyiapkan Kelas (Classes):

- Di tampilan proyek, akan ada dua kelas default (Class 1, Class 2). Ubah namanya menjadi "Multimeter" dan "Senter".



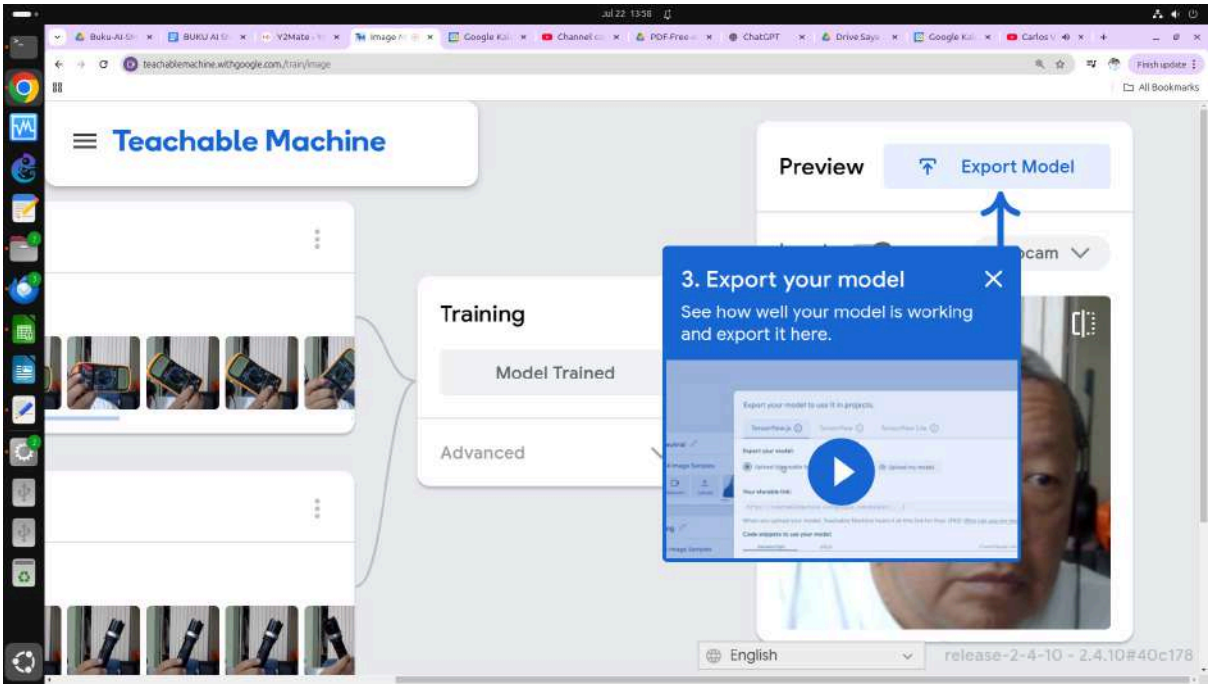

## Mengumpulkan Data Pelatihan (Training Data):

- Untuk kelas "Multimeter", klik "Webcam" dan ambil beberapa foto multimeter dari berbagai sudut atau pose. Lakukan hal yang sama untuk kelas "Senter" dengan foto-foto senter. Semakin banyak variasi, semakin baik!




## Melatih Model (Train Model):

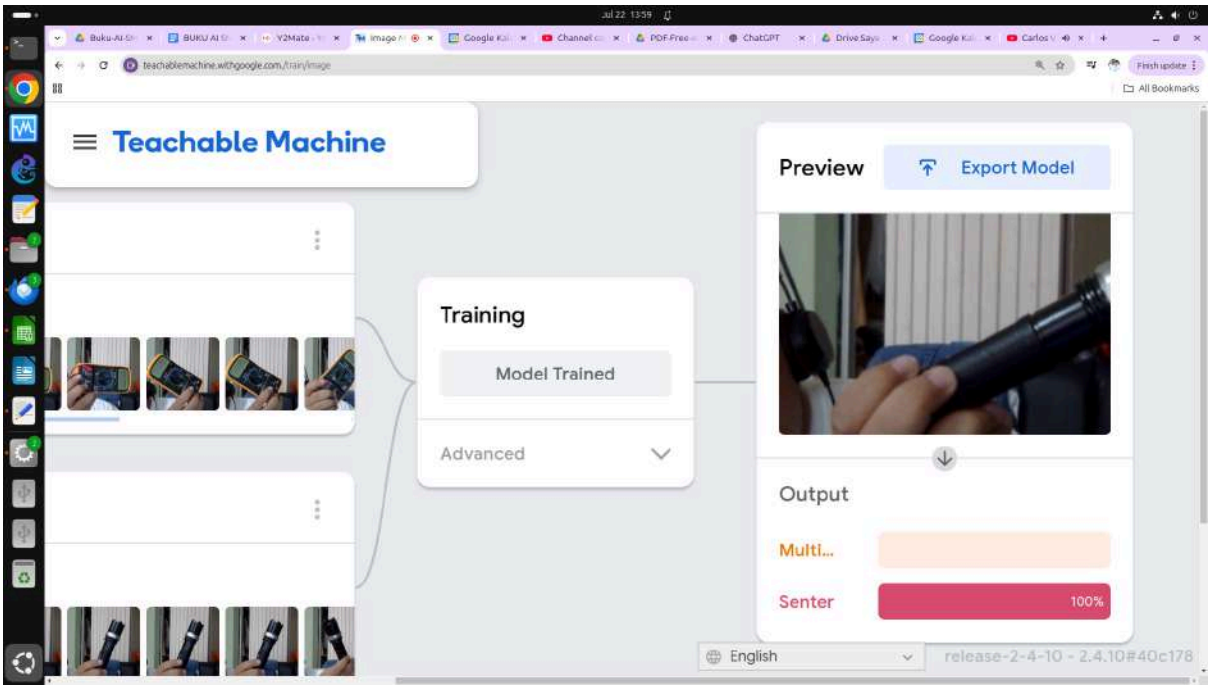
- Setelah data cukup, klik tombol "Train Model". Tunggu proses pelatihan selesai, sampai “Model Trained”. Ini seperti AI sedang belajar dari gambar-gambar yang kalian berikan.




## Menguji Model (Preview):

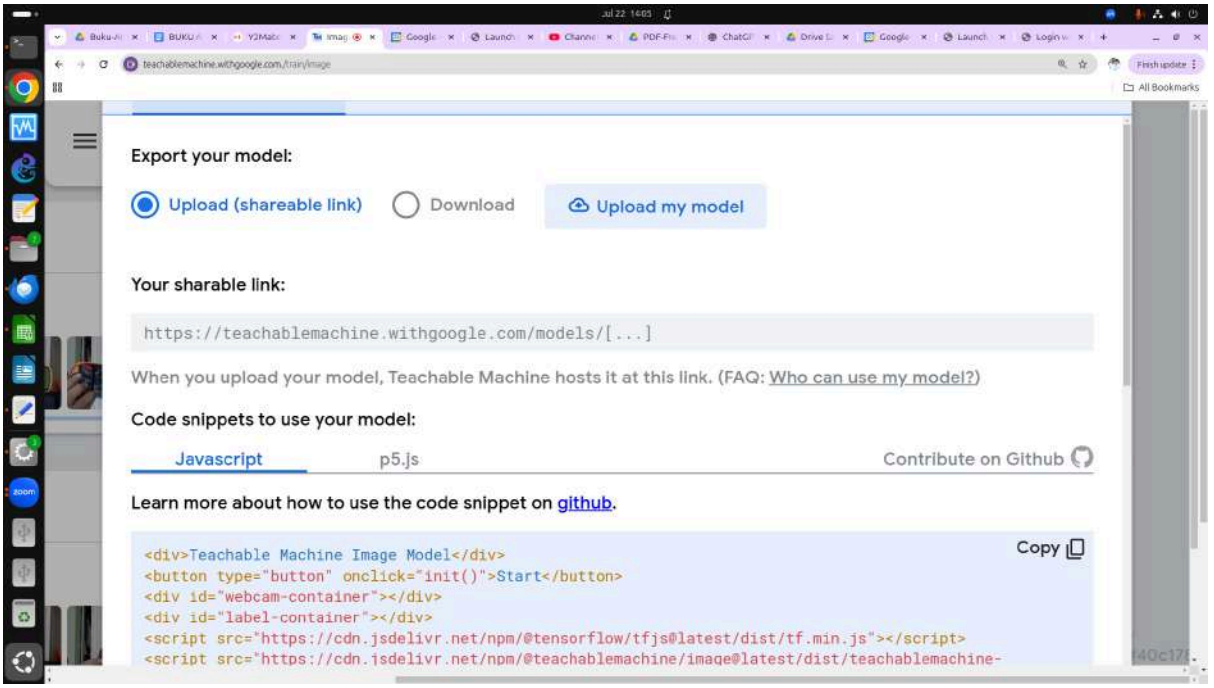
- Setelah pelatihan, kalian bisa menguji model di area "Preview". Gunakan webcam dan tunjukkan gambar multimeter atau senter. Lihat apakah model berhasil

memprediksi dengan benar!




## .Mengekspor Model (Export Model):

- Kalian bisa mengekspor model untuk digunakan di proyek lain (misal: di App Inventor atau Scratch, meskipun integrasinya lebih lanjut).




.

# Contoh Soal dan Pembahasan

**Soal:**

Kamu ingin membuat model Teachable Machine yang bisa membedakan "suara tepuk tangan" dan "suara batuk". Data seperti apa yang harus kamu kumpulkan untuk setiap kelas?

**Pembahasan:**

- Untuk kelas "Tepuk Tangan": Rekam berbagai jenis tepuk tangan (keras, pelan, cepat, lambat) dari berbagai orang.
- Untuk kelas "Batuk": Rekam berbagai jenis batuk (kering, berdahak) dari berbagai orang. Penting untuk memiliki variasi agar model belajar dengan baik.

**Soal:**

Setelah melatih model, Teachable Machine salah mengidentifikasi wajah temanmu sebagai "kucing". Apa yang mungkin menjadi penyebabnya dan bagaimana cara memperbaikinya?

**Pembahasan:**

**Penyebab:**

- Data pelatihan kurang (terlalu sedikit foto, kurang variasi).
- Ada "noise" atau gambar yang salah di kelas "kucing" (misal: ada wajah temanmu di foto kucing).
- Modelnya "overfit" (terlalu spesifik belajar hanya pada data yang ada).

**Perbaikan:**

- Tambahkan lebih banyak data pelatihan untuk kedua kelas ("kucing" dan "bukan kucing" atau "manusia").
- Pastikan tidak ada gambar yang salah masuk ke kelas yang tidak seharusnya.
- Jika masalahnya adalah Teachable Machine mengidentifikasi segala sesuatu sebagai kucing, tambahkan kelas ketiga seperti "Lain-lain" dan berikan banyak foto objek selain kucing dan anjing.

**Soal:**

Kamu ingin AI membedakan suara “siulan” dan “batuk”. Apa yang harus dilakukan?

**Pembahasan:**

- Rekam siulan dari beberapa orang (tinggi, rendah, pelan, keras).
- Rekam suara batuk dari berbagai orang dan jenis batuk.
- Latih AI menggunakan Teachable Machine dengan data itu.

**Soal:**

Model kamu salah mengenali suara "tepuk tangan" sebagai "siulan". Mengapa?

**Pembahasan:**

- Mungkin datanya kurang banyak.
- Mungkin suara terlalu mirip (suaranya samar).
- Solusi: tambahkan variasi suara dan periksa apakah rekaman sesuai.

**Soal :**

Apa perbedaan data pelatihan dan prediksi?

**Pembahasan:**

- Data pelatihan: Suara, gambar, atau pose yang kita masukkan agar AI belajar.
- Prediksi: Hasil saat AI menebak input baru (yang belum pernah dilihat).

## Fakta Menarik / Fun Facts

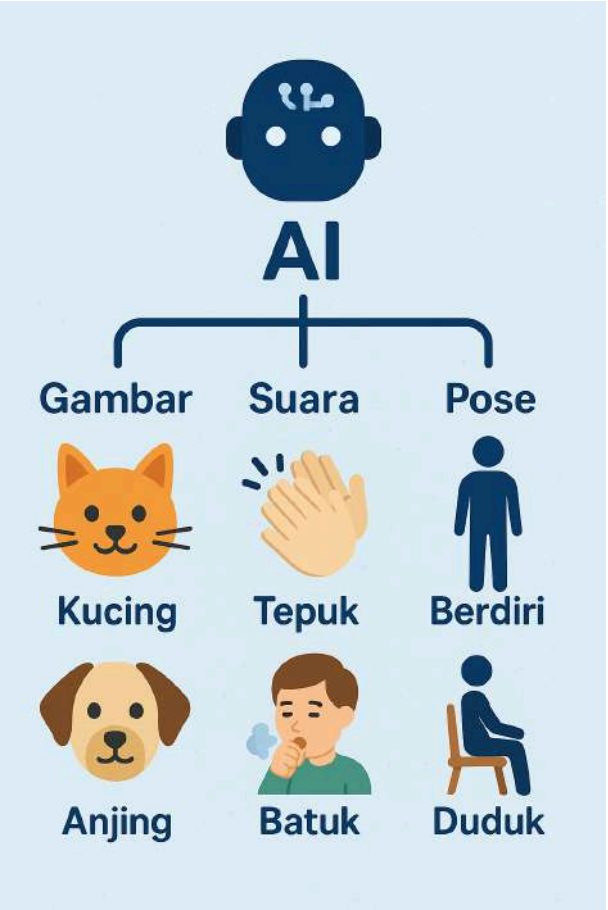
- **AI Sekarang Super Keren!**
  AI tidak cuma bisa mengenali gambar atau suara, tapi sekarang juga bisa **bikin gambar dan musik sendiri** yang hasilnya hampir seperti buatan manusia! Jadi, AI bisa jadi **"seniman digital**" juga, lho!

- **Belajar Sendiri Seperti Manusia!**
  Teknologi yang dipakai AI namanya **Machine Learning.** Ini yang bikin **mobil bisa jalan sendiri** tanpa sopir, dan juga yang bikin **rekomendasi film atau lagu** di YouTube dan Spotify pas banget sama selera kamu. Keren, kan?

## Tips Belajar atau Tips Cepat Menghafal

- **Ingat "AI belajar dari data":** Semakin banyak dan bervariasi data yang kalian berikan, semakin pintar AI-nya.
- **Ingat:** “Semakin banyak data, semakin pintar AI!”
- **Praktik Langsung:** Coba langsung bikin proyek kecil. Belajar langsung = lebih mudah mengerti.
- ngsung bikin proyek kecil. Belajar langsung = lebih mudah mengerti.

### Mind Map Sederhana:




### Cerita Mudah Diingat:

Bayangkan AI seperti murid baru. Kalau kita beri banyak contoh (gambar, suara, gerakan), dia jadi pintar. Tapi kalau cuma satu atau dua contoh, dia bingung! Jadi AI juga butuh belajar seperti kita.

## Aktivitas Siswa

### Aktivitas 1: Membuat Model Gambar di Teachable Machine

**Tujuan:**

Mengenali 3–4 ekspresi wajah (senang, sedih, marah, netral) **atau** benda-benda di kelas (buku, pensil, tas, penggaris).

**Langkah-langkah:**

1. Buka situs **Teachable Machine** di https://teachablemachine.withgoogle.com/
2. Pilih **Image Project** → Klik **Standard Image Model**
3. Buat **3–4 kelas (kategori)**, misalnya: Senang, Sedih, Marah, Netral

4. Gunakan kamera untuk merekam gambar ekspresi wajah atau objek di sekitar
5. Latih model dengan klik **Train Model**
6. Coba model kamu dengan ekspresi wajah/objek asli

**Pseudocode:**

```
buka_teachable_machine()
pilih_image_project()
buat_kelas(["Senang", "Sedih", "Marah", "Netral"])
rekam_gambar_untuk_setiap_kelas()
latih_model()
uji_model_dengan_input_gambar()
```

## Aktivitas 2: Membuat Model Suara di Teachable Machine

**Tujuan:**

Membedakan 2–3 suara seperti siulan, tepuk tangan, dan jentikan jari.

**Langkah-langkah:**

1. Masuk lagi ke Teachable Machine
2. Pilih **Audio Project**
3. Buat **2–3 kelas**, misalnya: Siulan, Tepuk, Jentikan
4. Rekam suara untuk tiap kelas dengan mikrofon
5. Klik **Train Model** untuk melatih
6. Uji model dengan mengeluarkan suara asli

**Pseudocode:**

```
buka_teachable_machine()
pilih_audio_project()
buat_kelas(["Siulan", "Tepuk", "Jentikan"])
rekam_suara_untuk_setiap_kelas()
latih_model()
uji_model_dengan_input_suara()
```

## Aktivitas 3: Uji Coba dan Perbaikan Model

**Tujuan:**

Memastikan model bekerja dengan baik dan memperbaiki jika perlu.

**Langkah-langkah:**

1. Ajak teman mencoba modelmu
2. Perhatikan apakah hasilnya sesuai
3. Jika tidak akurat, tambahkan data latihan lagi
4. Latih ulang model

**Pseudocode:**

```
uji_model_dengan_teman()
jika_model_tidak_akurat:
    tambahkan_data_latihan()
    latih_ulang_model()
```

### Aktivitas 4: Presentasi Model

**Tujuan:**

Menjelaskan dan mendemonstrasikan model buatan sendiri di depan kelas.

**Langkah-langkah:**

1. Siapkan presentasi singkat (judul model, fungsinya, cara kerja)
2. Tampilkan modelmu di layar
3. Tunjukkan cara menggunakan model
4. Jelaskan proses pembuatannya

**Pseudocode:**

```
siapkan_presentasi()
tampilkan_model()
demonstrasikan_penggunaan()
jelaskan_proses_pembuatan()
```

## Rangkuman

Pernah nggak kalian bertanya-tanya, kok HP bisa mengenali wajah kita? Atau kenapa YouTube tahu banget video apa yang kita suka? Nah, semua itu adalah hasil kerja dari **Kecerdasan Buatan** atau *Artificial Intelligence (AI)*! Teknologi ini membuat komputer bisa "berpikir" dan belajar seperti manusia. Serunya lagi, kita bisa mulai belajar bikin *AI* sendiri tanpa ribet pakai kode, lho! Di bab ini, kalian akan diajak mengenal dasar-dasar *AI* dan *Machine Learning*, serta mencoba serunya membuat proyek cerdas lewat **Google Teachable Machine**.

*Machine Learning* itu bisa dibilang "cara belajar" bagi si *AI*. Layaknya kita belajar dari buku atau pengalaman, komputer belajar dari **data**, misalnya gambar, suara, atau gerakan tubuh. Komputer akan membuat **model AI**, semacam otak kecil yang digunakan untuk mengenali hal-hal baru. Jadi, kalau kamu tunjukkan gambar anjing, model ini bisa menebak: "Apakah ini anjing?" Seru, kan? Semua proses ini terjadi karena adanya **data pelatihan**, **model**, dan **prediksi** yang bekerja sama.

Lalu, bagaimana kalau kita ingin langsung praktik? Tenang! Dengan **Google Teachable Machine**, kalian bisa membuat *AI* sendiri hanya dengan beberapa klik. Mau bikin proyek mengenali suara tepuk tangan? Atau membedakan ekspresi wajah senang dan marah? Semua bisa, tanpa harus jadi programmer! Kalian bisa pilih proyek gambar (*Image Project*),

suara (*Audio Project*), atau gerakan tubuh (*Pose Project*). Jadi, siapkah kalian memulai **petualangan cerdas** ini dan jadi bagian dari generasi pembuat teknologi masa depan?

# Latihan Soal

## A. Benar atau Salah (5 Soal)

1. Google Teachable Machine bisa digunakan tanpa harus menulis kode.
   **Jawaban: Benar**

2. AI tidak bisa mengenali suara manusia.
   **Jawaban: Salah**

3. Data pelatihan yang banyak dan bervariasi membuat model AI lebih pintar.
   **Jawaban: Benar**

4. Hasil dari pelatihan AI disebut model.
   **Jawaban: Benar**

5. Teachable Machine hanya bisa digunakan untuk gambar.
   **Jawaban: Salah**

## B. Pilihan Ganda (10 Soal)

6. Apa itu AI?
   A. Aplikasi untuk menggambar
   B. Kemampuan mesin meniru cara berpikir manusia
   C. Bahasa pemrograman baru
   D. Alat komunikasi online
   **Jawaban: B**

7. Machine Learning adalah...
   A. Mesin cuci otomatis
   B. Aplikasi edit foto
   C. Proses belajar mesin dari data
   D. Cara mengetik cepat di komputer
   **Jawaban: C**

8. Contoh AI dalam kehidupan sehari-hari adalah...
   A. Penggaris
   B. Kipas angin
   C. Siri dan Google Assistant
   D. Kertas gambar
   **Jawaban: C**

9. Untuk membuat model suara di Teachable Machine, kita memilih proyek...
   A. Image Project
   B. Audio Project
   C. Pose Project
   D. Voice Chat
   **Jawaban: B**

10. Fungsi tombol “Train Model” di Teachable Machine adalah...
   A. Mengedit foto
   B. Membuat suara robot
   C. Melatih AI dengan data yang dikumpulkan
   D. Menyimpan gambar
   **Jawaban: C**

11. Prediksi pada AI berarti...
   A. Menebak hasil berdasarkan data baru
   B. Menghapus data lama
   C. Membuat dokumen
   D. Merekam suara
   **Jawaban: A**

12. Jika ingin membedakan “kucing” dan “anjing”, kita harus...
   A. Menulis program panjang
   B. Memberi warna berbeda
   C. Mengumpulkan gambar kucing dan anjing
   D. Membuat suara lucu
   **Jawaban: C**

13. Apa yang akan terjadi jika data pelatihan terlalu sedikit?
   A. Model AI akan bekerja lebih cepat
   B. Model AI bisa salah mengenali
   C. Model AI jadi lebih keren
   D. Tidak berpengaruh
   **Jawaban: B**

14. Google Teachable Machine bisa membuat model untuk mengenali...
   A. Film
   B. Cuaca
   C. Pose tubuh
   D. Makanan
   **Jawaban: C**

15. Mengapa penting untuk menambah kelas “lain-lain” di Teachable Machine?
   A. Supaya AI bisa bernyanyi
   B. Agar AI tidak bingung dengan hal baru
   C. Untuk mempercepat proses pelatihan
   D. Supaya AI bisa tidur
   **Jawaban: B**

## C. Isian Singkat (5 Soal)

16. Hasil dari proses belajar AI disebut _____.
    **Jawaban: Model**

17. Data berupa gambar, suara, atau teks yang digunakan AI untuk belajar disebut _____.
    **Jawaban: Data pelatihan**

18. Teachable Machine dibuat oleh perusahaan _____.
    **Jawaban: Google**

19. Untuk membedakan suara tepuk tangan dan batuk, kita harus merekam _____ untuk setiap kelas.
    **Jawaban: Suara yang berbeda-beda**

20. AI belajar dari _____ yang kita berikan.
    **Jawaban: Data**

## D. Soal Eksplorasi

**Soal:**

Jelaskan langkah-langkah yang harus kamu lakukan jika ingin membuat model Teachable Machine yang bisa membedakan antara suara tepuk tangan dan suara siulan.

**Jawaban:** Pertama, buka situs teachablemachine.withgoogle.com dan pilih "Get Started". Lalu pilih "Audio Project" dan klik “Standard audio model”. Buat dua kelas, masing-masing diberi nama "Tepuk Tangan" dan "Siulan". Kemudian, rekam suara tepuk tangan dari berbagai variasi (cepat, lambat, keras, pelan) untuk kelas pertama, dan suara siulan dengan variasi yang sama untuk kelas kedua. Setelah data terkumpul cukup, klik "Train Model" dan tunggu hingga proses pelatihan selesai. Terakhir, gunakan fitur "Preview" untuk menguji model apakah berhasil membedakan suara tepuk tangan dan siulan.

**Soal:**

Bayangkan kamu membuat model yang bisa mengenali tiga ekspresi wajah: senang, marah, dan sedih. Apa saja tantangan yang kamu hadapi saat mengumpulkan data? Bagaimana cara kamu mengatasi tantangan tersebut?

**Jawaban:** Tantangan yang mungkin saya hadapi adalah mendapatkan variasi ekspresi yang cukup dari berbagai orang. Jika hanya menggunakan wajah saya sendiri, model mungkin tidak mengenali ekspresi orang lain. Selain itu, pencahayaan atau latar belakang bisa mempengaruhi akurasi model. Untuk mengatasinya, saya akan mengumpulkan foto dari beberapa teman dengan ekspresi yang sama namun wajah dan kondisi cahaya berbeda. Saya juga akan memastikan untuk mengambil foto dari berbagai sudut dan posisi wajah agar model bisa belajar lebih baik.

**Soal:**

Buatlah ide penggunaan model Teachable Machine untuk membantu kegiatan di sekolah. Misalnya, model yang bisa mengenali apakah siswa sedang duduk atau berdiri di kelas. Jelaskan cara kerjanya, apa manfaatnya, dan bagaimana kamu akan melatih model tersebut.

**Jawaban:** Saya ingin membuat model AI yang bisa mendeteksi apakah siswa sedang duduk atau berdiri. Caranya adalah dengan memilih “Pose Project” di Teachable Machine, lalu membuat dua kelas: "Duduk" dan "Berdiri". Saya akan merekam gerakan beberapa siswa saat duduk dan saat berdiri dari berbagai sudut dan posisi tubuh. Setelah itu, model dilatih menggunakan data tersebut. Manfaatnya adalah guru bisa memantau apakah siswa sudah duduk saat pelajaran dimulai, atau apakah ada yang terlalu lama berdiri. Ini juga bisa membantu dalam aktivitas interaktif atau permainan edukatif di kelas.

**Soal:**

Jelaskan langkah-langkah yang harus kamu lakukan jika ingin membuat model Teachable Machine yang bisa membedakan antara suara tepuk tangan dan suara siulan. Tulis dengan urutan yang benar, mulai dari memilih proyek hingga menguji model.

**Soal:**

Bayangkan kamu membuat model yang bisa mengenali tiga ekspresi wajah: senang, marah, dan sedih. Apa saja tantangan yang kamu hadapi saat mengumpulkan data? Bagaimana cara kamu mengatasi tantangan tersebut?

**Soal:**

Buatlah ide penggunaan model Teachable Machine untuk membantu kegiatan di sekolah. Misalnya, model yang bisa mengenali apakah siswa sedang duduk atau berdiri di kelas. Jelaskan cara kerjanya, apa manfaatnya, dan bagaimana kamu akan melatih model tersebut.

# Tugas Proyek

## Proyek 1: AI Gerakan Tangan untuk Mengendalikan Game

**Judul:** “Tanganmu, Joystick-mu!”

**Deskripsi:** Buat model AI yang dapat mengenali tiga gerakan tangan: **maju, mundur, dan berhenti**. Model ini bisa digunakan untuk mengendalikan game sederhana atau memberi perintah pada aplikasi lain.

**Langkah Saran:**

1. Buka Google Teachable Machine, pilih **Image Project > Standard**.
2. Buat tiga kelas: *Maju*, *Mundur*, dan *Berhenti*.

3. Kumpulkan gambar atau video pendek tangan kamu dari berbagai sudut dan pencahayaan.
4. Latih model, lalu uji coba apakah akurat saat mendeteksi gerakan.
5. Hubungkan ke Scratch atau aplikasi interaktif untuk simulasi pergerakan karakter.
6. Refleksikan: Bagaimana jika gerakan tangan kamu salah dikenali?

## Proyek 2: AI Pengenal Suara Hewan

**Judul:** “Tebak Suara Siapa?”

**Deskripsi:** Bangun model AI yang mampu mengenali tiga suara hewan peliharaan seperti **anjing, kucing, dan burung**. Cocok untuk digunakan dalam kuis interaktif atau sebagai alat bantu belajar anak kecil.

**Langkah Saran:**

1. Pilih **Audio Project > Standard** di Teachable Machine.
2. Buat tiga kelas: *Gonggongan Anjing*, *Meongan Kucing*, *Kicauan Burung*.
3. Rekam suara nyata dari hewan atau gunakan klip suara dari internet.
4. Latih model dan uji coba suara baru: Apakah AI bisa mengenali dengan benar?
5. Dokumentasikan jumlah sampel, kualitas suara, dan kesulitan saat pelatihan.
6. Bonus: Gunakan model ini untuk membuat kuis “tebak suara”.

### Proyek 3: AI Deteksi Posisi Tubuh (Duduk atau Berdiri)

**Judul:** “Siapa Berdiri, Siapa Duduk?”

**Deskripsi:** Kembangkan model AI yang dapat membedakan apakah seseorang **duduk atau berdiri**, misalnya untuk digunakan di ruang kelas pintar atau alarm kebugaran.

**Langkah Saran:**

1. Pilih **Image Project > Standard**.
2. Buat dua kelas: *Duduk* dan *Berdiri*.
3. Ambil gambar kamu atau temanmu saat duduk dan berdiri dari berbagai sudut.
4. Latih dan evaluasi model: Apakah bisa mengenali posisi orang berbeda?
5. Presentasikan hasil ke kelas: tunjukkan akurasi dan cara kerjanya.
6. Diskusikan: Apa yang bisa membuat model bingung (misalnya posisi setengah duduk)?

## Proyek 4: Eksplorasi AI di Sekitar Kita

**Judul:** “AI di Kehidupan Sehari-hari”

**Deskripsi:** Identifikasi **tiga contoh aplikasi AI** dalam kehidupan sehari-hari (seperti asisten suara, kamera otomatis, atau rekomendasi film) dan jelaskan **cara kerjanya secara sederhana**.

**Langkah Saran:**

1. Pilih tiga contoh nyata: bisa dari rumah, sekolah, atau internet.
2. Riset singkat tentang bagaimana AI bekerja dalam tiap aplikasi.
3. Buat infografis atau poster presentasi.
4. Tambahkan simulasi sederhana dengan Teachable Machine jika memungkinkan.
5. Diskusi kelas: Manakah AI yang paling bermanfaat? Apa risikonya?

## Proyek 5: Pentingnya Data yang Beragam

**Judul:** “Kenapa AI Butuh Banyak Wajah?”

**Deskripsi:** Buat eksperimen sederhana untuk membuktikan bahwa model AI membutuhkan data pelatihan yang **beragam**. Gunakan satu jenis data, lalu bandingkan dengan data dari berbagai orang/sudut.

**Langkah Saran:**

1. Buat model sederhana (misal: senyum vs. tidak senyum).
2. Latih hanya dengan wajah kamu, lalu uji coba ke teman lain.
3. Ulangi proses dengan menambah wajah teman dari berbagai sudut.
4. Bandingkan hasil akurasinya.
5. Refleksikan dan presentasikan: Apa yang terjadi jika data tidak bervariasi?

## Proyek 6: AI Pendeteksi Emosi

**Judul:** “Ekspresi Wajahmu Dibaca AI!”

**Deskripsi:** Buat AI yang bisa mengenali ekspresi wajah sederhana seperti **senang, sedih, dan marah**.

**Langkah Saran:**

1. Gunakan **Image Project > Standard**.
2. Kumpulkan gambar ekspresi wajah dari kamu dan temanmu.
3. Latih model dan lihat apakah bisa mengenali dengan benar.
4. Diskusikan: Apakah AI bisa salah mengenali ekspresi? Mengapa?
5. Kembangkan ide: Bisa tidak model ini digunakan untuk membantu orang dengan autisme?

# BAB 8: Mengenal Domain .id

## Apa Itu Domain?

Bayangkan internet seperti **kota besar** yang punya **jutaan rumah**. Setiap rumah di kota itu punya alamat agar bisa dikunjungi. Nah, nama **domain** itu seperti **alamat kita di internet**, nanti bisa digunakan sebagai alamat website maupun alamat email kita. Dan kita bisa mempunyai alamat email sendiri misalnya halo@meita.id. Misalnya alamat website

Contohnya:

- pandi.id
- home.s.id
- meita.my.id

Alamat itu disebut **nama domain**, dan tanpa domain, kamu harus mengingat angka-angka ribet yang disebut **IP address**. Pasti gak seru, kan?

## Apa Itu .id?

Sekarang coba perhatikan domain ini: www.sekolahku.id

Nah, yang bagian **".id"** itu disebut **ekstensi domain**. ".id" menunjukkan bahwa website itu **berasal dari Indonesia**. Setiap negara punya ekstensinya masing-masing, contohnya:

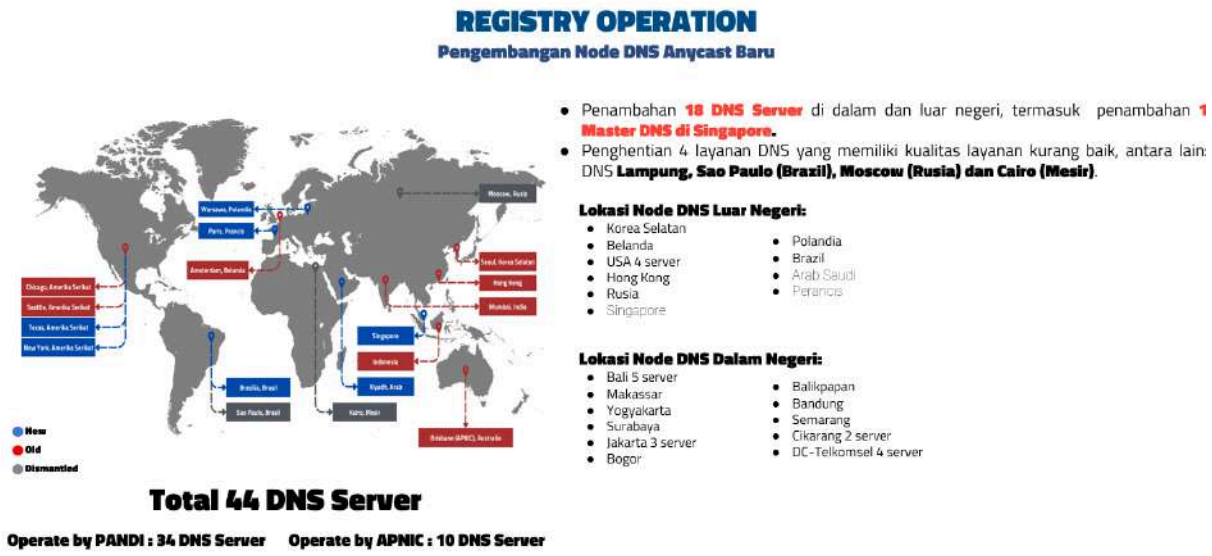
- .jp → Jepang
- .us → Amerika Serikat
- .id → Indonesia 🇮🇩

Menggunakan domain .id itu keren, karena:

- Menunjukkan kebanggaan sebagai orang Indonesia
- Membuat situsmu terlihat lebih terpercaya di mata orang Indonesia
- Cocok banget buat bisnis lokal, sekolah, komunitas, atau bahkan blog pribadi

## Keunggulan domain .id

- Representasi Indonesia
  Menunjukan bahwa ini entitas ataupun personal dari Indonesia karena .id adalah kode negara Indonesia
- Sebagai identity
  Selain reprensentasi Indonesia .id juga bisa berarti sebagai identity atau identitas, sehingga sangat menarik untuk dijadikan sebuah alamat website, email maupun brand di internet




## Siapa Itu PANDI?

**PANDI** adalah singkatan dari **Pengelola Nama Domain Internet Indonesia.**
Coba bayangkan **PANDI** seperti **penjaga gerbang dunia .id**. Mereka:

- Mengatur dan mengelola semua domain dengan **akhiran .id**
- Menjaga agar **domain .id aman, tertata, dan tidak disalahgunakan**
- Memberi kesempatan buat siapa saja — termasuk kamu — **punya domain sendiri!**

Misalnya kamu ingin punya website "tokohfavoritku.id", kamu bisa daftar lewat penyedia domain resmi, dan **PANDI** yang mengatur sistemnya agar website-mu bisa diakses semua orang.

## Kenapa Kamu Harus Tahu Ini?

- Siapa tahu **nanti kamu bikin website** untuk tugas sekolah, bisnis kecil-kecilan, atau komunitas temanmu
- Supaya kamu melek teknologi, karena **dunia digital adalah masa depan**
- Karena kamu bisa bikin identitas online-mu sendiri dan itu sangat keren!

Kalau kamu bisa punya akun Instagram, kamu juga bisa punya domain sendiri. Bayangkan: www.namamu.id – tempat kamu membagikan karya, foto, tulisan, atau bahkan jualan!

## Lebih dalam tentang PANDI

**PANDI** adalah Registry Nama Domain Indonesia yang berperan untuk mengoperasikan, memelihara dan mengelola Nama Domain Indonesia (.id)

**Pengelola Nama Domain Internet Indonesia (PANDI)** merupakan perkumpulan yang terdiri dari berbagai pemangku kepentingan, termasuk perwakilan dari pemerintah, operator industri internet, dan akademisi.

Didirikan pada tahun 2006, **PANDI** menerima Redelegasi dari **Internet Assigned Numbers Authority (IANA)** sebagai Registry .id pada tahun 2013.

Hingga 31 Desember 2024 jumlah domain .id yang terdaftar tumbuh menjadi yang terbesar di Asia Tenggara, dengan total 1.215.714 Nama Domain. Nama Domain .id sendiri bisa diartikan sebagai Indonesia, identitas, ide, dan lainnya. Sejalan dengan makna tersebut, **PANDI** juga memiliki kebijakan khusus dalam menyelesaikan perselisihan nama domain .id.

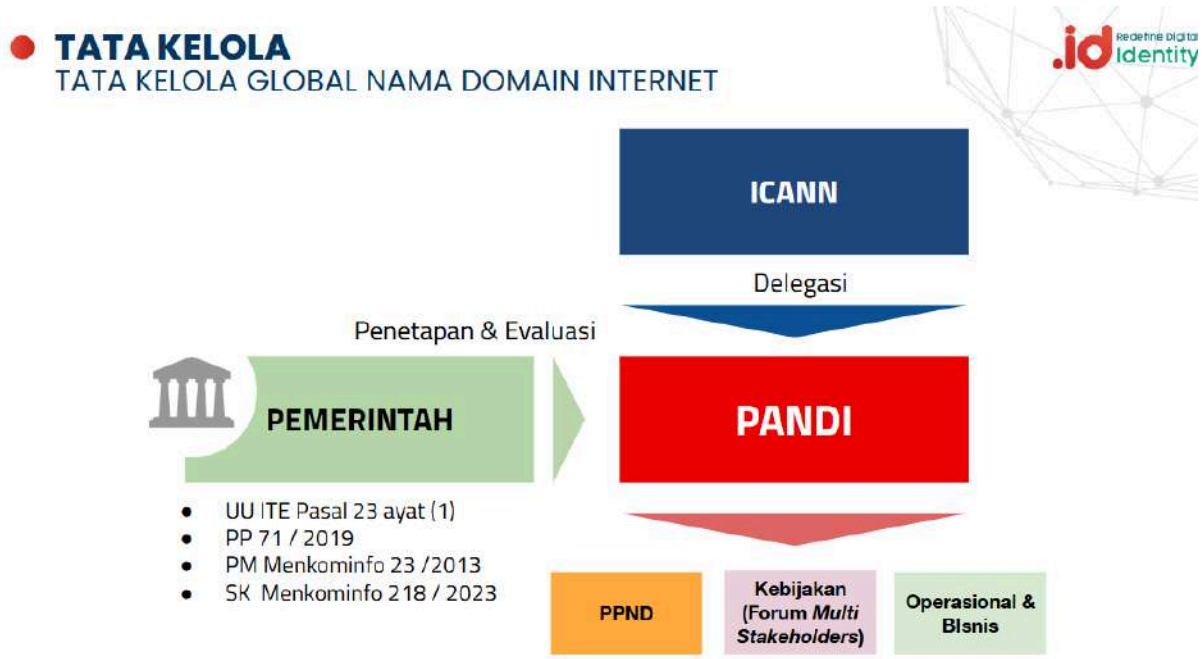
## Tugas dan Kewajiban PANDI

**Tugas**

- Merumuskan kebijakan di bidang pengelolaan Nama Domain Tingkat Tinggi Indonesia
- Menyiapkan, mengoperasikan, dan memelihara infrastruktur yang dibutuhkan serta menyediakan sistem elektronik untuk pengelolaan Nama Domain Tingkat Tinggi Indonesia
- Menyelenggarakan pendaftaran Nama Domain Tingkat Tinggi Indonesia sesuai dengan ketentuan peraturan perundang-undangan, kepatutan yang berlaku dalam masyarakat, dan prinsip kehati-hatian
- Melaksanakan seleksi Registrar Nama Domain
- Memberikan peringatan kepada Registrar Nama Domain jika terindikasi melakukan pelanggaran
- Mencabut hak operasional Registrar Nama Domain jika terbukti melakukan pelanggaran, dan
- Melakukan pengawasan operasional dan teknis Registrar Nama Domain

**Kewajiban**

- Menjamin sistem elektronik Registri Nama Domain Indonesia beroperasi dengan baik, stabil, aman didukung dengan layanan yang dapat diandalkan
- Menempatkan pusat data dan pusat pemulihan bencana di wilayah Indonesia
- Melakukan pengawasan terhadap Registrar Nama Domain
- Memfasilitasi penyelesaian perselisihan Nama Domain Indonesia
- Melaporkan daftar Registrar Nama Domain kepada Menteri
- Mengikuti ketentuan pengelolaan Nama Domain internasional dan peraturan perundang-undangan
- Menyampaikan laporan berkala kepada Menteri sekurang-kurangnya 1 (satu) kali dalam setahun
- Membayar pungutan biaya Pengelolaan Nama Domain Indonesia




## Tata Kelola Internet

PANDI berkomitmen penuh untuk terus berkontribusi aktif dalam tata kelola internet global dengan mendukung infrastruktur yang aman dan inovatif. Hal ini karena domain .id merupakan bagian penting dari ekosistem digital dunia yang inklusif dan berkelanjutan.

Di tahun 2024, PANDI menginisiasi berbagai wadah tata kelola internet nasional serta berpartisipasi aktif dalam forum tata kelola internet global, seperti:

- Forum Multi Stakeholder (FMS),
- PANDI Meeting 2024,
- ICANN 79 di Puerto Riko,
- ICANN 81 di Turki,
- APTLD 86 di Vietnam,
- APAC DNS Forum 2024 yang diselenggarakan oleh PANDI di Bali.

Selain APAC DNS Forum, PANDI juga menggelar program Indonesia Internet Governance Academy (IDIGA) yang bertujuan mencetak pemimpin muda di bidang tata kelola internet, baik di tingkat nasional maupun internasional.

# Institut Teknologi Tangerang Selatan (ITTS) dan Penulis.

Para penulis adalah **dosen aktif di Institut Teknologi Tangerang Selatan (ITTS)**, sebuah **institusi pendidikan tinggi berbasis teknologi** yang berperan aktif dalam pengembangan **solusi digital** dan *kecerdasan buatan (AI)*. Tidak hanya mengajar, mereka juga merupakan **praktisi dan inovator** yang menghasilkan **karya-karya nyata di bidang teknologi mandiri**.

Berlokasi di **Komplek Komersial BSD, Serpong Utara, Tangerang Selatan**, **ITTS hadir sebagai pusat inovasi** yang mengintegrasikan **pembelajaran, penelitian, dan pengembangan teknologi secara menyeluruh**. Fokus utama ITTS mencakup bidang strategis seperti *AI, Machine Learning, Deep Learning*, jaringan, *cloud*, keamanan siber, pemrograman, sistem informasi, dan multimedia.

**ITTS menawarkan lingkungan akademik yang dinamis dan kolaboratif.** Setiap minggu, diselenggarakan berbagai *workshop*, *demo teknologi*, seminar, hingga *webinar*—mayoritas **gratis dan tersedia secara terbuka di YouTube**. Ini menciptakan **kultur belajar yang aktif dan relevan dengan perkembangan industri**.

**Fasilitas ITTS menunjang kegiatan riset dan inovasi secara optimal.** Lab komputer dilengkapi *GPU RTX 4060* untuk eksperimen *AI*, serta **laboratorium teknologi dengan puluhan server, perangkat *IoT*, sistem komunikasi radio, dan infrastruktur jaringan jarak jauh**. Dari ekosistem ini lahirlah **produk-produk mandiri** seperti **ChatGPT versi lokal** untuk institusi pendidikan dan **sistem perpustakaan digital yang dapat diakses tanpa koneksi internet**.

**Daya saing ITTS** juga tercermin dari karakter dosen dan mahasiswa yang tidak hanya akademik, tetapi juga **produktif secara praktis**. Mahasiswa didorong aktif menjadi **narasumber, penulis, dan peneliti** sejak dini. Banyak diantaranya telah menghasilkan **karya buku, artikel ilmiah**, serta **terlibat langsung dalam proyek-proyek teknologi**.

**Kolaborasi strategis dengan berbagai mitra industri**—seperti *IDCloudHost, Ubuntu, XecureIT, Dinas KOMINFO Tangerang Selatan*, berbagai *ISP*, dan komunitas IT nasional—menjadi **jembatan nyata antara kampus dan dunia profesional**. Hasilnya, **lebih dari 75% mahasiswa ITTS telah bekerja bahkan sebelum lulus**, dan **hampir seluruh lulusan langsung terserap industri**.

**ITTS bukan hanya tempat belajar, tapi tempat mencipta.** Dosen sebagai **penulis dan praktisi**, mahasiswa sebagai **peneliti dan kontributor**—semuanya berperan **membangun ekosistem teknologi yang mandiri dan berdaya saing tinggi**.

**Institut Teknologi Tangerang Selatan (ITTS)**
Komplek Komersial BSD
Jl. Raya Serpong No. Kav. 9, Lengkong Karya, Serpong Utara
Kota Tangerang Selatan, Banten 15331
📞 (+62) 877-7277-1775 📧 info@itts.ac.id 🌐 www.itts.ac.id