UMM
umm.ac.id

IMPLEMENTASI 10 PROGRAM PENTING & STRATEGIS
UNIVERSITAS MUHAMMADIYAH MALANG

PROGRAM #9
P9.A06
Perubahan Materi Pelatihan Aplikasi Internet menjadi Pelatihan Dasar Teknologi Digital bagi Mahasiswa Baru

MODUL

PELATIHAN DASAR TEKNOLOGI DIGITAL UNTUK MAHASISWA BARU UNIVERSITAS MUHAMMADIYAH MALANG

Bahasa Pemrograman python™

BIRO SISTEM INFORMASI DAN DIGITALISASI UMM
OKTOBER, 2024

Kata Pengantar

Dengan penuh rasa syukur, modul "Panduan Teknologi Dasar Digital dengan Pemrograman Python" ini sebagai pendamping bagi mahasiswa baru dalam mempelajari dasar-dasar teknologi digital. Modul ini dirancang khusus untuk mahasiswa baru dari berbagai jurusan, dengan beragam latar belakang pengetahuan pemrograman, baik yang sudah memiliki pengalaman maupun yang baru memulai.

Kami memahami bahwa setiap mahasiswa memiliki pengalaman dan pemahaman yang berbeda dalam dunia pemrograman. Oleh karena itu, modul ini disusun dengan pendekatan yang sederhana, menyenangkan sehingga proses belajar menjadi lebih mudah diikuti oleh semua kalangan. Pemrograman Python dipilih karena bahasanya yang mudah dipahami, serta penggunaannya yang luas dalam berbagai bidang, seperti analisis data, pengembangan aplikasi, hingga kecerdasan buatan.

Melalui modul ini, kami berharap para mahasiswa baru UMM dapat belajar dengan cara yang interaktif dan menyenangkan. Kami percaya bahwa belajar pemrograman seharusnya menjadi pengalaman yang menginspirasi dan memicu rasa ingin tahu, bukan sesuatu yang rumit dan menakutkan. Dengan demikian, kami merancang setiap materi dalam modul ini agar dapat membantu mahasiswa membangun keterampilan pemrograman mereka dengan cara yang lebih santai, namun tetap efektif.

Semoga modul ini tidak hanya menjadi alat bantu belajar, tetapi juga menginspirasi mahasiswa untuk terus mengeksplorasi dunia teknologi digital dengan semangat dan antusiasme. Kami berharap pembaca dapat menemukan pengalaman belajar yang positif dan menyenangkan melalui modul ini.

Kami mengucapkan terima kasih kepada semua pihak yang telah berkontribusi dalam penyusunan modul ini. Semoga modul ini bermanfaat dan membawa kesuksesan dalam perjalanan akademik dan karir Anda.

Selamat belajar, dan nikmati prosesnya!

Penyusun

Daftar Isi

Modul 1: Pengenalan Pemikiran Komputasional

Computational Thinking (CT) atau Pemikiran komputasional adalah proses pemecahan masalah yang fundamental yang melibatkan berbagai keterampilan dan teknik kognitif yang digunakan untuk menangani masalah yang kompleks. Awalnya berakar dari ilmu komputer, pemikiran komputasional telah memperluas pengaruhnya di berbagai disiplin ilmu, termasuk pendidikan, sains, teknik, dan bisnis. Pada intinya, CT memungkinkan individu untuk memecah masalah, menciptakan algoritma, dan memahami mekanisme di balik sistem dan proses. Bagian ini akan menguraikan komponen utama pemikiran komputasional, menekankan relevansinya di era teknologi digital.

Komponen Pemikiran Komputasional

Pemikiran komputasional dapat dibagi menjadi empat komponen kunci:

- **Dekomposisi**:

  Kemampuan untuk memecah masalah kompleks menjadi bagian-bagian yang lebih kecil sehingga lebih mudah dipahami dan diselesaikan.

- **Pengenalan Pola**:

  Mengidentifikasi pola, tren, dan kesamaan dalam data atau masalah. Keterampilan ini membantu dalam memprediksi hasil dan membuat keputusan yang tepat.

- **Abstraksi**:

  Menyederhanakan masalah dengan fokus pada detail yang penting dengan mengabaikan informasi yang tidak relevan. Proses ini memungkinkan solusi yang lebih umum yang dapat diterapkan pada berbagai situasi.

- **Desain Algoritma**:

  Mengembangkan prosedur langkah demi langkah atau seperangkat aturan untuk memecahkan masalah. Algoritma sangat penting untuk pemrograman dan dapat diterapkan pada permasalahan sehari-hari.

Pentingnya Pemikiran Komputasional

Dengan melatih berpikir komputasional sejak dini, maka otak akan terbiasa berpikir logis, terstruktur, serta dapat memecahkan masalah dengan mengembangkan solusi yang tepat. Berikut adalah beberapa alasan mengapa pemikiran komputasional sangat penting:

- **Keterampilan Pemecahan Masalah**:

  CT mendorong pendekatan terstruktur untuk pemecahan masalah, meningkatkan keterampilan berpikir kritis dan analitis.

- **Aplikasi Interdisipliner**:

  Pemikiran komputasional dapat diterapkan di berbagai bidang, seperti biologi (bioinformatika), ekonomi (analisis data), dan ilmu sosial (analisis survei), menjadikannya seperangkat keterampilan yang serbaguna.

- **Kecakapan Teknologi**:

  Di dunia yang semakin digital, memahami pemikiran komputasional membantu individu menjadi mahir dalam menggunakan teknologi, meningkatkan daya saing di pasar kerja.

- **Persiapan untuk Karir Masa Depan**:

  Banyak pekerjaan di dunia kerja saat ini memerlukan pemahaman tentang konsep-konsep komputasional. Setiap orang yang memiliki keterampilan CT lebih siap untuk karir di berbagai bidang, termasuk teknologi, keuangan, kesehatan, dan penelitian.

Aplikasi Pemikiran Komputasional dalam Kehidupan Sehari-hari

Pemikiran komputasional tidak terbatas pada ilmu komputer. Hal ini dapat dilihat pada kehidupan sehari-hari. Berikut adalah beberapa contoh yang relevan pada institusi pendidikan tinggi:

- **Proyek Penelitian**:

  Saat melakukan penelitian, mahasiswa seringkali menghadapi masalah kompleks. Dengan menerapkan dekomposisi, mereka dapat memecah permasalahan penelitian menjadi komponen yang lebih kecil, sehingga prosesnya dapat dikelola dengan mudah.

- **Kerja Kelompok**:

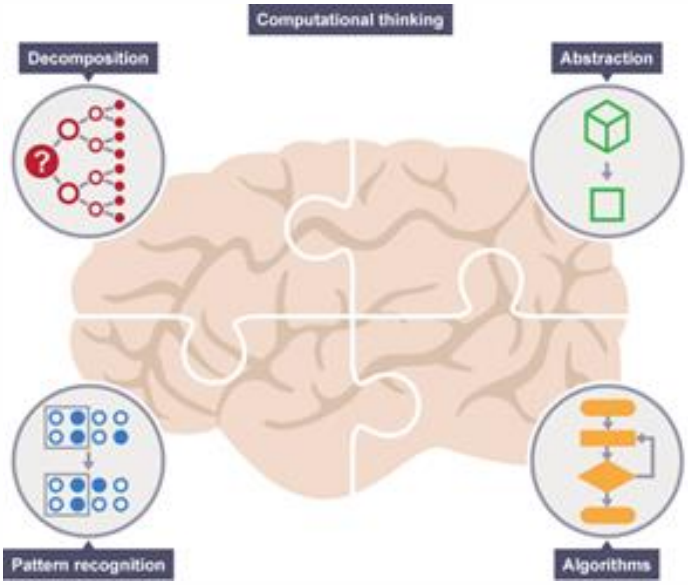
  Dalam proyek kolaboratif, mahasiswa dapat menggunakan pengenalan pola untuk mengidentifikasi peran dan tanggung jawab berdasarkan kelebihan dan kekurangan anggota tim.

- **Manajemen Waktu**:

  Mahasiswa dapat menerapkan abstraksi untuk memprioritaskan tugas dengan fokus pada tujuan dan memungkinkan untuk membuat jadwal yang efektif.

- **Analisis Data**:

  Saat menganalisis data, mahasiswa dapat mengembangkan algoritma untuk memproses data dengan efisien, mengidentifikasi tren yang mendukung kesimpulan mereka.



Studi Kasus CT

Studi Kasus 1: Menerapkan Pemikiran Komputasional dalam Bidang Biologi

Dalam bidang biologi, mahasiswa ditugaskan untuk menganalisis *dataset* besar yang terkait dengan perubahan lingkungan. Adapun cara menerapkan pemikiran komputasional sebagai berikut:

- **Mendekomposisi** *dataset* menjadi bagian-bagian yang lebih kecil, fokus pada variabel spesifik seperti suhu, kelembapan, dan populasi spesies.
- **Mengidentifikasi pola** dalam perilaku spesies sehubungan dengan perubahan suhu, memungkinkan mereka menarik kesimpulan yang berarti tentang dampak lingkungan.
- **Mengabstraksi** data untuk membuat model umum yang memprediksi perilaku spesies di berbagai kondisi lingkungan.
- **Mendesain algoritma** untuk mengotomatisasi pemrosesan data, yang meningkatkan efisiensi dan akurasi analisis.

Studi Kasus 2: Meningkatkan Keterampilan Pemecahan Masalah dalam Bidang Teknik

Bidang teknik mengintegrasikan pemikiran komputasional ke dalam kurikulumnya melalui proyek praktis. Mahasiswa diminta untuk:

- **Mendekomposisi** masalah teknik menjadi komponen individual, seperti desain, material, dan fungsionalitas.
- **Mengenali pola** dalam desain, belajar dari proyek sebelumnya.

- **Mengabstraksi** untuk membuat solusi yang dapat digunakan kembali, dan memfasilitasi kolaborasi antar tim yang bekerja pada masalah serupa.
- **Mengembangkan algoritma** untuk simulasi yang menguji desain dan meningkatkan pemahaman tentang permasalahan dalam bidang teknik.

Latihan Pemikiran Komputasional

Berikut adalah sepuluh latihan yang dirancang untuk membantu mahasiswa untuk mempraktikkan pemikiran komputasional beserta solusinya.

Latihan 1: Latihan Dekomposisi

Tugas: Dekomposisikan acara di perguruan tinggi menjadi sub tugas dengan menggunakan konsep CT.

Solusi:

1. Tentukan tujuan acara.
2. Pilih tanggal dan lokasi.
3. Buat anggaran.
4. Undang pembicara/tamu.
5. Atur pemasaran dan promosi.
6. Atur logistik (makanan, peralatan audio-visual).
7. Siapkan materi (program).
8. Tentukan lokasi.
9. Laksanakan acara.
10. Kumpulkan umpan balik dan evaluasi acara.

Latihan 2: Latihan Pengenalan Pola

Tugas: Analisis *dataset* nilai mahasiswa dan identifikasi pola terkait kebiasaan belajar.

Solusi:

1. Kumpulkan data tentang jam belajar, kehadiran, dan nilai.
2. Buat representasi visual (diagram/grafik).
3. Identifikasi tren (misalnya, siswa yang belajar lebih dari 10 jam per minggu umumnya mencapai nilai lebih tinggi).
4. Ringkas temuan dan usulkan rekomendasi untuk meningkatkan kebiasaan belajar.

Latihan 3: Latihan Abstraksi

Tugas: Buat model abstrak dari sistem lalu lintas sederhana.

Solusi:

1. Identifikasi komponen penting (mobil, lampu lalu lintas, persimpangan).
2. Abaikan detail yang tidak relevan (jenis mobil, kondisi cuaca).
3. Buat bagan alir (*flowchart*) untuk menggambarkan bagaimana mobil berinteraksi dengan lampu lalu lintas dan persimpangan.

Latihan 4: Latihan Desain Algoritma

Tugas: Rancang algoritma untuk mengurutkan daftar nama secara alfabetis.

Solusi:

1. Input: Daftar nama.
2. Langkah 1: Bandingkan dua nama pertama.
3. Langkah 2: Tukar jika perlu.
4. Langkah 3: Pindah ke nama berikutnya dan ulangi sampai seluruh daftar terurut.
5. Output: Daftar nama yang terurut.

Latihan 5: Latihan Pemecahan Masalah Kehidupan Nyata

Tugas: Terapkan pemikiran komputasional untuk memecahkan masalah umum, contoh: mengelola tugas harian Anda.

Solusi:

1. Dekomposisi tugas menjadi kategori (sekolah, kerja, pribadi).
2. Kenali pola dalam bagaimana tugas diselesaikan (misalnya, memprioritaskan tugas mendesak).
3. Abstrak untuk membuat jadwal harian.
4. Kembangkan algoritma untuk mengelola tugas berdasarkan kemampuan anggota tim.

Latihan 6: Latihan Analisis Data

Tugas: Analisa hasil survei yang diberikan tentang preferensi siswa untuk kelas online dibandingkan dengan tatap muka.

Solusi:

1. Masukkan data ke dalam *spreadsheet*.
2. Gunakan fungsi statistik untuk menghitung rata-rata, median, dan modus.
3. Buat alat bantu visual untuk mewakili temuan (misalnya: diagram lingkaran, grafik batang).
4. Buat ringkasan dan presentasikan dalam bentuk laporan.

Latihan 7: Latihan Desain Permainan

Tugas: Rancang permainan sederhana menggunakan prinsip pemikiran komputasional.

Solusi:

1. Dekomposisi permainan menjadi komponen kunci (aturan, tujuan, tindakan pemain).
2. Identifikasi pola dalam permainan untuk mencapai tujuan.
3. Buat abstrak untuk membuat kerangka umum dalam aturan permainan.
4. Kembangkan algoritma untuk pergerakan dan penilaian pemain.

Latihan 8: Latihan Simulasi

Tugas: Simulasikan ekonomi dasar menggunakan pemikiran komputasional.

Solusi:

1. Dekomposisi ekonomi menjadi elemen penting (permintaan, penawaran, harga).
2. Identifikasi pola dalam perilaku konsumen dan produsen.
3. Buat abstrak untuk menciptakan model yang menyederhanakan proses perilaku konsumen dan produsen.
4. Kembangkan algoritma untuk mensimulasikan bagaimana perubahan harga mempengaruhi permintaan dan penawaran.

Latihan 9: Latihan Analisis Keputusan

Tugas: Buat matriks keputusan untuk memilih antara dua pilihan karir.

Solusi:

1. Identifikasi kriteria penting (gaji, kepuasan kerja, lokasi, prospek pertumbuhan).
2. Berikan bobot pada setiap kriteria berdasarkan kepentingannya.
3. Berikan nilai setiap pilihan berdasarkan kriteria.
4. Hitung total skor untuk menentukan pilihan yang terbaik.

Latihan 10: Latihan Refleksi

Tugas: Tuliskan refleksi tentang bagaimana pemikiran komputasional telah membantu Anda menyelesaikan masalah di kehidupan sehari-hari.

Solusi:

1. Ceritakan pengalaman spesifik di mana Anda menerapkan CT.
2. Jelaskan bagaimana Anda menggunakan setiap komponen (dekomposisi, pola, abstraksi, algoritma).
3. Diskusikan hasil dan keterampilan yang Anda peroleh dari pengalaman tersebut.

Kesimpulan

Pemikiran komputasional adalah keterampilan penting bagi mahasiswa di era digital saat ini. Dengan mengembangkan keterampilan ini, mahasiswa tidak hanya mempersiapkan diri untuk tantangan akademis tetapi juga meningkatkan daya saing mereka di pasar kerja. Melalui pemahaman dan penerapan konsep-konsep CT, individu dapat menjadi pemecah masalah yang lebih efektif dan inovatif, berkontribusi secara positif dalam bidang mereka dan masyarakat secara keseluruhan.

Modul 2: Basic Python syntax

Tujuan Pembelajaran

Setelah menyelesaikan tutorial ini, mahasiswa diharapkan dapat:

1. Memahami sintaks dan struktur dasar Python.
2. Penggunaan variabel yang efektif.
3. Mengidentifikasi dan memanfaatkan berbagai tipe data.
4. Melakukan operasi input dan output dasar.
5. Menggunakan operator untuk memanipulasi data.
6. Menerapkan pengetahuan mereka melalui latihan praktis.

Python adalah bahasa pemrograman yang serbaguna dan banyak digunakan di berbagai bidang, termasuk pengembangan web, analisis data, kecerdasan buatan, dan komputasi ilmiah. Modul ini bertujuan untuk memperkenalkan sintaks dasar Python, mencakup konsep kunci yang membentuk fondasi untuk studi pemrograman lebih lanjut.

Mempersiapkan lingkungan Kompilasi Python

Berikut langkah-langkah Untuk menjalankan program Python menggunakan Google Collaboratory :

1. Akses Google Colab:
2. Buka browser dan masuk ke akun Google Anda.
3. Buka Google Drive
4. Di dalam Google Drive klik kanan pilih Google Collaboratory
5. Tulis Kode Python:

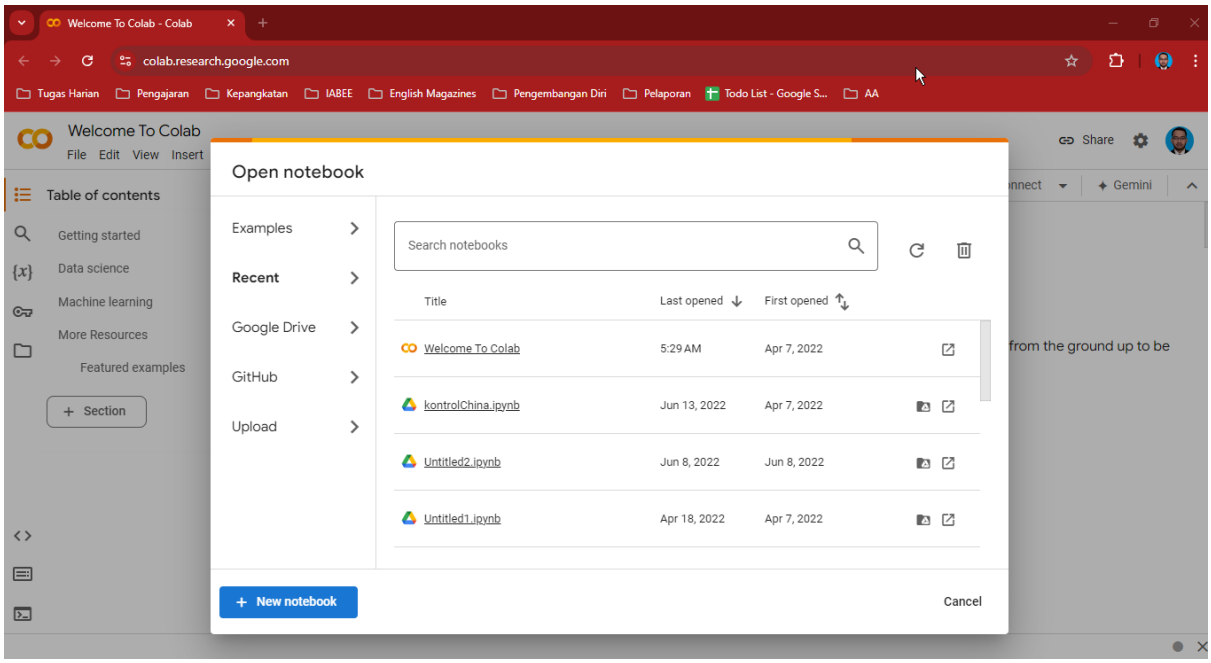
Setelah notebook terbuka, Anda akan melihat sel (cell) tempat Anda bisa menulis kode. Ketik kode Python yang ingin dijalankan di dalam sel. Untuk menjalankan kode, klik tombol play (ikon ▶) di sebelah kiri sel, atau tekan Shift + Enter setelah menulis kode.

Menjalankan Google Collab

Setelah Registrasi Google Collab berhasil, maka bisa membukanya dengan langkah-langkah berikut:

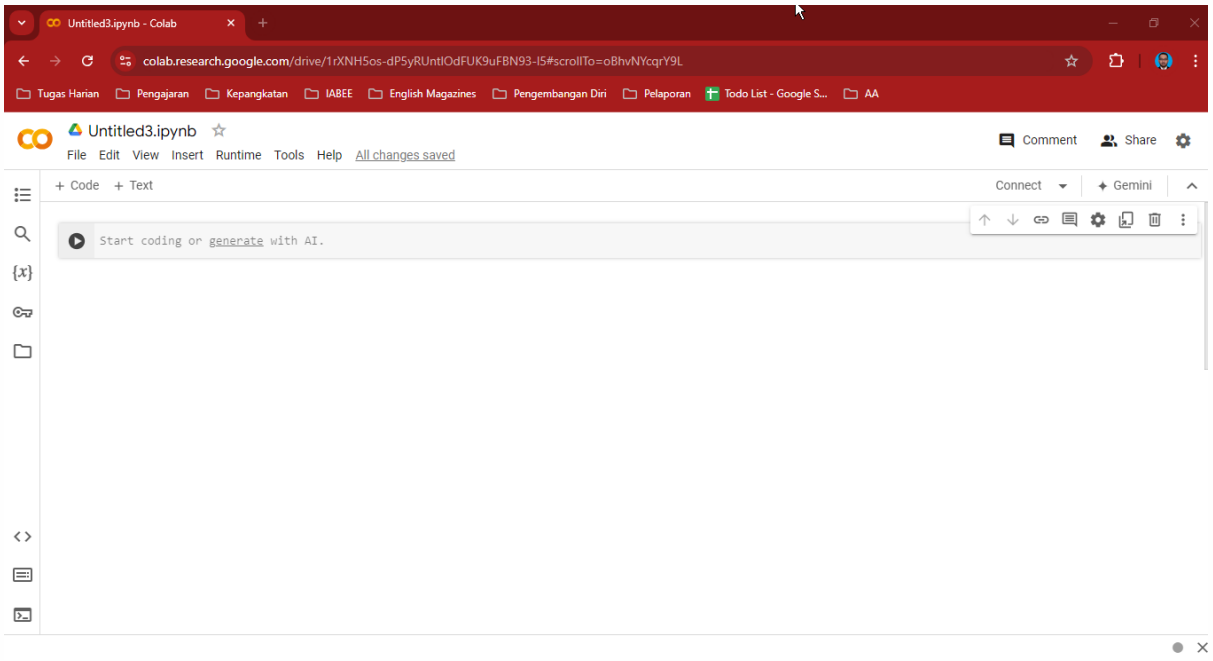
1. Buka alamat [Google Colab] dengan mengetikkan domain berikut https://colab.research.google.com/)

2. Klik **New Notebook** di bagian kiri bawah.



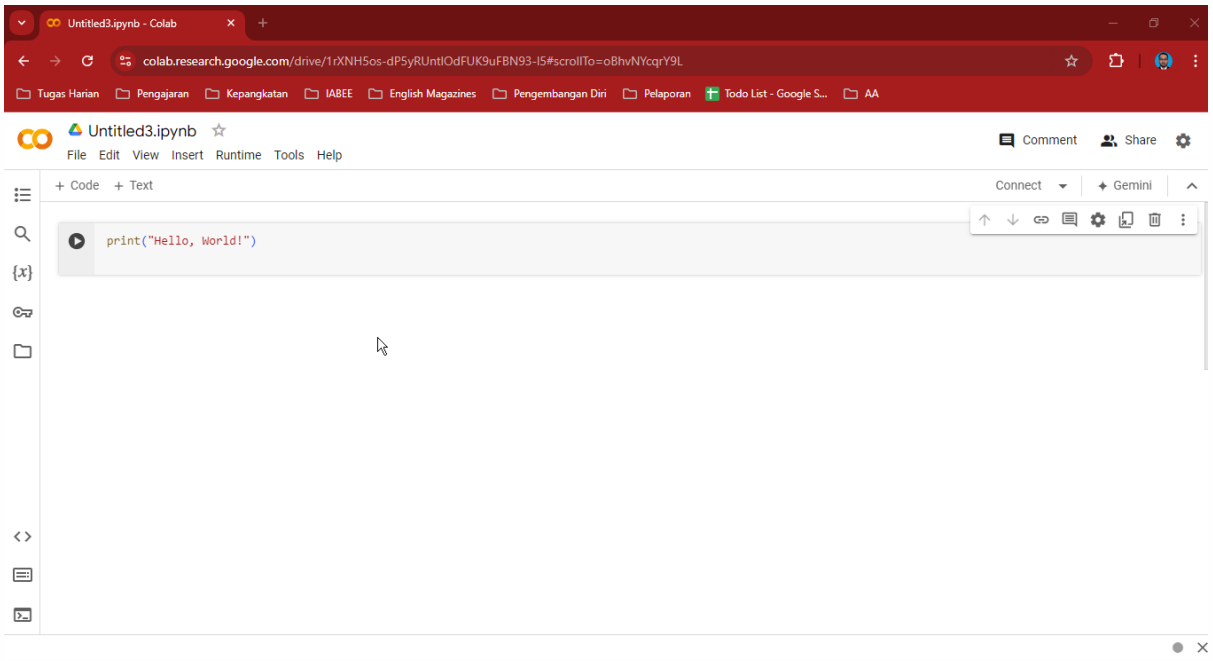
3. Menjalankan Kode di Jupyter Notebook**

   Sekarang, bisa menulis dan menjalankan kode Python di Jupyter Notebook:

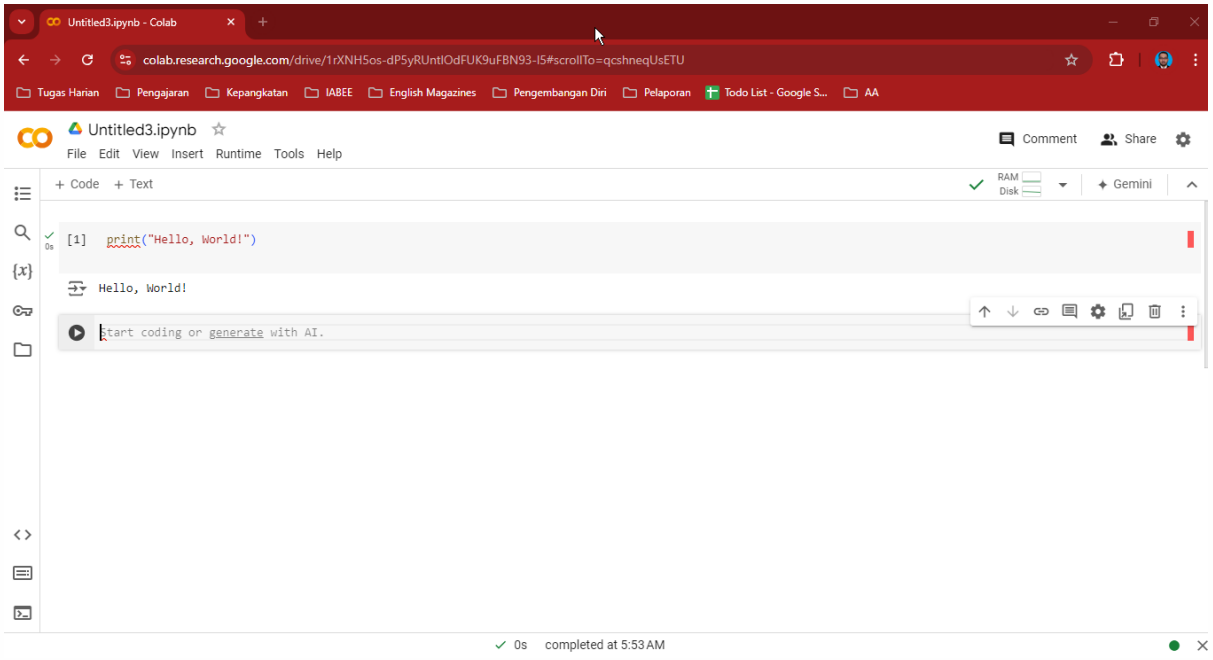


4. Di dalam sel (cell) yang tersedia, ketikkan kode Python, misalnya:

   print("Hello, World!")

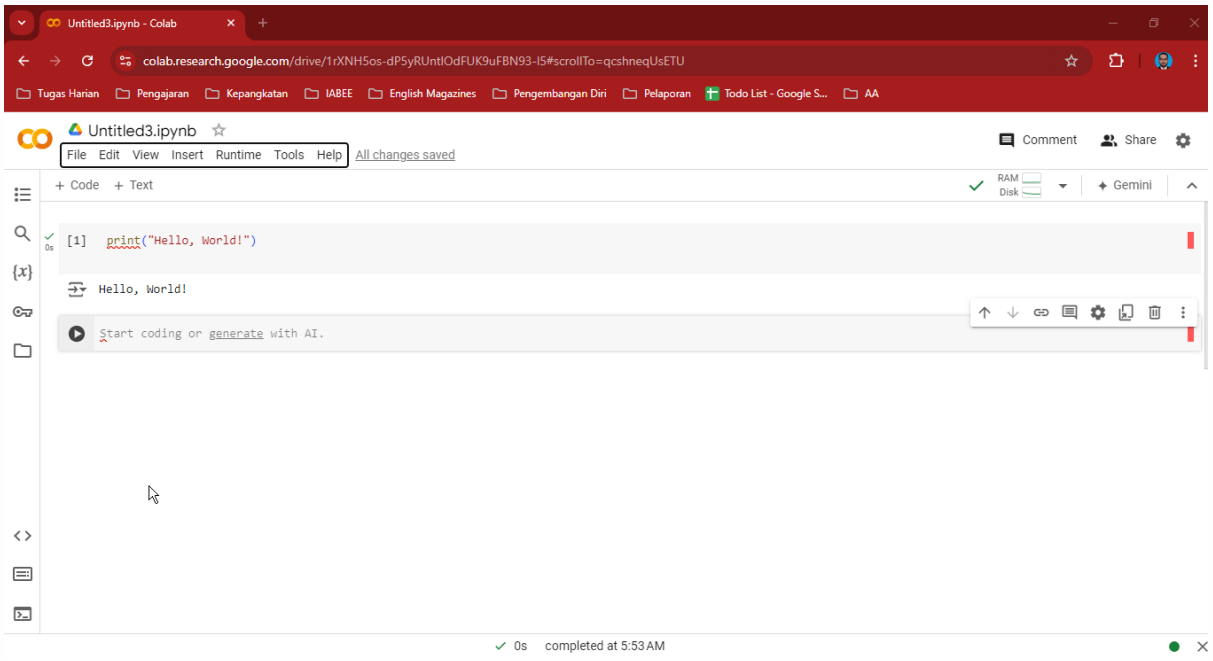


5. Tekan **Shift + Enter** untuk menjalankan kode tersebut. Hasilnya akan ditampilkan di bawah sel.



6. **Menyimpan Notebook**

Untuk menyimpan pekerjaan kamu, klik **File** > **Save** atau tekan **Ctrl + S**. Notebook kamu akan disimpan dengan ekstensi `.ipynb`, yang bisa dibuka kembali di Google Collab.



Syntax Python

1. Sintaks Dasar Python

Sintaks Python mengacu pada seperangkat aturan yang menentukan bagaimana program Python ditulis dan dipahami. Di bawah ini adalah contoh sederhana dari program Python:

```
Python Code
# Ini adalah program Python sederhana yang mencetak pesan sapaan.
print("Halo, Dunia!")
```

Poin Kunci:

- Komentar dimulai dengan `#` dan diabaikan oleh interpreter Python.
- Fungsi `print()` mengeluarkan teks ke konsol.

2. Variabel

Variabel digunakan untuk menyimpan nilai data. Dalam Python, Anda dapat menyatakan variabel hanya dengan memberikan nilai padanya:

```
Python Code
# Menyatakan variable
nama = "Alice"  # Variabel string
usia = 21       # Variabel integer
tinggi = 5.6    # Variabel float
```

Poin Kunci:

- Nama variabel harus deskriptif dan mengikuti konvensi penamaan (misalnya, `snake_case`).
- Python bersifat dinamis terketik, yang berarti Anda tidak perlu menyatakan tipe secara eksplisit.

3. Tipe Data

Python memiliki beberapa tipe data bawaan, termasuk:

- **Integer**: Angka bulat (misalnya, `5`, `-3`)
- **Float**: Angka desimal (misalnya, `3.14`, `-0.001`)
- **String**: Teks (misalnya, `"Halo"`, `'Python'`)
- **Boolean**: Nilai True atau False (`True`, `False`)

Contoh penggunaan berbagai tipe data:

```
Python Code
# Contoh tipe data yang berbeda
nilai_integer = 10
nilai_float = 10.5
nilai_string = "Halo, Python!"
nilai_boolean = True
```

4. Operator

Operator digunakan untuk melakukan operasi pada variabel dan nilai. Jenis operator yang paling umum di Python adalah:

- **Operator Aritmatika**: `+`, `-`, `*`, `/`, `%` (modulus)
- **Operator Perbandingan**: `==`, `!=`, `<`, `>`, `<=`, `>=`
- **Operator Logika**: `and`, `or`, `not`

Contoh Operator Aritmatika:

```
Python Code
# Melakukan operasi aritmatika
a = 10
b = 5
hasil_penjumlahan = a + b  # Penjumlahan
selisih = a - b            # Pengurangan
produk = a * b             # Perkalian
```

```
kuotien = a / b              # Pembagian
modulus = a % b              # Modulus
```

5. Input dan Output

Python menyediakan fungsi bawaan untuk operasi input dan output. Fungsi `input()` digunakan untuk mengambil input dari pengguna, sementara fungsi `print()` mengeluarkan data ke konsol.

Contoh Input dan Output:

```
Python Code
# Mengambil input dari pengguna dan menampilkannya
nama_pengguna = input("Masukkan nama Anda: ")  # Input dari pengguna
print("Halo, " + nama_pengguna + "!")          # Output ke konsol
```

Latihan

Berikut adalah beberapa latihan untuk mempraktikkan konsep yang dibahas dalam tutorial ini. Setiap latihan diikuti dengan solusinya.

Latihan 1: Deklarasi Variabel

Tugas: Deklarasikan tiga variabel: nama Anda, usia Anda, dan warna favorit Anda.

Solusi:

```
Python Code

nama = "Alice"

usia = 21

warna_favorit = "Biru"
```

Latihan 2: Aritmatika Dasar

Tugas: Hitung luas persegi panjang dengan lebar 5 dan tinggi 10.

Solusi:

```
Python Code

lebar = 5

tinggi = 10

luas = lebar * tinggi

print("Luas persegi panjang:", luas)
```

Latihan 3: Penggabungan String

Tugas: Buat pesan sapaan yang mencakup nama dan usia Anda.

Solusi:

```
Python Code

nama = "Alice"

usia = 21

sapaan = "Halo, nama saya " + nama + " dan saya berusia " + str(usia) +
" tahun."

print(sapaan)
```

Latihan 4: Kalkulator Sederhana

Tugas: Buat kalkulator sederhana yang menjumlahkan dua angka yang diberikan oleh pengguna.

Solusi:

```
Python Code

angka1 = float(input("Masukkan angka pertama: "))

angka2 = float(input("Masukkan angka kedua: "))

hasil_penjumlahan = angka1 + angka2

print("Hasil penjumlahan adalah:", hasil_penjumlahan)
```

Latihan 5: Konverter Suhu

Tugas: Konversi suhu dari Celsius ke Fahrenheit.

Solusi:

```
Python Code

celsius = float(input("Masukkan suhu dalam Celsius: "))

fahrenheit = (celsius * 9/5) + 32

print("Suhu dalam Fahrenheit:", fahrenheit)
```

Latihan 6: Memeriksa Genap atau Ganjil

Tugas: Periksa apakah angka yang dimasukkan oleh pengguna adalah genap atau ganjil.

Solusi:

```
Python Code

angka = int(input("Masukkan sebuah angka: "))

if angka % 2 == 0:

    print(angka, "adalah genap.")

else:

    print(angka, "adalah ganjil.")
```

Latihan 7: Daftar Buah

Tugas: Buat daftar dari tiga buah favorit Anda dan cetak daftar tersebut.

Solusi:

```
Python Code

buah = ["Apel", "Pisang", "Ceri"]

print("Buah favorit saya adalah:", buah)
```

Latihan 8: Loop Sederhana

Tugas: Cetak angka dari 1 hingga 5 menggunakan loop.

Solusi:

```
Python Code

for i in range(1, 6):

    print(i)
```

Latihan 9: Perhitungan Faktorial

Tugas: Hitung faktorial dari angka yang dimasukkan oleh pengguna.

Solusi:

```
Python Code

angka = int(input("Masukkan angka untuk menghitung faktorial: "))

faktorial = 1

for i in range(1, angka + 1):

    faktorial *= i

print("Faktorial dari", angka, "adalah:", faktorial)
```

Latihan 10: Fungsi Dasar

Tugas: Buat fungsi yang menerima dua angka sebagai input dan mengembalikan hasil kali mereka.

Solusi:

```
Python Code

def kali(x, y):

    return x * y



angka1 = float(input("Masukkan angka pertama: "))

angka2 = float(input("Masukkan angka kedua: "))

hasil = kali(angka1, angka2)

print("Hasil kali adalah:", hasil)
```

Kesimpulan

Dalam tutorial ini, kita telah membahas dasar-dasar sintaks Python, termasuk deklarasi variabel, tipe data, operator, dan operasi input/output. Dengan berlatih pada latihan yang diberikan, mahasiswa dapat memperkuat pemahaman mereka tentang konsep-konsep fundamental ini. Menguasai dasar-dasar ini akan menjadi fondasi untuk topik pemrograman yang lebih lanjut.

Modul 3: Pernyataan Kondisional

Pendahuluan

Bab ini, akan dipelajari **pernyataan kondisional** dalam Bahasa Pemrograman Python, yaitu sebuah konsep penting untuk mengontrol alur program. Pernyataan kondisional ini memungkinkan program untuk membuat keputusan berdasarkan kondisi tertentu dan memungkinkan eksekusi kode yang dinamis dan fleksibel. Bab ini dirancang untuk mahasiswa yang mungkin pernah belajar atau belum memiliki memiliki pengetahuan tentang Bahasa Pemrograman Python.

Pada akhir tutorial ini, Anda akan memahami:

1. Pernyataan **if, elif, dan else**.
2. Cara menggunakan **kondisional bersarang**.
3. Menerapkan **logika kondisional** dengan contoh kehidupan nyata.

Bab ini juga menyediakan **10 latihan** beserta solusinya di akhir bab ini untuk memastikan mahasiswa dapat menerapkan apa yang telah Anda pelajari.

1. Apa itu Pernyataan Kondisional?

Dalam pemrograman, pernyataan kondisional digunakan untuk mengeksekusi kode hanya ketika kondisi tertentu terpenuhi. Python menyediakan konstruksi kondisional berikut:

- **if statement**: Menjalankan blok kode jika kondisi benar.
- **elif statement**: Memeriksa kondisi tambahan jika kondisi sebelumnya salah.
- **else statement**: Menjalankan blok kode jika semua kondisi sebelumnya salah.

Sintaks:

```
Python Code

if kondisi:

    # Blok kode yang dijalankan jika kondisi benar

elif kondisi_lain:

    # Blok kode yang dijalankan jika kondisi sebelumnya salah, tetapi
yang ini benar

else:

    # Blok kode yang dijalankan jika semua kondisi salah
```

2. Contoh: Pernyataan Kondisional Dasar

Mari kita mulai dengan contoh sederhana untuk menggambarkan pernyataan kondisional:

```
Python Code

# Ini adalah pernyataan kondisional sederhana untuk memeriksa apakah
suatu angka positif, negatif, atau nol.

number = int(input("Masukkan sebuah angka: "))  # Mengambil input dari
pengguna dan mengubahnya menjadi integer

if number > 0:

    print("Angka tersebut positif.")

elif number < 0:

    print("Angka tersebut negatif.")

else:

    print("Angka tersebut nol.")
```

Dalam contoh ini, program memeriksa apakah angka tersebut lebih besar dari, kurang dari, atau sama dengan nol. Berdasarkan kondisi, ia akan mencetak pesan yang sesuai.

3. Pernyataan Kondisional Bersarang

Anda dapat meletakkan satu pernyataan `if` di dalam pernyataan lain untuk membuat logika pengambilan keputusan yang lebih kompleks.

```
Python Code

# Contoh pernyataan kondisional bersarang

age = int(input("Masukkan umur Anda: "))

if age >= 18:

    if age >= 60:

        print("Anda adalah warga senior.")

    else:

        print("Anda adalah orang dewasa.")

else:

    print("Anda adalah anak di bawah umur.")
```

Di sini, setelah memeriksa apakah seseorang berusia 18 tahun atau lebih, pemeriksaan lain dilakukan untuk melihat apakah mereka berusia 60 tahun atau lebih, menciptakan percabangan kondisi yang lebih terperinci.

4. Operator Logika dalam Kondisional

Anda dapat menggabungkan beberapa kondisi menggunakan **operator logika** seperti `and`, `or`, dan `not`.

- `and`: Mengembalikan `True` jika kedua kondisi benar.
- `or`: Mengembalikan `True` jika setidaknya satu kondisi benar.

- `not`: Membalik nilai kebenaran dari kondisi.

```
Python Code

# Contoh menggunakan operator logika

temp = int(input("Masukkan suhu dalam Celcius: "))

if temp > 20 and temp < 30:

    print("Cuacanya hangat.")

elif temp <= 20:

    print("Cuacanya dingin.")

else:

    print("Cuacanya panas.")
```

Latihan

Berikut adalah 10 latihan untuk Anda praktikkan. Setiap latihan termasuk solusi di akhir tutorial.

Latihan 1:

Buatlah program untuk memeriksa apakah sebuah angka adalah genap atau ganjil.

Latihan 2:

Buatlah program yang menerima input tahun dan memeriksa apakah tahun tersebut adalah tahun kabisat.

Latihan 3:

Buatlah program untuk menentukan angka terbesar dari tiga angka.

Latihan 4:

Buatlah program untuk mengklasifikasikan nilai siswa berdasarkan skor mereka (A, B, C, D, atau F).

Latihan 5:

Buatlah program untuk memeriksa apakah suatu karakter adalah huruf vokal atau konsonan.

Latihan 6:

Buatlah program yang menerima dua angka sebagai input dan memeriksa apakah keduanya dapat dibagi oleh 5.

Latihan 7:

Buatlah program yang memeriksa apakah sebuah string adalah palindrome.

Latihan 8:

Buatlah program yang menerima input kata sandi dan memeriksa apakah memenuhi kriteria berikut: minimal 8 karakter, mengandung huruf besar dan kecil, serta menyertakan angka.

Latihan 9:

Buatlah program untuk mengklasifikasikan seseorang berdasarkan umur mereka sebagai anak-anak, remaja, dewasa, atau warga senior.

Latihan 10:

Buatlah program yang menerima tinggi seseorang (dalam cm) sebagai input dan mengkategorikan mereka sebagai pendek, rata-rata, atau tinggi.

Solusi

Solusi 1:

```
Python Code
# Memeriksa apakah angka genap atau ganjil

num = int(input("Masukkan sebuah angka: "))
if num % 2 == 0:
    print("Angka tersebut genap.")
else:
    print("Angka tersebut ganjil.")
```

Solusi 2:

```
Python Code
# Memeriksa apakah suatu tahun adalah tahun kabisat

year = int(input("Masukkan sebuah tahun: "))
if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
    print(f"{year} adalah tahun kabisat.")
else:
    print(f"{year} bukan tahun kabisat.")
```

Solusi 3:

```
Python Code
# Menemukan angka terbesar dari tiga angka

num1 = int(input("Masukkan angka pertama: "))
num2 = int(input("Masukkan angka kedua: "))
num3 = int(input("Masukkan angka ketiga: "))
```

```
if num1 >= num2 and num1 >= num3:
    print(f"Angka terbesar adalah {num1}")
elif num2 >= num1 and num2 >= num3:
    print(f"Angka terbesar adalah {num2}")
else:
    print(f"Angka terbesar adalah {num3}")
```

Solusi 4:

```
Python Code
# Mengklasifikasikan nilai siswa berdasarkan skor

score = float(input("Masukkan skor Anda: "))

if score >= 90:
    grade = 'A'
elif score >= 80:
    grade = 'B'
elif score >= 70:
    grade = 'C'
elif score >= 60:
    grade = 'D'
else:
    grade = 'F'

print(f"Nilai Anda adalah {grade}.")
```

Penjelasan: Program ini menerima skor sebagai input dan menentukan grade (A, B, C, D, atau F) berdasarkan rentang skor yang ditentukan menggunakan pernyataan `if`, `elif`, dan `else`.

Solusi 5:

```
Python Code
# Memeriksa apakah karakter adalah vokal atau konsonan

char = input("Masukkan sebuah karakter: ").lower()

vowels = ['a', 'e', 'i', 'o', 'u']

if char in vowels:
    print(f"'{char}' adalah huruf vokal.")
elif char.isalpha():
    print(f"'{char}' adalah huruf konsonan.")
else:
    print("Input bukan huruf alfabet.")
```

Penjelasan: Program ini memeriksa apakah karakter yang dimasukkan adalah vokal atau konsonan. Pertama, karakter diubah menjadi huruf kecil untuk memudahkan pemeriksaan. Kemudian, menggunakan `if` untuk memeriksa apakah karakter tersebut ada dalam daftar vokal. Jika tidak, dan karakter tersebut adalah huruf alfabet, maka diklasifikasikan sebagai konsonan. Jika tidak, program menyatakan bahwa input bukan huruf alfabet.

Solusi 6:

```
Python Code
# Memeriksa apakah dua angka dapat dibagi oleh 5

num1 = int(input("Masukkan angka pertama: "))
num2 = int(input("Masukkan angka kedua: "))

if num1 % 5 == 0 and num2 % 5 == 0:
    print("Kedua angka dapat dibagi oleh 5.")
else:
    print("Salah satu atau kedua angka tidak dapat dibagi oleh 5.")
```

Penjelasan: Program ini menerima dua angka sebagai input dan memeriksa apakah kedua angka tersebut habis dibagi oleh 5 menggunakan operator logika `and`. Jika kedua kondisi terpenuhi, program mencetak bahwa kedua angka dapat dibagi oleh 5. Jika tidak, program menyatakan sebaliknya.

Solusi 7:

```
Python Code
# Memeriksa apakah sebuah string adalah palindrome

string = input("Masukkan sebuah kata atau kalimat: ").replace(" ", "").lower()

if string == string[::-1]:
    print("String tersebut adalah palindrome.")
else:
    print("String tersebut bukan palindrome.")
```

Penjelasan: Program ini memeriksa apakah sebuah string adalah palindrome, yaitu kata atau kalimat yang dibaca sama dari depan maupun belakang. Pertama, program menghapus spasi dan mengubah semua huruf menjadi huruf kecil untuk memastikan pemeriksaan yang akurat. Kemudian, menggunakan slicing `[::-1]` untuk membalik string dan membandingkannya dengan string asli.

Solusi 8:

```
Python Code
# Memeriksa apakah kata sandi memenuhi kriteria keamanan

password = input("Masukkan kata sandi: ")

is_valid = True
errors = []

# Memeriksa panjang kata sandi
```

```
if len(password) < 8:
    is_valid = False
    errors.append("Kata sandi harus minimal 8 karakter.")

# Memeriksa adanya huruf besar
if not any(char.isupper() for char in password):
    is_valid = False
    errors.append("Kata sandi harus mengandung huruf besar.")

# Memeriksa adanya huruf kecil
if not any(char.islower() for char in password):
    is_valid = False
    errors.append("Kata sandi harus mengandung huruf kecil.")

# Memeriksa adanya angka
if not any(char.isdigit() for char in password):
    is_valid = False
    errors.append("Kata sandi harus mengandung angka.")

if is_valid:
    print("Kata sandi memenuhi semua kriteria.")
else:
    print("Kata sandi tidak valid:")
    for error in errors:
        print(f"- {error}")
```

Penjelasan: Program ini memeriksa apakah kata sandi yang dimasukkan memenuhi kriteria keamanan: minimal 8 karakter, mengandung huruf besar dan kecil, serta menyertakan angka. Program menggunakan fungsi `any()` bersama dengan generator expressions untuk memeriksa keberadaan karakter yang memenuhi kriteria tersebut. Jika kata sandi tidak

memenuhi salah satu kriteria, program akan mencantumkan alasan mengapa kata sandi tersebut tidak valid.

Solusi 9:

```
Python Code
# Mengklasifikasikan seseorang berdasarkan umur

age = int(input("Masukkan umur Anda: "))

if age < 13:
    classification = 'Anak-anak'
elif 13 <= age < 20:
    classification = 'Remaja'
elif 20 <= age < 60:
    classification = 'Dewasa'
else:
    classification = 'Warga senior'

print(f"Anda diklasifikasikan sebagai {classification}.")
```

Penjelasan: Program ini menerima umur sebagai input dan mengklasifikasikan seseorang sebagai anak-anak, remaja, dewasa, atau warga senior berdasarkan rentang umur yang ditentukan menggunakan pernyataan `if`, `elif`, dan `else`.

Solusi 10:

```
Python Code
# Mengkategorikan tinggi badan seseorang

height_cm = float(input("Masukkan tinggi badan Anda (dalam cm): "))

if height_cm < 160:
```

```
    category = 'Pendek'
elif 160 <= height_cm <= 175:
    category = 'Rata-rata'
else:
    category = 'Tinggi'

print(f"Tinggi badan Anda dikategorikan sebagai {category}.")
```

Penjelasan: Program ini mengkategorikan tinggi badan seseorang sebagai pendek, rata-rata, atau tinggi berdasarkan input tinggi dalam sentimeter. Rentang kategori dapat disesuaikan sesuai kebutuhan. Program menggunakan pernyataan `if`, `elif`, dan `else` untuk menentukan kategori yang sesuai.

Kesimpulan

Memahami pernyataan kondisional adalah kunci untuk menulis program Python yang efektif. Dengan menguasai konsep ini, Anda dapat mengontrol alur program dan menangani berbagai situasi secara dinamis. Latihan-latihan di atas diharapkan dapat membantu Anda memperkuat pemahaman tentang penggunaan pernyataan kondisional dalam berbagai konteks. Selalu ikuti **praktik penulisan kode terbaik** agar kode Anda tetap bersih, mudah dibaca, dan efisien.

Modul 4: Pengulangan dalam Python

Pengulangan adalah konsep dasar dalam pemrograman yang memungkinkan untuk mengeksekusi sekelompok kode berulang kali berdasarkan kondisi tertentu. Dalam Python, pengulangan biasanya dicapai menggunakan loop `for` dan `while`. Memahami cara menggunakan loop sangat penting untuk menulis kode yang efisien dan efektif, terutama saat menangani koleksi data atau melakukan tugas berulang.

Jenis-jenis Loop di Python

1. Loop For

Loop `for` dalam Python melakukan iterasi pada sebuah urutan (seperti daftar, tuple, atau string) dan mengeksekusi sekelompok kode untuk setiap item dalam urutan tersebut. Sintaks dasar adalah:

```
Python Code

for variabel in urutan:

    # Kode yang dieksekusi
```

Contoh:

```
Python Code
# Menggunakan loop for untuk iterasi melalui daftar angka
angka = [1, 2, 3, 4, 5]
for num in angka:
    print(f"Angka saat ini: {num}")
```

2. Loop While

Loop `while` terus mengeksekusi sekelompok kode selama kondisi tertentu bernilai `True`. Sintaks untuk loop while adalah:

```
Python Code

while kondisi:

    # Kode yang dieksekusi
```

Contoh:

```
Python Code
# Menggunakan loop while untuk hitung mundur dari 5 ke 1
hitung = 5
while hitung > 0:
    print(f"Hitung mundur: {hitung}")
    hitung -= 1  # Mengurangi hitungan
```

Latihan

Berikut adalah sepuluh latihan yang memperkuat konsep pengulangan dalam Python. Setiap latihan mencakup solusi untuk membantu memahami implementasinya.

Latihan 1: Jumlah dari Sebuah Daftar

Tugas: Tulis program yang menghitung jumlah semua angka dalam sebuah daftar menggunakan loop `for`.

Solusi:

```
Python Code

angka = [10, 20, 30, 40, 50]

total = 0
```

```
for num in angka:
    total += num
print(f"Jumlah total: {total}")
```

Latihan 2: Cetak Angka Genap

Tugas: Gunakan loop `for` untuk mencetak semua angka genap dari 1 hingga 20.

Solusi:

```
Python Code
for num in range(1, 21):
    if num % 2 == 0:
        print(num)
```

Latihan 3: Faktorial dari Sebuah Angka

Tugas: Hitung faktorial dari sebuah angka yang diberikan menggunakan loop `while`.

Solusi:

```
Python Code
n = 5
faktorial = 1
while n > 1:
    faktorial *= n
    n -= 1
print(f"Faktorial: {faktorial}")
```

Latihan 4: Hitung Mundur dengan While Loop

Tugas: Tulis program yang menghitung mundur dari 10 ke 1 dan mencetak "Liftoff!" di akhir.

Solusi:

```
Python Code

hitung = 10

while hitung > 0:

    print(hitung)

    hitung -= 1

print("Liftoff!")
```

Latihan 5: Membalik String

Tugas: Gunakan loop `for` untuk mencetak sebuah string secara terbalik.

Solusi:

```
Python Code

teks = "Halo, Dunia!"

teks_terbalik = ""

for char in teks:

    teks_terbalik = char + teks_terbalik

print(teks_terbalik)
```

Latihan 6: Mencari Angka Maksimum

Tugas: Tulis program yang mencari angka maksimum dalam sebuah daftar menggunakan loop `for`.

Solusi:

```
Python Code

angka = [15, 22, 8, 19, 35]

angka_maksimum = angka[0]

for num in angka:

    if num > angka_maksimum:

        angka_maksimum = num

print(f"Angka maksimum: {angka_maksimum}")
```

Latihan 7: Deret Fibonacci

Tugas: Hasilkan deret Fibonacci hingga angka tertentu menggunakan loop `while`.

Solusi:

```
Python Code

n = 10

a, b = 0, 1

count = 0

while count < n:

    print(a)

    a, b = b, a + b

    count += 1
```

Latihan 8: Jumlah Angka Ganjil

Tugas: Hitung jumlah semua angka ganjil dari 1 hingga 100 menggunakan loop `for`.

Solusi:

```
Python Code

total = 0

for num in range(1, 101):

    if num % 2 != 0:

        total += num

print(f"Jumlah angka ganjil: {total}")
```

Latihan 9: Menghitung Vokal

Tugas: Tulis program yang menghitung jumlah vokal dalam sebuah string.

Solusi:

```
Python Code

teks = "Pemrograman itu menyenangkan!"

vokal = "aeiouAEIOU"

count = 0

for char in teks:

    if char in vokal:

        count += 1

print(f"Jumlah vokal: {count}")
```

Latihan 10: Tabel Perkalian

Tugas: Hasilkan tabel perkalian untuk angka tertentu.

Solusi:

```
Python code

angka = 5

for i in range(1, 11):

    print(f"{angka} x {i} = {angka * i}")
```

Kesimpulan

Pengulangan adalah konsep penting dalam pemrograman Python yang memungkinkan eksekusi kode secara efisien dan manipulasi data. Dengan menguasai loop `for` dan `while`, Anda dapat melakukan berbagai tugas mulai dari iterasi sederhana hingga penanganan data yang kompleks. Latihan yang disediakan di atas bertujuan untuk memperkuat pemahaman Anda tentang pengulangan dalam Python.

Modul 5 dan 6: Proyek Sederhana untuk Pemula

Selamat datang di modul pemrograman Python! Modul ini dirancang untuk memperkenalkan Bahasa Pemrograman Python melalui latihan praktis. Python adalah bahasa pemrograman yang serbaguna dan banyak digunakan di berbagai bidang, termasuk pengembangan web, analisis data, kecerdasan buatan, dan lainnya. Dalam modul ini, mahasiswa akan belajar dengan membuat proyek sederhana yang akan membantu untuk mengembangkan keterampilan pemrograman.

Tujuan Tutorial Ini

1. **Memahami Dasar-Dasar Python**: Menjadi familiar dengan sintaks dan struktur Python.
2. **Membangun Proyek Sederhana**: Menerapkan pengetahuan Anda dengan membuat program yang fungsional.
3. **Mengembangkan Keterampilan Pemecahan Masalah**: Belajar memecah masalah menjadi langkah-langkah yang dapat dikelola.
4. **Mengikuti Praktik Terbaik**: Menulis kode yang bersih, terdokumentasi dengan baik, dan mudah dipelihara.

Proyek 1: Kalkulator

Tujuan Proyek

Membuat kalkulator sederhana yang dapat melakukan operasi aritmatika dasar: penjumlahan, pengurangan, perkalian, dan pembagian.

Panduan Langkah demi Langkah

1. **Definisikan Fungsi**: Buat fungsi untuk setiap operasi.
2. **Input Pengguna**: Minta pengguna untuk memasukkan angka dan operasi yang diinginkan.
3. **Tampilkan Hasil**: Tunjukkan hasil perhitungan.

Contoh Kode

```
Python Code
# Program kalkulator sederhana

def add(x, y):
    """Mengembalikan jumlah x dan y."""
    return x + y

def subtract(x, y):
    """Mengembalikan selisih x dan y."""
    return x - y

def multiply(x, y):
    """Mengembalikan hasil kali x dan y."""
    return x * y

def divide(x, y):
    """Mengembalikan hasil bagi x dan y. Memeriksa pembagian dengan
nol."""
    if y == 0:
        return "Error! Pembagian dengan nol."
    return x / y

# Input pengguna
print("Pilih operasi:")
print("1. Tambah")
```

```
print("2. Kurang")
print("3. Kali")
print("4. Bagi")

choice = input("Masukkan pilihan (1/2/3/4): ")
num1 = float(input("Masukkan angka pertama: "))
num2 = float(input("Masukkan angka kedua: "))

if choice == '1':
    print(f"{num1} + {num2} = {add(num1, num2)}")
elif choice == '2':
    print(f"{num1} - {num2} = {subtract(num1, num2)}")
elif choice == '3':
    print(f"{num1} * {num2} = {multiply(num1, num2)}")
elif choice == '4':
    print(f"{num1} / {num2} = {divide(num1, num2)}")
else:
    print("Input tidak valid!")
```

Proyek 2: Permainan Tebak Angka

Tujuan Proyek

Membuat permainan sederhana di mana pengguna harus menebak angka yang dihasilkan secara acak.

Panduan Langkah demi Langkah

1. **Hasilkan Angka Acak**: Gunakan modul `random` untuk menghasilkan angka.
2. **Tebakan Pengguna**: Izinkan pengguna untuk memasukkan tebakan mereka.
3. **Berikan Umpan Balik**: Indikasikan apakah tebakan terlalu tinggi, terlalu rendah, atau benar.

Contoh Kode

```
Python Code
import random


def guess_the_number():
    """Permainan tebak angka sederhana di mana pengguna harus menebak angka."""
    number_to_guess = random.randint(1, 100)
    guess = None

    print("Tebak angka antara 1 dan 100.")

    while guess != number_to_guess:
        guess = int(input("Masukkan tebakan Anda: "))
        if guess < number_to_guess:
            print("Terlalu rendah! Coba lagi.")
        elif guess > number_to_guess:
            print("Terlalu tinggi! Coba lagi.")
        else:
            print("Selamat! Anda telah menebak angka.")

# Mulai permainan
guess_the_number()
```

Proyek 3: Aplikasi Daftar Tugas

Tujuan Proyek

Membuat aplikasi daftar tugas berbasis teks sederhana dimana pengguna dapat menambah dan menghapus tugas.

Panduan Langkah demi Langkah

1. **Simpan Tugas**: Gunakan daftar untuk menyimpan tugas.
2. **Input Pengguna**: Izinkan pengguna untuk menambah atau menghapus tugas.
3. **Tampilkan Tugas**: Tunjukkan daftar tugas saat ini.

Contoh Kode

```
Python Code
def todo_list():
    """Aplikasi daftar tugas sederhana."""
    tasks = []

    while True:
        print("\nDaftar Tugas:")
        for i, task in enumerate(tasks, start=1):
            print(f"{i}. {task}")

        print("\nPilihan: [1] Tambah tugas [2] Hapus tugas [3] Keluar")
        choice = input("Masukkan pilihan Anda: ")

        if choice == '1':
            task = input("Masukkan tugas: ")
            tasks.append(task)
            print("Tugas ditambahkan.")
        elif choice == '2':
            task_number = int(input("Masukkan nomor tugas yang ingin
dihapus: "))
```

```
            if 0 < task_number <= len(tasks):
                removed_task = tasks.pop(task_number - 1)
                print(f"Tugas yang dihapus: {removed_task}")
            else:
                print("Nomor tugas tidak valid.")
        elif choice == '3':
            break
        else:
            print("Pilihan tidak valid!")

# Mulai aplikasi daftar tugas
todo_list()
```

Proyek 4: Jam Alarm Sederhana

- **Tujuan**: Membuat jam alarm berbasis teks yang memberi tahu pengguna setelah sejumlah detik tertentu.
- **Konsep Kunci**: Penanganan waktu, input pengguna, dan perulangan.

Proyek 5: Konverter Mata Uang Dasar

- **Tujuan**: Mengonversi antara berbagai mata uang menggunakan nilai tukar statis.
- **Konsep Kunci**: Input pengguna, operasi matematika, dan definisi fungsi.

Proyek 6: Konverter Suhu

- **Tujuan**: Mengkonversi suhu antara Celcius dan Fahrenheit.
- **Konsep Kunci**: Input pengguna dan pernyataan bersyarat.

Proyek 7: Permainan Petualangan Berbasis Teks Sederhana

- **Tujuan**: Membuat permainan berbasis teks dimana pengguna dapat membuat pilihan untuk menjelajahi cerita.
- **Konsep Kunci**: Alur kontrol dan input pengguna.

Proyek 8: Aplikasi Quiz Dasar

- **Tujuan**: Membuat kuis sederhana yang mengajukan pertanyaan pilihan ganda kepada pengguna.
- **Konsep Kunci**: Daftar, kamus, dan perulangan.

Proyek 9: Permainan Batu, Kertas, Gunting

- **Tujuan**: Menerapkan permainan sederhana di mana pengguna dapat bermain melawan komputer.
- **Konsep Kunci**: Pilihan acak dan input pengguna.

Proyek 10: Buku Kontak Sederhana

- **Tujuan**: Membuat aplikasi berbasis teks untuk menyimpan dan mengambil informasi kontak.
- **Konsep Kunci**: Kamus dan fungsi.

Kesimpulan

Selamat! Anda telah menyelesaikan semua modul Python! Anda telah belajar bagaimana membangun proyek sederhana, memahami syntax Python, dan mengembangkan keterampilan pemecahan masalah. Ingatlah untuk mengikuti praktik terbaik dalam perjalanan pengkodean Anda dan terus bereksperimen dengan proyek-proyek baru. Selamat meng-coding dengan menyenangkan!