



Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	71230977
Nama Lengkap	MICHAEL HOSEA
Minggu ke / Materi	05 / STRUKTUR KONTROL PERULANGAN

SAYA MENYATAKAN BAHWA LAPORAN PRAKTIKUM INI SAYA BUAT DENGAN USAHA SENDIRI TANPA MENGGUNAKAN BANTUAN ORANG LAIN. SEMUA MATERI YANG SAYA AMBIL DARI SUMBER LAIN SUDAH SAYA CANTUMKAN SUMBERNYA DAN TELAH SAYA TULIS ULANG DENGAN BAHASA SAYA SENDIRI.

SAYA SANGGUP MENERIMA SANKSI JIKA MELAKUKAN KEGIATAN PLAGIASI, TERMASUK SANKSI TIDAK LULUS MATA KULIAH INI.

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS KRISTEN DUTA WACANA
YOGYAKARTA
2024

BAGIAN 1: MATERI MINGGU INI (40%)

Pada bagian ini, tuliskan kembali semua materi yang telah anda pelajari minggu ini. Sesuaikan penjelasan anda dengan urutan materi yang telah diberikan di saat praktikum. Penjelasan anda harus dilengkapi dengan contoh, gambar/ilustrasi, contoh program (source code) dan outputnya. Idealnya sekitar 5-6 halaman.

MATERI 1 : Perulangan

Perulangan (atau loop) dalam pemrograman adalah proses pengulangan atau eksekusi berulang dari serangkaian pernyataan atau blok kode tertentu selama kondisi tertentu terpenuhi. Perulangan digunakan ketika kita perlu melakukan tugas yang sama beberapa kali atau ketika kita ingin menjalankan suatu blok kode secara berulang berdasarkan kondisi tertentu. Ini memungkinkan penggunaan kode yang lebih efisien dan tersusun secara terstruktur, serta memungkinkan pemrosesan data secara otomatis dalam jumlah besar. Dengan perulangan, kita dapat menghindari penulisan kode yang berulang-ulang, meningkatkan kinerja program, dan membuat program lebih dinamis. Pada pemrograman Python, perulangan dapat dilakukan dengan menggunakan “for”, “while”, dan cara rekursif.

MATERI 2 : Bentuk Perulangan “for”

Perulangan for digunakan dalam Python ketika kita memiliki situasi di mana kita ingin melakukan suatu tugas berulang kali, terutama dalam kondisi-kondisi berikut:

1. Jumlah perulangan sudah diketahui sejak awal.
2. Perulangan terjadi karena operasi yang sama pada suatu rentang data atau rentang nilai.

Fungsi range() sangat dibutuhkan ketika kita ingin melakukan perulangan for pada rentang tertentu dalam Python. Fungsi range() menghasilkan urutan bilangan bulat secara berurutan, yang memudahkan dalam membuat rentang nilai untuk iterasi. Bentuk umum dari fungsi “range()” adalah sebagai berikut:

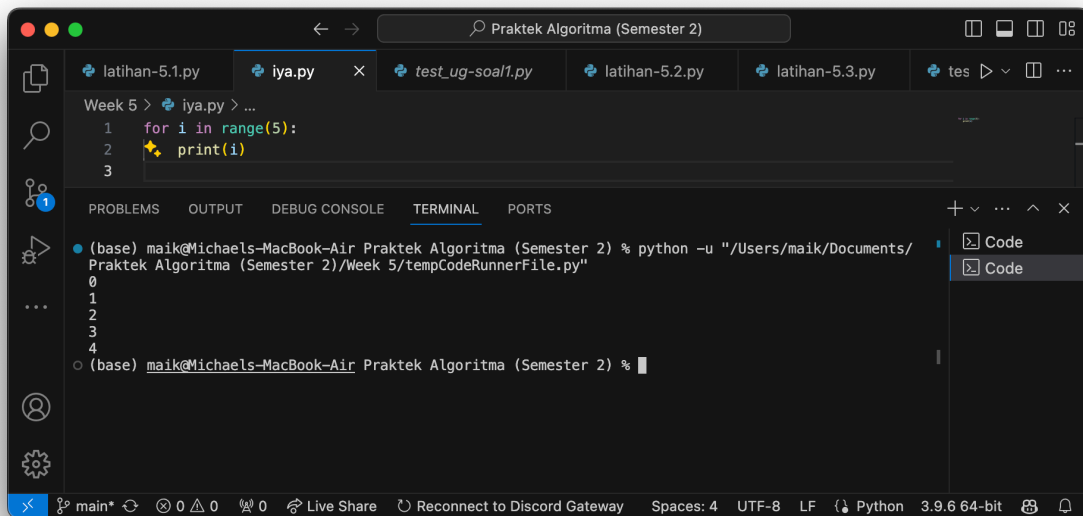
range(start, stop, step)

- start: Nilai awal dari rentang (opsional, nilai default adalah 0).
- stop: Nilai akhir dari rentang (nilai ini tidak akan disertakan dalam urutan).
- step: Jarak antara setiap dua nilai berturut-turut dalam urutan (opsional, nilai default adalah 1).

Contohnya beserta outputnya:

- Membuat Iterasi dari 0 hingga 4:

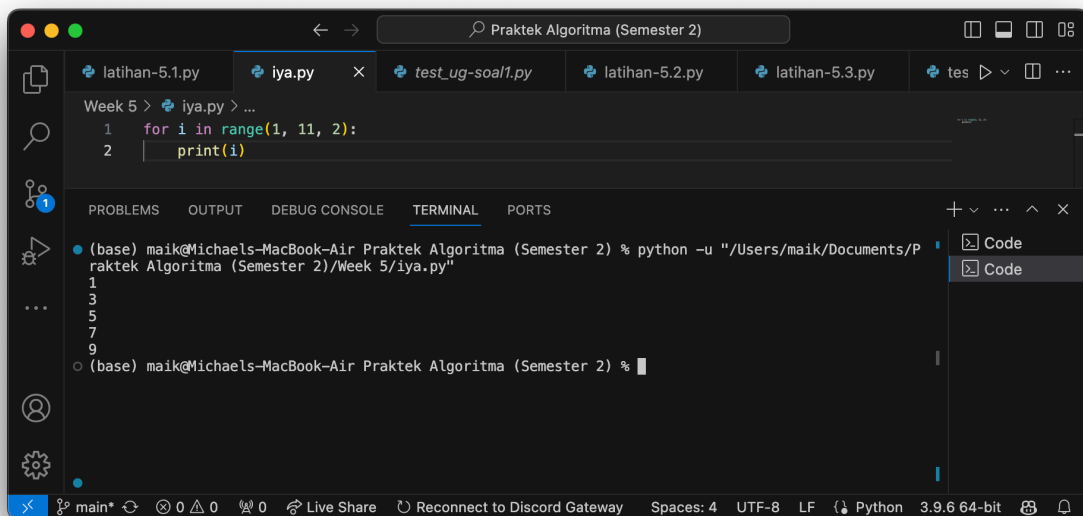
```
for i in range(5):  
    print(i)
```



Gambar 1: Membuat Iterasi dari 0 hingga 4

- Iterasi dari 1 hingga 10 dengan lompatan 2:

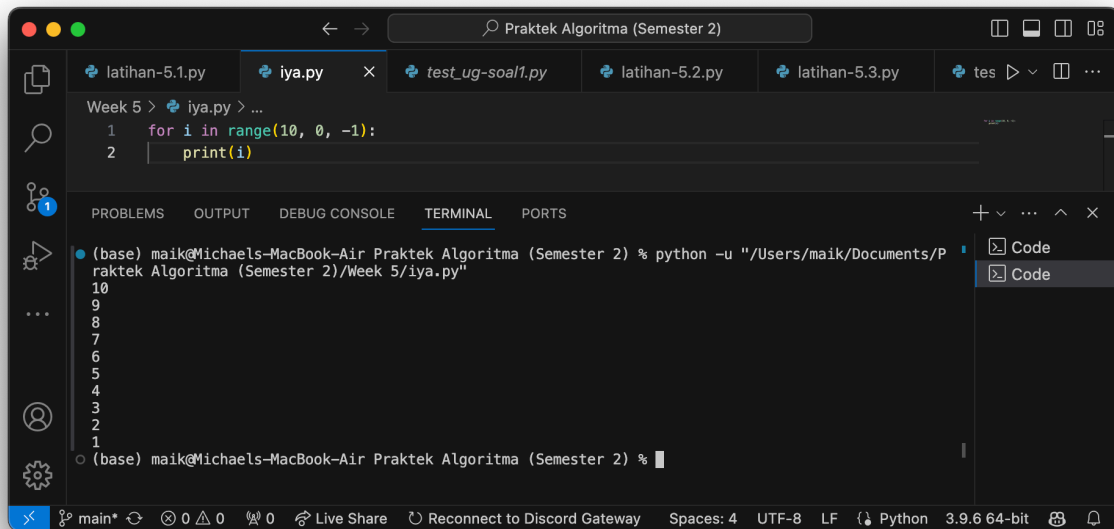
```
- for i in range(1, 11, 2):  
-     print(i)
```



Gambar 2: Membuat Iterasi dari 1 hingga 10 dengan lompatan 2

- Iterasi mundur dari 10 hingga 1:

```
- for i in range(10, 0, -1):  
-     print(i)
```



Gambar 3: Membuat Iterasi mundur dari 10 hingga 1

Materi 3: Bentuk Perulangan “while”

Secara umum, bentuk perulangan while dalam Python adalah sebagai berikut:

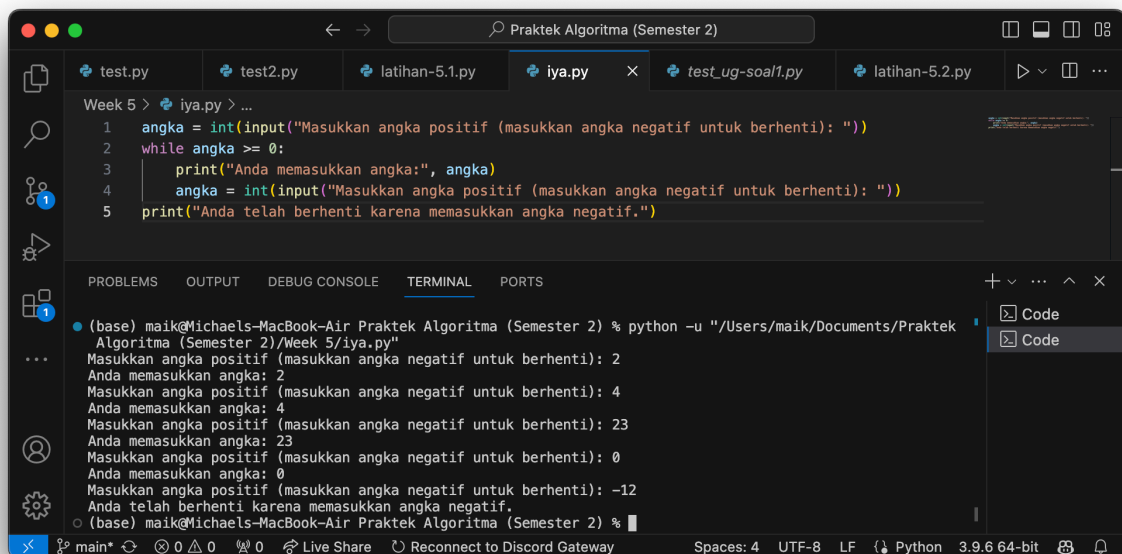
```
<start>
while <stop>:
    operation
    operation
    <step (optional)>
```

- <start>: Awal dari perulangan, yang dapat berupa inisialisasi variabel atau kondisi awal lainnya sebelum memasuki perulangan.
- <stop>: Kondisi yang dievaluasi sebelum setiap iterasi perulangan. Perulangan akan terus berlanjut selama kondisi ini benar (True). Ketika kondisi ini menjadi salah (False), perulangan akan berhenti.
- <operation>: Serangkaian perintah atau operasi yang akan dieksekusi di setiap iterasi perulangan.
- <step> (opsional): Langkah yang diambil setelah setiap iterasi perulangan. Ini sering digunakan untuk mengubah nilai variabel atau kondisi yang digunakan dalam kondisi perulangan.

Perulangan while biasanya digunakan ketika jumlah perulangan tidak diketahui sebelumnya, atau ketika perulangan harus terus berlanjut sampai kondisi tertentu terpenuhi. Ini memungkinkan Anda untuk menjalankan serangkaian perintah berulang kali selama kondisi yang diberikan terus terpenuhi, tanpa harus menentukan jumlah iterasi di awal.

Contoh umum penggunaan perulangan while dalam kondisi di mana jumlah perulangan tidak diketahui sebelumnya adalah ketika kita meminta masukan pengguna dan perlu mengeksekusi serangkaian perintah sampai pengguna memberikan input yang menandakan akhir dari iterasi. Berikut adalah contoh sederhana:

```
angka = int(input("Masukkan angka positif (masukkan angka negatif untuk berhenti): "))
while angka >= 0:
    print("Anda memasukkan angka:", angka)
    angka = int(input("Masukkan angka positif (masukkan angka negatif untuk berhenti): "))
print("Anda telah berhenti karena memasukkan angka negatif.")
```



The screenshot shows a code editor window titled 'Praktek Algoritma (Semester 2)' with several tabs open. The active tab is 'iya.py', which contains the following Python code:

```
1 angka = int(input("Masukkan angka positif (masukkan angka negatif untuk berhenti): "))
2 while angka >= 0:
3     print("Anda memasukkan angka:", angka)
4     angka = int(input("Masukkan angka positif (masukkan angka negatif untuk berhenti): "))
5 print("Anda telah berhenti karena memasukkan angka negatif.")
```

Below the code editor is a terminal window showing the execution of the script. The output is as follows:

```
(base) maik@Michaels-MacBook-Air Praktek Algoritma (Semester 2) % python -u "/Users/maik/Documents/Praktek Algoritma (Semester 2)/Week 5/iya.py"
Masukkan angka positif (masukkan angka negatif untuk berhenti): 2
Anda memasukkan angka: 2
Masukkan angka positif (masukkan angka negatif untuk berhenti): 4
Anda memasukkan angka: 4
Masukkan angka positif (masukkan angka negatif untuk berhenti): 23
Anda memasukkan angka: 23
Masukkan angka positif (masukkan angka negatif untuk berhenti): 0
Anda memasukkan angka: 0
Masukkan angka positif (masukkan angka negatif untuk berhenti): -12
Anda telah berhenti karena memasukkan angka negatif.
(base) maik@Michaels-MacBook-Air Praktek Algoritma (Semester 2) %
```

Gambar 4: Membuat perulangan dengan menggunakan "while"

Dalam contoh ini, perulangan while akan terus berjalan selama pengguna memasukkan angka positif. Ketika pengguna memasukkan angka negatif, kondisi `angka >= 0` akan menjadi False, dan perulangan akan berhenti.

Perulangan while sangat berguna ketika Anda perlu menjalankan serangkaian perintah berulang kali tanpa mengetahui jumlah iterasi di awal atau ketika iterasi harus terus berlanjut sampai kondisi tertentu terpenuhi. Namun, Anda harus berhati-hati agar tidak terjebak dalam perulangan tak berujung jika kondisi yang diberikan tidak pernah menjadi False.

Materi 3: Penggunaan “break” dan “continue”

Break dan continue adalah dua pernyataan kontrol yang sering digunakan dalam perulangan di Python untuk mengatur alur eksekusi perulangan. Secara umum break digunakan untuk menghentikan perulangan, sedangkan continue digunakan untuk melanjutkan perulangan ke iterasi berikutnya. Berikut adalah penjelasan tentang keduanya:

1. Break

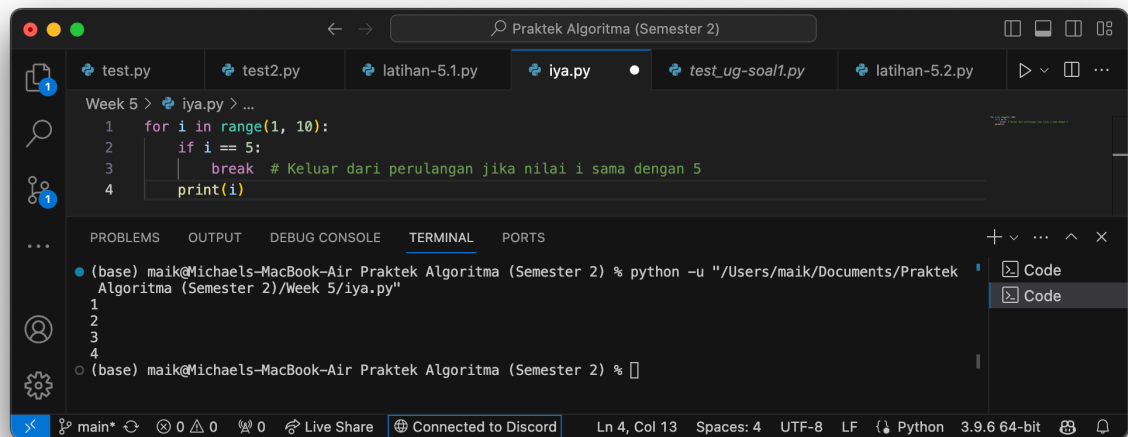
Break adalah pernyataan kontrol yang digunakan untuk menghentikan perulangan secara paksa bahkan jika kondisi perulangan masih terpenuhi. Ketika pernyataan break dieksekusi di dalam sebuah perulangan, program akan keluar dari perulangan dan melanjutkan eksekusi pada baris kode setelah perulangan.

Kapan menggunakan break?

- Ketika Anda ingin menghentikan perulangan secara prematur, bahkan sebelum kondisi perulangan selesai.
- Biasanya digunakan untuk keluar dari perulangan jika suatu kondisi tertentu terpenuhi.

Contoh penggunaan “break” dan juga outputnya:

```
for i in range(1, 10):  
    if i == 5:  
        break # Keluar dari perulangan jika nilai i sama dengan 5  
    print(i)
```



Gambar 5: Contoh penggunaan “break”

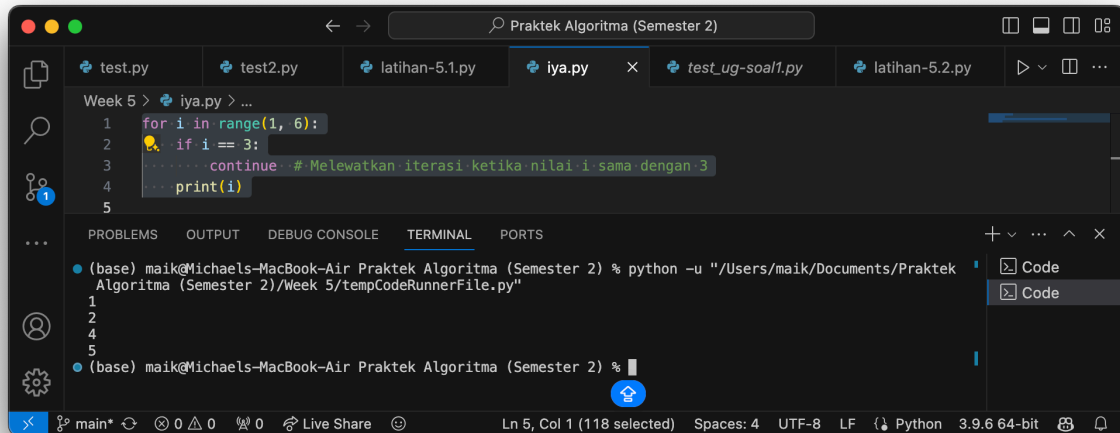
2. Continue

Continue adalah pernyataan kontrol yang digunakan untuk melompati sisa iterasi dalam perulangan saat ini dan melanjutkan ke iterasi berikutnya. Ketika pernyataan continue dieksekusi, program akan mengabaikan perintah-perintah yang ada setelahnya dalam iterasi saat ini, dan akan langsung memulai iterasi berikutnya.

Kapan menggunakan continue?

Ketika Anda ingin melewati iterasi tertentu dalam perulangan berdasarkan kondisi tertentu tanpa menghentikan perulangan secara keseluruhan.

```
for i in range(1, 6):  
    if i == 3:  
        continue # Melewatkan iterasi ketika nilai i sama dengan 3  
    print(i)
```



Gambar 6: Contoh penggunaan "continue"

BAGIAN 2: LATIHAN MANDIRI (60%)

Pada bagian ini anda menuliskan jawaban dari soal-soal Latihan Mandiri yang ada di modul praktikum. Jawaban anda harus disertai dengan source code, penjelasan dan screenshot output.

SOAL 1

Buatlah program yang menerapkan perhitungan perkalian dengan menggunakan penjumlahan. Buatlah fungsi perkalian() dalam program tersebut! Berikut ini adalah beberapa contoh perhitungan yang diharapkan:

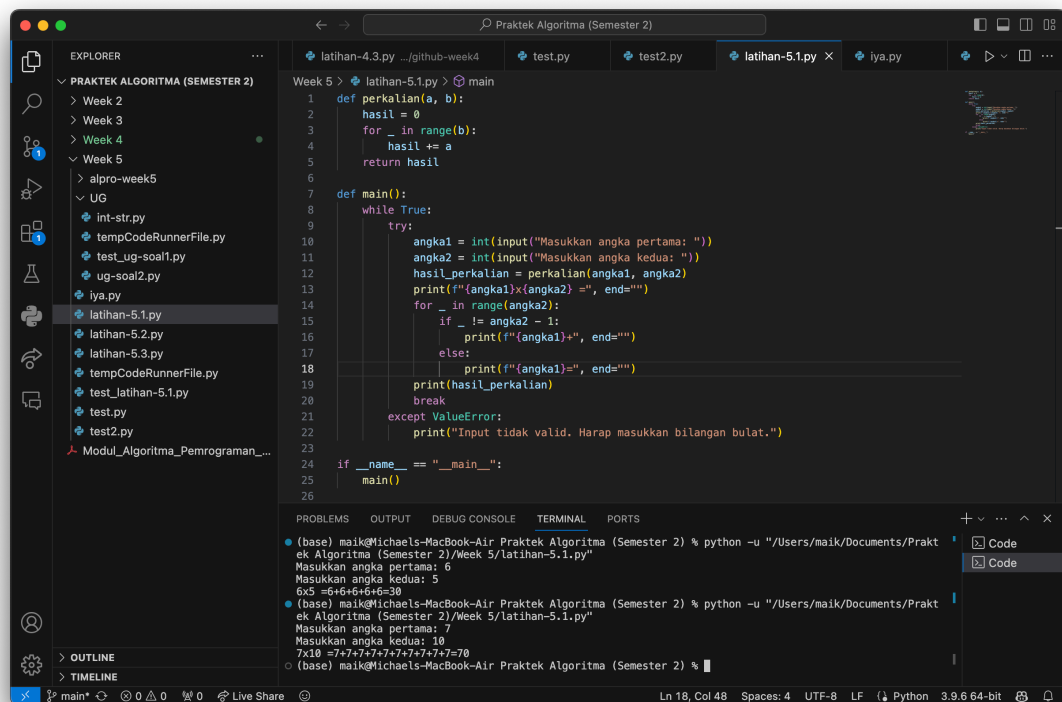
- $6 \times 5 = 5 + 5 + 5 + 5 + 5 + 5 = 30$.
- $7 \times 10 = 10 + 10 + 10 + 10 + 10 + 10 + 10 = 70$.

- Source Code dan Output

```
def perkalian(a, b):
    hasil = 0
    for _ in range(b):
        hasil += a
    return hasil

def main():
    while True:
        try:
            angka1 = int(input("Masukkan angka pertama: "))
            angka2 = int(input("Masukkan angka kedua: "))
            hasil_perkalian = perkalian(angka1, angka2)
            print(f"{angka1}x{angka2} =", end="")
            for _ in range(angka2):
                if _ != angka2 - 1:
                    print(f"{angka1}+", end="")
                else:
                    print(f"{angka1}=", end="")
            print(hasil_perkalian)
            break
        except ValueError:
            print("Input tidak valid. Harap masukkan bilangan bulat.")

if __name__ == "__main__":
    main()
```

Gambar 7: Contoh penggunaan “continue”

- Penjelasan

Berikut adalah penjelasan per baris dari kode yang diberikan:

1. “perkalian(a, b)”: Ini adalah fungsi yang melakukan operasi perkalian antara dua bilangan “a” dan “b”. Cara kerjanya adalah dengan melakukan penjumlahan berulang sebanyak “b” kali, dimana setiap kali “a” ditambahkan ke dalam variabel “hasil”. Hasil akhir dari penjumlahan ini dikembalikan sebagai hasil perkalian.

2. “main()”: Fungsi ini adalah fungsi utama yang mengontrol alur program. Pertama-tama, program memasuki loop “while True”, yang berarti akan terus berjalan sampai ada perintah untuk keluar. Di dalam loop tersebut, program mencoba untuk mengambil input dari pengguna dalam bentuk dua bilangan bulat, “angka1” dan “angka2”. Kemudian, program memanggil fungsi “perkalian(angka1, angka2)” untuk menghitung hasil perkalian dari kedua bilangan tersebut. Setelah itu, program mencetak hasil perkalian secara terperinci, yaitu menampilkan rumus perkalian “angka1 x angka2 = hasil_perkalian”. Jika terjadi kesalahan saat pengguna memasukkan input yang tidak valid (misalnya, input tidak berupa bilangan bulat), program akan menangkap kesalahan tersebut dan memberikan pesan kesalahan kepada pengguna, lalu meminta input ulang. Loop akan terus berlanjut sampai input yang valid diberikan oleh pengguna.

Pernyataan “if __name__ == “__main__”: main()” digunakan untuk memastikan bahwa fungsi “main()” akan dijalankan hanya jika skrip dieksekusi langsung (bukan diimpor sebagai modul ke skrip lain).

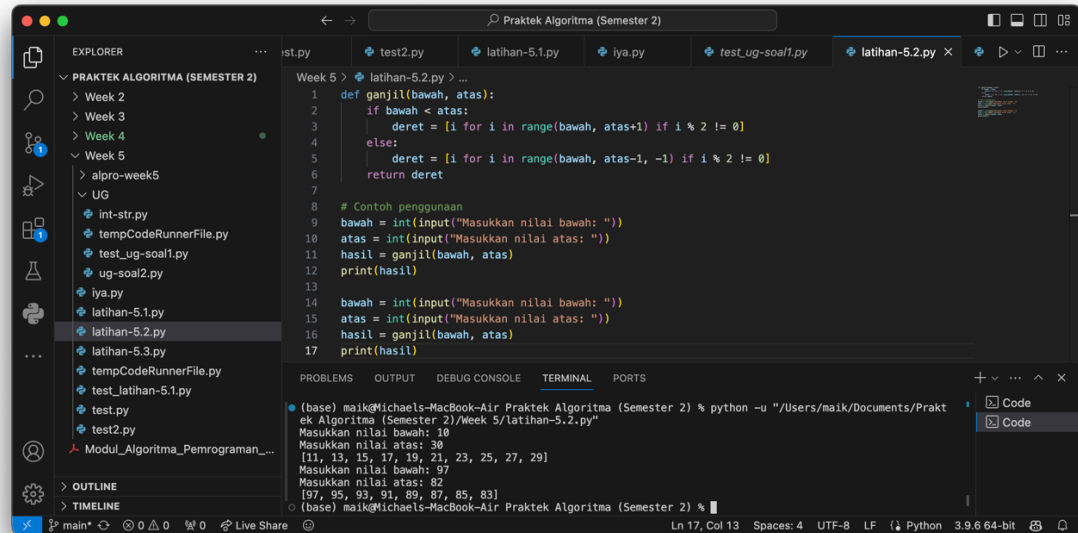
Dengan demikian, fungsi-fungsi dalam kode tersebut mengimplementasikan operasi perkalian, mengambil input dari pengguna, dan menangani kesalahan input.

SOAL 2

Buatlah program yang dapat menampilkan deret bilangan ganjil dari batas bawah dan batas atas yang diberikan oleh pengguna. Jika ternyata batas atas < batas bawah, berarti deret tersebut dimulai dari batas atas, sampai batas bawah (negatif range). Buatlah fungsi ganjil() dalam program tersebut! Berikut ini adalah contoh hasil yang diharapkan:

- bawah = 10, atas = 30. Karena bawah < atas, berarti dari kecil ke besar, maka hasilnya adalah: 11, 13, 15, 17, 19, 21, 23, 25, 27, 29.
- bawah = 97, atas = 82. Karena bawah > atas, berarti dari besar ke kecil, maka hasilnya adalah: 97, 95, 93, 91, 89, 87, 85, 83. Input = 30, 20, 18. Output yang diharapkan = True
- Source Code dan Output

```
def ganjil(bawah, atas):  
    if bawah < atas:  
        deret = [i for i in range(bawah, atas+1) if i % 2 != 0]  
    else:  
        deret = [i for i in range(bawah, atas-1, -1) if i % 2 != 0]  
    return deret  
  
# Contoh penggunaan  
bawah = int(input("Masukkan nilai bawah: "))  
atas = int(input("Masukkan nilai atas: "))  
hasil = ganjil(bawah, atas)  
print(hasil)  
  
bawah = int(input("Masukkan nilai bawah: "))  
atas = int(input("Masukkan nilai atas: "))  
hasil = ganjil(bawah, atas)  
print(hasil)
```



- Penjelasan

Fungsi “ganjil(bawah, atas)” dalam kode tersebut memiliki tujuan untuk menghasilkan deret bilangan ganjil dalam rentang tertentu antara dua nilai, yaitu “bawah” dan “atas”. Berikut adalah penjelasan singkat mengenai fungsi tersebut:

1. Pengecekan Rentang: Pertama-tama, fungsi melakukan pengecekan apakah nilai “bawah” lebih kecil daripada nilai “atas”. Jika iya, maka deret bilangan ganjil akan dibentuk dari nilai “bawah” hingga “atas” (inklusif). Jika tidak, artinya nilai “bawah” lebih besar dari nilai “atas”, maka deret akan dibentuk dari “atas” hingga “bawah” (inklusif), dengan langkah mundur (dari nilai tinggi ke rendah).
2. Pembentukan Deret: Deret bilangan ganjil dibentuk dengan menggunakan list comprehension, dimana setiap bilangan dalam rentang tersebut dicek apakah merupakan bilangan ganjil atau tidak (“i % 2 != 0”). Jika iya, bilangan tersebut dimasukkan ke dalam list “deret”.
3. Pengembalian Deret: Setelah deret bilangan ganjil terbentuk, deret tersebut dikembalikan sebagai output dari fungsi.

Pada bagian "Contoh penggunaan", pengguna diminta untuk memasukkan nilai “bawah” dan “atas”, kemudian fungsi “ganjil()” dipanggil untuk menghasilkan deret bilangan ganjil antara kedua nilai tersebut, dan hasilnya dicetak.

Dengan demikian, fungsi tersebut memungkinkan pengguna untuk dengan mudah memperoleh deret bilangan ganjil dalam rentang tertentu yang ditentukan oleh “bawah” dan “atas”.

SOAL 3

Buatlah sebuah program penghitung nilai Indeks Prestasi Semester (IPS). Input bagi program:

- Jumlah mata kuliah

- Nilai A, B, C, dan D untuk setiap mata kuliah mahasiswa. Diasumsikan sks setiap mata kuliah selalu 3. Kemudian bobot dari masing-masing nilai adalah: A=4, B=3, C=2, D=1. Output program ialah hasil IPS yang didapatkan. Jalannya program seperti pada Gambar 5.7. Tips: Gunakan kontrol percabangan di dalam perulangan.

- Source Code dan Output

```
def hitung_ips(jumlah_matkul):
    total_sks = jumlah_matkul * 3
    total_bobot = 0

    for i in range(1, jumlah_matkul + 1):
        nilai = input(f"Masukkan nilai untuk mata kuliah ke-{i} (A/B/C/D):").upper()

        # Menghitung bobot berdasarkan nilai
        if nilai == 'A':
            bobot = 4
        elif nilai == 'B':
            bobot = 3
        elif nilai == 'C':
            bobot = 2
        elif nilai == 'D':
            bobot = 1
        else:
            print("Nilai tidak valid. Silakan masukkan nilai A, B, C, atau D.")
            return None

        # Menambah total bobot nilai
        total_bobot += bobot * 3 # setiap mata kuliah memiliki 3 SKS

    # Menghitung IPS
    ips = total_bobot / total_sks
    return ips

def main():
    # Input jumlah mata kuliah
    while True:
        try:
            jumlah_matkul = int(input("Masukkan jumlah mata kuliah: "))
            if jumlah_matkul > 0:
                break
            else:
                print("Jumlah mata kuliah harus lebih dari 0.")
        except ValueError:
            print("Masukkan angka yang valid untuk jumlah mata kuliah.")
```

```

# Hitung IPS
ips = hitung_ips(jumlah_matkul)
if ips is not None:
    print(f"Indeks Prestasi Semester (IPS) Anda adalah: {ips:.2f}")

if __name__ == "__main__":
    main()

```

```

Week 5 > lathan-5.3.py >
1 def hitung_ips(jumlah_matkul):
2     total_sks = jumlah_matkul * 3
3     total_bobot = 0
4
5     for i in range(1, jumlah_matkul + 1):
6         nilai = input(f"Masukkan nilai untuk mata kuliah ke-{i} (A/B/C/D): ").upper()
7
8         # Menghitung bobot berdasarkan nilai
9         if nilai == 'A':
10             bobot = 4
11         elif nilai == 'B':
12             bobot = 3
13         elif nilai == 'C':
14             bobot = 2
15         elif nilai == 'D':
16             bobot = 1
17         else:
18             print("Nilai tidak valid. Silakan masukkan nilai A, B, C, atau D.")
19             return None
20
21     # Menambah total bobot nilai
22     total_bobot += bobot * 3 # setiap mata kuliah memiliki 3 SKS
23
24     # Menghitung IPS
25     ips = total_bobot / total_sks
26     return ips
27
28 def main():
29     # Jumlah mata kuliah
30     while True:
31         try:
32             jumlah_matkul = int(input("Masukkan jumlah mata kuliah: "))
33             if jumlah_matkul > 0:
34                 break
35             else:
36                 print("Jumlah mata kuliah harus lebih dari 0.")
37         except ValueError:
38             print("Masukkan angka yang valid untuk jumlah mata kuliah.")
39
40     # Hitung IPS
41     ips = hitung_ips(jumlah_matkul)
42     if ips is not None:
43         print(f"Indeks Prestasi Semester (IPS) Anda adalah: {ips:.2f}")
44
45 if __name__ == "__main__":
46     main()
47
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Masukkan jumlah mata kuliah: 6
Masukkan nilai untuk mata kuliah ke-1 (A/B/C/D): a
Masukkan nilai untuk mata kuliah ke-2 (A/B/C/D): b
Masukkan nilai untuk mata kuliah ke-3 (A/B/C/D): c
Masukkan nilai untuk mata kuliah ke-4 (A/B/C/D): a
Masukkan nilai untuk mata kuliah ke-5 (A/B/C/D): a
Masukkan nilai untuk mata kuliah ke-6 (A/B/C/D): c
Indeks Prestasi Semester (IPS) Anda adalah: 2.67
(base) msi-gtchen1s-MacBook-Air:Praktek Algoritma (Semester 2)

```

- Penjelasan

1. Fungsi “hitung_ips(jumlah_matkul)”: Fungsi ini mengambil jumlah mata kuliah sebagai argumen. Di dalamnya, terdapat loop yang berjalan sebanyak “jumlah_matkul”, di mana setiap iterasi loop meminta pengguna untuk memasukkan nilai (A/B/C/D) untuk setiap mata kuliah. Nilai tersebut kemudian diubah menjadi bobot berdasarkan skala yang ditentukan (A=4, B=3, C=2, D=1). Setiap bobot dikalikan dengan jumlah SKS setiap mata kuliah (dalam kasus ini, diasumsikan setiap mata kuliah memiliki 3 SKS). Total bobot nilai untuk semua mata kuliah dihitung dan disimpan. Setelah loop selesai, IPS dihitung dengan membagi total bobot nilai dengan total SKS dari semua mata kuliah.

2. Fungsi “main()”: Fungsi utama program ini meminta pengguna untuk memasukkan jumlah mata kuliah. Program akan terus meminta input sampai pengguna memasukkan jumlah mata kuliah yang valid (lebih dari 0). Setelah itu, program memanggil fungsi “hitung_ips()” untuk menghitung IPS berdasarkan jumlah mata kuliah yang dimasukkan. Jika IPS telah dihitung dengan sukses, hasilnya akan dicetak.

Dengan kode ini, pengguna dapat menghitung IPS mereka dengan memasukkan nilai untuk setiap mata kuliah dan jumlah mata kuliah yang diambil.