



Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	71230971
Nama Lengkap	James Marvin Santoso
Minggu ke / Materi	06 / Percabangan dan Perulangan Kompleks

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

Struktur Percabangan Kompleks

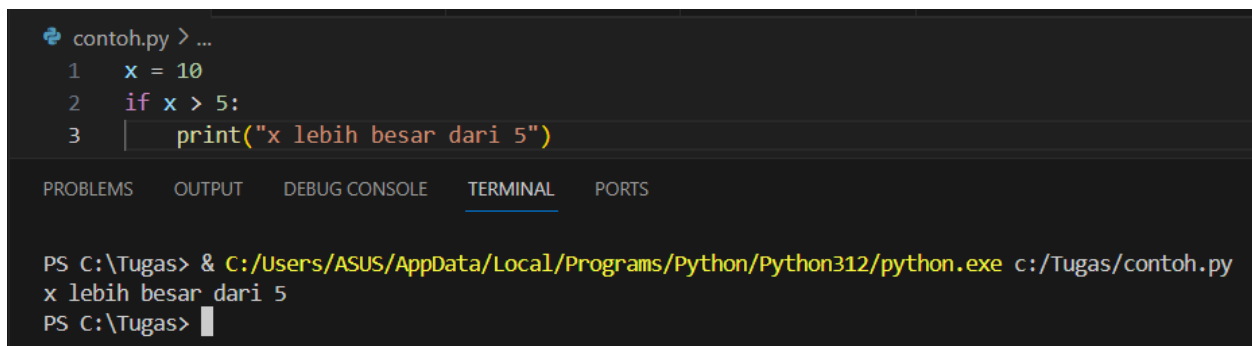
Struktur percabangan kompleks dalam Python memberikan kemampuan untuk mengevaluasi beberapa kondisi secara berurutan dan menjalankan blok kode yang sesuai dengan kondisi yang terpenuhi. Struktur ini berguna ketika kita perlu melakukan pemilihan tindakan berdasarkan berbagai kondisi yang mungkin terjadi.

Masing-masing bagian dalam struktur dan contohnya sebagai berikut:

- **If**

Digunakan untuk mengevaluasi kondisi pertama. Jika kondisi pertama terpenuhi (evaluasi menghasilkan nilai True), maka blok kode di dalam if akan dieksekusi, dan program akan keluar dari struktur percabangan.

Berikut adalah contoh penggunaan 'if' sebagai berikut :



```
contoh.py > ...
1  x = 10
2  if x > 5:
3      print("x lebih besar dari 5")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh.py
x lebih besar dari 5
PS C:\Tugas> 
```

Pada contoh di atas, if digunakan untuk mengevaluasi apakah nilai x lebih besar dari 5. Jika kondisi tersebut benar (True), maka blok kode di dalam if akan dieksekusi dan mencetak "x lebih besar dari 5".

- **Elif**

Singkatan dari else if. Digunakan untuk mengevaluasi kondisi kedua (dan seterusnya) jika kondisi sebelumnya tidak terpenuhi. Jika kondisi pada elif terpenuhi, blok kode di dalam elif akan dieksekusi, dan program akan keluar dari struktur percabangan.

Berikut adalah contoh penggunaan 'elif' sebagai berikut :

```
contoh.py > ...
1  x = 5
2  if x > 5:
3      print("x lebih besar dari 5")
4  elif x == 5:
5      print("x sama dengan 5")
6  else:
7      print("x kurang dari 5")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh.py
x sama dengan 5
PS C:\Tugas>
```

elif digunakan untuk mengevaluasi kondisi tambahan jika kondisi pada if tidak terpenuhi. Pada contoh di atas, jika nilai x tidak lebih besar dari 5, maka Python akan mengecek apakah nilai x sama dengan 5. Jika ya, maka blok kode di dalam elif akan dieksekusi.

- **Else**

Digunakan sebagai kondisi terakhir yang akan dievaluasi jika semua kondisi sebelumnya tidak terpenuhi. Jika tidak ada kondisi elif yang terpenuhi, maka blok kode di dalam else akan dieksekusi.

Berikut adalah contoh penggunaan 'else' sebagai berikut :

```
contoh.py > ...
1  x = 3
2  if x > 5:
3      print("x lebih besar dari 5")
4  else:
5      print("x kurang dari atau sama dengan 5")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh.py
x kurang dari atau sama dengan 5
PS C:\Tugas>
```

Pada contoh di atas, else digunakan untuk mengevaluasi kondisi yang berbeda dari kondisi if. Jika kondisi pada if tidak terpenuhi, maka blok kode di dalam else akan dieksekusi.

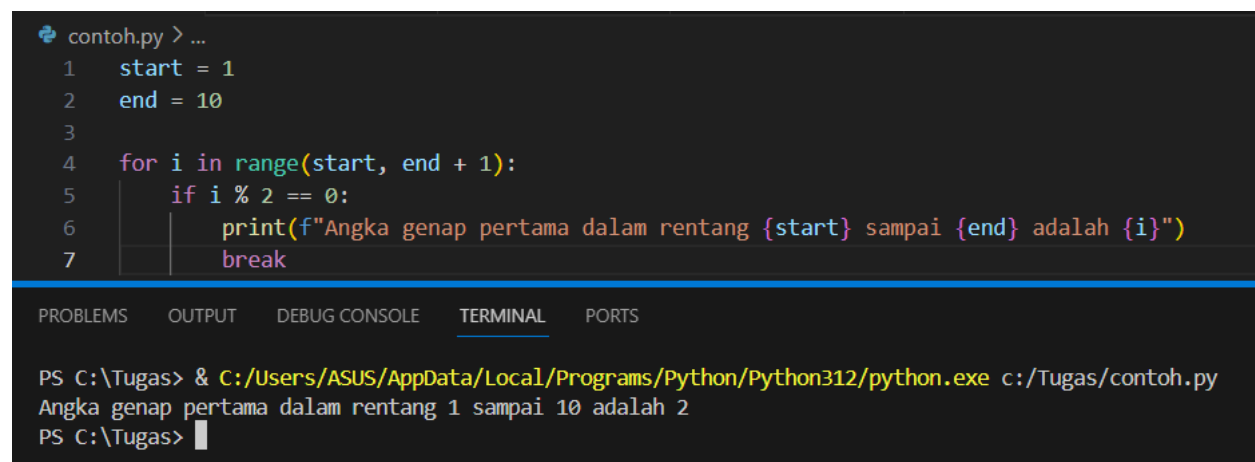
Struktur Perulangan Kompleks

Struktur perulangan kompleks dalam Python memungkinkan pengontrolan yang lebih fleksibel dan adaptif terhadap alur eksekusi program, terutama saat kita perlu menangani kondisi-kondisi yang kompleks. Salah satu fitur utama dari struktur perulangan kompleks ini adalah penggunaan pernyataan `break` dan `continue`, serta perulangan bertingkat.

- Break :

Pernyataan `break` dalam Python digunakan untuk menghentikan iterasi secara paksa dalam suatu perulangan saat kondisi tertentu terpenuhi. Ketika `break` dieksekusi, program akan keluar dari perulangan tersebut dan melanjutkan eksekusi kode setelah perulangan.

Berikut contoh penggunaan `break`:



```
contoh.py > ...
1  start = 1
2  end = 10
3
4  for i in range(start, end + 1):
5      if i % 2 == 0:
6          print(f"Angka genap pertama dalam rentang {start} sampai {end} adalah {i}")
7          break

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

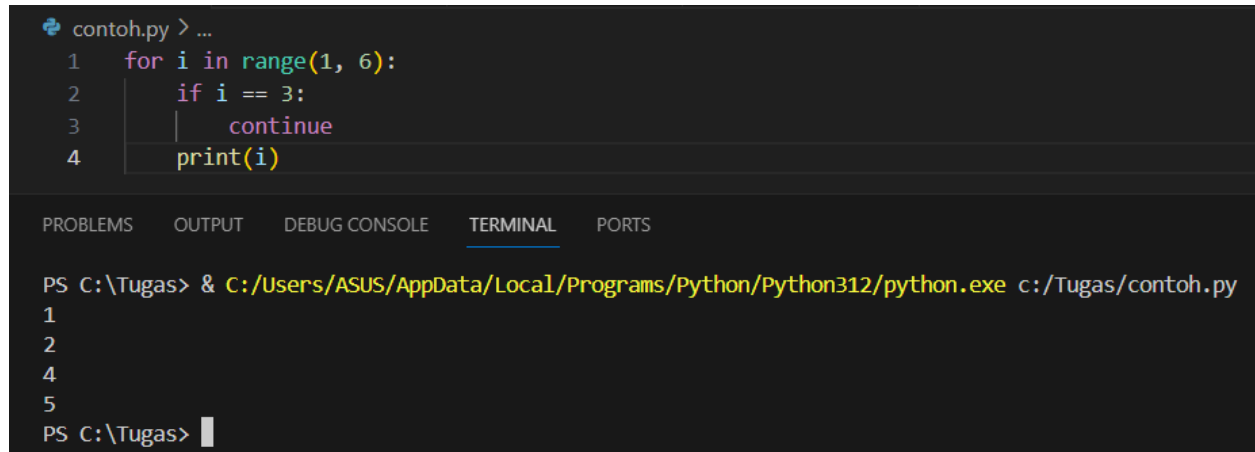
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh.py
Angka genap pertama dalam rentang 1 sampai 10 adalah 2
PS C:\Tugas>
```

Pada contoh ini, kita menggunakan perulangan `for` untuk mengiterasi angka dari `start` hingga `end`. Ketika kita menemukan angka genap pertama (`i % 2 == 0`), kita mencetak pesan yang menyatakan angka genap pertama yang ditemukan dalam rentang tersebut dan menggunakan pernyataan `break` untuk menghentikan perulangan.

- Continue

Pernyataan continue digunakan dalam Python untuk melanjutkan ke iterasi berikutnya dalam perulangan tanpa menjalankan kode di bawahnya dalam iterasi saat ini. Ini memungkinkan untuk melewati bagian tertentu dari perulangan berdasarkan kondisi yang diberikan.

Berikut adalah contoh penggunaan continue:



```
contoh.py > ...
1  for i in range(1, 6):
2      if i == 3:
3          continue
4      print(i)

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

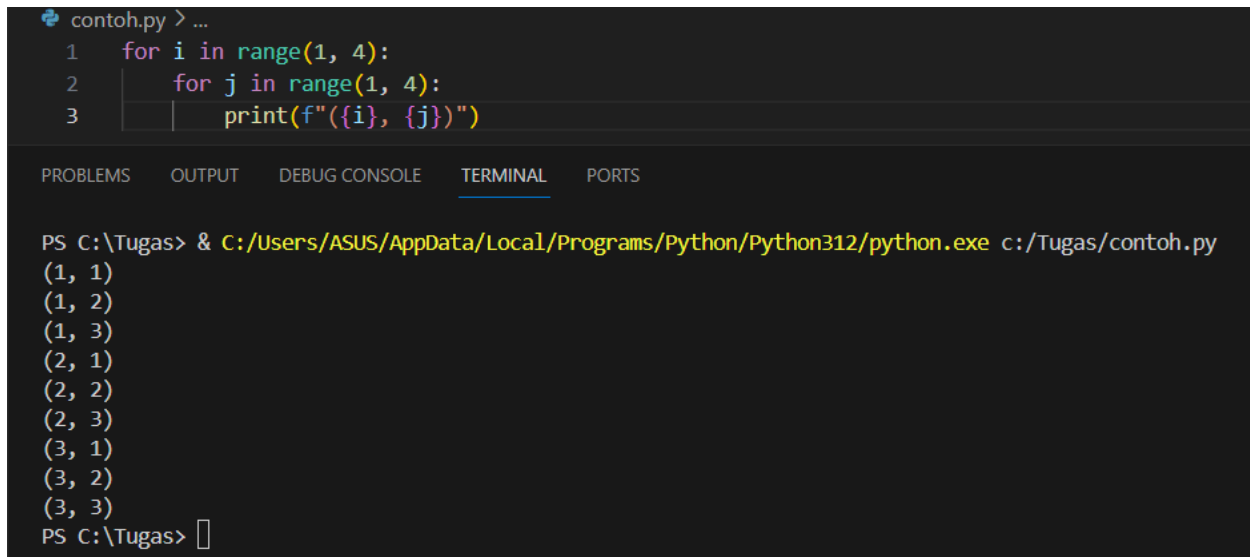
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh.py
1
2
4
5
PS C:\Tugas> 
```

Pada contoh di atas, kita menggunakan perulangan for untuk mengiterasi nilai dari 1 hingga 5. Saat nilai i sama dengan 3, pernyataan continue dieksekusi, dan kode di bawahnya dalam iterasi saat ini tidak dijalankan. Sehingga, angka 3 tidak dicetak, dan iterasi langsung melanjutkan ke angka berikutnya. Jadi, continue memungkinkan kita untuk mengabaikan atau melewati bagian tertentu dari perulangan berdasarkan kondisi tertentu tanpa menghentikan perulangan sepenuhnya.

- Perulangan bertingkat

Perulangan bertingkat dalam Python adalah konsep di mana satu atau lebih perulangan ditempatkan di dalam perulangan lain. Dengan menggunakan perulangan bertingkat, kita dapat melakukan iterasi dalam pola yang lebih kompleks, seperti mengakses dan memanipulasi elemen dalam struktur data bersarang atau membuat pola pengulangan yang rumit. Setiap iterasi dari perulangan dalam akan dieksekusi untuk setiap iterasi dari perulangan luar.

Berikut adalah contoh penggunaan perulangan bertingkat :



```
contoh.py > ...
1  for i in range(1, 4):
2      for j in range(1, 4):
3          print(f"({i}, {j})")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe c:/Tugas/contoh.py
(1, 1)
(1, 2)
(1, 3)
(2, 1)
(2, 2)
(2, 3)
(3, 1)
(3, 2)
(3, 3)
PS C:\Tugas> 
```

Pada contoh di atas, kita memiliki dua perulangan for bersarang. Perulangan pertama (for i) mengontrol nilai i dari 1 hingga 3. Setiap kali nilai i berubah, perulangan kedua (for j) akan mengontrol nilai j dari 1 hingga 3. Sehingga, setiap kombinasi nilai i dan j akan dicetak sebagai pasangan nilai yang bersesuaian.

BAGIAN 2: LATIHAN MANDIRI

SOAL 6.1

- Source Code :

```
Latihan 6,1.py > ...
1  def is_prime(num):
2      if num < 2:
3          return False
4      return all(num % i != 0 for i in range(2, num))
5
6  def cari_prima_terdekat_dibawah_n(n):
7      return next((i for i in range(n - 1, 1, -1) if is_prime(i)), None)
8
9  n = int(input("Masukkan bilangan n: "))
10 prima_dekat = cari_prima_terdekat_dibawah_n(n)
11 if prima_dekat:
12     print(f"Bilangan prima terdekat yang lebih kecil dari {n} adalah {prima_dekat}")
13 else:
14     print(f"Tidak ada bilangan prima terdekat yang lebih kecil dari {n}")
```

- Output :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe "c:/Tugas/Latihan 6,1.py"
Masukkan bilangan n: 6
Bilangan prima terdekat yang lebih kecil dari 6 adalah 5
PS C:\Tugas> 
```

- Penjelasan :

Kode tersebut berfungsi untuk mencari dan mencetak bilangan prima terdekat yang lebih kecil dari bilangan yang dimasukkan, dengan menghasilkan dan memeriksa bilangan prima secara berurutan dari bilangan `n-1` ke bawah hingga menemukan bilangan prima terdekat atau mencapai angka 2.

SOAL 6.2

- Source Code :

```
Latihan 6,2.py > ...
1  def faktorial(num):
2      result = 1
3      for i in range(1, num + 1):
4          result *= i
5      return result
6
7  n = int(input("Masukkan nilai n: "))
8  for i in range(n, 0, -1):
9      print(faktorial(i), end=" ")
10     for j in range(i, 0, -1):
11         print(j, end=" ")
12     print()
```

- Output :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe "c:/Tugas/Latihan 6,2.py"
Masukkan nilai n: 6
720 6 5 4 3 2 1
120 5 4 3 2 1
24 4 3 2 1
6 3 2 1
2 2 1
1 1
PS C:\Tugas> 
```

- Penjelasan :

Kode tersebut mendefinisikan fungsi `faktorial(num)` untuk menghitung nilai faktorial dari suatu bilangan `num`. Selanjutnya, kode meminta input `n` dari pengguna dan mencetak nilai faktorial dari setiap bilangan dari `n` hingga 1, bersama dengan urutan angka dari `n` hingga 1 dalam setiap baris.

SOAL 6.3

- Source Code :

```
Latihan 6,3.py > ...
1  def display_series(T, L):
2      start = 1
3      for i in range(T):
4          for j in range(L):
5              print(start, end=" ")
6              start += 1
7          print()
8
9  T = int(input("Masukkan tinggi: "))
10 L = int(input("Masukkan lebar: "))
11 display_series(T, L)
```

- Output :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Tugas> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe "c:/Tugas/Latihan 6,3.py"
Masukkan tinggi: 5
Masukkan lebar: 6
1 2 3 4 5 6
7 8 9 10 11 12
13 14 15 16 17 18
19 20 21 22 23 24
25 26 27 28 29 30
PS C:\Tugas> █
```

- Penjelasan :

Kode tersebut adalah sebuah fungsi Python yang mencetak urutan bilangan berurutan dalam bentuk matriks berukuran $T \times L$, dimulai dari angka 1 dan terus bertambah hingga mencapai $T \times L$. Input T dan L dimasukkan. Setiap baris matriks mencetak bilangan secara berurutan dari kiri ke kanan, dan setiap kolom matriks merupakan bagian dari urutan tersebut.

- Link GitHub :

<https://github.com/Jamesmarvins/Tugas-Alpro-6.git>