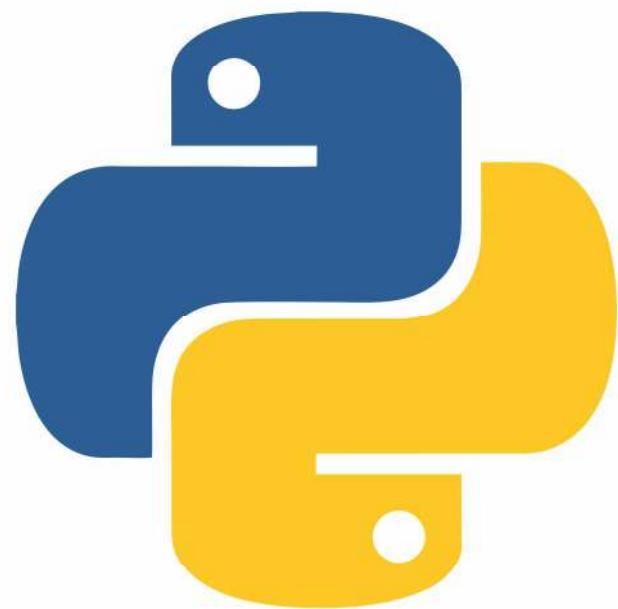


PYTHON

Pemrograman IV



PYTHON

Percabangan

Percabangan adalah cara yang digunakan untuk mengambil keputusan apabila di dalam program dihadapkan pada kondisi tertentu. Jumlah kondisinya bisa satu, dua atau lebih.

Percabangan mengevaluasi kondisi atau ekspresi yang hasilnya benar atau salah. Kondisi atau ekspresi tersebut disebut ekspresi boolean. Hasil dari pengecekan kondisi adalah **True** atau **False**. Bila benar (True), maka pernyataan yang ada di dalam blok kondisi tersebut akan dieksekusi. Bila salah (False), maka blok pernyataan lain yang dieksekusi.

Di Python ada 3 jenis pernyataan yang digunakan untuk percabangan, yaitu sebagai berikut :

- **if**. Pernyataan **if** terdiri dari ekspresi boolean diikuti oleh satu baris atau lebih pernyataan.
- **if...else**. Bila pernyataan **if** benar, maka blok pernyataan **if** dieksekusi. Bila salah, maka blok pernyataan **else** yang dieksekusi.
- **if...elif...else**. Disebut juga **if** bercabang. Bila ada kemungkinan beberapa kondisi bisa benar maka digunakan pernyataan **if...elif** atau **if...elif...else**

9.1 Pernyataan if

Pernyataan if menguji **satu buah kondisi**. Bila hasilnya benar maka pernyataan di dalam blok if tersebut dieksekusi. Bila salah, maka pernyataan tidak dieksekusi. Sintaksnya adalah seperti berikut :

if tes kondisi:

blok pernyataan if

```
percabangan_if.py ▶ ...
1  angka = 4
2  if angka > 0:
3      print(angka, "adalah Bilangan Positif.")
```

Gambar 78 Contoh Pernyataan if

9.2 Pernyataan if...else

Pernyataan if...else menguji **2 kondisi**. Kondisi pertama kalau benar, dan kondisi kedua kalau salah. Sintaksnya adalah seperti berikut :

if tes kondisi:

blok pernyataan if

else:

blok pernyataan else

PYTHON

```
percabangan_if_else.py ▶ ...
1  bilangan = -1
2
3  if bilangan >= 0:
4      print("Positif atau Nol")
5  else:
6      print("Bilangan negatif")
```

Gambar 79 Contoh Pernyataan if else

9.3 Pernyataan if...elif...else...

Pernyataan if...elif...else digunakan untuk menguji **lebih dari 2 kondisi**. Bila kondisi pada if benar, maka pernyataan di dalamnya yang dieksekusi. Bila salah, maka masuk ke pengujian kondisi elif. Terakhir bila tidak ada if atau elif yang benar, maka yang dijalankan adalah yang di blok else. Sintaksnya adalah seperti berikut :

if tes kondisi:

blok pernyataan if

elif tes kondisi:

blok pernyataan elif

else:

blok pernyataan else

```
percabangan_if elif_else.py ▶ ...
1  bilangan = 5.5
2
3  if bilangan > 0:
4      print("Bilangan positif")
5  elif bilangan == 0:
6      print("Nol")
7  else:
8      print("Bilangan negatif")
```

Gambar 80 Contoh Pernyataan if elif else

9.4 Tambahan : if Bersarang

Sebuah kondisional dapat disimpan di dalam **if** lain. Berikut ini adalah contoh kode if bersarang di Python :

PYTHON

```
❸ percabangan_if_bersarang.py ▶ ...
1  gaji = 10000000
2  berkeluarga = True
3  punya_rumah = True
4
5  if gaji > 3000000:
6      print ("Gaji sudah diatas UMR")
7      if berkeluarga:
8          print ("Wajib ikut asuransi dan menabung untuk pensiun")
9      else:
10         print ("Tidak perlu ikut asuransi")
11
12     if punya_rumah:
13         print ("Wajib bayar pajak rumah")
14     else:
15         print ("Tidak wajib bayar pajak rumah")
16 else:
17     print ("Gaji belum UMR")
```

Gambar 81 Contoh Pernyataan if Bersarang

9.5 Contoh Program Percabangan Indeks Nilai Statis

Ketentuan : Nilai 85 s/d 100 indeks A, nilai 70 s/d 84 indeks B, nilai 55 s/d 69 indeks C, nilai dibawah 55 indeks D.

```
❸ latihan_percabangan_indeks_nilai.py ▶ ...
1  nilai = 80
2
3  if nilai >= 85 and nilai <=100:
4      print("Nilai A")
5  elif nilai >= 70 and nilai <= 84:
6      print("Nilai B")
7  elif nilai >= 55 and nilai <= 69:
8      print("Nilai C")
9  else:
10     print("Nilai D")
```

Gambar 82 Contoh Program Percabangan Indeks Nilai Statis

9.6 Latihan

Buatlah program indeks nilai diatas menjadi dinamis.

PYTHON

Perulangan

Secara umum, Python mengeksekusi program baris perbaris. Mulai dari baris satu, dua, dan seterusnya. Ada kalanya, kita perlu mengeksekusi satu baris atau satu blok kode program beberapa kali.

Di python, perulangan bisa dilakukan dengan dua cara atau metode, yaitu :

- Menggunakan **for**
- Menggunakan **while**

10.1 Perulangan dengan Menggunakan for

Perulangan dengan menggunakan for memiliki sintaks seperti berikut :

for var in sequence:

 body of for

var adalah variabel yang digunakan untuk penampung sementara nilai dari sequence pada saat terjadi perulangan. **Sequence** adalah tipe data berurut seperti string, list, dan tuple.

Perulangan terjadi sampai looping mencapai elemen atau anggota terakhir dari sequence. Bila loop sudah sampai ke elemen terakhir dari sequence, maka program akan keluar dari looping.

```
perulangan_for.py ▶ ...
1  nomor = [5, 5, 2]
2
3  jumlah = 0
4
5  for tampung in nomor:
6      jumlah = jumlah + tampung
7
8  print("Jumlah semuanya :", jumlah)
```

Gambar 83 Contoh Perulangan for

10.1.1 Perulangan for dengan range

Fungsi **range()** dapat digunakan untuk menghasilkan deret bilangan. **range(10)** akan menghasilkan bilangan dari **0 sampai dengan 9 (10 bilangan)**.

PYTHON

```
perulangan_for_range.py x
perulangan_for_range.py ▶ ...
1 for hitung in range(5):
2     print("Hitung :", hitung)
```

Gambar 84 Contoh Perulangan for dengan Range

10.2 Perulangan Menggunakan while

Perulangan menggunakan **while** akan menjalankan blok pernyataan terus menerus selama kondisi bernilai **benar**.

Adapun sintaks dari perulangan menggunakan while adalah :

while expression:

 statement (s)

Di sini, **statement (s)** bisa terdiri dari satu baris atau satu blok pernyataan. **Expression** merupakan ekspresi atau kondisi apa saja, dan untuk nilai selain nol dianggap True. Iterasi akan terus berlanjut selama kondisi benar. Bila kondisi salah, maka program akan keluar dari while dan lanjut ke baris pernyataan di luar while.

```
perulangan_while.py ▶ ...
1 hitung = 0
2
3 while (hitung < 5):
4     print("Hitung :", hitung)
5     hitung = hitung + 1
```

Gambar 85 Contoh Perulangan while

10.3 Contoh Program Kelipatan Bilangan Genap

Ketentuan : Program pengulangan dengan for. Tampilkan bilangan genap dari 0 hingga batas terakhir bilangan input. Misalnya, apabila diinput 10, maka yang tampil adalah : 0 2 4 6 8.

```
latihan_perulangan_kelipatan_genap.py ▶ ...
1 i = 0
2 n = int(input("Masukkan Batas : "))
3
4 for i in range(n):
5     if i%2 == 0:
6         print("Bilangan :",i)
7
8     i = i + 1
```

Gambar 86 Contoh Program Kelipatan Bilangan Genap

PYTHON

10.4 Latihan

Buatlah program kelipatan bilangan genap dengan menampilkan banyaknya jumlah. Misalnya, apabila diinput 10, maka yang tampil adalah 0 2 4 6 8 10 12 14 16 18 (**10 bilangan**).

PYTHON

Fungsi

Fungsi adalah grup/blok program untuk melakukan tugas tertentu yang berulang. Fungsi membuat kode program menjadi reusable, artinya hanya di definisikan sekali saja, dan kemudian bisa digunakan berulang kali dari tempat lain di dalam program.

11.1 Mendefinisikan Fungsi

Berikut adalah sintaks yang digunakan untuk membuat fungsi :

```
def function_name(parameters):
    """function_docstring"""
    statement(s)
    return [expression]
```

Penjelasannya dari sintaks fungsi di atas :

- Kata kunci **def** diikuti oleh **function_name** (nama fungsi), tanda kurung dan tanda titik dua (:) menandai header (kepala) fungsi.
- **Parameter**/ argumen adalah input dari luar yang akan diproses di dalam tubuh fungsi.
- "**function_docstring**" bersifat opsional, yaitu sebagai string yang digunakan untuk dokumentasi atau penjelasan fungsi. "function_docstring" diletakkan paling atas setelah baris **def**.
- Setelah itu diletakkan baris-baris pernyataan (**statements**). Jangan lupa indentasi untuk menandai blok fungsi.
- **return** bersifat opsional. Gunanya adalah untuk mengembalikan suatu nilai expression dari fungsi.

```
funksi1.py ▶ ...
1  def sapa(nama):
2      print("Hai, " + nama + ". Apa kabar?")
3      return nama
4
5  # pemanggilan fungsi
6  # output: Hai, Anna. Apa kabar?
7  sapa("Anna")
```

Gambar 87 Contoh Fungsi

11.2 Docstring

Docstring adalah singkatan dari documentation string. Ini berfungsi sebagai dokumentasi atau keterangan singkat tentang fungsi yang kita buat. Meskipun bersifat opsional, menuliskan docstring adalah kebiasaan yang baik.

Untuk contoh di atas kita menuliskan docstring. Cara mengaksesnya adalah dengan menggunakan format **namafungsi._doc_**

PYTHON

```
❖ fungsi_docstring.py ▶ ...
1  def sapa(nama):
2      "contoh cetak keterangan"
3      print("Hai, " + nama + ". Apa kabar?")
4      return nama
5
6  sapa("Anna")
7  print(sapa.__doc__)
```

Gambar 88 Contoh Docstring

11.3 Contoh Program Luas Persegi Panjang dengan Fungsi

```
❖ latihan_fungsi_persegi_panjang.py ▶ ...
1  def persegi_panjang(panjang,lebar):
2      luas = panjang * lebar
3      print("Luasnya :",luas)
4      return luas
5
6  print("Menghitung Luas Persegi Panjang")
7  persegi_panjang(4,6)
```

Gambar 89 Contoh Program Statis

```
❖ latihan_fungsi_persegi_panjang_dinamis.py ▶ ...
1  def persegi_panjang(panjang,lebar):
2      luas = panjang * lebar
3      print("Luasnya :",luas)
4      return luas
5
6  print("Menghitung Luas Persegi Panjang")
7  a = int(input("Masukkan Panjang : "))
8  b = int(input("Masukkan Lebar : "))
9  persegi_panjang(a,b)
```

Gambar 90 Contoh Program Dinamis

11.4 Latihan

Buatlah program dinamis menghitung luas persegi panjang dan persegi dengan menggunakan 1 fungsi. Misalnya, apabila diinput panjang = 4 dan lebar 3, maka tampil luas persegi panjang = 12. Dan apabila diinput sisi persegi = 3, maka tampil luas persegi = 9. Contoh tampilan terminal seperti gambar dibawah.

PYTHON

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
lient' '--host' 'localhost' '--port' '54851
Menghitung Luas Persegi Panjang & Persegi
Persegi Panjang
Masukkan Panjang : 4
Masukkan Lebar : 3
Luasnya : 12

Persegi
Masukkan Sisi : 3
Luasnya : 9
PS C:\Python Projects\sintaks_ok> █
s_ok)
```

Gambar 91 Output Latihan yang Tampil di Terminal