

# **MODUL *PYHTON* 14**

## **Python Perulangan**

### **1. Pokok bahasan Praktikum**

1. Pengenalan Perulangan
2. Penggunaan For dan While

### **2. Tujuan Instruksional Praktikum**

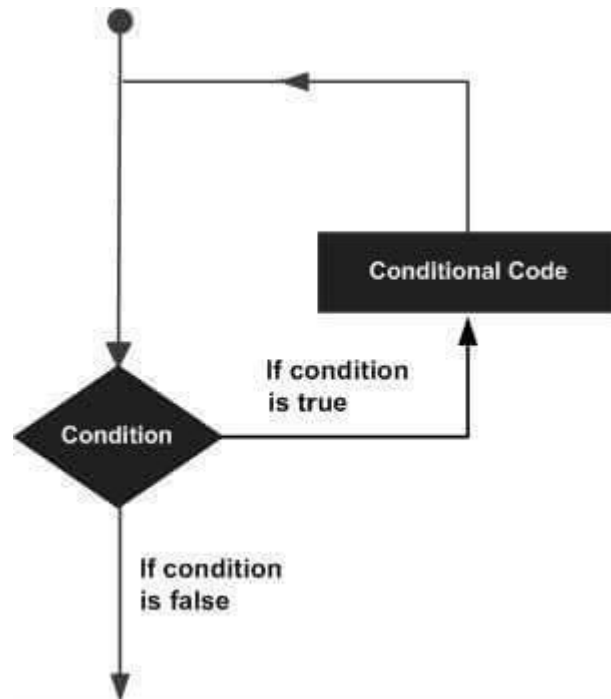
1. Mahasiswa dapat mengenal tentang dasar pemrograman
2. Mahasiswa mampu berpikir logis dalam hal pembangunan program
3. Mahasiswa mampu mengerjakan soal dasar pemrograman

### **3. Durasi**

60 Menit

## 4. 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. Hal ini disebut dengan perulangan atau biasa disebut looping atau iterasi. Untuk lebih jelasnya perhatikan gambar berikut:



Pada gambar dapat dilihat bahwa perulangan juga memerlukan tes kondisi. Bila hasil tes kondisi True, maka blok kode kembali dieksekusi. Tapi jika False, maka keluar dari perulangan. Dalam python, perulangan bisa dilakukan dengan dua cara atau metode, yaitu:

1. Menggunakan for
2. Menggunakan while

## 5. Penggunaan 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

```
# Program untuk menemukan jumlah bilangan dalam satu list

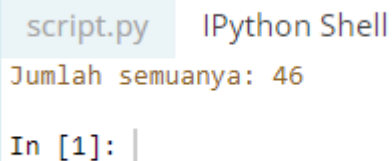
# List number
numbers = [7, 5, 9, 8, 4, 2, 6, 4, 1]

# variabel untuk menyimpan jumlah
sum = 0

# iterasi
for each in numbers:
    sum = sum + each

# Output: Jumlah semuanya: 46
print("Jumlah semuanya:", sum)
```

Bila program tersebut kita jalankan, maka hasilnya adalah seperti berikut:



```
script.py  IPython Shell
Jumlah semuanya: 46
In [1]: |
```

Bisa dikatakan penggunaan for dilakukan jika sudah diketahui jumlah perulangan yang akan dilakukan.

## 6. Fungsi range()

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

Kita juga bisa menentukan batas bawah, batas atas, dan interval dengan format **range(batas bawah, batas atas, interval)**. Bila interval dikosongkan, maka nilai default 1 yang akan digunakan.

Fungsi range tidak menyimpan semua nilai dalam memori secara langsung. Ia hanya akan mengingat batas bawah, batas atas, dan interval dan membangkitkan hasilnya satu persatu hanya bila dipanggil. Untuk membuat fungsi ini langsung menampilkan semua item, kita bisa menggunakan fungsi **list()**. Untuk jelasnya perhatikan contoh berikut:

```
# Output: range(0,10)
print(range(10))

# Output: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(list(range(10)))

# Output: [2, 3, 4, 5, 6, 7]
print(list(range(2,8)))

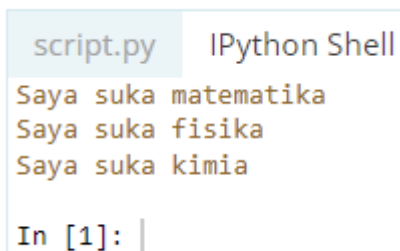
# Output: [2, 5, 8, 11, 14, 17]
print(list(range(2, 20, 3)))
```

Kita bisa menggunakan fungsi range() dalam perulangan menggunakan for untuk iterasi bilangan berurut. Hal ini dengan cara mengkombinasikan fungsi range() dengan fungsi **len()**. Fungsi **len()** berfungsi untuk mendapatkan panjang atau jumlah elemen suatu data sekuensial atau berurut.

```
# Program untuk iterasi list menggunakan pengindeksan
mapel = ['matematika', 'fisika', 'kimia']

# iterasi list menggunakan indeks
for i in range(len(mapel)):
    print("Saya suka", mapel[i])
```

Kalau program dijalankan, hasilnya akan jadi seperti berikut:



```
script.py  IPython Shell
Saya suka matematika
Saya suka fisika
Saya suka kimia

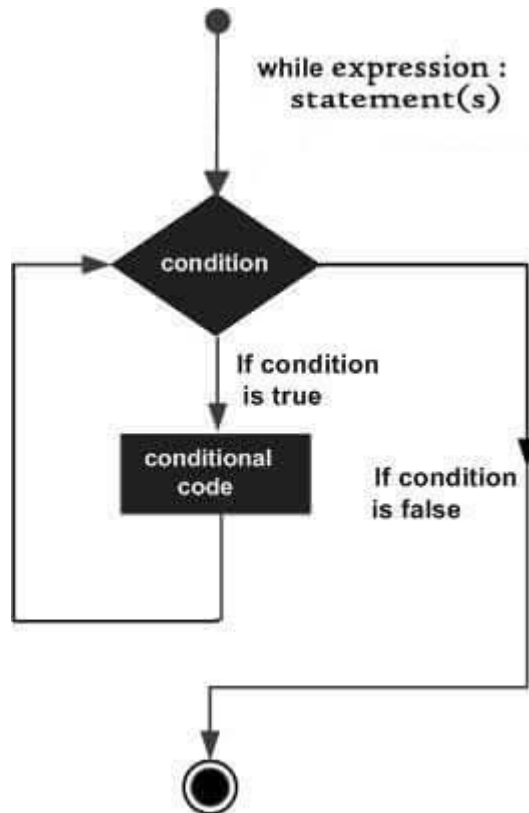
In [1]: |
```

## 7. Penggunaan *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*. Adapun diagram alir *while* adalah seperti gambar berikut:



Perhatikan bahwa bila kondisi yang diuji bernilai salah, maka loop tidak akan pernah dieksekusi.

```
count = 0
while (count < 5):
    print('The count is:', count)
    count = count + 1
print('Good bye!')
```

Bila program tersebut dijalankan, maka hasilnya seperti berikut:

```
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
Good bye!
```

Di sini, blok pernyataan **print('The count is:', count)**, dijalankan terus selama **count** masih lebih kecil dari 5. Count ditambah 1 setiap kali iterasi. Pada saat nilai count mencapai 5, maka kondisi menjadi False dan program keluar dari looping while dan melanjutkan baris selanjutnya yaitu **print("Good bye")**.

## 8. Infinite Loop

Sebuah kondisi dimana loop selalu benar dan tidak pernah salah disebut loop tidak terbatas (infinite loop). Terkadang hal ini menjadi masalah. Tapi sering juga infinite loop berguna, misalnya untuk program client/server dimana server perlu menjaga komunikasi tetap hidup dan tidak terputus. Pada contoh program while di atas, bila kita lupa menuliskan kode **count = count + 1**, maka akan jadi infinite loop. Hasilnya akan jadi seperti berikut:

```
The count is: 0  
The count is: 0  
The count is: 0  
The count is: 0  
The count is: 0  
The count is: 0  
The count is: 0  
The count is: 0  
The count is: 0  
The count is: 0  
The count is: 0  
The count is:Traceback (most recent call last):  
  File "E:/loopingerrors.py", line 3, in <module>  
    print('The count is:', count)  
KeyboardInterrupt  
>>>
```

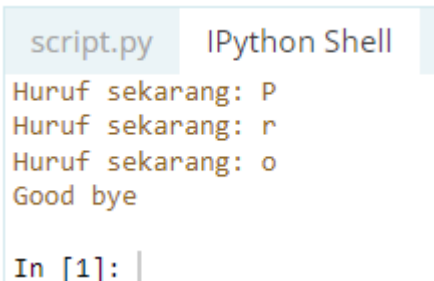
Kita perlu menekan CTRL+C untuk menghentikan program.

## 9. Kendali Looping (*break*, *continue*)

Looping umumnya akan berhenti bila kondisi sudah bernilai salah. Akan tetapi, seringkali kita perlu keluar dari looping di tengah jalan tergantung keperluan. Hal ini bisa kita lakukan dengan menggunakan kata kunci **break** dan **continue**. Statement **break** memaksa program keluar dari blok looping di tengah jalan. Sedangkan statement **continue** menyebabkan program langsung melanjut ke step / interval berikutnya dan mengabaikan (skip) baris kode di bawahnya (yang satu blok). Jelasnya perhatikan contoh berikut:

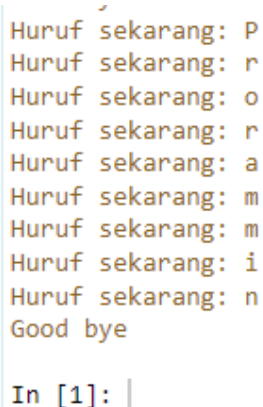
```
# contoh penggunaan statement break
for letter in "Programming":
    if letter == "g":
        break
    print("Huruf sekarang:", letter)
print("Good bye")
```

Hasil kode program:



The screenshot shows a code editor with two tabs: 'script.py' and 'IPython Shell'. The 'script.py' tab contains the Python code from the previous block. The 'IPython Shell' tab shows the output of the script: 'Huruf sekarang: P', 'Huruf sekarang: r', 'Huruf sekarang: o', and 'Good bye'. Below the output, the prompt 'In [1]:' is visible.

Bila pada program di atas kita ganti kode **break** menjadi **continue**, maka hasilnya akan jadi seperti berikut:



The screenshot shows a code editor with two tabs: 'script.py' and 'IPython Shell'. The 'script.py' tab contains the Python code from the previous block, but with 'break' replaced by 'continue'. The 'IPython Shell' tab shows the output of the script: 'Huruf sekarang: P', 'Huruf sekarang: r', 'Huruf sekarang: o', 'Huruf sekarang: r', 'Huruf sekarang: a', 'Huruf sekarang: m', 'Huruf sekarang: m', 'Huruf sekarang: i', 'Huruf sekarang: n', and 'Good bye'. Below the output, the prompt 'In [1]:' is visible.

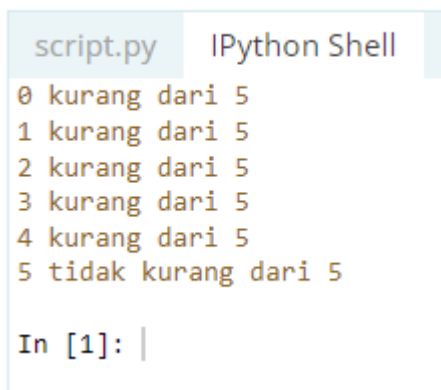
Perhatikan bahwa huruf **g** tidak pernah ditampilkan karena diabaikan karena kode **continue**.

## 10. While Else

Python mendukung penggunaan **else** sebagai pasangan dari **while**. Blok pernyataan **else** hanya akan dieksekusi bila kondisi **while** bernilai salah.

```
count = 0
while (count < 5):
    print(count, "kurang dari 5")
    count = count + 1
else:
    print(count, "tidak kurang dari 5")
```

Bila dijalankan program tersebut akan menghasilkan tampilan seperti berikut:



```
script.py  IPython Shell
0 kurang dari 5
1 kurang dari 5
2 kurang dari 5
3 kurang dari 5
4 kurang dari 5
5 tidak kurang dari 5

In [1]: |
```



## Latihan

Buat program untuk menampilkan output seperti berikut ini!

1.

```
*  
**  
***  
****  
*****  
*****  
*****  
*****  
*****  
*****  
*****
```

2.

Masukkan angka :5

```
* * * * *  
 * * * *  
  * * *  
   * *  
    *
```

3.

```
1 * 2 * 3 * 4 * 5  
1 * 2 * 3 * 4  
1 * 2 * 3  
1 * 2  
1
```