



UNS
UNIVERSITAS
SEBELAS MARET



MODUL PRAKTIKUM PROGRAMA KOMPUTER

**PROGRAM STUDI TEKNIK INDUSTRI
UNIVERSITAS SEBELAS MARET**

**TIM ASISTEN LABORATORIUM
PERANCANGAN DAN OPTIMASI
SISTEM INDUSTRI 2020**

MODUL V ITERASI

A. Tujuan

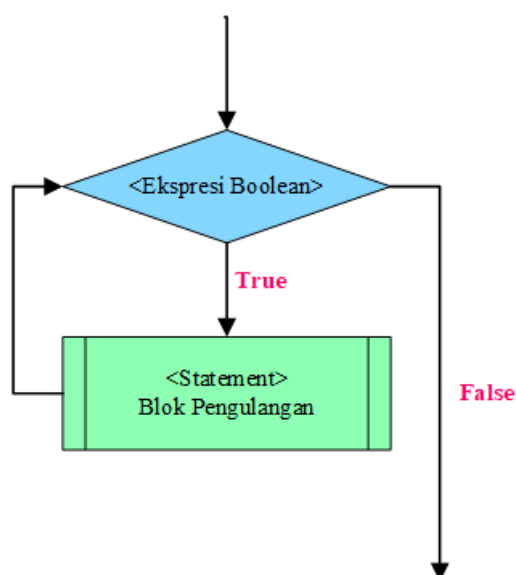
Berikut merupakan tujuan Praktikum Programa Komputer Modul IV.

1. Memahami struktur iterasi menggunakan **for**.
2. Memahami struktur iterasi menggunakan **while**.
3. Memahami struktur iterasi bersarang atau *nested loop*.
4. Memahami pernyataan loncat.

B. Iterasi atau *Looping*

Iterasi atau *looping* adalah baris kode atau instruksi yang dieksekusi oleh komputer secara berulang-ulang dengan jumlah tertentu selama sebuah kondisi yang telah ditentukan masih terpenuhi.

Dalam bahasa pemrograman Python, terdapat dua *syntax* yang digunakan untuk iterasi, yaitu **for** dan **while**. Iterasi **for** disebut *counted loop* (perulangan yang terhitung), sementara perulangan **while** disebut *uncounted loop* (perulangan yang tak terhitung).



Gambar 1 Ilustrasi Iterasi

C. **for** Loop

Iterasi **for** adalah perintah yang digunakan untuk melakukan pengulangan dari sebuah nilai sequence atau tipe data koleksi pada Python seperti List, Tuple, String dan lain-lain. Pada

iterasi **for**, eksekusi terhadap blok kode dilakukan berulang kali sesuai dengan variabel yang mengatur perulangan.

1. **for** untuk Penelusuran Tipe Data Koleksi

Format umum dari iterasi **for** untuk penelusuran item adalah sebagai berikut.

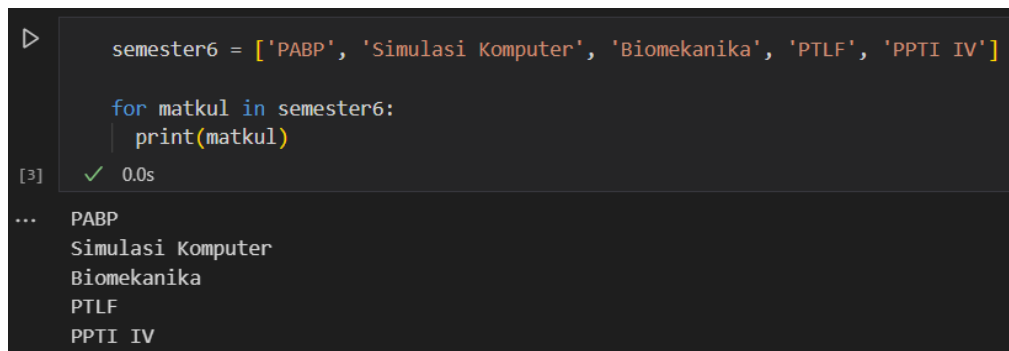
```
#Untuk penelusuran elemen dalam tipe data koleksi
for nilai in sequence:
    statement 1
    statement 2
    ...

#Menggunakan indeks yang dihasilkan fungsi range
for index in range(nilai_awal,nilai_akhir):
    statement 1
    statement 2
    ...
```

Keterangan:

sequence : tipe data koleksi
 nilai : setiap item dalam sequence
 statement : perintah tertentu yang akan dieksekusi secara berulang

Berikut merupakan contoh iterasi **for** untuk penelusuran item dalam tipe data koleksi.



```
> semester6 = ['PABP', 'Simulasi Komputer', 'Biomekanika', 'PTLF', 'PPTI IV']

for matkul in semester6:
    print(matkul)

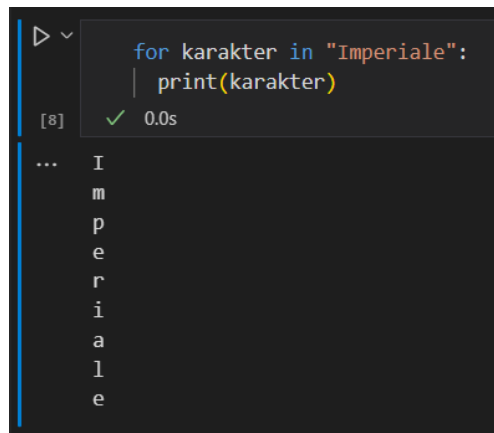
[3] ✓ 0.0s

... PABP
    Simulasi Komputer
    Biomekanika
    PTLF
    PPTI IV
```

Gambar 2 **for** untuk penelusuran

Berdasarkan contoh di atas, `semester6` merupakan `sequence` yang berupa tipe data list, `matkul` merupakan `nilai`, dan `print(matkul)` merupakan `statement`. Baris kode pada contoh di atas meminta untuk menampilkan setiap item (berupa `matkul`) yang ada pada list `semester6`.

Berikut merupakan contoh lain iterasi **for** untuk penelusuran item dalam tipe data koleksi.



```

for karakter in "Imperiale":
    print(karakter)

```

[8] ✓ 0.0s

... I
m
p
e
r
i
a
l
e

Gambar 3 **for** untuk penelusuran

Dalam penelusuran item, dapat juga diketahui urutan iterasi menggunakan fungsi **enumerate()**. Fungsi **enumerate()** digunakan untuk mengetahui urutan iterasi. Format umum dari iterasi **for** menggunakan fungsi **enumerate()** adalah sebagai berikut.

```

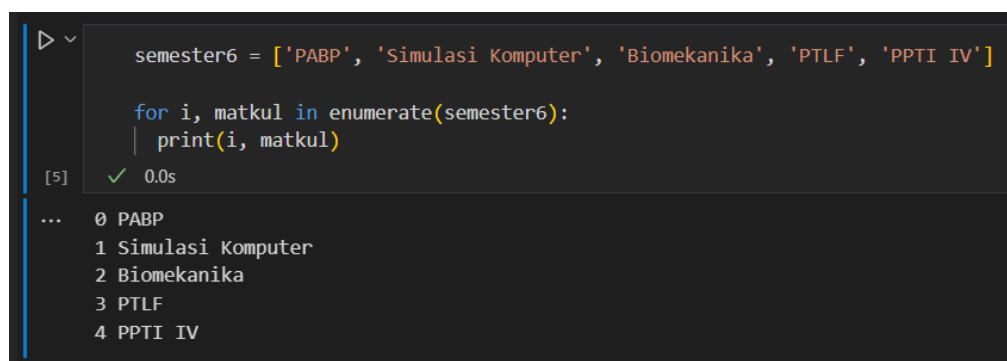
#Untuk penelusuran elemen dalam tipe data koleksi
for i, nilai in enumerate(sequence):
    statement 1
    statement 2
    ...

```

Keterangan:

i : indeks atau urutan iterasi ke-berapa
 enumerate : fungsi untuk mengetahui urutan iterasi

Berikut merupakan contoh iterasi **for** menggunakan fungsi **enumerate()**.



```

semester6 = ['PABP', 'Simulasi Komputer', 'Biomekanika', 'PTLF', 'PPTI IV']

for i, matkul in enumerate(semester6):
    print(i, matkul)

```

[5] ✓ 0.0s

... 0 PABP
1 Simulasi Komputer
2 Biomekanika
3 PTLF
4 PPTI IV

Gambar 4 **for** menggunakan fungsi **enumerate()**

2. **for** dengan fungsi **range()**

Fungsi **range()** digunakan untuk melakukan iterasi sebanyak nilai *range* yang ditetapkan. Format umum dari iterasi **for** dengan fungsi **range()** adalah sebagai berikut.

```
#Untuk melakukan iterasi dari item berindeks i=0 hingga i<y
for nilai in range(y):
    statement 1
    statement 2
    ...
```

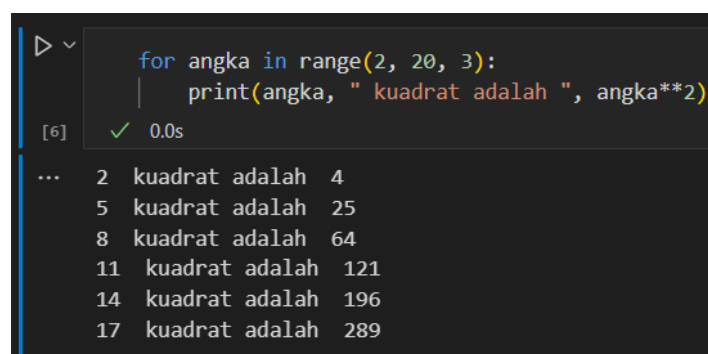
```
#Untuk melakukan iterasi dari item berindeks i=x hingga i<y
for nilai in range(x,y):
    statement 1
    statement 2
    ...
```

```
#Untuk melakukan iterasi dari item berindeks i=x hingga i<y
dengan step inkremen z
for nilai in range(x,y,z):
    statement 1
    statement 2
    ...
```

Keterangan:

range : fungsi untuk menentukan *range* iterasi
 x : indeks dimulainya iterasi ($i=x$)
 y : indeks berakhirnya iterasi ($i<y$)
 z : step inkremen

Berikut merupakan contoh iterasi **for** menggunakan fungsi **range()**.



```
for angka in range(2, 20, 3):
    print(angka, " kuadrat adalah ", angka**2)
```

[6] ✓ 0.0s

```
... 2 kuadrat adalah 4
     5 kuadrat adalah 25
     8 kuadrat adalah 64
    11 kuadrat adalah 121
    14 kuadrat adalah 196
    17 kuadrat adalah 289
```

Gambar 5 **for** menggunakan fungsi **range()**

Berdasarkan contoh di atas, program diminta untuk menampilkan kuadrat dari angka dimulai dari 2 hingga sebelum 20 dengan step inkremen 3.

Berikut merupakan contoh lain iterasi **for** untuk iterasi menggunakan fungsi **range()**.

```

total = 0.0
n = 5
for i in range(1, n + 1):
    print("nilai total sebelum ditambahkan ", total)
    total = total + i
    print(total)

print("Jumlah keseluruhan ", total)
print("Rata-rata : ", total/n)

```

[7] ✓ 0.0s

```

... nilai total sebelum ditambahkan 0.0
1.0
nilai total sebelum ditambahkan 1.0
3.0
nilai total sebelum ditambahkan 3.0
6.0
nilai total sebelum ditambahkan 6.0
10.0
nilai total sebelum ditambahkan 10.0
15.0
Jumlah keseluruhan 15.0
Rata-rata : 3.0

```

Gambar 6 **for** menggunakan fungsi **range()**

D. **while** Loop

Iterasi **while** adalah perintah yang digunakan untuk melakukan pengulangan suatu blok kode program selama sebuah kondisi terpenuhi. Iterasi **while** pengulangan yang bersifat *indefinite* alias tidak pasti, atau bahkan tidak terbatas .

Format umum dari iterasi **for** untuk penelusuran item adalah sebagai berikut.

```

while <kondisi>:
    statement 1
    statement 2
    ...

```

Keterangan:

Kondisi : kondisi syarat pengulangan dilakukan (dapat berupa Boolean atau ekspresi logika)

statement : perintah tertentu yang akan dieksekusi secara berulang

1. **while** untuk Iterasi Tanpa Batas

Selama suatu kondisi itu selalu benar, maka iterasi akan selalu di eksekusi. Berikut merupakan contoh iterasi **while** untuk iterasi tanpa batas.

Gambar 7 **while** untuk iterasi tanpa batas

2. **while** untuk Iterasi Terbatas

```

▷ ✓
    i = 7

    while i <= 10:
        print(i)
        i = i+1

    print("selesai")

[10] ✓ 0.0s
... 7
    8
    9
    10
    selesai

```

Gambar 8 **while** untuk iterasi terbatas

Berdasarkan contoh di atas, program diminta untuk menampilkan isi dari variabel i selama i kurang dari sama dengan 10. Sebelumnya telah didefinisikan bahwa $i = 7$. Baris kode $i=i+1$ merupakan penambahan *counter*. Hal ini berguna untuk membatasi iterasi yang terjadi dimana program tidak terus menerus mendefinisikan bahwa $i = 7$.

Berikut merupakan contoh lain iterasi **while** untuk iterasi terbatas.

```

jumlahBilangan = input("Inputkan jumlah bilangan : ")
jumlahBilangan = int(jumlahBilangan)

total = 0.0
bilanganBerikutnya = 1

while bilanganBerikutnya <= jumlahBilangan:
    total = total + bilanganBerikutnya
    print("menambahkan :", bilanganBerikutnya, "Total saat ini : {:.1f}".format(total))
    bilanganBerikutnya = bilanganBerikutnya + 0.1

print("Total penjumlahan bilangan : ", total)
print("Rataan bilangan : ", total/jumlahBilangan)

```

[11] ✓ 12.5s

```

...
menambahkan : 1 Total saat ini : 1.0
menambahkan : 1.1 Total saat ini : 2.1
menambahkan : 1.2000000000000002 Total saat ini : 3.3
menambahkan : 1.3000000000000003 Total saat ini : 4.6
menambahkan : 1.4000000000000004 Total saat ini : 6.0
menambahkan : 1.5000000000000004 Total saat ini : 7.5
menambahkan : 1.6000000000000005 Total saat ini : 9.1
menambahkan : 1.7000000000000006 Total saat ini : 10.8
menambahkan : 1.8000000000000007 Total saat ini : 12.6
menambahkan : 1.9000000000000008 Total saat ini : 14.5
Total penjumlahan bilangan : 14.500000000000004
Rataan bilangan : 7.250000000000002

```

Gambar 9 **while** untuk iterasi terbatas

```

ulangi = True
counter = 1

while ulangi == True:
    print("Blok Proses ke-", counter, "Dilaksanakan")
    print(" - - - - - ")

    inputPengguna = input("Apakah Anda akan mengulangi program ? (y / n) ")
    if inputPengguna == 'y':
        ulangi = True
        counter = counter + 1
    else:
        ulangi = False
    print("")

    print("Selesai")

```

[2] ✓ 3.7s

```

...
Blok Proses ke- 1 Dilaksanakan
- - - - -

Blok Proses ke- 2 Dilaksanakan
- - - - -

Selesai

```

Gambar 10 **while** untuk iterasi terbatas

E. Iterasi Bersarang atau *Nested Loop*

Iterasi bersarang atau *nested loop* merupakan jenis *loop* pada Python yang mengizinkan penggunaan *loop* didalam *loop*. Berikut merupakan contoh iterasi bersarang.

```

> ~
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
    for y in fruits:
        print(x, y)
[3] ✓ 0.0s

... red apple
    red banana
    red cherry
    big apple
    big banana
    big cherry
    tasty apple
    tasty banana
    tasty cherry

```

Gambar 11 Iterasi bersarang **for** dalam **for**

```

> ~
for i in range (1,11):
    for j in range (1,i+1):
        print('%d ' %(i*j), end= '')
    print()
[4] ✓ 0.0s

... 1
    2 4
    3 6 9
    4 8 12 16
    5 10 15 20 25
    6 12 18 24 30 36
    7 14 21 28 35 42 49
    8 16 24 32 40 48 56 64
    9 18 27 36 45 54 63 72 81
    10 20 30 40 50 60 70 80 90 100

```

Gambar 12 Iterasi bersarang **for** dalam **for**

```

> ~
i=7
while i<=10:
    j=1
    while j<=i:
        print("%d" %(i*j),end="")
        j=j+1
    print()
    i+=1
[2] ✓ 0.0s

... 7
    8
    9
    10

```

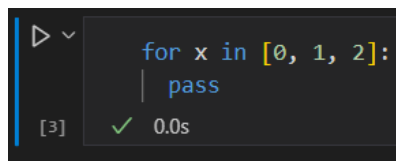
Gambar 13 Iterasi bersarang **while** dalam **while**

F. Pernyataan Loncat

Pernyataan loncat adalah perintah yang digunakan memindahkan eksekusi program dari satu bagian tertentu kebagian lain.

1. **pass**

Pernyataan **pass** digunakan sebagai *placeholder* untuk *future code* yang mungkin dibuat di masa mendatang. Hal ini berguna untuk menghindari terjadinya error oleh adanya kode kosong. Dengan kata lain, **pass** digunakan sebagai pernyataan *dummy*. Berikut merupakan contoh penggunaan **pass**.



```

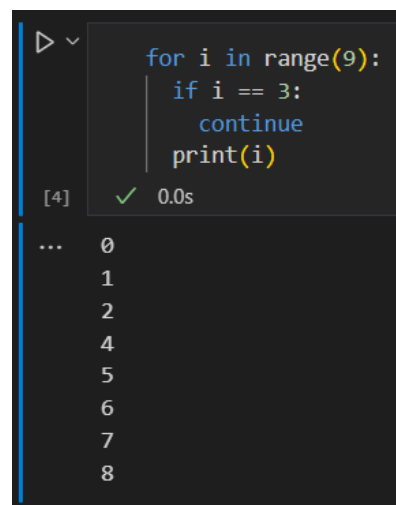
for x in [0, 1, 2]:
    pass
[3] ✓ 0.0s

```

Gambar 14 Penggunaan **pass**

2. **continue**

Pernyataan **continue** digunakan sebagai untuk mengakhiri iterasi yang terjadi saat ini dan berlanjut ke iterasi berikutnya. Berikut merupakan contoh penggunaan **continue**.



```

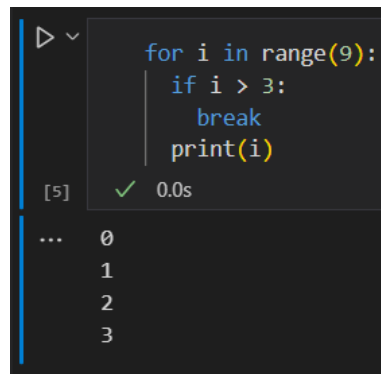
for i in range(9):
    if i == 3:
        continue
    print(i)
[4] ✓ 0.0s
... 0
    1
    2
    4
    5
    6
    7
    8

```

Gambar 15 Penggunaan **continue**

3. **break**

Pernyataan **break** digunakan menghentikan secara paksa proses iterasi, meskipun kondisi iterasi sebenarnya masih memenuhi syarat iterasi. Berikut merupakan contoh penggunaan **break**.



```
for i in range(9):  
    if i > 3:  
        break  
    print(i)
```

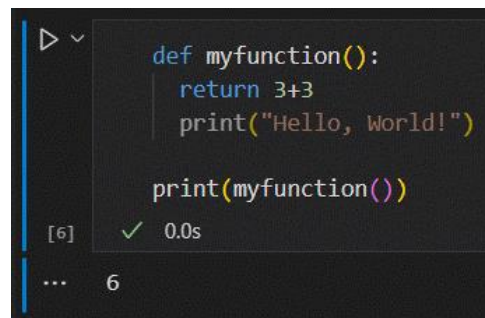
[5] ✓ 0.0s

... 0
1
2
3

Gambar 16 Penggunaan **break**

4. **return**

Pernyataan **return** digunakan sebagai untuk keluar dari fungsi dan mengembalikan nilai. Perintah **return** akan menghentikan proses eksekusi kode di dalam fungsi dan eksekusi akan diarahkan ke baris pemanggil. Berikut merupakan contoh penggunaan **return**.



```
def myfunction():  
    return 3+3  
    print("Hello, World!")  
  
print(myfunction())
```

[6] ✓ 0.0s

... 6

Gambar 17 Penggunaan **return**