PERULANGAN PADA PYTHON

Perintah perulangan di gunakan untuk mengulang pengeksekusian statemen-statemen hingga berkali-kali sesuai dengan iterasi yang diinginkan. Dalam python, perintah untuk perulangan (loop) adalah *while* dan *for*.

1. Perintah While

Perintah while pada python merupakan perintah yang paling umum digunakan untuk proses iterasi. Konsep sederhana dari perintah while adalah ia akan mengulang mengeksekusi statemen dalam blok while selama nilai kondisinya benar. Dan ia akan keluar atau tidak melakukan eksekusi blok statemen jika nilai kondisinya salah.

Bentuk umum statemen *while,*

```
while (kondisi) :
     statemen
```

Contoh penggunaan *while :*

contoh 1 :

```
>>> while True :
...     print "Tekan CTRL + C  untuk Stop"
...
```

Pada contoh 1, merupakan contoh sederhana penggunaan *while.* Pada contoh di atas program akan terus mengeksekusi statemen dalam badan while, dikarenakan kondisinya selalu benar (true). Kondisi seperti ini disebut *infinite loop.i*

contoh 2 :

```
>>> x = "Gunadarma"
>>> while x:
...     print x, ' '
...     x = x[1:]
...
Gunadarma
unadarma
nadarma
adarma
darma
arma
rma
ma
a
```

contoh 3 :

```
>>> a = 0; b = 10
>>> while a < b :
...     print a,
...     a = a + 1
...
0 1 2 3 4 5 6 7 8 9
```

2. Perintah For

Perintah *for* dalam python mempunyai ciri khas tersendiri dibandingkan dengan bahasa pemrograman lain. Tidak hanya mengulang bilangan-bilangan sebuah ekspresi aritmatik, atau memberikan keleluasaan dalam mendefinisikan iterasi perulangan dan menghentikan perulangan pada

saat kondisi tertentu. Dalam python, statemen *for* bekerja mengulang berbagai macam tipe data sekuensial seperti List, String, dan Tuple.

Bentuk umum perintah *for*,

```
for (variabel) in (objek) :
     statemen
else:
     statemen
```

Contoh penggunaan *for :*

Contoh 1 :
```
>>> for i in [5, 4, 3, 2, 1]:
...     print i,
...
5 4 3 2 1
```

Pada contoh 1, perintah perulangan terjadi dimana data-data untuk iterasi (objek) berada dalam List. Jadi elemen-elemen yang berada dalam List akan di masukkan (assign) ke dalam variabel target yaitu i.

Contoh 2 :
```
>>> T = [(1,2), (3,4), (5,6)]
>>> for (a,b) in T :
...     print (a,b)
...
(1, 2)
(3, 4)
(5, 6)
```

Pada contoh 2, merupakan penggunaan tipe data Tuple untuk proses perulangan. Elemen pada tuple akan di assign kedalam variabel a dan b.

Contoh 3 :
```
>>> nama = ['budi', 'andi', 'rudi', 'sandi']
>>> usia = [20, 18, 22, 19]
>>> for i in range(len(nama)) :
...    print nama[i], ' berusia ', usia[i], ' tahun'
...
budi  berusia  20  tahun
andi  berusia  18  tahun
rudi  berusia  22  tahun
sandi  berusia  19  tahun
```

3. Perintah Break, Continue dan Pass

Perintah Break

Perintah *break* digunakan untuk menghentikan jalannya proses iterasi pada statemen *for atau while.* Statemen yang berada di bawah *break* tidak akan di eksekusi dan program akan keluar dari proses looping.

Contoh *break :*
```
>>> x = 1
>>> while x < 5:
...    if x == 3:
...      break
...    print x
...    x = x+1
```

```
... else:
        print "Loop sdh selesai dikrjkn"
...
1
2
```

Perintah Continue

Statemen *continue* menyebabkan alur program kembali ke perintah looping. Jadi jika dalam sebuah perulangan terdapat statemen *continue,* maka program akan kembali ke perintah looping untuk iterasi selanjutnya.

Contoh *continue :*

```
>>> n = 10
>>> while n:
...     n = n - 1
...     if n % 2 != 0:
...             continue
...     print n
...
8
6
4
2
```

Perintah Pass

Statemen *pass* mengakibatkan program tidak melakukan tindakan apa-apa. Perintah *pass* biasanya digunakan untuk mengabaikan suatu blok statemen perulangan, pengkondisian, class, dan fungsi yang belum didefinisikan badan programnya agar tidak terjadi error ketika proses compilasi.

Contoh program *pass :*

```
#program tidak akan melakukan
#proses looping
while True : pass
```

LATIHAN MEMBUAT PROGRAM DENGAN FOR, WHILE, DAN BREAK

```
>>> a=1
>>> while a<5:
...     print a,
...     a+=1
...
1 2 3 4

>>> a=0
>>> while a<10:
...     a+=1
...     if a%2:
...             print '%d bilangan ganjil'%a
...     else:
...             continue
...
1 bilangan ganjil
3 bilangan ganjil
```

```
5 bilangan ganjil
7 bilangan ganjil
9 bilangan ganjil

>>> a=1
>>> while a<10:
...         print a,
...         a+=1
...         if a>6:
...                 break
1 2 3 4 5 6
>>> while 1:
...         print 'perulangan tiada batas, tekan ^C
            Untuk berhenti'
...
            perulangan tiada batas, tekan ^C untuk berhenti
            perulangan tiada batas, tekan ^C untuk berhenti
......

>>> a=1
>>> while a<5:
...         print a,
...         a+=1
...     else:
...         print 'selesai'
...
1 2 3 4 selesai

>>> for a in range(1,5):
...         print a,
...
1 2 3 4

>>> for a in range(1,10,2):
...         print '%d bilangan ganjil'%a
...
1 bilangan ganjil
3 bilangan ganjil
5 bilangan ganjil
7 bilangan ganjil
9 bilangan ganjil

>>> bulan={1:'januari',2:'februari',3:'maret',4:'april',5:'mei'}
>>> for a in bulan.values():
...         print a
...
januari
februari
```

```
maret
april
mei

>>> s='ini sebuah string'
>>> for a in s:
...     print a,
...
i n i s e b u a h s t r i n g

>>> t=1,'aku',5.6,6666666,['a','b','c']
>>> for a in t:
...     print a,
...
1 aku 5.6 6666666 ['a', 'b', 'c']

>>> for a in t[4]:
...     print a,
...
a b c
```

Coba perhatikan bagian perulangan for, python melakukannya dengan sangat teliti dan efisien. Kita dapat memecah elemen data pada objek string, list, tuple, dan dictionary dengan sangat mudah.

Cetak Segitiga *

```
x = "**********"
while x:
        print x + ' '
        x = x[1:]

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

Perulangan

Perulangan dengan kata kunci while mempunyai format umum sebagai berikut:

```
while kondisi:
   perintah_jika_kondisi_benar
else:
   perintah_lain
```

Perintah-perintah antara while dan else akan selalu dijalankan jika kondisi benar. Perintah-perintah di bawah else akan dijalankan jika perulangan selesai dengan normal. Yang dimaksud dengan normal di sini adalah proses perulangan tidak berhenti karena bertemu kata kunci break.

```
x = 1
while x<5:
   print x
   x = x + 1
else:
   print 'Loop sudah selesai dikerjakan!'
```

Jika skrip di atas ditambahkan suatu kondisi yang diikuti perintah break, maka blok perintah di bawah else tidak akan pernah dijalankan. Perhatikan perbedaannya dengan skrip berikut:

```
x = 1
while x<5:
   if x==3:
      break
   print x
   x = x + 1
else:
   print 'Loop sudah selesai dikerjakan!'
```

Perintah break menyebabkan dijalankannya perintah-perintah setelah blok while dan else ini.

Perintah perulangan selain while adalah for. Format dasar perulangan for adalah:

```
for variabel in objek:
   perintah-perintah
else:
   perintah_jika_tidak_bertemu_break
```

Langsung saja ke contoh penggunaan pernyataan for ini.

```
for x in range(1,5):
   print x
else:
   print 'Perulangan selesai'
```

Fungsi builtin range() dalam skrip ini menghasilkan bilangan 1 sampai 4. Hasilnya skrip akan mencetak bilangan dari 1 sampai 4 dan string Perulangan selesai.

Keistimewaan perulangan dengan for di Python adalah dapat memroses array. Seperti contoh di bawah ini:

```
y = [10,20,30,40,50,60,70,80,90]
for x in y:
   if x==50:
      continue
   if x>70:
      break
```

```
    print x
else:
    print 'Perulangan selesai'
```

Hasil yang akan didapat jika skrip ini dijalankan:

10 20 30 40 60 70

Pernyataan continue akan menyebabkan proses berlanjut ke awal perulangan dan melewatkan perintah-perintah yang ada di antara continue dan akhir blok perulangan.

```
count = 0
while True:
    count += 1
    # end loop if count is greater than 10
    if count > 10:
        break
    # skip 5
    if count == 5:
        continue
    print count
raw_input("\n\nPress the enter key to exit.")
```

1
2
3
4
6
7
8
9
10

Press the enter key to exit.

The `continue` statement means "jump back to the top of the loop." At the top of the loop, the `while` condition is tested and the loop is entered again if it's true. So when `count` is equal to `5`, the program does not get to the `print count` statement.
Instead it goes right back to the top of the loop and `5` is skipped and never printed.

Perulangan for

Perintah for dalam python mempunyai ciri khas tersendiri dibandingkan dengan perulangan for pada bahasa pemograman C ataupun Pascal. Tidak hanya mengulang bilangan - bilangan sebuah ekspresi aritmatik(dalam Pascal), atau memberikan keleluasaan si user untuk mendefinisikan perulangan iterasi dan menghentikan perulangan pada saat kondisi tertentu (dalam C). Dalam Python mengulang berbagai macam tipe data sekuensial seperti list, string, dan tuple.

```
>>> a = ['satu', 'dua', 'tiga', 'empat'] >>> for i in a :
...     print i
satu
dua
```

```
tiga
empat
```

Contoh diatas berarti fungsi for <iterasi> in <objek>:.

Perulangan while

Perulangan while akan mengulang didalam ruang lingkup while, selama suatu kondisi terpenuhi.

```
>>> n = 9
>>> while n < 20 :
...     print n
...     n = n + 1

9
10
11
12
13
14
15
16
17
18
19
```

Pada contoh diatas, nilai variabel n akan ditambahkan 1 secara terus menerus sampai kondisi n lebih kecil dari 20.

Fungsi range()

Jika Anda ingin melakukan perulangan sejumlah yang diinginkan, fungsi built-in range sangat membantu. Fungsi tersebut menghasilkan sejumlah indeks dari nilai yang telah ditentukan. Contohnya :

```
>>> range(15)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

Ataupun sebagian angka yang diinginkan. Contohnya :

```
>>> range (8, 15)
[8, 9, 10, 11, 12, 13, 14]
>>> range(0,9,3)
[0, 3, 6]
>>> range(0, 20, 3)
[0, 3, 6, 9, 12, 15, 18]
```

Contoh diatas menunjukan kelipatan dari suatu interval bilangan yang mempunyai sintaks range(<nilai-awal>, <nilai-akhir>, <kelipatan-angka>). Selanjutnya adalah Contoh perulangan for dengan range() :

```
>>> for i in range(10):
...     print i

0
1
2
3
```

```
4
5
6
7
8
9
```

Mengulang perulangan for sejumlah anggota elemen suatu tipe data sekuensial. Contohnya :

```
>>> a
['satu', 'dua', 'tiga', 'empat']
>>> for i in range(len(a)):
...     print i
0
1
2
3
```

Perintah break, continue dan else

Perintah break seperti dalam bahasa C, berarti keluar dari ruang lingkup yang terkecil dari kondisi for atau while.

Perintah continue sama halnya dengan di C, yang berfungsi melanjutkan kalimat perintah berikutnya dalam kondisi perulangan.

Pada kondisi perulangan juga diperbolehkan untuk menggunakan kalimat perintah else, yang dijalankan pada saat kondisi perulangan for tidak menemui suatu kondisi atau jika suatu kondisi tersebut mengalami kesalahan / false (dengan while), tetapi bukan pada saat kondisi perulangan dihentikan dengan perintah break. Berikut adalah contohnya :

```
for n in range(2, 10):
    for x in range(2, n):
        if n % x == 0:
            print n, 'sama dengan', x, '*', n/x break

else:

    print n, 'adalah bilangan prima'
```

```
2 adalah bilangan prima
3 adalah bilangan prima 4 sama dengan 2 * 2
5 adalah bilangan prima 6 sama dengan 2 * 3
7 adalah bilangan prima 8 sama dengan 2 * 4
9 sama dengan 3 * 3
```

Penjelasannya adalah apabila suatu kondisi dalam perulangan for x in range(2, n) tidak ada yang terpenuhi maka alur perulangannya akan lari ke ruang lingkup perintah else.