Pertemuan2

Percabangan & Perulangan pada Python

Objektif:

1. Mahasiswa mengetahui percabangan dan perulangan pada Python.
2. Mahasiswa mengetahui bentuk umum dari percabangan dan perulangan pada Python.
3. Mahasiswa dapat membuat program sederhana dengan menggunakan bahasa pemrograman Python, pada percabangan dan perulangan.

P2.1 Teori

Percabangan

Pada umummnya dalam membuat program selalu ada seleksi, dimana diperlukan pengecekan suatu kondisi untuk mengarahkan program berjalan sesuai keinginan. Pada Python untuk melakukan suatu pengecekan kondisi, terdapat tiga macam statemen. Antara lain :

1. Perintah if
2. Perintah if – else
3. Perintah if – else – elif
4. Perintah if bersarang

Perintah If

Bentuk umum perintah if :

if (kondisi) :

statemen

Statemen*if* digunakan untuk melakukan penyeleksian dimana jika kondisi bernilai benar, maka program akan mengeksekusi statemen dibawahnya. Dalam Python, untuk penulisan pengkondisian dan statemen dipisahkan oleh tanda titik dua (:). Contohnya :

```
>>> nama = “python”
>>> if nama == “python” :
...     print “Hello “ + nama
...
Hello python
```

Untuk setiap penulisan perintah *if* setelah penentuan kondisi, maka dilanjutkan dengan penulisan tanda titik dua (:). Tanda titik dua ini berarti jika kondisi bernilai benar, maka statemen-statemen setelah tanda titik dua akan dijalankan.

Perintah if – else

Statemen*if – else* digunakan untuk melakukan penyeleksian kondisi dimana jika kondisi bernilai benar maka program akan mengeksekusi statemen 1. Namun jika nilai kondisi bernilai salah, maka statemen 2 yang akan dieksekusi.

Bentuk umum perintah *if – else* :

```
if (kondisi) :
        statemen 1
else :
        statemen 2
```

Contoh program :

```
>>> kunci = “python”
>>> password = raw_input(“Masukkan Password : “)
Masukkan Password : saya
>>> if password == kunci :
...     print “Password Benar”
...  else :
...     print “Password Salah”
...
Password Salah
```

Perintah if – else – elif

Statemen *if – else – elif* digunakan untuk melakukan penyeleksian kondisi dimana kondisi yang diberikan lebih dari 1 kondisi atau memiliki beberapa kondisi. Jika kondisi pertama bernilai benar, maka lakukan seleksi kondisi kedua dan seterusnya.

Bentuk umum perintah *if – else – elif*

```
if (kondisi 1) :
        statemen
elif (kondisi 2) :
        statemen
else:
        statemen
```

Contoh program

```
>>> angka = input(“Masukkan bilangan : “)
Masukkan bilangan : 5
>>> if angka > 0 :
...     print “Angka merupakan bilangan positif”
...elif angka < 0 :
...     print “Angka merupakan bilangan negatif”
...else :
```

```
...     print “Angka merupakan 0”
...
Angka merupakan bilangan positif
```

Perintah if bersarang

Kondisi bersarang adalah suatu kondisi didalam kondisi tertentu. Jika terdapat 2 cabang kondisi, maka didalam salah satu cabang kondisi tersebut dapat pula diisi suatu kondisi tertentu, misalnya :

```
if x == y:
        print x, y “mempunyai nilai yang sama”
else :
        if x > y :
                print x, “lebih besar dari”, y
        if x < y :
                print x, “lebih kecil dari”, y
```

Kondisi pertama mempunyai 2 pilihan kondisi. Kondisi pertama mempunyai perintah baris yang sederhana, sedangkan kondisi kedua mempunyai 2 pilihan kondisi lagi didalamnya. Walaupun pengidentifikasian dalam Python sangat mudah dibaca, akan tetapi akan lebih sulit untuk membacanya secara cepat. Pada umumnya lebih baik menghindari kondisi bersarang seperti ini. Misalnya kita dapat menjalankan perintah berikut dengan menggunakan satu kondisi :

```
if 0 < x :
        if x < 10 :
                print x, “bilangan positif terdiri dari satu digit”
```

Perintah print akan dijalankan jika kedua kondisi diatas terpenuhi, jadi kita dapat menulisnya dengan carfa menggunakan operator logika “and” :

```
if 0 < x and x < 10 :
        print x, “bilangan positif terdiri dari satu digit”
```

Python juga menyediakan struktur kalimat matematika pada umumnya, seperti :

```
if 0 < x < 10 :
        print x, “bilangan positif terdiri dari satu digit”
```

Contoh diatas sama artinya dengan contoh-contoh sebelumnya yang menggunakan kondisi berantai dan operator logika.

Untuk menguji kondisi, dapat menggunakan operator ==, <, <=, >, >=, dan !=.

Perhatikan cara penulisan blok-blok program dalam Python blok-blok perintah ditandai dengan penulisan kode program yang menjorok ke dalam. Setiap perintah yang mempunyai batas kiri yang sama dianggap satu blok. Sebisa mungkin harus konsisten menggunakan karakter spasi atau karakter tabulasi untuk membuat indentasi. Kesalahan yang sering terjadi dengan indentasi ini adalah terlihat dalam penampilan editor sudah lurus pada batas kiri tapi ada perbedaan dalam jumlah karakter tabulasi atau spasi.

Perulangan

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

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* :

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

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 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 diatas, perintah perulangan terjadi dimana data-data untuk iterasi (objek) berada dalam List. Jadi, elemen-elemen yang berada dalam List akan dimasukkan (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.

Perintah break, continue dan pass

Perintah break

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

Contoh :

```
>>> x = 1
>>> while x < 5 :
...     if x == 3 :
...         break
```

```
...     print x
...     x = x + 1
... else :
...     print “Loop sudah selesai dikerjakan”
...
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 :

```
>>> 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 eror ketika proses kompilasi.

Contoh program pass :

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

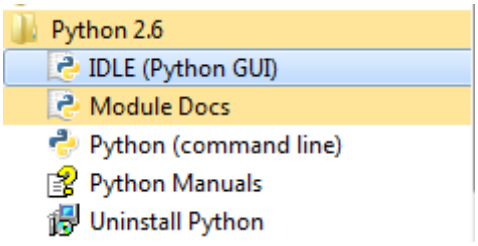
P2.2 Contoh Kasus

Kasus 1 :

Buat program untuk memasukkan nama, NPM, nilai UTS dan nilai UAS, yang memiliki perhitungan nilai rata-rata (dari nilai UTS dan UAS) dan kondisi nilai akhir dari nilai rata-rata yang ada (nilai A s/d E).

Langkah 1.

Buka IDLE (Python GUI) yang ada di menu start, lalu klik menu file pilih new window (Ctrl + N).



Langkah 2.

Ketik listing program berikut :

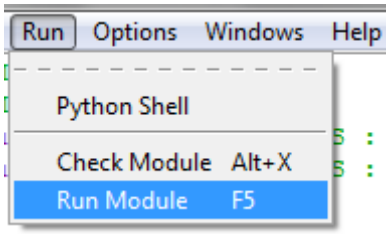
INGAT! Penggunaan spasi dan tabulasi dalam penulisan listing. Salah indentasi maka program tidak bisa running!

```
a=raw_input("Masukkan Nama : ")
b=raw_input("Masukkan NPM : ")
c=int(raw_input("Masukkan Nilai UTS : "))
d=int(raw_input("Masukkan Nilai UAS : "))
e = (c+d)/2
print
print "Nama Anda " +a
print "NPM Anda " +b
print "Nilai rata-rata Anda %d" %(e)
if e>=80:
    print "Anda Mendapatkan A"
elif e>=70:
    print "Anda Mendapatkan B"
elif e>=60:
    print "Anda Mendapatkan C"
elif e>=40:
    print "Anda Mendapatkan D"
else :
    print "Anda Mendapatkan E"
```

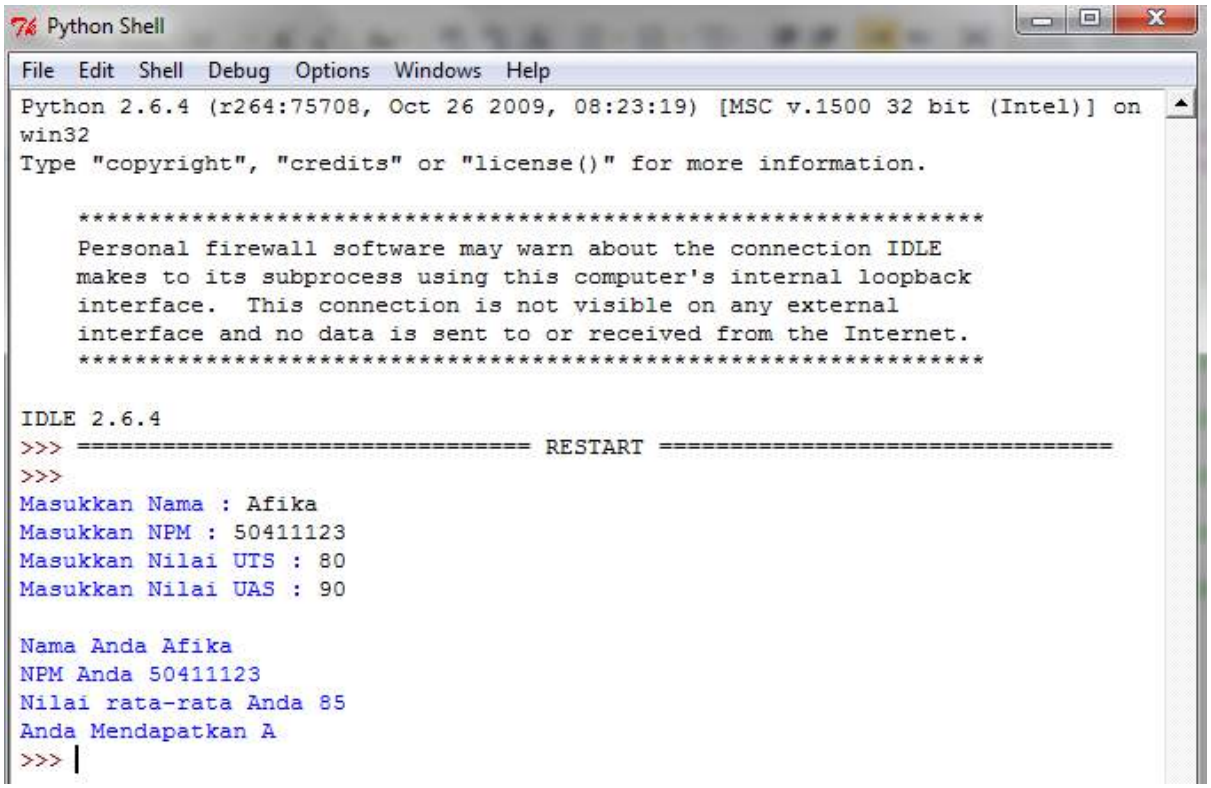
Setelah selesai klik menu file → save. Simpan dengan nama nilai.py.

Langkah 3.

Untuk menjalankan listing program diatas klik menu Run → Run Module F5 seperti gambar dibawah ini.



Kemudian input data dan tekan enter, maka contoh hasil outputnya adalah sebagai berikut.

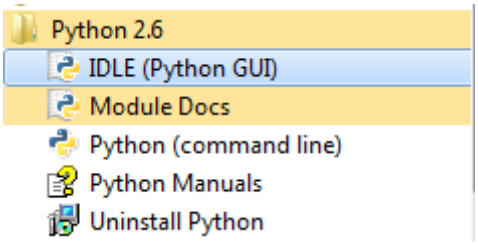


Kasus 2 :

Buat program dengan output segitia siku-siku yang terbalik dengan menggunakan perintah while.

Langkah 1.

Buka IDLE (Python GUI) yang ada di menu start, lalu klik menu file pilih new window (Ctrl + N).



Langkah 2.

Ketik listing program berikut :

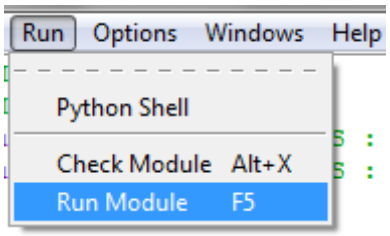
INGAT! Penggunaan spasi dan tabulasi dalam penulisan listing. Salah indentasi maka program tidak bisa running!

```
x = "@@@@@@@@@@"
while x:
        print x, ' '
        x = x[1:]
```

Setelah selesai klik menu file → save. Simpan dengan nama segitiga.py

Langkah 3.

Untuk menjalankan listing program diatas klik menu Run → Run Module F5, maka akan muncul output seperti gambar dibawah ini.




```
IDLE 2.6.4
>>> ================================ RESTART ================================
>>>
@@@@@@@@@@
@@@@@@@@
@@@@@@@
@@@@@@
@@@@@
@@@@
@@@
@@
@
>>>
```

P2.3 Latihan

1. Buat persegi simbol ‘$’ dengan menggunakan perintah perulangan for dimana user dapat memasukkan panjang dan lebar sisi persegi tersebut.

   Jawaban :

```
s = input("Masukkan sisi yang Anda inginkan : ")
for i in range(0,s,1):
    for j in range(0,s,1):
        print"$",
    print""
print'\n'
```

```
IDLE 2.6.4
>>> ================================ RESTART
>>>
Masukkan sisi yang Anda inginkan : 5
$ $ $ $ $
$ $ $ $ $
$ $ $ $ $
$ $ $ $ $
$ $ $ $ $


>>>
```

2. Buat segitiga siku-siku simbol ‘*’ dengan menggunakan perintah perulangan for dimana user dapat memasukkan tinggi segitiga siku-siku tersebut.

   Jawaban :

```
t = input("Masukkan tinggi yang Anda inginkan : ")
t = t + 1
for i in range(0,t,1):
    for j in range(0,i,1):
        print"*",
    print""
print'\n'
```

```
IDLE 2.6.4
>>> ================================ RESTART
>>>
Masukkan tinggi yang Anda inginkan : 4

*
* *
* * *
* * * *


>>>
```

3. Buat program menginput nama dan umur dengan kondisi sebagai berikut :
    a. Jika 10 <= umur <= 17 maka cetak nama “termasuk pemain anak-anak/remaja“
    b. Jika 18 <= umur <= 30 maka cetak nama “termasuk pemain muda/dewasa”
    c. Jika umur > 30 maka cetak nama “termasuk pemain tua/veteran”
    d. Selain dari ketentuan diatas cetak nama “termasuk pemain dibawah umur”

    Contoh hasil output :

```
IDLE 2.6.4
>>> ================================ RESTART
>>>
Masukkan Nama : Afika
Masukkan Umur : 6

Afika termasuk pemain dibawah umur.
>>>
```

    Jawaban :

```
nama = raw_input("Masukkan Nama : ")
umur = input("Masukkan Umur : ")
print
if 10<=umur<=17:
    print nama + " termasuk pemain anak-anak/remaja."
elif 18<=umur<=30:
    print nama + " termasuk pemain muda/dewasa."
elif umur>30:
    print nama + " termasuk pemain tua/veteran."
else:
    print nama + " termasuk pemain dibawah umur."
```

P2.4 Daftar Pustaka

[1] http://www.master.web.id/mwmag/issue/01/content/tutorial-python-1/tutorial-python-1.html, 27 Februari 2012

[2] http://www.scribd.com/doc/30882425/Tutorial-Python-2, 28 Februari 2012

[3] http://id.wikibooks.org/wiki/Python_Selayang_Pandang, 29 Februari 2012