#KampusKuningGading

STRUKTUR DATA (PYTHON)

"Instruksi Utama Python"

[@SUARGA| [Pertemuan 03]

OutLine


Instruksi Seleksi if/else
Instruksi Perulangan
Mendefinisikan Fungsi
Menangani Error
Akses File Teks

Instruksi Utama Python

1. Instruksi Seleksi: digunakan untuk memilih instruksi yang akan di-eksekusi selanjutnya berdasarkan pada suatu syarat / kondisi. Bilamana syarat dipenuhi maka instruksi di-laksanakan, bila tidak maka hal lain dilakukan
2. Instruksi Perulangan: digunakan untuk mengulang serangkaian instruksi pada perulangan yang terbatas atau tidak terbatas, namun dapat dihentikan berdasarkan suatu kondisi.
3. Pemakaian File : instruksi untuk membaca file dan merekam ke file
4. Pembuatan Fungsi : Pembuatan Fungsi merupakan bagian penting dari pemrograman Python

Instruksi Seleksi if/else

- Instruksi **if** digunakan untuk memilih instruksi yang akan dilaksanakan berdasarkan suatu syarat atau relasi, bentuknya sebagai berikut:

```
if (relasi) :
        true statements #kerjakan bila benar
else:
        false statements #kerjakan bila salah
```

- Contoh apabila anda ingin membuat pilihan antara menampilkan nilai (x+10) bila x bernilai kurang dari 5, dengan nilai (x*10) bila x bernilai lebih dari 5, maka secara algoritmik anda menuliskan:

<u>Algoritma:</u>

Bila x < 5:

maka tampilkan (x + 10)

selain-itu tampilkan (x * 10).

<u>Python:</u>

```
if ( x < 5) :
    print( x + 10)
else:
    print( x * 10)
```

pemilihan multi-syarat

- Instruksi seleksi dapat dibuat multi syarat/relasi, dengan memakai *elif* diantaranya, bentuknya sebagai berikut.

```
if (relasi-1):
        statement-1
elif (relasi-2):
        statement-2
elif (relasi-3):
        statement-3
else:
        statement-4
```

contoh

```
x = int(input('Masukkan satu angka : '))
if x < 0 :
    x = 0
    print('Angka negatif dijadikan nol!!')
elif x == 0:
    print('Anda memasukkan angka nol!')
else :
    print('Angka positif ',x)

Contoh eksekusi :
>>>
Masukkan satu angka : 10
Angka positif  10
>>>
Masukkan satu angka : 0
Anda memasukkan angka nol!
>>>
```

Contoh

- Buat sebuah program yang menerima masukkan berupa koefisien dari persamaan kuadrat $ax^2 + bx + c = 0$, kemudian menampilkan dua akar dari persamaan tersebut.
-
- Algoritma:

```
Masukkan koefisien a, b, dan c
Hitung diskriminan D = b² – 4ac
Bila D > 0 maka:          # akar riel
    Hitung akar dari D, D = sqrt(D)
    x1 = (-b + D)/(2a)
    x2 = (-b – D)/(2a)
Bila D < 0 maka:          # akar kompleks
    Hitung akar dari –D, D = sqrt(-D)
    x1 = -b/2a + D/2a j
    x2 = -b/2a – D/2a j
Bila D=0 maka:            # akar kembar
    x1 = -b/2a
    x2 = -b/2a
Tampilkan x1,x2
```

```python
#akar.py - mencari akar pers ax**2 + bx + c
import math
a = float(input('Masukkan koefisien a : '))
b = float(input('Masukkan koefisien b : '))
c = float(input('Masukkan koefisien c : '))

D = b**2 - 4*a*c
if (D > 0):                 #akar riel
    D = math.sqrt(D)
    x1 = (-b + D)/(2*a)
    x2 = (-b - D)/(2*a)
elif (D < 0):               #akar complex
    D = math.sqrt(-D)
    x1 = (-b/2*a) + (D/2*a)*1j
    x2 = (-b/2*a) - (D/2*a)*1j
else:                       #akar kembar
    x1 = -b/(2*a)
    x2 = -b/(2*a)

print('Akar persamaan kuadrat:')
print('x1 = ', x1)
print('x2 = ', x2)
```

```
>>>
Masukkan koefisien a : 1
Masukkan koefisien b : 10
Masukkan koefisien c : 30
Akar persamaan kuadrat:
x1 =  (-5+2.23606797749979j)
x2 =  (-5-2.23606797749979j)
>>>
Masukkan koefisien a : 2
Masukkan koefisien b : 10
Masukkan koefisien c : 12
Akar persamaan kuadrat:
x1 =  -2.0
x2 =  -3.0
>>>
```

Perulangan for

- Python memiliki bentuk for yang unik, dibanding dengan for pada bahasa C/C++ atau Java. For pada python berbasis pada struktur data “list”, bentuk pertama adalah:

for x in listvar:

do-something

dimana listvar adalah variabel bertipe list (list akan dibicarakan pada bagian lain). Instruksi ini sepadan dengan kalimat algoritma berikut ini:
Untuk setiap nilai x dalam listvar, kerjakan: do-something

- Bentuk kedua:

for i in range(intvar):
**   do-something**

dimana intvar adalah variabel bertipe integer. Instruksi ini sepadan dengan kalimat algoritma berikut ini:

```
Untuk setiap nilai i mulai dari 0 hingga (intvar-1),
      kerjakan do-something
```

Berarti *do-something* akan dikerjakan berulang sebanyak (invar) kali, misalnya bila intvar = 10 maka akan diulangi 10 kali dengan nilai i = 0, 1, 2, …, 9.

contoh

```
# bentuk pertama
a =['pintu', 'jendela', 'kusen', 'teralis']
for x in a:
    print (x, len(x))

print()
print('i \t 2i \t i^2')
# bentuk kedua
for i in range(len(x)):
    print(i,'\t', 2*i, '\t', i**2)
```

Hasil-nya:

```
>>>
```

- pintu 5
- jendela 7
- kusen 5
- teralis 7
-
- i 2i i^2
- 0 0 0
- 1 2 1
- 2 4 4
- 3 6 9
- 4 8 16
- 5 10 25
- 6 12 36

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

hasilnya adalah : 0 1 2 3 4 5 6 7 8 9

```
b = ['guru', 'mengajar', 'pelajaran', 'budi', 'pekerti']
for k in range(len(b)):
   print (k, b[k])
```

hasilnya adalah:
guru
mengajar
pelajaran
budi
pekerti

- Instruksi for dari python juga memiliki option **else**, yang bisa melaksanakan instruksi ketika perulangan for berakhir, atau di-akhiri dengan **break**.

```
for n in range(2,10):       # n bernilai 2 s/d 9
    for x in range(2,n):    # x bernilai 2 s/d n
        if ( n % x) == 0:   # bila n habis dibagi oleh x
            print (n, ' = ', x, '*', n/x)
            break     # loop x berakhir
    else:             # keluar loop x, else dari for
        print (n, ' adalah bilangan prima')
```

Hasilnya adalah:

```
>>>
2  adalah bilangan prima
3  adalah bilangan prima
4  =  2 * 2.0
5  adalah bilangan prima
6  =  2 * 3.0
7  adalah bilangan prima
8  =  2 * 4.0
9  =  3 * 3.0
>>>
```

Perulangan while

- Selain dengan instruksi for, serangkaian instruksi dapat diulang memakai instruksi while, bentuknya:

While (relasi) :
do-something

Instruksi tersebut sepadan dengan kalimat berikut:

```
Selama (relasi) ini masih benar maka
lakukan: do-something
```

Contoh Program

Disain program untuk menebak angka dengan interface dialog sebagai berikut:

```
Halo kawan, siapa nama anda?
Eko
Ok, Eko saya memikirkan satu angka antara 1 s/d 20
Coba anda tebak,
10
Angka tersebut lebih kecil dari 10
Tebak lagi,
2
Angka tersebut lebih besar dari 2
Tebak lagi,
4
Bagus, anda telah menebaknya dalam 3 langkah.
```

```
# program tebakan angka
#tebakan.py
import random              # import pustaka fungsi acak
maxi = 6                   # jumlah tebakan maksimum

jumTebakan = 0
print('Hallo kawan, siapa nama anda?')
nama = input()
angka = random.randint(1, 20) # menciptakan angka acak antara 1 s/d 20

print('Ok ' + nama + ' saya memikirkan satu angka antara 1 s/d 20')
while (jumTebakan < maxi):
    print('Coba anda tebak, ')
    tebak = input()
    tebak = int(tebak)
    jumTebakan = jumTebakan + 1
    if (tebak < angka) :
        print('Angka tersebut lebih besar dari ' + str(tebak))
    if (tebak > angka):
        print('Angka tersebut lebih kecil dari ' + str(tebak))
    if (tebak == angka):
        break

if (tebak == angka):
    print ('Bagus, anda telah menebaknya dalam ' + str(jumTebakan) \
                + ' langkah')
if (tebak != angka):
    print ('Maaf, anda tidak berhasil menebak angka ' + str(angka))
```

Contoh Run :

>>>

Hallo kawan, siapa nama anda?

EKO

Ok EKO saya memikirkan satu angka antara 1 s/d 20

Coba anda tebak,

4

Angka tersebut lebih besar dari 4

Coba anda tebak,

20

Angka tersebut lebih kecil dari 20
Coba anda tebak,
12
Angka tersebut lebih kecil dari 12
Coba anda tebak,
8
Angka tersebut lebih kecil dari 8
Coba anda tebak,
6
Angka tersebut lebih kecil dari 6
Coba anda tebak,
5
Bagus, anda telah menebaknya dalam 6 langkah

Menangani Error

- Program sering kali menimbulkan error baik karena adanya kesalahan logik program, maupun akibat yang lain. Error yang diperkirakan dapat terjadi dalam program dapat ditangani dengan perintah **raise** suatu kesalahan, misalnya:

```
def sqrt(x):
      if not isinstance(x, (int, float)):
            raise TypeError(  'x harus numerik ')
      elif x < 0:
            raise ValueError( 'x harus positif ' )
      else:
            return(math.sqrt(x))
```

- Pada contoh fungsi sqrt(x), yaitu mencari akar kuadrat dari bilangan x, maka ada dua kemungkinan error, pertama x harus numerik (float) dan kedua x harus positif. Oleh sebab itu, pada fungsi ini diperkirakan kemungkinan dua macam error yaitu: kesalahan tipe-data (type error) dan kesalahan nilai data (value error).
- Perlu diperhatikan bahwa TypeError dan ValueError merupakan error standard yang disediakan oleh Python, jenis error yang lain disajikan dalam tabel berikut ini.

Class	Description
Exception	A base class for most error types
AttributeError	Raised by syntax obj.foo, if obj has no member named foo
EOFError	Raised if "end of file" reached for console or file input
IOError	Raised upon failure of I/O operation (e.g., opening file)
IndexError	Raised if index to sequence is out of bounds
KeyError	Raised if nonexistent key requested for set or dictionary
KeyboardInterrupt	Raised if user types ctrl-C while program is executing
NameError	Raised if nonexistent identifier used
StopIteration	Raised by next(iterator) if no element; see Section 1.8
TypeError	Raised when wrong type of parameter is sent to a function
ValueError	Raised when parameter has invalid value (e.g., sqrt(−5))
ZeroDivisionError	Raised when any division operator used with 0 as divisor

Jenis-jenis error Python

try … except

- Cara yang lain adalah dengan memakai pasangan instruksi **try** … **except** misalnya pada pembacaan file berikut ini:

```
try :
  fp = open( 'sample.txt')
except IOError as e :
  print( 'tidak dapat membuka file: ', str(e) )
```

- Berarti file ‘sample.txt’ akan coba dibuka, bila tidak bisa, akan tampil error (‘tidak dapat membuka file’).

Kuiz

- Buat sebuah program yang meminta user memasukkan angka bulat X, kemudian menghitung S1 sebagai jumlah semua bilangan ganjil antara 1 dan X, dan S2 sebagai jumlah semua bilangan genap antara 1 dan X.

Definisi Fungsi

- Suatu fungsi dalam python didefinisikan melalui keyword "**def**", diikuti oleh nama fungsi dan argumen-nya. Sebagai contoh akan dibuat fungsi yang menghitung nilai suku ke-n bilangan Fibonacci, dari suatu deret Fibonacci. Deret Fibonacci hingga suku ke-10 adalah sebagai berikut:

0 1 1 2 3 5 8 13 21 35

- Terlihat bahwa dua nilai awal adalah 0 dan 1, kemudian nilai berikut-nya adalah 1 = 0 + 1, lalu 2 = 1+1, kemudian 3 = 1+2, dan seterusnya. Andaikan nilai awal adalah a=0, dan b=1, kemudian nilai berikutnya adalah a + b, kemudian nilai a, b digeser ke depan, yaitu b digeser ke a, dan (a+b) digeser ke b, sehingga dapat dibuat suatu fungsi fibo(n) sebagai berikut:

```
def fibo(n):
  # menampilkan deret fibonacci sampai nilainya <= n
  a, b = 0, 1
  while (b < n):
     print (b),
     a, b = b, a+b
```

setelah di-save dengan nama **fibo.py**, maka dapat dicoba dengan memanggil namanya disertai parameter, misalnya untuk menampilkan bilangan Fibonacci hingga maksimum bernilai 2000,

```
>>> fibo(2000)
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
>>>
```

```
def fibo2(n):
    # menampilkan deret fibonacci sampai
    # nilainya <=-n.
    # hasil ditampilkan horisontal
    hasil = []
    a, b = 0, 1
    while (b < n):
        hasil.append(b)
        a, b = b, a+b
    return hasil
```

- Agar hasilnya bisa ditampilkan lakukan sebagai berikut.

```
>>> x=fibo2(2000)
>>> x
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597]

>>>
```

Akses File Teks

- Membaca File Teks:

Membaca suatu file dapat dilakukan dengan urutan instruksi sebagai berikut:

1. Menyatakan variable nama file : `infile = "nama-file"`
2. Membuka file untuk dibaca : `infh = open(infile, mode)`
3. Membaca baris-demi-baris :

```
line = infh.readline()
while line:
    # proses baris data
    line = infh.readline()
```

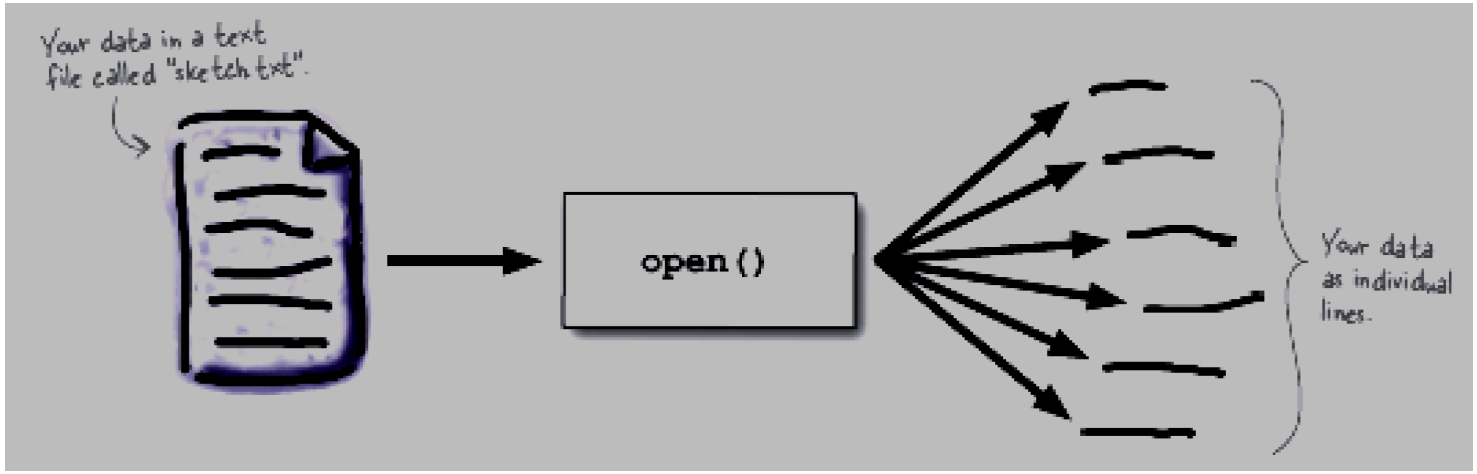
4. Menutup file : `infh.close()`

dimana mode-file untuk membaca adalah:

"r" - membaca file text
"r+" - membaca sekaligus bisa merekam
"rb" - membaca file binary

Contoh baca file

```
path="D:\\USER\\Python\\sample.txt"
myfile = open(path, 'r')
for line in (myfile):
    print(line, end='')
myfile.close()
```



Merekam Teks ke File

- Prosedur-nya

1. Menyatakan variable nama file : outfile = “nama-file”
2. Membuka file untuk diisi : outfh = open(outfile, mode)
3. Rekam data baris-demi-baris : outfh.write(‘…data…’)
4. Menutup file : outfh.close()

Modus perekam ke file adalah:

```
'w'   : merekam ke file, selalu mulai dari awal
'w+'  : merekam ke file, mulai dari awal, dan juga membaca
'a'   : merekam ke file, mulai dari akhir file
'a+'  : menambah isi file, dan dapat dibaca
```

Contoh merekam ke File

```
path="D:\\USER\\Python\\sample.txt"
myfile = open(path, 'a')
print('Ketik kalimat yang akan direkam')
lagi=True
while lagi:
    print('Ketik stop bila selesai')
    baris = input()
    if baris == 'stop':
        lagi = False
    else:
        baris = baris + '\n'
        myfile.write(baris)

myfile.close()
```