Struktur Pemrograman Python (Bagian 2)

1. Nilai dan Tipe data

Sebuah nilai adalah hal yang paling mendasar seperti sebuah huruf atau sebuah angka yang akan di manipulasi oleh program. Nilai 2 (hasil ini didapat, ketika menambahkan 1 + 1), dan "Hello Python!".

Nilai - nilai tersebut berbeda tipe data, yakni 2 sebagai sebuah integer, dan "Hello Python!" sebagai sebuah string, disebut string, karena terdiri dari sebuah kata yang terdiri dari beberapa huruf - huruf. Diidentifikasikan string karena kata-kata tersebut di dalam tanda kutip dua("). Perintah print juga dapat menampilkan integer

```
>>> print 4
4
```

Interpreter dapat memberitahu tipe data dari nilai yang dituliskan, yaitu dengan menggunakan fungsi built_in type() yang ada bersama interpreter.

```
>>> type ("Hello Python!")
<'type string'>

>>> type 5
<'type int'>
```

Angka desimal dengan tanda (.) dibelakang angka dikenal dengan bilangan pecahan atau float karena angka tersebut merepresentasikan suatu bentuk dengan nama floating point.

```
>>> type (6.5)
<'type float'>
```

Contoh dibawah ini adalah tipe string karena berada didalam tanda kutip (”).

```
>>> type ("17.5")
<'type string'>

>>> type ("5")
<'type string'>

>>> a=”Belajar Python”
>>> type a
<'type string'>
```

```
>>> phi=3.14
>>> type phi
<'type float'>
```

1.1 Tipe Data

1.1.1 Number

Tipe data Number merepresentasikan nilai-nilai berupa angka. Python menggolongkan beberapa tipe data umum seperti, Integer (bilangan bulat) dan Floating-point (bilangan desimal) ke dalam tipe data Number.

Contoh :

```
>>> 123 + 789
912
>>> 6 * 34
204
>>> 3 ** 2
9
>>> 3.245 * 3
9.734999999999994
>>> print (5.21 + 6.234)
11.444
```

Untuk perhitungan aritmatika yang menghasilkan nilai desimal antara 0.0 – 0.9 maka akan dilakukan pembulatan ke bawah.

```
>>> 9 / 2
4
```

Operator penugasan (=) digunakan untuk memasukkan nilai kedalam variabel. Tidak ada hasil yang akan muncul sampai statemen selanjutnya.

```
>>> a = 8
>>> b = 3.5
>>> a * b
28.0
```

Nilai dapat di masukkan kedalam beberapa variabel secara simultan.

```
>>> x = y = z = 20
>>> x
20
>>> y
20
```

```
>>> z
20
```

1.1.2 String

Selain angka, python juga mampu melakukan manipulasi string, yang dapat di ekspresikan dengan beberapa cara. Penulisan nilai string pada python menggunakan tanda petik satu (') atau tanda petik dua (“). Contohnya,

```
>>> "Selamat datang"
'Selamat datang'
>>> 'Selamat datang'
'Selamat datang'
```

String literal juga dapat menggabungkan beberapa baris dalam berbagai cara. Dengan menggunakan operator (\n\ ) di akhir kalimat untuk menyambung kalimat selanjutnya yang berada di baris selanjutnya.

```
>>> text = "ini adalah contoh \n\
... penggunaan multiple line\n\
... di python"
>>> print text

ini adalah contoh
penggunaan multiple line
di python
```

Penulisan string untuk multiple line juga dapat dilakukan dengan menggunakan tanda petik dua atau satu sebanyak 3 kali, (“ “ “ atau ''').

```
>>> print """
... my name is python
... i'm an object oriented programming language
... this is an example in using triple quotes
... """

my name is python
i'm an object oriented programming language
this is an example in using triple quotes
```

1.1.2.1 Operasi pada String

Pada umumnya tidak dapat melakukan operasi matematika pada string, walaupun string tersebut berupa angka. Berikut adalah contoh - contoh yang salah.

```
"Belajar Python!" + 1   nama * 5    "5" + 2
```

Tetapi operator tambah (+) dapat berlaku sesama string, walaupun tidak seperti yang dilakukan pada operasi matematika. Pada operator tambah (+) dalam operasi string, operator tambah (+) dapat diasumsikan sebagai penggabungan antara dua string atau lebih. Contohnya :

```
>>> hadir = "Peserta sebanyak 1"
>>> banyak = "100"
>>> print "hadir" + hadir + banyak
hadir Peserta sebanyak 1100

>>> 'universitas' + 'gunadarma'
'universitasgunadarma'
```

Bisa kita lihat dalam penggabungan tersebut, antara string dengan string yang lainnya langsung digabungkan tanpa tanda pemisah, seperti spasi atau [tab].

```
>>>"Belajar" "python"
'Belajarpython'
```

Operator perkalian (*) juga berlaku dalam operasi string, tetapi tidak dapat melakukan perkalian string antar string, melainkan string dengan integer. Operator perkalian ini di analogikan dengan penggandaan string, Misalnya :

```
>>> "ulang" * 3
'ulangulangulang'
```

Penggabungan dan penggandaan string di analogikan dengan penambahan dan perkalian, seperti 4*3 sama dengan 4+4+4, sama halnya seperti ulang*3 dengan ulang+ulang+ulang.Tanda koma (,) dalam operasi string sebagai tanda pemisah (spasi) di antara string.Misalnya :

```
>>> print "ulang", 3, 4, 5
ulang 3 4 5
```

String dalam pemograman bahasa C dianggap sebagai array of character , Karakter pertama pada sebuah string berindex 0, karakter ke-dua berindex 1 dan seterusnya. hal ini juga berlaku di pemograman bahasa Python. Misalnya :

```
>>> kata = "Gunadarma"
>>> kata[0]
'G'
```

```
>>> kata[5]
'a'
>>> kata[0:5]
'Gunad'
>>> kata[4:8]
'darm'
>>> kata[:7]
'Gunadar'
>>> kata[5:]
'arma'
```

Tidak seperti List, elemen anggota karakter dalam string tidak dapat digantikan,

```
>>> kata[2] = 'p'
Traceback (most recent call last):
              File "<stdin>", line 1, in ?
TypeError: object doesn't support item assignment
```

Akan tetapi untuk melakukan penambahan karakter pada string dapat dilakukan dengan operator tambah (+). Misalnya:

```
>>> 'pert' + kata[5:]
'pertama'
```

1.1.3 List

List sering disebut juga array pada bahasa pemrograman lain. List adalah jenis data campuran yang bisa memiliki komponen penyusun yang berbeda-beda. Sebuah list dapat dibuat dengan menggunakan tanda kurung siku [ ]. Anggota list didaftar dalam kurung siku tersebut dan masing-masing dipisahkan oleh tanda koma. Sifat-sifat list bisa didaftar seperti ini:

- Komponen penyusunnya bisa diganti.
- Komponen penyusunya dapat dibaca dan dimanipulasi secara langsung.
- Komponen penyusunnya bisa ditambah.
- Komponen penyusunnya dapat diambil dengan menunjukkan indeksnya atau dengan notasi slice.
- Komponen penyusun sebuah list dapat juga berupa list yang lain.

Contohnya,

```
>>> a = ['lab', 'TI', 2010, 'J1']
>>> a
['lab', 'TI', 2010, 'J1']
>>> a[3]
'J1'
>>> a[2]
2010
>>> a[1:-1]
['TI', 2010]
>>> a[:3] + ['gunadarma', 'ATA', 2009, 2010]
['lab', 'TI', 2010, 'gunadarma', 'ATA', 2009, 2010]
>>> len (a)
4
```

Python dapat mengelompokan beberapa tipe data yang berbeda menjadi satu kelompok, yang kemudian dikenal sebagai List pemisah tanda koma ",".

```
>>> a = ["satu", 2, 3.0, "empat"]
>>> print a
['satu', 2, 3.0, 'empat']
```

Lists bisa dianalogikan sebagai array dan urutan pengaksesannya dimulai dari 0.

```
>>> a[0]
'satu'
>>> a[1]
2
>>> a[-2]
3.0
>>> a[3]
'empat'
```

pengaksesan List pada urutan terakhir dengan nilai -1 .

```
|'satu' |  2   | 3.0  |'empat'|
| a[0]  | a[1] | a[2] | a[3]  |
|'satu' |   2  | 3.0  |'empat'|
| a[-4] | a[-3]| a[-2]| a[-1] |
```

List juga dapat dipisah - pisahkan dan dapat digabungkan, ditambahkan dan lainnya.

```
>>> a[0:2]
['satu', 2]
>>> a[-4:-1]
['satu', 2, 3.0]
```

Tanda titik dua ":" mempunyai argumen [<indeks>:<indeks-n>], berarti dimulai dari indeks sampai indeks ke -n (batas indeks-n, tidak ditampilkan). Di tambahkan, Misalnya :

```
>>> a + ['lima', 'enam']
['satu', 2, 3.0, 'empat', 'lima', 'enam']
```

Penambahannya hanya dapat dilakukan antar lists. Begitupun operasi penggandaan suatu lists, sebagian anggota list ataupun salah satu anggota list.

```
>>> 3*a[:3] + ['tujuh']
['satu', 2, 3.0, 'satu', 2, 3.0, 'satu', 2, 3.0, 'tujuh']
>>> [a[3]] + [a[2]] + ['delapan']
['empat', 3.0, 'delapan']
>>> print a[3]
'empat'
```

Untuk melakukan perubahan terhadap satu anggota atau sebagian anggota list , kita hanya meng-assignkan nilainya, Misalnya :

```
>>> a[2]
3.0
>>> a[2] = a[1] + 5
>>> a
['satu', 2, 7, 'empat']
```

yang berarti nilai a[2] digantikan menjadi nilai a[1] = 2 ditambahkan dengan 5, maka nilai a[2] menjadi 7. Untuk menggantikan sebagian anggota list secara berurutan juga diperbolehkan. Misalnya :

```
>>> a[0:2] = [1,'dua'] #Menggantikan elemen a[0], a[1]
     #Menjadi a[0] = 1, a[1] = 'dua'
>>> print a
[1, 'dua', 7, 'empat']
```

Menghilangkan beberapa elemen anggota.

```
>>> a[0:2] = []
```

```
>>> print a
[7, 'empat']
```

Menyisipkan suatu nilai.

```
>>> a[0:-1] = ['satu']
>>> a
['satu', 'empat']
```

Contoh diatas, berarti menempatkan elemen di antara 0,1 sampai -1. Untuk mengetahui jumlah elemen anggota List, digunakan fungsi built-in len yang berlaku juga untuk menghitung character suatu string.

```
>>> len(a)
2
```

Untuk menambahkan anggota elemen list digunakan metode append yang berlaku pada list. Misalnya :

```
>>> a.append('lima')
>>> a.append('enam')
>>> a
['satu', 'empat', 'lima', 'enam']
List di dalam List.
>>> b = ['tujuh']
>>> a.append(b)
>>> a
['satu', 'empat', 'lima', 'enam', ['tujuh']]
```

Berikut metode - metode yang dapat dilakukan dengan object List :

append (x) : Menambahkan satu elemen anggota dan diletakkan di bagian indeks akhir pada segment LIST

extend (L) : Menggantikan seluruh anggota elemen pada List menjadi seluruh elemen list L

insert(i, x) : Menyisipkan satu elemen anggota List pada posisi tertentu

remove(x) : Menghilangkan satu anggota list

pop([i]) : Menghilangkan salah satu anggota tertentu yang telah ditentukan posisinya

index(x) : Mengembalikan nilai indeks suatu anggota list

count(x) : Memeriksa jumlah x di dalam List

sort() : Mensorting list atau mengurutkan anggota list

reverse() : Kebalikan dari fungsi sort()

LATIHAN

Buatlah program kalender bulan, dengan tampilan sebagai berikut :

Bulan apa [1-12]? 2

Bulan yang Anda pilih ? Februari

Petunjuk : Menggunakan List sebagai daftar nama bulan.

Jawabannya :

```
Bulan = ['Januari', 'Februari', 'Maret', 'April', 'Mei', 'Juni', 'Juli',
'Agustus', 'September','Oktober', 'Nopember', 'Desember']
Pilih = input ("Bulan apa [1-12]? ")
if 1 <= Pilih <= 12 :
     print "Bulan yang Anda pilih ?", Bulan[Pilih-1]
```

1.1.4 Dictionary

Berbeda dengan list yang memakai indeks angka untuk merujuk pada isi variabel, dictionary memakai *key* untuk merujuk pada isi variabelnya. Sifat kedua jenis data ini hanya berbeda dalam beberapa hal saja. Untuk mendeklarasikan sebuah dictionary, Python memakai tanda `{ }`.

```
>>> D = { 'food' : 'spam', 'quality' : 4, 'color' : 'blue'}
>>> D
{'food': 'spam', 'color': 'blue', 'quality': 4}
>>> D['color']
'blue'

>>> status = {}
>>> status['nama']='python'
>>> status['desc']='programming language'
>>> status['age']=6
>>> status
{'nama': 'python', 'age': 6, 'desc': 'programming language'}
```

2. Operator dan Operand

Operator adalah simbol-simbol khusus yang merepresentasikan komputasi seperti penambahan dan perkalian. Nilai yang digunakan oleh operator, kemudian disebut sebagai operand.

Berikut adalah ekspresi - ekspresi yang benar dalam Python.

```
20+3   hour-1   hour*60+minute   minute/60 5**2 (5+9)*(15-7)
```

Simbol-simbol +, -, *, / dan kurung buka dan kurung tutup adalah ekspresi matematika sehari - hari dan dapat berlaku di Python, tanda asteriks (*) berarti perkalian dan tanda asteriks 2(**) berarti tanda eksponen (pangkat).

Pada saat variabel ditempatkan sebagai operand, maka variabel tersebut digantikan dengan nilai dari variabel sebelum perintah tersebut dijalankan.

Operasi berikut menghasilkan hasil yang tidak diinginkan.

```
>>> minute = 66
>>> minute / 60
1
```

nilai yang seharusnya muncul adalah 1.1 bukan 1. Hal ini dikarenakan Python melakukan pembagian integer. Karena integer di bagi integer akan menghasilkan nilai integer, maka harus melakukan pembagian pecahan dengan cara membuat salah satu operand menjadi float.

```
>>> minute = 66.0
>>> minute / 60
1.1000000000000001
```

2.1 Operator Logika

Terdapat 3 operator logika, yaitu and, or, dan not . Arti ketiga operator logika tersebut sama halnya dengan arti yang sebenarnya dalam bahasa inggris, Misalnya x > 8 and x < 20 adalah benar jika kedua kondisi tersebut terpenuhi keduanya dalam arti jika x lebih besar dari 8 dan lebih kecil dari 20 .

x % 2 == 0 or x % 3 == 0 menghasilkan nilai true jika salah satu di antara kedua kondisi tersebut benar, dan nilai itu akan benar jika nilai x dapat dibagi dengan 2 atau 3.

Operator not me-negasikan sebuah ekspresi boolean, jadi not (x > y) mempunyai nilai true, jika if (x > y) mempunyai nilai false .

Operan - operan dalam operator logika harus dalam bentuk ekspresi boolean, tetapi Python tidak terlalu menegaskan hal tersebut. Semua angka yang bukan merupakan bilangan nol (0) di interpretasikan sebagai kondisi true (benar) atau mempunyai nilai 1(satu). Misalnya :

```
>>> x = 5
>>> x and 1
1
>>> y = 0
>>> y and 1
0
```

2.2 Operator Modulus

Operator modulus bekerja pada bilangan integer (dan ekspresi integer) yang berarti bahwa menghasilkan nilai sisa hasil operan pertama dibagi dengan operan kedua. Di Python, operator modulus di wakili simbol persentase (%). Sintaksnya sama dengan operator - operator lain pada umumnya.

```
>>> pembagian = 5 / 3
>>> print pembagian
1
>>> sisa = 5 % 3
>>> print sisa
2
```

Jadi 5 dibagi 3 hasilnya adalah 1 sisa 2.

2.3 Aturan pada operasi

Jika terdapat lebih dari satu operator dalam sebuah ekspresi, maka aturan pada operasi tergantung dari aturan presedansi. Python mengikuti aturan presedansi dari presedansi matematika pada umumnya :

1. Operasi yang berada di dalam kurung memiliki nilai presedensi yang tinggi, dan operasi yang di dalam kurung tersebut di proses terlebih dahulu. Misalnya 4 * (5+4) maka hasilnya sama dengan 36, ditambahkan terlebih dahulu 5 dan 4, kemudian baru dikalikan dengan 4.
2. Kemudian nilai presedensi yang tinggi setelah dalam kurung, adalah tanda pangkat, misalnya 3**2 + 8 adalah 17 bukan 13. dan 3 *1**4 hasilnya sama dengan 3 bukan 12.
3. Pembagian dan perkalian memiliki nilai presedansi yang sama, didahulukan terlebih dahulu dibandingkan dengan penambahan dan pengurangan, misalnya 1 + 3 * 2 hasilnya adalah 7 bukan 8.Penambahan dan pengurangan juga memiliki nilai presedensi yang sama.
4. Apabila terdapat satu atau lebih operator yang memiliki presedensi yang sama, maka yang diproses terlebih dahulu adalah bagian sebelah kiri sampai ke kanan, dalam kata lain di evaluasi dari kiri ke kanan, misalnya 4 + 5 - 2 hasilnya adalah 7, 4 *3 / 2 hasilnya adalah 6.

Operator Aritmatika

Operator	Deskripsi	Contoh	Hasil
*	Perkalian	7 * 3	21

/	Pembagian	7 / 3	2
%	Modulus	7 % 3	1
+	Penjumlahan	7 + 3	10
-	Pengurangan	7 - 3	4

Operator Perbandingan

Operator	Deskripsi	Contoh	Hasil
>=	Lebih besar atau sama dengan	7 >= 9	FALSE
<=	Lebih kecil atau sama dengan	3 <= 8	TRUE
!=	Tidak sama dengan	1 != 10	TRUE
<	Lebih kecil	14 < 6	FALSE
>	Lebih besar	5 > 3	TRUE
==	Sama dengan	4 == 4	TRUE

Operator Penugasan

Operator	Contoh	Sama dengan
*=	x *= 100	x = x * 100
/=	x /= 100	x = x / 100
%=	x %= 100	x = x % 100
+=	x += 100	x = x + 100
-=	x -= 100	x = x - 100