1 Modul 1

1.1 Pendahuluan

Pada modul ini, kita menggunakan Python versi 3.6. Anda dapat mendownload Python di https://www.python.org/downloads/release/python-360/, pilihan "Windows x86-64 executable installer".

Beberapa karakteristik dari bahasa ini:

- Python *case-sensitive*, artinya perbedaan huruf besar dan huruf kecil menyebabkan perbedaan makna.
- Python sangat memerhatikan indentasi dan pergantian baris. Kesalahan indentasi dapat menyebabkan gagal *compile* hingga kesalahan program.
- Variabel di python bersifat implisit dan dinamis. Artinya, sebuah variabel tidak perlu dideskripsikan tipe datanya. Namun di modul ini kita tetap mempelajari tipe data yang ada.

1.2 Input dan Output

Dalam python, program untuk menulis `"Hello, World!"` ke layar adalah seperti berikut:

```
print("Hello, World!")            # (1)

print("Ini program", end="")      # (2)
print(" pertama saya")            # (3)

# Ini adalah sebuah komentar.

# Semua yg ada setelah tanda pagar (#)
# akan diabaikan oleh interpreter.

'''
Selain itu, kita juga bisa membuat komentar
multi baris dengan tiga petik ('''')
'''
```

Bagian yang ditandai nomor 1 bertugas menuliskan `"Hello, World!"` ke layar. Sintaks `print` akan otomatis memindahkan baris setelah selesai menulis ke layar, kecuali disertai parameter `end=""`. Berarti, output program di atas adalah `"Hello, World!"` diikuti dengan `"Ini program pertama saya"` di baris selanjutnya.

Untuk melakukan input, kita membutuhkan penampung untuk menyimpan data yang diinputkan. Sebagai contoh, di bawah ini adalah program yang menerima input dan menuliskan ulang yang dimasukkan.

```
S = input("Masukkan kalimat: ")                    # 1
print("Anda memasukkan kalimat " + S)              # 2

N = int(input("Masukkan sebuah angka: "))          # 3
```

```
print("Jika angka Anda ditambah 5, hasilnya " + str(N + 5)) # 4
```

Pada bagian nomor (1) dan (3), program membaca input dari user dan dimasukkan ke variabel `S` dan `N`. Lalu, bila ingin program menerima sebuah bilangan (sehingga bisa dijumlah / dikali / lainnya), kita bungkus `input()` dengan `int()`. Selain `int`, `string` (kumpulan karakter), Python dapat menerima tipe data `float` (bilangan real), dan masih banyak lagi.

1.3 Tipe Data

Ada beberapa macam tipe data, namun dalam PTI-B kita hanya akan banyak menggunakan tipe data berikut:

`bool`	Boolean	`True` atau `False`
`int`	Bilangan bulat	seluruh bilangan bulat
`float`	Bilangan real	seluruh bilangan real
`string`	Teks	kumpulan karakter

Contoh penggunaan:

```
B = True          # Boolean
I = 123456789012  # Bilangan bulat
R = 2.331973      # Bilangan real
S = "abc"         # Teks
S = 'def'
```

Perhatkan pada variabel `S`, Anda bebas menggunakan petik satu (') atau petik dua (")

1.4 Operator

1.4.1 Operator Aritmatika

Operator	Deskripsi	Contoh
`+`	Penjumlahan	`2 + 3` bernilai `5`
`-`	Pengurangan	`1 - 8` bernilai `-7`
`*`	Perkalian	`5 * 6` bernilai `30`
`/`	Pembagian	`13 / 5` bernilai `2.6`
`//`	Pembagian (dibulatkan ke bawah)	`13 // 5` bernilai `2`
`%`	Sisa Bagi / Modulo	`13 % 5` bernilai `3`

1.4.2 Operator Assignment

Operator	Deskripsi	Contoh
`=`	Assignment	`N = 5`
`+=`	Penjumlahan	`N += 5`, N akan ditambah 5.
`-=`	Pengurangan	`N -= 5`, N akan dikurang 5.
`*=`	Perkalian	`N *= 5`, N akan dikali 5.
`/=`	Pembagian	`N /= 5`, N akan dibagi 5.
`//=`	Pembagian (dibulatkan ke bawah)	`N //= 5`, N akan dibagi 5 (dibulatkan ke bawah).
`%=`	Sisa Bagi / Modulo	`N %= 5`, N akan dimodulo 5.

1.4.3 Operator Relasional

Operator	Deskripsi	Contoh True	Contoh False
`==`	Sama dengan	`2 == 2`	`2 == 3`
`!=`	Tidak sama dengan	`3 != 2`	`3 != 3`
`<`	Kurang dari	`2 < 3`	`2 < 2`
`>`	Lebih dari	`3 > 2`	`3 > 2`
`<=`	Kurang dari sama dengan	`2 <= 2`	`1 <= 3`
`>=`	Lebih dari sama dengan	`6 >= 5`	`2 >= 4`

1.4.4 Operator Logika

Operator	Deskripsi	Contoh True	Contoh False
`and`	Dan	`(1 < 2)and (3 == 3)`	`(1 == 2)and (3 == 3)`
`or`	Atau	`(1 < 2)or (4 == 3)`	`(3 < 2)or (2 == 3)`
`not`	Negasi	`not (3 < 2)`	`not (1 > 2)`

1.5 Percabangan

Dalam pemrograman, terdapat percabangan. Dengan demikian, program kita dapat berperilaku tergantung input user. Misal kita buat program yang memeriksa apakah sebuah bilangan positif:

```
print("Masukkan nilai N: ", end="")
N = int(input())

if (N > 0):
    print(str(N) + " bilangan positif")
```

Lalu, jika kita ingin menuliskan kebalikannya:

```
...
    if (N > 0):
        print(str(N) + " bilangan positif")
    else: # N <= 0
        print(str(N) + " bilangan bukan positif")
...
```

Namun, kita tahu kadang bilangan bisa nol atau negatif, jadi perlu kita tambahkan:

```
...
    if (N > 0):
        print(str(N) + " bilangan positif")
    elif (N < 0):
        print(str(N) + " bilangan negatif")
    else: # N == 0
        print(str(N) + " bilangan nol")
...
```

Perhatikan juga kalau kita bisa membuat else ini berulang sampai yang kita mau. Selain itu, kita juga bisa meletakkan `if` di dalam `if`.

```
...
    if (N >= 0):
        if (N > 0):
            print(str(N) + " bilangan positif")
        else: # N == 0
            print(str(N) + " bilangan nol")
    else: # N < 0
        print(str(N) + " bilangan negatif")
...
```

2 Modul 2

2.1 Pengulangan

Pada pemrograman, sering kali dibutuhkan pemrosesan berulang-ulang untuk mencapai suatu hasil tertentu. Apabila pengulangan ini dilakukan secara manual ukuran file program akan menjadi terlalu besar. Contoh sederhana adalah jika kita ingin menuliskan Hello World di layar sebanyak 1000 kali, maka akan dibutuhkan paling tidak 1000 baris perintah. Menggunakan sintaks pengulangan, persoalan tersebut dapat diselesaikan hanya menggunakan beberapa baris program.

```
for i in range(1000):
    print("Hello world")
```

2.1.1 While Loop

Salah satu sintaks yang looping / pengulangan yang sering digunakan adalah sintaks While-Do. Program akan mengecek sebuah kondisi yang diberikan terlebih dahulu sebelum menjalankan statement yang ada di dalamnya.

Berikut adalah program yang menerima a dan b dan menuliskan $a, a+1, a+2, \ldots, b-1, b$.

```
a = int(input())
b = int(input())
i = a
while (i <= b):
    print(i)
    i += 1
```

2.1.2 For Loop

Bentuk looping yang kedua adalah bentuk For. Bentuk ini umumnya digunakan untuk pengulangan yang sudah diketahui jumlahnya. Namun, for loop juga dapat dibuat menggunakan while-do loop.

Berikut adalah program yang menerima a dan b dan menuliskan $a, a+1, a+2, \ldots, b-1, b$.

```
a = int(input())
b = int(input())
for i in range(a, b+1):
    print(i)
```

`range` dapat digunakan dengan satu, dua, atau tiga parameter. Untuk lebih jelasnya, perhatikan potongan kode berikut:

```
range(10)          # 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
range(2, 5)        # 2, 3, 4
```

```
range(7, 2, -1)    # 7, 6, 5, 4, 3
range(2, 8, 2)     # 2, 4, 6
```

2.2 Perulangan bersarang

Perhatikan pula, perulangan dapat dilakukan di dalam perulangan. Sebagai contoh, berikut adalah program untuk membuat pola persegi.

```
n = int(input())
for i in range(n):
    for j in range(n):
        print('*', end='')
    print()
```

2.3 Array

Array adalah variabel dengan satu buah nama, tetapi mengandung banyak nilai. Akses nilai-nilainya dilakukan dengan indeks.

Perhatikanlah contoh berikut!

Indeks	0	1	2	3	4	5	6	7	8	9
A	3	10	5	7	11	19	23	35	37	12

- `A[0] = 3`
- `A[1] = 10`
- `A[7] = 35`

Pada contoh diatas, kita memiliki sebuah variabel yang bernama A. Variabel A tersebut memiliki 10 buah nilai, dimana nilai-nilai tersebut dapat diakses dengan indeks. Untuk mengakses indeks ke x, gunakanlah `A[x]`. Dan nilai `A[x]` itu bisa kita anggap sebagai variabel yang berdiri sendiri. Konsep inilah yang kita sebut dengan array. Perhatikan pula bahwa indeks dimulai dari 0.

2.3.1 Deklarasi Array

Karena array juga merupakan sebuah variabel, maka array juga memerlukan deklarasi seperti variabel lainnya.

Contoh deklarasi array:

- `x = [0 for x in range(n)]` membuat array berukuran `n` dengan isi 0.
- `x = ['*'for x in range(100)]` membuat array berukuran `100` dengan isi karakter `'*'`.

Untuk contoh tersebut array `A` yang terdefinisi adalah `A[0]`, `A[1]`, `A[2]`, ..., `A[9]` Mengakses nilai indeks di luar batasan tersebut akan menyebabkan runtime error. Oleh karena itu, tentukanlah rentang indeks yang akan digunakan saat deklarasi dengan tepat (sesuai dengan kebutuhan).

2.3.2 Array dan Variabel

Suatu array dapat kita anggap sebagai variabel, sehingga segala jenis operasi pada variabel juga berlaku pada array. Sebagai contoh, kita memiliki suatu array

```
tabel = [0 for i in range(10)]
```

Maka array tabel tersebut akan terdefinisi untuk indeks 0 sampai dengan indeks 9. Maka kita bisa melakukan instruksi

```
tabel[2] = int(input())
```

Jika diberikan 5 buah bilangan , dan kita perlu menyimpan bilangan tersebut pada tabel, kita bisa melakukan

```
tabel[0] = int(input('Masukkan nilai ke-0: '))
tabel[1] = int(input('Masukkan nilai ke-1: '))
tabel[2] = int(input('Masukkan nilai ke-2: '))
tabel[3] = int(input('Masukkan nilai ke-3: '))
tabel[4] = int(input('Masukkan nilai ke-4: '))
```

Namun, cara menginput tersebut kurang efisien. Akan lebih efisien jika kita menginput menggunakan perulangan (looping).

```
for i in range(5):
    tabel[i] = int(input('Masukkan nilai ke-' + str(i) + ': '))
```