Input/Output pada Python

Di bagian ini Anda akan mempelajari tentang mekanisme Input/Output, misalnya meminta masukan dari pengguna, menyimpan nilai pada variabel dan mencetak nilai ke layar.

Setelah sebelumnya mempelajari tipe data, selanjutnya Anda akan belajar tentang variabel. Variabel adalah sebuah tempat (di memori komputer) untuk menyimpan nilai dengan tipe data tertentu.

Untuk memberikan nilai pada sebuah variabel, kita menggunakan operator "=", antara nama variabel dengan nilai yang ingin disimpan.

Misalnya: *x = 1.*

Artinya kita akan menyimpan nilai 1 (tipe int) ke variabel x.

Output

Print

Seperti dicontohkan dalam beberapa sample code sebelumnya, fungsi print() adalah cara output langsung ke konsol/layar.

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

Output:

```
Hello, World!
```

Memasukkan nilai variabel pada string

Untuk memasukkan nilai variabel pada string, Python memiliki dua cara. Cara yang pertama adalah langsung menggabungkan variabel pada statement print().

```
1. x = 100
2. print('Nilai x adalah', x)
```

Output:

Untuk menampilkan text (string), bisa menggunakan mekanisme string format. Misalnya yang pertama:

```
1. print('hai {}'.format('bro'))
```

Cara yang kedua mirip dengan sintaks C/C++, yakni menggunakan operator “%” yang ditambahkan dengan "argument specifiers", misalnya "%s" and "%d". Contohnya saat kita ingin menambahkan nama kita pada string hello:

```
1. nama = "Dicoding"
2. print("Halo, %s!" % nama)
```

Output:

```
Halo, Dicoding!
```

Contoh menambahkan string dan integer:

```
1. nama = "Dicoding"
2. umur = 5
3. print("Umur %s adalah %d tahun." % (nama, umur))
```

Output:

```
Umur Dicoding adalah 5 tahun.
```

Contoh menambahkan objek selain string (otomatis dikonversi):

```
1. angka = [7, 9, 11, 13]
2. print("Angka saya: %s" % angka)
```

Output:

```
Angka saya: [7, 9, 11, 13]
```

Beberapa argument specifier yang umum digunakan:

```
1. %s - String
2. %d - Integers
3. %f - Bilangan Desimal
4. %.<digit>f - Bilangan desimal dengan sejumlah digit angka dibelakang koma.
5. %x/%X - Bilangan bulat dalam representasi Hexa (huruf kecil/huruf besar)
```

Contoh mencetak representasi Hexa (bilangan basis 16):

```
1. a, b = 10, 11
2. print('10: %x and 11: %X' % (a, b))
```

Output:

```
10: a and 11: B
```

Referensi yang dapat dipelajari:

https://docs.python.org/id/3.8/library/string.html#format-specification-mini-language

Input

input()

Untuk memungkinkan user memberikan input pada program Anda, gunakan fungsi input(), dengan argumen dalam kurung () adalah teks yang ingin ditampilkan (*prompt*) dan variabel sebelum tanda sama dengan (=) adalah penampung hasil dari input pengguna:

```
1. nilai = input('Masukkan angka : ')
```

Output:

```
Masukkan angka : 90
```

```
1. print(nilai)
```

```
'90'
```

Secara default, input dari user adalah string (walaupun pada contoh di atas, 90 sebenarnya dimaksudkan sebagai integer) yang ditandai dengan petik. Untuk itu diperlukan fungsi konversi yang akan dibahas pada modul-modul selanjutnya, misalnya int() dan float().

```
1. print(int(nilai))
```

Output:

```
90
```

```
1. print(float(nilai))
```

Output:

```
90.0
```

Jika input merupakan string berisi ekspresi matematika, maka konversi dengan **int()** atau **float()** akan menghasilkan error. Anda dapat menggunakan fungsi **eval()** yang sekaligus juga berfungsi menyelesaikan ekspresi matematika. Anda akan mempelajari lebih jauh mengenai fungsi pada modul Fungsi.

```
1. print(int('90+10'))
```

Output:

```
Traceback (most recent call last):
File "<string>", line 301, in runcode
File "<interactive input>", line 1, in <module>
ValueError: invalid literal for int() with base 10: '90+10'
```

Output:

```
100
```

Command-line arguments

Python memungkinkan Anda untuk membuat sebuah "skrip" berupa deretan kode program kemudian disimpan dalam sebuah berkas dengan nama akhiran .py (misal: skrip.py).

Berkas ini dapat dipanggil sebagai skrip di konsol atau command prompt, serta dapat ditambahkan parameter tambahan saat memanggil skrip tersebut.

```
$ python test.py arg1 arg2 arg3
```

Hal ini difasilitasi oleh module sys yang telah dibawa secara default pada Python. Untuk menggunakannya, jangan lupa lakukan import terlebih dahulu:

```
import sys
```

Utamanya fungsi yang akan digunakan adalah sys.argv yang memuat seluruh argumen yang diterima. Anda juga dapat menggunakan len(sys.argv) untuk mengetahui banyaknya argumen yang ditampung.

Contoh, sebuah berkas test.py yang akan menambahkan tiga argumen:

```
import sys
print('Jumlah arguments:', len(sys.argv), 'arguments.')
print('Argument List:', str(sys.argv))
print(sys.argv[1])
```

Jalankan pada konsol/terminal/command prompt:

```
$ python test.py arg1 arg2 arg3
```

Output:

Argument List: ['test.py', 'arg1', 'arg2', 'arg3']

arg1

Sebelumnya

Selanjutnya

dicoding

Dicoding Space
Jl. Batik Kumeli No.50, Sukaluyu,
Kec. Cibeunying Kaler, Kota Bandung
Jawa Barat 40123

Penghargaan

image
click bila
belum muncul

DIBANTU