

MATA KULIAH : DASAR PEMROGRAMAN
SESI PERTEMUAN : V (LIMA)
MATERI : INPUT/OUTPUT & STRING
DOSEN : ALUN SUJJADA, S.KOM., M.T

A. INPUT

Input (dalam konteks pemrograman) merupakan sebuah data, informasi, atau nilai apa pun yang dikirimkan oleh user kepada komputer untuk diproses lebih lanjut. User melakukan proses input melalui media atau perangkat masukan seperti keyboard, mouse, kamera, mikrofon dan lain sebagainya. Proses input dapat dilakukan dengan menggunakan cara statis maupun dinamis. Secara statis berarti setiap variabel diberikan nilai menggunakan *hard code* atau memberikan nilai langsung kepada variabel pada kode program, sehingga nilainya akan tetap. Perhatikan contoh kode program 5.1 berikut ini:

```
1. panjang = 10
2. lebar = 5
3. luas = panjang * lebar
4. print(f"luas={luas}")
```

Kode Program 5.1 Input nilai statis

Jika kode program di atas dijalankan, maka hasil output yang didapatkan adalah **luas = 50**. Lalu bagaimana jika ingin menghitung luas persegi panjang selain 10 dan 5? Maka yang dapat dilakukan adalah mengubah kode program, mengganti nilai dari variabel panjang dan lebar lalu kembali mengeksekusi program dari awal. Tapi jika ingin tanpa harus mengubah kode program, maka jawabannya adalah meminta user untuk melakukan input data, yang dikenal dengan istilah input dinamis.

B. MEMBUAT INPUT

Bahasa pemrograman python memberikan fitur untuk membuat sebuah input dari *user* yaitu dengan cara memanggil fungsi dari python yang bernama fungsi **input()**. Fungsi **input()** menerima satu buah parameter string, yang mana parameter tersebut akan ditampilkan di layar sebelum user memasukkan sebuah data atau dapat disebut sebagai *label* seperti pada contoh kode program 5.2 berikut ini:

```
1. nama = input("Isikan nama anda : ")
2. print(f"{nama} adalah nama yang bagus")
```

Kode Program 5.2 Input nilai dinamis

Sehingga jika kode program tersebut dijalankan maka akan tampil output seperti gambar 5.1 berikut ini:

```
nama = input("Isikan nama anda : ")
print(f"{nama} adalah nama yang bagus")

Isikan nama anda : 
```

Gambar 5.1 Tampilan program penggunaan fungsi input

Jika pada box input tersebut diisi dengan “Rudi” maka akan tampil output “Rudi adalah nama yang bagus”. Tetapi perlu diingat bahwa tipe data dari sebuah input adalah string meskipun nilai yang diisikan adalah angka. Perhatikan kode program 5.3 berikut ini:

```
1. nilai_program = input("Isikan Nilai Pemrograman : ")
2. print(f"Nilai Pemrograman :{nilai_program}")
3. print(f"Tipe data dari nilai_program adalah {type(nilai_program)}")
```

Kode Program 5.3 Pengecekan tipe data dari input

Jika kode program 5.3 tersebut dijalankan, maka akan tampil output seperti pada gambar 5.2 berikut ini:

```
nilai_program = input("Isikan Nilai Pemrograman : ")
print(f"Nilai Pemrograman :{nilai_program}")
print(f"Tipe data dari nilai_program adalah {type(nilai_program)}")

Isikan Nilai Pemrograman : 90
Nilai Pemrograman :90
Tipe data dari nilai_program adalah <class 'str'>
```

Gambar 5.2 Hasil tipe data string

Sehingga ketika nilai dari sebuah input akan dilakukan proses operasi aritmatika, maka harus dilakukan proses konversi (casting) dari tipe data string ke integer. Jika hal tersebut tidak dilakukan, maka untuk operator (+) akan dianggap sebagai *concat* atau menggabungkan string seperti pada contoh kode program 5.4 berikut ini:

```
1. nilai_satu = input("Isikan nilai 1 : ")
2. nilai_dua = input("Isikan nilai 2 : ")
3. hasil_jumlah = nilai_satu + nilai_dua
4. print(f"Hasil penjumlahan dari {nilai_satu} dan {nilai_dua} adalah {hasil_jumlah}")
```

Kode Program 5.4 Penjumlahan input tanpa *casting*

Jika kode program 5.4 tersebut dijalankan, maka akan tampil output seperti pada gambar 5.3 berikut ini:

```
nilai_satu = input("Isikan nilai 1 : ")
nilai_dua = input("Isikan nilai 2 : ")
hasil_jumlah = nilai_satu + nilai_dua
print(f"Hasil penjumlahan dari {nilai_satu} dan {nilai_dua} adalah {hasil_jumlah}")

Isikan nilai 1 : 5
Isikan nilai 2 : 6
Hasil penjumlahan dari 5 dan 6 adalah 56
```

Gambar 5.3 Output string

Ketika operator aritmatika yang digunakan adalah selain (+), maka akan terjadi *error* program seperti pada gambar 5.4 berikut ini:

```
nilai_satu = input("Isikan nilai 1 : ")
nilai_dua = input("Isikan nilai 2 : ")
hasil_kali = nilai_satu * nilai_dua
print(f"Hasil perkalian dari {nilai_satu} dan {nilai_dua} adalah {hasil_kali}")

Isikan nilai 1 : 5
Isikan nilai 2 : 6

-----
TypeError                                Traceback (most recent call last)
<ipython-input-23-0494bd85469c> in <module>
      1 nilai_satu = input("Isikan nilai 1 : ")
      2 nilai_dua = input("Isikan nilai 2 : ")
----> 3 hasil_kali = nilai_satu * nilai_dua
      4 print(f"Hasil perkalian dari {nilai_satu} dan {nilai_dua} adalah {hasil_kali}")

TypeError: can't multiply sequence by non-int of type 'str'
```

Gambar 5.4 Error program

Lalu bagaimana solusinya jika ingin melakukan operasi aritmatika dari dua buah bilangan hasil dari input user? Caranya adalah dengan mengkonversi (*casting*) tipe data. Konversi tipe data **string** menjadi **integer** menggunakan fungsi **int("string")** seperti pada contoh kode program 5.5 berikut ini:

```
1. nilai_satu = input("Isikan nilai 1 : ")
2. nilai_dua = input("Isikan nilai 2 : ")
3. hasil_kali = int(nilai_satu) * int(nilai_dua)
4. print(f"Hasil perkalian dari {nilai_satu} dan {nilai_dua} adalah {hasil_kali}")
```

Kode Program 5.5 Proses konversi data (casting)

C. OUTPUT

Keluaran (*Output*) merupakan setiap nilai atau data atau informasi yang dikirimkan oleh mesin komputer kepada user setelah tahap pemrosesan tertentu.

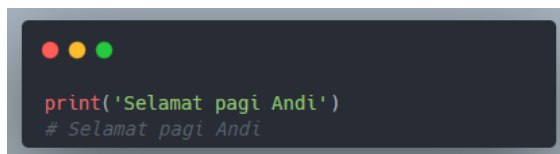
Secara umum *output* bisa berupa teks, gambar, suara, atau bahkan berupa informasi yang dicetak di atas kertas, dan sebagainya. Adapun jenis *output* pada python adalah:

1. *Output* yang ditampilkan di **layar (CLI)**
2. *Output* yang dikeluarkan (ditulis) **dalam bentuk file**

Pada materi ke 5 (lima) kali ini hanya akan membahas jenis *output* yang pertama yaitu *output* yang ditampilkan di layar (secara CLI), sehingga pembahasannya akan lebih banyak mengenai variasi cara menggunakan fungsi `print()` pada python. Adapun fungsi yang digunakan untuk mencetak *output* adalah:

1. `print()`

Untuk membuat output di layar, perintah atau fungsi yang paling sering digunakan adalah fungsi `print()` seperti pada contoh berikut ini:



```
print('Selamat pagi Andi')  
# Selamat pagi Andi
```

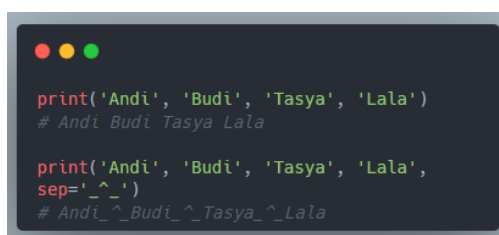
2. Multi argumen dengan pemisah koma



```
print('Selamat pagi', 'Budi!')  
# Selamat pagi Budi!  
  
print('Selamat', 'pagi', 'Doni!')  
# Selamat pagi Doni!
```

Pada model `print` dengan pemisah koma, python akan menambahkan pemisah berupa spasi secara default.

3. Pemisah pada fungsi `print()`



```
print('Andi', 'Budi', 'Tasya', 'Lala')  
# Andi Budi Tasya Lala  
  
print('Andi', 'Budi', 'Tasya', 'Lala',  
      sep='^_^')  
# Andi_^Budi_^Tasya_^Lala
```

Ketika kita memanggil fungsi `print()` untuk menampilkan multi argumen, python akan otomatis menambahkan karakter spasi sebagai pemisah antar argumen tersebut. Jika tidak ingin pemisah spasi, maka dapat menambahkan parameter `sep` (separator) saat memanggil `print()`.

4. Karakter akhir pada fungsi print()

Selain parameter sep, python juga bisa menggunakan parameter lain yaitu parameter end. Parameter end berfungsi untuk mengganti karakter terakhir bawaan yang dicetak di layar. Jadi secara bawaan, setiap kali memanggil fungsi print() untuk mencetak sesuatu, python akan mencetak karakter ganti baris (\n) di setiap output.

```
print('Merkurius')
print('Venus')
print('Bumi')
```

Sehingga *output* yang dihasilkan adalah

```
Merkurius
Venus
Bumi
```

Sedangkan dengan parameter end, python bisa mengganti karakter ganti baris bawaan python dengan karakter lain sesuai keinginan *user*.

```
print('Merkurius', end=' ')
print('Venus', end=' ')
print('Bumi', end=' ')
```

Python juga memungkinkan penggunaan parameter end dan sep secara bersamaan.

```
print('1', '2', '3', sep="*", end="@")
# 1*2*3@

print('1', '2', '3', sep="🐶", end="🐱\n")
# 1🐶2🐶3🐱
```

5. String format

Selain dengan operator +, python juga dapat menyisipkan sebuah variabel atau nilai kedalam sebuah string dengan memanggil fungsi str.format().

```
a = 5
b = 3

print(a, '+', b, '=', a + b)
# 5 + 3 = 8

print('{} + {} = {}'.format(a, b, a + b))
# 5 + 3 = 8
```

6. Indeks Parameter

Selain menggunakan {}, python juga dapat mendefinisikan index supaya yang ditampilkan tidak sesuai urutan parameter.

```
print('{} dan {}'.format('Huda', 'Budi'))  
# Huda dan Budi  
  
print('{1} dan {0}'.format('Huda', 'Budi'))  
# Budi dan Huda
```

7. Key String

Selain menggunakan index seperti {0} atau {1} dan seterusnya, python juga bisa menggunakan kunci atau key berupa string.

```
print('Halo {namaDepan} {namaBelakang}'  
      .format(namaDepan='Agus', namaBelakang='Priyono'))  
# Halo Agus Priyono  
  
print('Halo {namaDepan} {namaBelakang}'  
      .format(namaBelakang='Priyono', namaDepan='Agus'))  
# Halo Agus Priyono
```

D. STRING

Pada dunia pemrograman, ada satu tipe data yang berfungsi untuk menyimpan kumpulan dari karakter-karakter. Karakter-karakter tersebut tersusun menjadi satu-kesatuan membentuk sebuah kata, kalimat, atau paragraf yang bahkan bisa terbentuk dari digit dan juga numerik. Pada python, String dibuat dengan kombinasi tanda petik tunggal (') atau tanda petik dua (").

```
nama = 'Wahid Abdulloh'  
asal = "Indonesia"
```

Python dapat mengambil karakter pada index ke-i pada string seperti ini:

```
nama = 'Lendis Fabri'  
print(nama[4]) # output: i  
print(nama[7]) # output: F  
print(nama[-1]) # output: i  
print(nama[-3]) # output: b
```

Selain itu python juga dapat melakukan slicing string

```
judul = 'Pelajaran Matematika Untuk SD'

print(judul[0:5]) # output: Pelaj
print(judul[:10]) # output: Pelajaran
print(judul[10:15]) # output: Matem
print(judul[-1:-3]) # output:
print(judul[:-3]) # output: Pelajaran Matematika Untuk
print(judul[-5:]) # output: uk SD
```

Untuk melakukan slicing atau pemotongan string dapat menggunakan range of index yang diapit oleh dua kurung siku ([]) dan dipisahkan oleh tanda titik dua (:).

D. MODIFIKASI STRING

Python mempunyai beberapa fungsi bawaan yang dapat digunakan untuk memodifikasi string diantaranya yaitu:

1. **upper()** digunakan untuk mengubah string menjadi huruf besar

```
#Code here
mata_kuliah = 'Dasar Pemrograman Python'
print(mata_kuliah.upper())

DASAR PEMROGRAMAN PYTHON
```

2. **lower()** digunakan untuk mengubah string menjadi huruf kecil

```
#Code here
mata_kuliah = 'Dasar Pemrograman Python'
print(mata_kuliah.lower())

dasar pemrograman python
```

3. **strip()** digunakan untuk menghilangkan spasi sebelum maupun sesudah teks

```
#Code here
mata_kuliah = ' Dasar Pemrograman Python '
print(mata_kuliah.strip())

Dasar Pemrograman Python
```

4. **replace()** digunakan untuk mengganti string dengan string yang lainnya

```
#Code here
mata_kuliah = ' Dasar Pemrograman Python '
print(mata_kuliah.replace('D','P'))

Pasar Pemrograman Python
```

5. **split()** digunakan untuk memisahkan string dan menampung data dalam bentuk list

```
#Code here
mata_kuliah = ' Dasar Pemrograman Python '
print(mata_kuliah.split())

['Dasar', 'Pemrograman', 'Python']
```

```
#Code here
mata_kuliah = ' Dasar-Pemrograman-Python '
print(mata_kuliah.split('-'))

[' Dasar', 'Pemrograman', 'Python ']
```