2. Operator, Input, dan Output

Operator

Apa itu Operator?

Operator di dalam Python adalah **simbol khusus** yang berfungsi untuk **menjalankan suatu operasi tertentu**, baik operasi aritmatika maupun operasi logika. Sedangkan **nilai yang dioperasikan** oleh operator **dinamakan sebagai operan**.

Berikut ini salah satu contoh paling sederhana dari operator aritmatika pada Python:

```
>>> 10 + 5
15
```

Pada kode program di atas, tanda `+` adalah sebuah operator. Sedangkan angka `10` dan `5` keduanya merupakan operan.

Dari operasi tersebut, didapatkanlah sebuah hasil akhir berupa nilai integer yaitu `15`.

Jenis-jenis operator pada python

Terdapat 7 jenis operator pada python:

- Operator aritmatika
- Operator komparasi atau perbandingan
- Operator penugasan
- Operator logika
- Operator keanggotaan (*membership*)
- Operator identitas
- Operator *bitwise*

Operator aritmatika

Operator matematika adalah operator yang kita gunakan untuk menghitung operasi matematika, mulai dari penjumlahan, pengurangan, perkalian, perpangkatan, dan lain sebagainya.

Berikut ini tabel operator aritmatika pada python.

Simbol	Nama	Tugas	Contoh
+	Penjumlahan	Menjumlahkan nilai operan	5 + 1 = 6
-	Pengurangan	Mengurangkan nilai operan	10 - 10 = 0
*	Perkalian	Mengalikan nilai operan	2 * 2 = 4
/	Pembagian	Membagi nilai operan	100 / 5 = 50
%	Modulus	Menghitung sisa hasil bagi dari operan	10 / 3 = 1
**	Perpangkatan	Menghitung pangkat dari operan	2 ** 3 = 8
//	Pembagian Bulat	Membagi operan lalu membulatkannya dengan menghapus angka dibelakang koma	10 // 3 = 3

Silakan dioba satu-persatu dari masing-masing operator di atas.

```
a, b = 10, 3

print(a, '+', b, '=', a + b)
print(a, '-', b, '=', a - b)
print(a, '*', b, '=', a * b)
print(a, '/', b, '=', a / b)
print(a, '%', b, '=', a % b)
print(a, '**', b, '=', a ** b)
print(a, '//', b, '=', a //b)
```

Output:

```
10 + 3 = 13
10 - 3 = 7
10 * 3 = 30
10 / 3 = 3.3333333333333335
10 % 3 = 1
10 ** 3 = 1000
10 // 3 = 3
```

Operator komparasi atau perbandingan

Operator perbandingan adalah operator yang bertugas untuk membandingkan antar dua operan. Apakah operan 1 **lebih besar** dari pada operan 2, atau **apakah keduanya sama**? Dan lain sebagainya.

Berikut ini adalah tabel operator perbandingan pada Python.

Simbol	Nama	Contoh	Hasil
`>`	Lebih dari	`5 > 5`	`False`
`<`	Kurang dari	`2 < 4`	`True`
`==`	Sama dengan	`10 == 10`	`True`
`!=`	**Tidak** sama dengan	`5 != 5`	`False`
`>=`	Lebih dari **atau** sama dengan	`10 >= 10`	`True`
`<=`	Kurang dari **atau** sama dengan	`9 <= 10`	`True`

Agar menjadi lebih jelas, silakan dicoba satu persatu

```
# buat variabel a dan b dengan teknik squence ordering
a, b = 5, 10

print(a, '>', b, '=', a > b)
print(a, '<', b, '=', a < b)
print(a, '==', b, '=', a == b)
print(a, '!=', b, '=', a != b)
print(a, '>=', b, '=', a >= b)
print(a, '<=', b, '=', a <= b)
```

Output dari kode program di atas adalah:

```
5 > 10 = False
5 < 10 = True
5 == 10 = False
5 != 10 = True
5 >= 10 = False
5 <= 10 = True
```

Operator penugasan

Operator penugasan adalah operator yang digunakan untuk **memberikan sebuah tugas** terhadap suatu variabel. Atau dalam bahasa yang lebih manusiawi: operator penugasan adalah operator yang **berfungsi untuk memberikan nilai** ke dalam sebuah variabel.

Sebenarnya operator penugasan ini **hanya ada 1 saja**, yaitu operator `=`.

Akan tetapi, **ada banyak *variant shortcut*** yang memudahkan kita untuk melakukan operasi aritmatika atau operasi *bitwise* bersamaan dengan operasi penugasan.

Berikut ini adalah tabel operator penugasan pada Python.

Operator	Contoh	Sama dengan
=	a = 10	a = 10
+=	a += 5	a = a + 5
-=	a -= 3	a = a - 3
*=	a *= 6	a = a * 6
/=	a /= 8	a = a / 8
%=	a %= 9	a = a % 9
//=	a //= 6	a = a // 6
**=	a **= 1	a = a ** 1
&=	a &= 2	a = a & 2
|=	a |= 3	a = a | 3
^=	a ^= 4	a = a ^ 4
>>=	a »= 4	a = a » 4
<<=	a «= 2	a = a « 2

Silakan dicoba masing-masing dari operator penugasan di atas.

```
# penugasan pertama
a = 10
print('a = 10 -> ', a)

a += 5
print('a += 5 -> ', a)

a -= 3
print('a -= 3 -> ', a)

a *= 6
print('a *= 6 -> ', a)

a /= 8
print('a /= 8 -> ', a)

# karena a jadi float, kita ubah lagi menjadi integer
a = int(a)

a %= 9
print('a %= 9 -> ', a)

a //= 6
print('a //= 6 -> ', a)

a **= 1
print('a **= 1 -> ', a)
```

```
a &= 2
print('a &= 2 -> ', a)

a |= 3
print('a |= 3 -> ', a)

a ^= 4
print('a ^= 4 -> ', a)

a >>= 4
print('a >>= 4 -> ', a)

a <<= 2
print('a <<= 4 -> ', a)
```

Output dari kode program di atas adalah:

```
a = 10 ->  10
a += 5 ->  15
a -= 3 ->  12
a *= 6 ->  72
a /= 8 ->  9.0
a %= 9 ->  0
a //= 6 ->  0
a **= 1 ->  0
a &= 2 ->  0
a |= 3 ->  3
a ^= 4 ->  7
a >>= 4 ->  0
a <<= 4 ->  0
```

NB: beberapa operator terakhir di atas mengandung operasi bitwise. Operasi bitwise ada di bagian akhir dari pembahasan operator.

Operator logika

Operator logika adalah operator yang sangat penting. Operator ini sangat **berkaitan erat** dengan operator perbandingan. Dan kedua-duanya juga mengembalikan nilai dengan tipe data yang sama yaitu boolean.

Berikut ini tabel dari operator logika pada python.

Simbol	Tugas	Contoh
`and`	Mengembalikan `True` jika dua statement sama-sama benar	`True and True`
`or`	Mengembalikan `True` jika salah satu statement bernilai benar	`2 > 5 or 1 < 3`
`not`	Menegasikan hasil. `True` menjadi `False` dan sebaliknya	`not(1 > 5)`

Silakan dicoba masing-masing dari operator di atas.

```
print(True and True)
print(1 + 2 == 3 and True)
print('----')
print(False or 1 > 5)
print(False or 5 > 2)
print('----')
print(not(1 > 5))
print(not(1 < 5))
```

Jika kita jalankan, kita akan mendapatkan output sebagai berikut:

```
True
True
----
False
True
----
True
False
```

Operator keanggotaan

Python adalah bahasa pemrograman yang terbilang unik, ia memiliki operator khusus atau juga sering dinamakan sebagai operator spesial. Dinamakan spesial **karena memang operator ini hanya ada di Python** dan tidak ada di bahasa pemrograman lainnya.

Di antara operator spesial tersebut adalah **operator keanggotaan**. Atau di dalam bahasa inggris ia dinamakan *membership operator*.

Operator keanggotaan dalam python **hanya memiliki dua varian**, yaitu `in` dan `not in`. Berikut ini tabel lebih lengkapnya:

Simbol	Tugas
`in`	Bernilai `true` jika suatu nilai **ada** di dalam *sequence*
`not in`	Bernilai `false` jika suatu nilai **tidak ada** di dalam *sequence*

Agar lebih jelas, mari kita coba kode program berikut ini:

```
perusahaan = 'Microsoft'
list_pulau = ['Jawa', 'Sumatra', 'Sulawesi']

# ini adalah dictionary, akan kita pelajari
# di pertemuan-pertemuan yang akan datang
mahasiswa = {
  'nama': 'Lendis Fabri',
  'asal': 'Lamongan'
}

print(
  "Apakah 'c' ada di variabel perusahaan?",
  'c' in perusahaan
)
print(
  "Apakah 'z' tidak ada di variabel perusahaan?",
  'c' not in perusahaan
)
print(
  "Apakah 'Madura' ada di variabel list_pulau?",
  'Madura' in perusahaan
)
print(
  "Apakah 'Madura' tidak ada di variabel list_pulau?",
  'Madura' not in perusahaan
)
print(
  "Apakah atribut 'nama' ada di variabel mahasiswa?",
  'nama' in mahasiswa
)
```

Jika kita jalankan program di atas, kita akan mendapatkan output sebagai berikut:

```
Apakah 'c' ada di variabel perusahaan? True
Apakah 'z' tidak ada di variabel perusahaan? False
Apakah 'Madura' ada di variabel list_pulau? False
Apakah 'Madura' tidak ada di variabel list_pulau? True
Apakah atribut 'nama' ada di variabel mahasiswa? True
```

Operator identitas

Selain operator keanggotaan, python juga masih memiliki operator spesial lainnya: yaitu **operator identitas**. Operator ini **didefinisikan dengan** `is` dan `is not`.

Tugasnya adalah **untuk mengetahui apakah dua buah variabel** memiliki **nilai**yang sama dan **posisi** yang sama di dalam memori. Karena **tidak semua** nilai yang sama **memiliki** tempat / posisi yang sama di dalam memori.

Simbol	Tugas
`is`	Bernilai `true` jika dua variabel bersifat identik baik dari segi nilai mau pun penempatan lokasi di memory
`is not`	Bernilai `false` jika dua variabel tidak identik baik dari segi nilai mau pun penempatan lokasi di memory

Untuk lebih jelasnya, silakan tulis dan jalankan kode program di bawah ini:

```
a = 5
b = 5
list_a = [1, 2, 3]
list_b = [1, 2, 3]
nama_a = 'budi'
nama_b = 'budi'

# output True
print('a is b:', a is b)
# output False
print('a is not b:', a is not b)

# output False
print('list_a is list_b:', list_a is list_b)
# output True
print('list_a == list_b:', list_a == list_b)

# output True
print('nama_a is nama_b:', nama_a is nama_b)
# output False
print('nama_a is not nama_b:', nama_a is not nama_b)
```

Output:

```
a is b: True
a is not b: False
list_a is list_b: False
list_a == list_b: True
nama_a is nama_b: True
nama_a is not nama_b: False
```

Penjelasan

Kita coba perhatikan output dari kode program di atas. Dari 3 pasang variabel yang kita tes, masing-masing memiliki nilai yang sama.

1. Variabel `a` dan `b` sama-sama memiliki nilai `5`. Dan ketika kita periksa, mereka *equal* dan *identical*. Alias interpreter python menaruh nilai dari kedua variabel tersebut pada lokasi yang sama dalam memory.

2. Pada pasangan kedua, variabel `list_a` dan `list_b` juga memiliki nilai yang sama. Setelah kita tes, ternyata mereka berdua tidak *identical* meskipun nilai mereka berdua *equal*. Artinya, interpreter python menaruh nilai dari 2 variabel tersebut pada tempat yang berbeda pada memory.
3. Untuk pasangan yang terakhir, variabel `nama_a` dan `nama_b` yang bertipe data string, mereka berdua memiliki nilai yang *equal* dan juga *identical*.

Untuk mengetahui id atau lokasi penyimpanan suatu nilai pada python, kita bisa memanggil fungsi `id()`.

```
>>> id('merah')
140442358081648
>>> id('merah')
140442358081776
>>>
```

Pada kode di atas, kita mendefinisikan nilai `merah` sebanyak 2x, dan python menempatkan dua nilai tersebut di lokasi yang berbeda di dalam memory.

Contoh berikutnya untuk nilai integer:

```
>>> id(10)
140442360371792
>>> id(10)
140442360371792
>>>
```

Di dalam kode di atas, kita mendefinisikan nilai `10` sebanyak 2x, akan tetapi python menempatkan keduanya dalam posisi yang sama.

Operator bitwise

Operator terakhir yang akan kita bahasa dalam pertemuan ini adalah operator bitwise.

Operator bitwise **adalah operator yang berhubungan dengan angka-angka biner**.

Angka-angka biner adalah angka `0` dan `1`. Dan pada hakikatnya hanya ini lah angka yang dipahami oleh mesin.

Sebelum kita mulai, kita bisa mengetahui nilai biner dari suatu angka desimal dengan melakukan perintah `format()` dengan parameter kedua berupa string `'08b'`. Berikut ini demonstrasi menggunakan python mode interaktif.

```
>>> # biner dari angka 0
>>> print(format(0, '08b'))
00000000
>>> # biner dari angka 1
>>> print(format(1, '08b'))
00000001
>>> # biner dari angka 2
>>> print(format(2, '08b'))
00000010
>>> # biner dari angka 37
>>> print(format(37, '08b'))
00100101
>>>
```

Setelah sedikit perkenalan dengan binary, berikut ini adalah tabel yang menjelaskan tentang operator bitwise pada python.

Simbol	Nama	Tugas
&	Bitwise AND	Mengembalikan bit 1 **jika dua bit** bernilai 1
|	Bitwise OR	Mengembalikan bit 1 jika **salah satu** bit bernilai 1
^	Bitwise XOR	Mengembalikan bit 1 jika **hanya satu bit saja** yang bernilai 1
~	Bitwise NOT	Membalikkan semua bit
>>	Bitwise right shift	Menggeser bit ke kanan dengan mendorong salinan digit sebelah kiri dan membiarkan digit sebalah kanan terlepas
<<	Bitwise left shift	Menggeser bit ke kiri dengan mendorong digit 0 dan membiarkan bit paling kiri terlepas

Mari kita coba satu persatu dari kode operator bitwise di atas.

```
a = 1
b = 64

print('a =', a, '=', format(a, '08b'))
print('b =', b, '=', format(b, '08b'), '\n')

print('[and]')
print('a & b =', a & b)
print(format(a, '08b'), '&', format(b, '08b'), '=', format(a & b, '08b'),
'\n')

print('[or]')
print('a | b =', a | b)
print(format(a, '08b'), '|', format(b, '08b'), '=', format(a | b, '08b'),
'\n')

print('[xor]')
print('a ^ b =', a ^ b)
print(format(a, '08b'), '^', format(b, '08b'), '=', format(a ^ b, '08b'),
'\n')
```

```
print('[not]')
print('~a ~b =', ~a, ~b)
print('~' + format(a, '08b'), '~' + format(b, '08b'), '=', format(~a,
'08b'), format(~b, '08b'), '\n')

print('[shift right]')
print('a >> b =', a >> b)
print(format(a, '08b'), '>>', format(b, '08b'), '=', format(a >> b,
'08b'), '\n')

print('[shift left]')
print('b << a =', b << a)
print(format(b, '08b'), '<<', format(a, '08b'), '=', format(b << a,
'08b'), '\n')
```

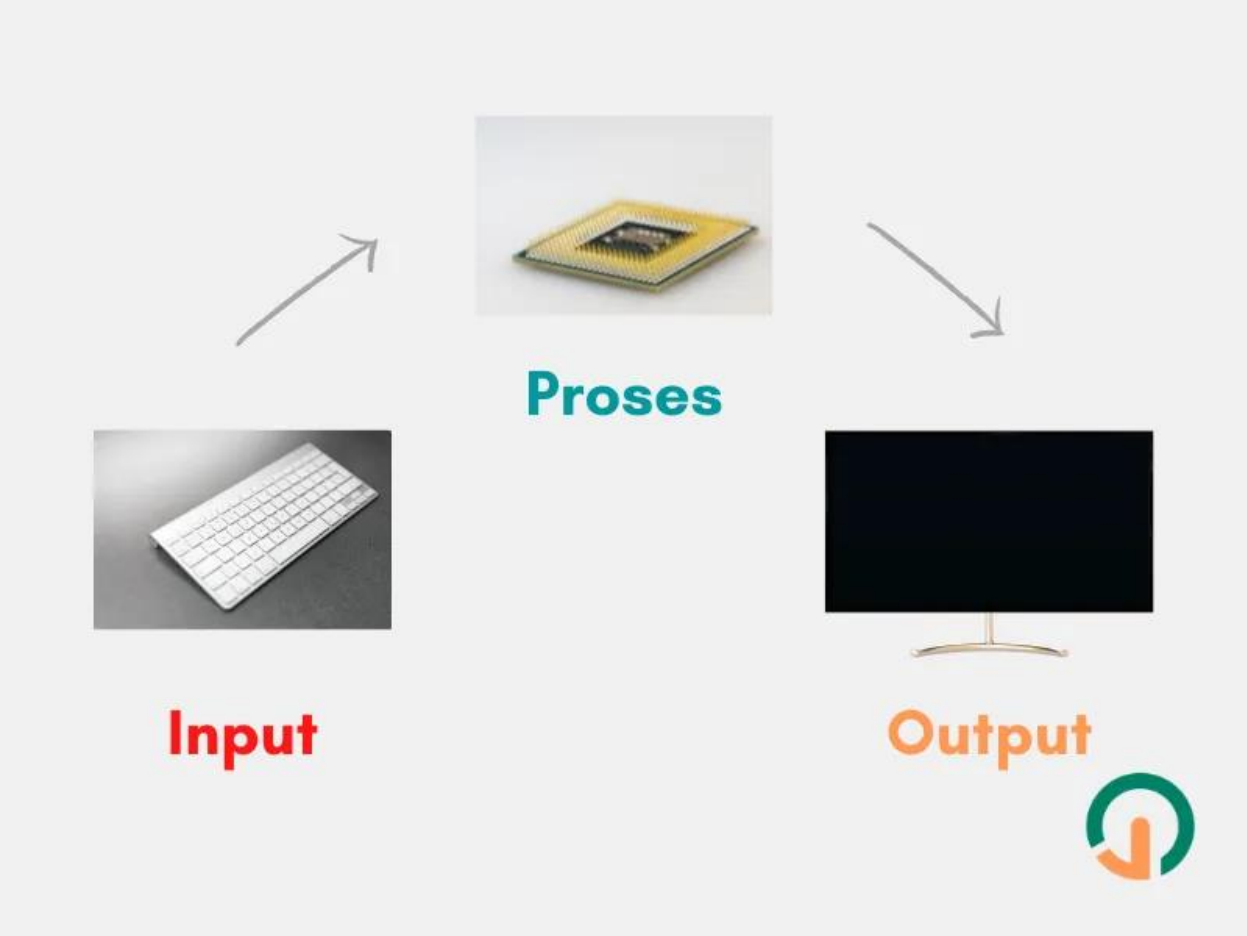
Output dari program di atas adalah:

```
a = 1 = 00000001
b = 64 = 01000000

[and]
a & b = 0
00000001 & 01000000 = 00000000

[or]
a | b = 65
00000001 | 01000000 = 01000001

[xor]
a ^ b = 65
00000001 ^ 01000000 = 01000001

[not]
~a ~b = -2 -65
~00000001 ~01000000 = -0000010 -1000001

[shift right]
a >> b = 0
00000001 >> 01000000 = 00000000

[shift left]
b << a = 128
01000000 << 00000001 = 10000000
```

Input

Apa Itu Input?

Input –atau *inputan*– (dalam konteks pemrograman) merupakan sebuah data, informasi, atau nilai apa pun yang dikirimkan oleh user kepada komputer untuk diproses lebih lanjut [1]. User melakukan proses input melalui media atau perangkat masukan seperti kibor, mouse, kamera, mikrofon dan lain sebagainya.



Nilai Statis vs Nilai Inputan User

Sejauh ini, semua kode program yang kita buat bersifat statis. Setiap variabel yang kita buat memiliki nilai sesuai dengan yang kita inginkan, tapi tidak seperti yang user inginkan.

Perhatikan contoh kode program di bawah ini:

```
panjang = 10
lebar = 5
luas = panjang * lebar

print("luas = ", luas)
```

Jika kode program di atas kita jalankan, kita akan mendapatkan output:

```
luas = 50
```

Lalu bagaimana kalau misal kita ingin menghitung luas persegi panjang selain 10 dan 5?

Maka yang kita lakukan adalah mengedit kode program, mengganti nilai dari variabel `panjang` dan `lebar` lalu kembali mengeksekusi program dari awal.

Tapi kalau kita ingin tanpa harus mengedit kode program, bagaimana?

Jawabannya adalah: minta user untuk meng-input-kan data!

Membuat Input Pada Python

Di dalam bahasa pemrograman python, kita bisa membuat sebuah inputan dengan cara memanggil fungsi bawaan 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.

Perhatikan contoh berikut:

```
nama = input('Masukkan nama anda: ')

print('Halo,', nama, '.. selamat datang!')
```

Pada kode program di atas:

1. Kita membuat sebuah variabel yaitu variabel `nama`.
2. Kemudian kita mengisi variabel `nama` tersebut berdasarkan inputan dari user.

Tipe data dari nilai input adalah string

Jika kita perhatikan pada program di atas, user **bisa memasukkan nilai atau data apa pun**. Bisa berupa angka, tanggal lahir, bahkan emoji (seperti yang dilakukan pada video di atas).

Lalu apa tipe data nilai kembalian dari fungsi `input()`?

Tipe data kembaliannya adalah `string`!

Mari kita buktikan.

```
nama = input('Masukkan nama anda: ')

print('Variabel nama bertipe data:', type(nama))
```

Jika kita masukkan “Nurul Huda” sebagai input, maka kode program di atas akan menghasilkan output:

```
Masukkan nama anda: Huda
Variabel nama bertipe data: <class 'str'>
```

Bahkan, meskipun kita memasukkan nilai `10`, variabel `nama` tetaplah sebuah string.

Untuk lebih menguatkan lagi, kita bisa mencoba kode program di bawah.

```
print('Kalkulator luas persegi panjang\n')
panjang = input('Masukkan panjang: ')
lebar = input('Masukkan lebar: ')

print('Luas =', panjang * lebar)
```

Jalankan kode program di atas, lalu masukkan angka `10` sebagai `panjang`, dan angka `5` sebagai `lebar`.

Program akan menghasilkan error:

```
Kalkulator luas persegi panjang

Masukkan panjang: 10
Masukkan lebar: 5
Traceback (most recent call last):
  File "02-int.py", line 5, in <module>
    print('Luas =', panjang * lebar)
TypeError: can't multiply sequence by non-int of type 'str'
```

Kenapa error? karena kita berusaha mengalikan dua buah nilai yang bertipe data `string`, dan hal itu akan menyebabkan error seperti di atas pada bahasa pemrograman python.

Input data selain string

Lalu bagaimana solusinya jika kita ingin mengalikan dua buah bilangan hasil dari input user?

Caranya adalah dengan **mengkonversi tipe data**!

Kita bisa mengkonversi tipe data `string` menjadi `integer`menggunakan fungsi `int()`.

Perhatikan contoh berikut dengan python mode interaktif:

```
>>> type(10)
<class 'int'>
>>> type('10')
<class 'str'>
>>> type(int('10'))
<class 'int'>
>>>
```

Pada percobaan ketika di atas, kita mencoba untuk mengubah nilai string `'10'` menjadi integer dengan fungsi `int('10')`.

Sehingga program kalkulator kita bisa *working well* kalau kode programnya kita ganti seperti berikut:

```
print('Kalkulator luas persegi panjang\n')
panjang = input('Masukkan panjang: ')
lebar = input('Masukkan lebar: ')

print('Luas =', int(panjang) * int(lebar))
```

Ketika kita jalankan, sistem tidak lagi menampilkan error

```
Kalkulator luas persegi panjang

Masukkan panjang: 10
Masukkan lebar: 5
Luas = 50
```

Output

Apa itu output?

Output atau keluaran (dalam bahasa Indonesia) merupakan setiap nilai atau data atau informasi yang dikirimkan oleh mesin / komputer kepada user (manusia) setelah tahap pemrosesan tertentu.

Secara umum output bisa berupa teks, gambar, suara, atau bahkan berupa informasi yang dicetak di atas kertas, dan sebagainya.

Jenis Output Pada Python

Pada python, output bisa berbagai macam seperti yang telah disebutkan sebelumnya. Akan tetapi yang paling dasar ada 2 jenis yaitu:

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

Pada kesempatan kali ini, kita 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.

Perintah `print()` yang Paling Dasar

Untuk membuat output di layar, perintah atau fungsi yang paling sering kita gunakan adalah fungsi `print()`. Dari awal seri tutorial python dasar ini pun kita sudah memakai `print()` berkali-kali dan di berbagai tempat.

Kalau kita perhatikan lagi, kita bisa menggunakan print dalam pola berikut ini:

Pola Biasa:

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

Multi argumen dengan pemisah koma:

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

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

Penjelasan

Seperti yang bisa kita lihat di atas, bahwa memanggil fungsi `print()` bisa dengan berbagai cara:

1. Cara pertama yaitu cara biasa, hanya menggunakan **satu buah string** saja.
2. Dan cara kedua yaitu menampilkan **lebih dari satu argumen string**, dan **memisahkan** antar string tersebut **menggunakan tanda koma**.

Pada model kedua ini, python akan menambahkan pemisah berupa spasi secara default.

Pemisah pada fungsi print()

Ketika kita memanggil fungsi `print()` untuk menampilkan multi argumen, python akan otomatis menambahkan karakter spasi sebagai pemisah antar argumen tersebut.

Jika kita tidak ingin pemisah spasi, maka kita bisa **menambahkan parameter** `sep` (*separator*) saat memanggil `print()`.

Perhatikan contoh berikut:

```
print('Andi', 'Budi', 'Tasya', 'Lala')
# Andi Budi Tasya Lala
```

```
print('Andi', 'Budi', 'Tasya', 'Lala', sep='_^_')
# Andi_^_Budi_^_Tasya_^_Lala
```

Pada contoh di atas, menampilkan 4 buah string:

1. `Andi`
2. `Budi`
3. `Tasya`
4. dan `Lala`

Perintah `print()` yang pertama tanpa parameter `sep`, sehingga python otomatis menjadikan karakter spasi sebagai pemisah.

Sedangkan perintah `print()` yang kedua, menggunakan parameter `sep` dengan nilai `_^_`. Hal itu akan membuat python mengganti karakter pemisah bawaan-nya dengan karakter yang didefinisikan sendiri.

Karakter Akhir Pada fungsi print()

Selain parameter `sep`, kita 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 kita memanggil fungsi `print()` untuk mencetak sesuatu, python akan mencetak karakter ganti baris (`\n`) di setiap output.

Perhatikan contoh berikut:

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

Output:

```
Merkurius
Venus
Bumi
```

Sedangkan dengan parameter `end`, kita bisa mengganti karakter ganti baris bawaan python dengan karakter lain sesuai keinginan kita:

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

Output:

```
Merkurius Venus Bumi
```

Kita juga bisa menggunakan 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🦕
```

Fungsi escape karakter `\n` pada contoh kedua adalah untuk menampilkan ganti baris. Karena tanpa diakhiri `\n`, perintah print selanjutnya tidak akan dimulai dari baris baru.

Memformat Output

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

Perhatikan contoh berikut:

```
a = 5
b = 3

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

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

Dua contoh di atas menghasilkan dua buah output yang sama.

Format Dengan Index

Selain menggunakan `{}`, kita juga bisa mendifinisikan index supaya yang ditampilkan tidak sesuai urutan parameter.

Perhatikan contoh berikut:

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

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

Pada contoh di atas, kita menggunakan `{}` tanpa index, dan juga menggunakan `{0}` dan `{1}` yang berupa index dari argumen fungsi `format()`.

Format dengan kunci

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

Perhatikan contoh berikut:

```
print('Halo {namaDepan} {namaBelakang}'.format(namaDepan='Ahza',
namaBelakang='Farezky'))
# Halo Ahza Farezky

print('Halo {namaDepan} {namaBelakang}'.format(namaBelakang='Farezky',
namaDepan='Ahza'))
# Halo Ahza Farezky
```

Dua baris kode pada contoh di atas menghasilkan output yang sama meskipun urutan parameternya berbeda. Itu karena yang kita gunakan adalah format menggunakan kunci.

Tips

Agar kode program tidak terlalu memanjang kesamping, kita bisa ganti baris saat pemanggilan fungsi. Perhatikan kode program berikut:

```
print('Halo {namaDepan} {namaBelakang}'
  .format(namaDepan='Ahza', namaBelakang='Farezky'))
# Halo Ahza Farezky

# begini juga bisa:
print('Halo {namaDepan} {namaBelakang}'
  .format(
    namaBelakang='Farezky',
    namaDepan='Ahza'
  )
)
# Halo Ahza Farezky
```

Latihan

Buat program untuk kasus-kasus berikut:

1. Menjumlahkan 4 bilangan bulat.
2. Menentukan bilangan terbesar dan terkecil dari tiga bilangan bulat.

3. Menentukan sisa pembagian suatu bilangan bulat dengan x (x > 0).
4. Menghitung luas:

    - Lingkaran
    - Bujur sangkar
    - Segitiga

Referensi

[1] https://www.programiz.com/python-programming/operators – diakses tanggal 15 Mei 2021

[2] https://www.microfocus.com/documentation/silk-test/205/en/silktestclassic-help-en/STCLASSIC-F02D493B-ARITHMETICOPERATORS-REF.html – diakses tanggal 15 Mei 2021

[3] https://www.computerhope.com/jargon/i/input.htm – diakses tanggal 15 Mei 2021

[4] https://techmonitor.ai/what-is/what-is-input-4935519 – diakses tanggal 15 Mei 2021

[5] https://www.computerhope.com/jargon/o/output.htm – diakses tanggal 16 Mei 2021