01153 - ALGORITMA DAN STRUKTUR DATA 1

INPUT DAN OUTPUT

04

OLEH : ANDRI HERYANDI, M.T.

INPUT – PROSES - OUTPUT

- Input – Proses – Output (IPO) adalah operasi utama dalam komputer.
- Komputer dapat menerima input/masukan (I), melakukan proses (P) terhadap data masukan, dan menghasilkan output/keluaran (O).
- Input bisa didapat dari berbagai sumber :
  - Keyboard
  - File (termasuk dari server di internet)
  - Perangkat Sensor
- Output dapat disajikan dalam bentuk :
  - Tampilan di layar monitor.
  - File
  - Print / Cetak
- Bab ini hanya menjelaskan bagaimana mendapatkan data input dari keyboard dan menampilkan data output ke layar monitor.

Algoritma
(Notasi Pseudo-Code

INPUT (ALGORITMA)

- Dalam algoritma (notasi pseudo-code), Langkah input dilakukan dengan menggunakan pernyataan/perintah **input**, disertai dengan variable yang akan menampungnya.
- Contoh :
  - `Input(nama)`
    - Baca masukan dari pengguna, simpan di variable nama
  - `Input(harga)`
    - Baca masukan dari pengguna, simpan di variable harga
  - `Input(Panjang, lebar)`
    - Baca masukan dari pengguna, simpan di variable Panjang dan lebar

OUTPUT (ALGORITMA)

- Dalam algoritma (notasi pseudo-code), Langkah ouput dilakukan dengan menggunakan pernyataan/perintah **output**, disertai dengan variable yang akan dikeluarkan.
- Contoh :
  - `output(nama)`
    - Tampilkan isi variable nama ke perangkat keluaran (monitor)
  - `output("Harga : ",harga)`
    - Tampilkan isi varable harga disertai dengan tulisan “Harga : “. Jika variable harga berisi 10000, maka akan menampilkan “Harga : 10000”.
  - `Input("Dimensi Persegi : ", panjang, " x ", lebar)`
    - Menampilkan isi variable Panjang dan lebar disertai dengan pengaturan tampilan. Jika Panjang berisi 10 dan lebar berisi 15 maka akan menampilkan “Dimensi Persegi : 10 x 15”

INPUT – PROSES - OUTPUT

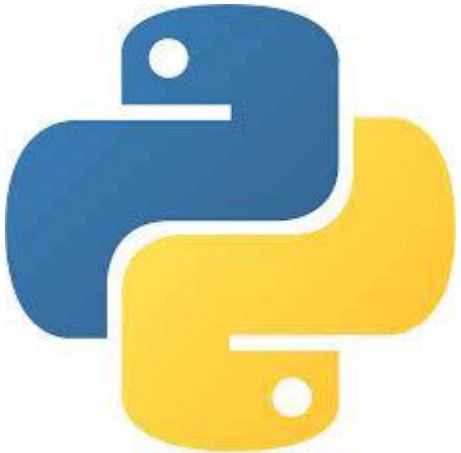
- Contoh :

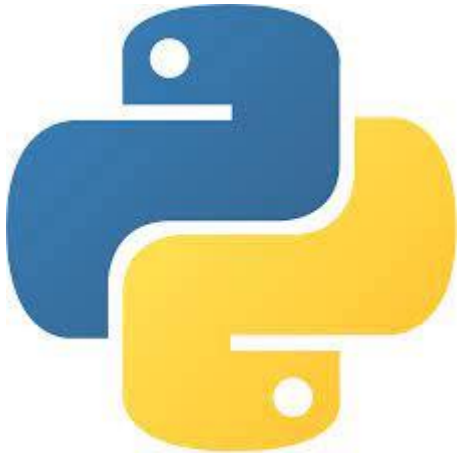
```
Algoritma Perhitungan_Persegi
{I.S: Panjang dan Lebar diinputkan oleh pengguna}
{F.S: Keliling dan Luas Persegi ditampilkan}

Kamus:
panjang, lebar, keliling, luas : integer

Algoritma:
input(panjang)
input(lebar)
keliling ← 2 * panjang + 2 * lebar
luas ← panjang * lebar
output("Keliling : ",keliling)
output("Luas     : ",luas)
```

Bahasa Pemrograman (python)





INPUT

- Untuk mendapatkan masukan data dari pengguna, gunakan function **input**
- Sintak dasar input adalah :

```
input([prompt])
```

- Keterangan :
  - Function input akan menghasilkan data bertipe string (walau pun diisi dengan data angka atau rumus).
  - Parameter **prompt** digunakan untuk menampilkan teks pertanyaan ketika menunggu input dari pengguna. Parameter ini sifatnya optional (boleh diisi boleh tidak).

INPUT

- Contoh :

```
>>> a = input()
Python
>>> type(a)
<class 'str'>
>>> b = input("Isikan Nama Anda : ")
Isikan Nama Anda : Budi
>>> type(b)
<class 'str'>
>>> c = input("Masukan angka : ")
Masukan angka : 123
>>> type(c)
<class 'str'>
```

```
>>> d = input("Masukan Rumus : ")
Masukan Rumus : 5 + 7
>>> print(d)
5 + 7
>>> type(d)
<class 'str'>
```

Keterangan :

- Function `type()` digunakan untuk mengetahui tipe data / class dari suatu data/variable

KONVERSI

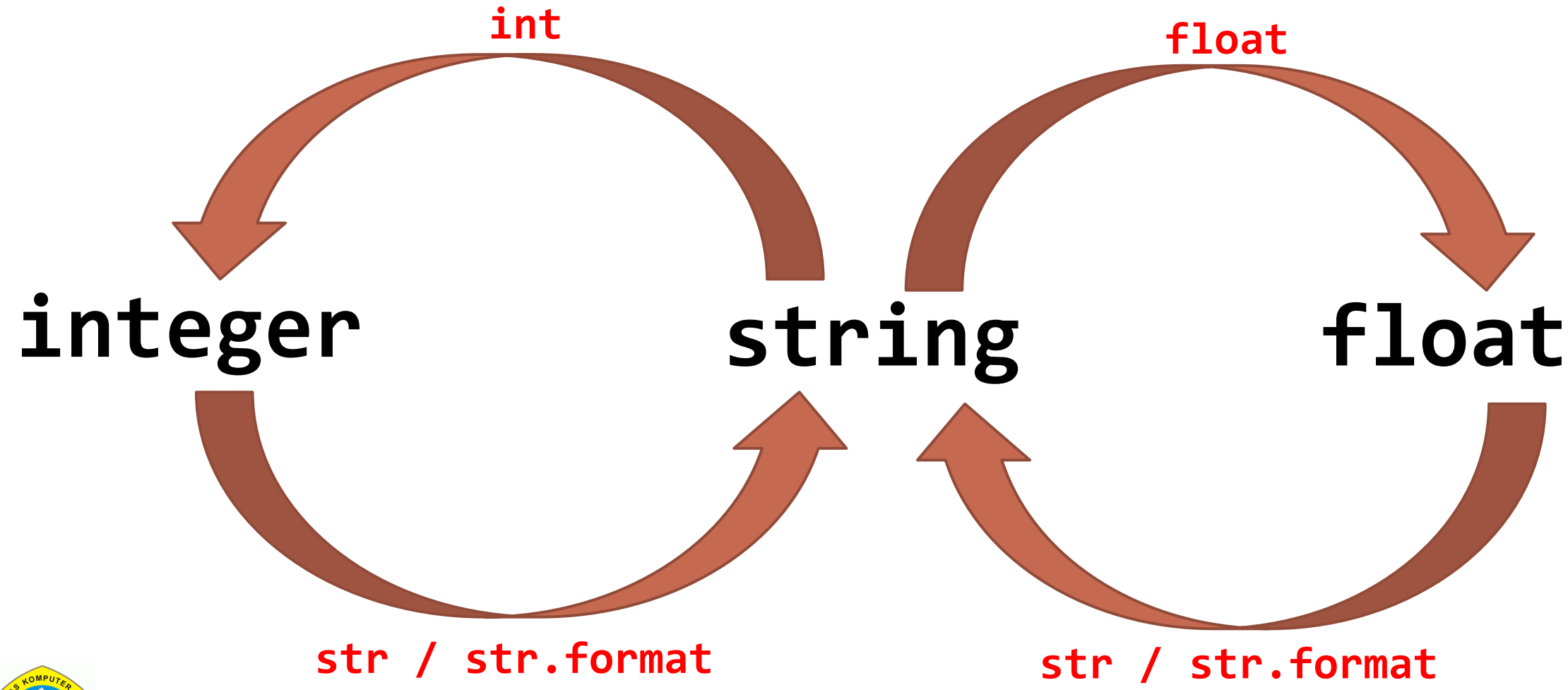
- Dikarenakan function input hanya akan memberikan data bertipe string maka jika anda ingin mengoperasikan data dengan operator aritmetika maka data tersebut harus dikonversi terlebih dahulu.
- Berikut adalah **contoh salah** :

```
>>> a = input("Masukan A : ")
Masukan A : 5
>>> b = input("Masukan B : ")
Masukan B : 10
>>> c = a + b
>>> print(c)
510
```

Keterangan :

1. Variable a akan bertipe string
2. Variable b akan bertipe string
3. Variable c adalah gabungan string a dan string b (bukan penjumlahan aritmetika), sehingga variable c akan bertipe string
4. “510” adalah penggabungan dari “5” + “10”.

KONVERSI


int
float
integer
string
float
str / str.format
str / str.format

KONVERSI (STRING → INTEGER)

- Jika anda menginginkan sebuah data yang bertipe integer, maka data string yang anda miliki harus dikonversikan terlebih dahulu.
- Untuk mengkonversi data string menjadi integer, maka menggunakan construktur **int**
- Sintaknya adalah :

```
class int([x])
```

- Keterangan :
  - Constructor **int** akan menghasilkan objek yang bertipe/kelas **int**.
  - Parameter x adalah string yang akan dikonversikan ke dalam integer. Jika x tidak diisi maka akan menghasilkan nilai 0.

KONVERSI (STRING → INTEGER)

- Contoh :

```
>>> a = "20"
>>> type(a)
<class 'str'>
>>> b = int(a)
>>> type(b)
<class 'int'>
>>> print(b)
20
>>> c = int()
>>> type(c)
<class 'int'>
>>> print(c)
0
```

Keterangan :

- Variable a diisi string “20”
- Tipe variable a adalah str
- Konversikan isi variable a menjadi integer, kemudian isikan ke variable b. Sehingga b berisi nilai 20
- Variable b bertipe int
- Variable b dicetak ke layar, menampilkan 20
- Variable c diisi dari int kosong
- Variable c bertipe int
- Isi variable c adalah 0 (karena tidak ada yang dikonversikan)

KONVERSI (STRING → INTEGER)

- Contoh :

```
>>> str_a = input("Masukan A : ")
Masukan A : 20
>>> a = int(str_a)
>>> b = int(input("Masukan B : "))
Masukan B : 7
>>> c = a + b
>>> print(c)
27
```

Keterangan :

1. Variable a diisi string “20”
2. Tipe variable a adalah str
3. Konversikan isi variable a menjadi integer, kemudian isikan ke variable b. Sehingga b berisi nilai 20
4. Variable b bertipe int
5. Variable b dicetak ke layar, menampilkan 20
6. Variable c diisi dari int kosong
7. Variable c bertipe int
8. Isi variable c adalah 0 (karena tidak ada yang dikonversikan)

KONVERSI (STRING → FLOAT)

- Jika anda menginginkan sebuah data yang bertipe float, maka data string atau integer yang anda miliki harus dikonversikan terlebih dahulu.
- Untuk mengkonversi data string menjadi float, maka menggunakan construktur **float**
- Sintaknya adalah :

```
class float([x])
```

- Keterangan :
  - Constructor **float** akan menghasilkan objek yang bertipe/kelas **float**.
  - Parameter x adalah string atau integer yang akan dikonversikan ke dalam float. Jika x tidak diisi maka akan menghasilkan nilai 0.

KONVERSI (STRING → FLOAT)

- Contoh :

```
>>> a = float(“-123.456")
>>> type(a)
<class 'float'>
>>> print(a)
-123.456
>>> b = float()
>>> print(b)
0.0
>>> c = float("1.56e+6")
>>> print(c)
1560000.0
```

Keterangan :

1. Konversikan string “-123.456” menjadi float dan menyimpannya dalam variable a.
2. Tipe variable a adalah float
3. Cetak isi variable a ke layar, sehingga menampilkan -123.456
4. Konversikan string kosong menjadi float, kemudian isikan ke variable b. Sehingga b berisi nilai 0
5. Variable b dicetak ke layar, menampilkan 0.0
6. Konversikan string “1.56e+6” menjadi float. “e+6” artinya 10 pangkat 6 sehingga “1.56e+6” = 1.56×10^6
7. Cetak variable c, sehingga menampilkan 1560000,0

KONVERSI (INTEGER / FLOAT → STRING)

- Untuk mengkonversi suatu objek menjadi string maka bisa menggunakan constructor **str**.
- Sintaknya adalah :

```
class str([x])
```

- Keterangan :
  - Constructor **str** akan menghasilkan objek yang bertipe/kelas **string**.
  - Parameter x adalah objek (boleh integer, float atau objek lainnya) yang akan dikonversikan ke dalam string. Jika x tidak diisi maka akan menghasilkan string kosong (‘’).

KONVERSI (INTEGER → STRING)

- Contoh :

```
>>> i = 12345
>>> type(i)
<class 'int'>
>>> i
12345
>>> str_i = str(i)
>>> type(str_i)
<class 'str'>
>>> str_i
'12345'
```

Keterangan :

1. Variable i diisi data integer.
2. Tipe variable i bertipe int
3. Cetak isi variable i ke layar, sehingga menampilkan 12345
4. Konversikan variable i menjadi string, kemudian isikan ke variable str_i. Sehingga str_i berisi nilai “12345”
5. Variable str_i bertipe str
6. Tampilkan isi Variable str_i dicetak ke layar, menampilkan “12345”

KONVERSI (FLOAT → STRING)

- Contoh :

```
>>> f = -12345.67890
>>> type(f)
<class 'float'>
>>> f
-12345.6789
>>> str_f = str(f)
>>> type(str_f)
<class 'str'>
>>> str_f
'-12345.6789'
```

Keterangan :

1. Variable f diisi data float.
2. Tipe variable f bertipe float
3. Cetak isi variable f ke layar, sehingga menampilkan `-12345.6789`
4. Konversikan variable f menjadi string, kemudian isikan ke variable str_f. Sehingga str_f berisi nilai "`-12345.6789`"
5. Variable str_f bertipe str
6. Tampilkan isi Variable str_f dicetak ke layar, menampilkan "`-12345.6789`"

OUTPUT

- Untuk mengeluarkan (output) informasi hasil proses yangdilakukan oleh puthon, maka salah satu function yang dapat digunakan adalah **print**. (jika output ingin ditampilkan dalam layar/monitor)
- Sintak dasar print adalah :

```
print(objects)
```

- Keterangan :
  - Function **print** akan menampilkan isi objek (atau objek-objek) dalam bentuk teks yang secara default dikirim ke layar monitor. Secara default setiap selesai cetak, maka akan pindah baris.
  - Parameter **objects** diisi dengan objek/variable yang akan ditampilkan. Objek/variable yang akan digunakan boleh hanya 1 atau banyak. Jika data yang dicetak banyak maka secara default ada karakter pemisah yaitu 1 buah spasi.

Referensi Lanjut : https://docs.python.org/3/library/functions.html#print

OUTPUT

- Contoh :

```
>>> a = 12345
>>> b = 678.901234
>>> c = "Kota Bandung"
>>> print(a)
12345
>>> print(b)
678.901234
>>> print(a,b,c)
12345 678.901234 Kota Bandung
>>> print("a=",a,", b= ",b,", c=",c)
a= 12345 , b=  678.901234 , c= Kota Bandung
```

Keterangan :

1. Variable a diisi integer 12345.
2. Variable b diisi float 678.901234
3. Variable c diisi string “Kota Bandung”
4. Cetak sebuah variable (a) ke layar
5. Cetak sebuah variable (b) ke layar
6. Cetak 3 buah variable (a, b, c) ke layar
7. Menampilkan 3 variable dengan menambahkan string pelengkap

OUTPUT
(MENULIS PRINT BANYAK OBJEK)

- Contoh :

```
>>> nama_lengkap = "Cecep Gorbachev"
>>> panggilan = "Cep"
>>> tempat_lahir = "Bandung"
>>> umur = 25
>>> print("Hai, nama saya", nama_lengkap, "biasa dipanggil dengan \"", panggilan,"\",
lahir di ", tempat_lahir, "dan sekarang saya berumur", umur,"tahun")
Hai, nama saya Cecep Gorbachev biasa dipanggil dengan " Cep ", lahir di  Bandung dan
sekarang saya berumur 25 tahun
```

Keterangan :

1. \" digunakan untuk menuliskan tanda kutip.
2. Bagian yang diberi warna merah, adalah literal string tambahan

Perhatian!
Pada bagian nama panggilan menghasilkan " Cep ". Ada spasi yang tidak berada pada tempat seharusnya. Seharusnya munculnya "Cep".
Itu dikarenakan di setiap antara objek akan otomatis diberi pemisah (separator) yaitu 1 spasi.

OUTPUT
(MENGGANTI PEMISAH DI FUNCTION PRINT)

- Untuk mengganti pemisah antar objek/variable Ketika menulis ke layar menggunakan print, maka tambahkankan parameter sep disertai string pemisah yang diinginkan.

```
>>> a = 5
>>> b = 7
>>> c = "Kota Bandung"
>>> print(a,b,c)
5 7 Kota Bandung
>>> print(a,b,c,sep='-')
5-7-Kota Bandung
>>> print(a,b,c,sep='')
57Kota Bandung
```

Keterangan :

1. Pengisian variable a
2. Pengisian variable b
3. Pengisian variable c
4. Tulis ke layar 3 variable (a, b, c) dengan pemisah 1 spasi (default)
5. Tulis ke layar 3 variable (a, b, c) dengan pemisah tanda minus
6. Tulis ke layar 3 variable (a, b, c) dengan pemisah spasi kosong (tanpa pemisah)

OUTPUT
(MENGGANTI AKHIR FUNCTION PRINT)

- Secara default, setiap selesai melaksanakan penulisan objek, maka print akan mengakhiri dengan tanda pindah baris (\n).
- Tanda akhir ini bisa anda ganti jika tidak menginginkan tanda pindah baris digunakan sebagai akhir print.
- Gunakan parameter **end** diikuti string penutup print.

```
>>> print("AAAAAA")
AAAAAA
>>> print("AAAAAA",end=".")
AAAAAA.>>>
>>> print("AAAAAA",end=".\n")
AAAAAA.
>>> print("BBBBBB",end="")
BBBBBB>>>
>>> print("Sarah","Hasan","Alica","Idris",sep=", ",end=".")
Sarah, Hasan, Alica, Idris.>>>
```

Keterangan :

1. Penulisan diakhiri \n (default)
2. Penulisan diakhiri titik (.), sehingga tidak pindah baris
3. Penulisan diakhiri titik dan pindah baris
4. Penulisan diakhiri spasi kosong (sehingga pindah baris)
5. Penulisan dipisah koma dan spasi dan diakhiri dengan titik.

OUTPUT
(MENGATUR FORMAT TAMPILAN)

- Menampilkan banyak variable dalam sebuah pernyataan print bisa sangat merepotkan jika hanya dengan mengirimkan seluruh variable yang dipisahkan dengan tanda koma.
- Dalam python, anda dapat menggabungkan isi dari banyak variable ke dalam sebuah string.
- Proses menyusun/mengatur string agar tersusun dalam bentuk tertentu disebut memformat string (string formatting)
- Setidaknya ada 3 cara untuk memformat string
  - Memformat string menggunakan % (versi lama)
  - Memformat string menggunakan str.format
  - Memformat string menggunakan f-string (formatted string literal)

OUTPUT
(MEMFORMAT STRING MENGGUNAKAN %)

- Dalam format ini, tanda % digunakan diantara format string dan nilai yang akan mengisinya.
- Operasi ini disebut sebagai interpolasi string.
- Untuk setiap tanda % yang berada dalam string, maka akan diganti dengan variable yang ditulis setelah tanda %.
- Sebagai tanda posisi data, gunakan tipe konversi.
- Ada banyak tipe konversi, tapi yang paling sering digunakan adalah :
  - %d atau %i mewakili data bilangan integer
  - %s digunakan untuk mewakili data string
  - %f digunakan untuk mewakili data float.

Referensi Lanjut :

1. https://docs.python.org/id/3.8/library/stdtypes.html#old-string-formatting
2. https://docs.python.org/id/3.8/library/stdtypes.html#printf-style-string-formatting

OUTPUT (MEMFORMAT STRING MENGGUNAKAN %)

- Dalam format ini, tanda % digunakan diantara format string dan nilai yang akan mengisinya.

```
>>> a = 123
>>> b = 456.789
>>> c = "Kota Bandung"
>>> print("A berisi %i" % a)
A berisi 123
>>> print("B berisi %f dan C berisi %s" % (b, c))
B berisi 456.789000 dan C berisi Kota Bandung
```

Keterangan :

1. Pengisian variable a
2. Pengisian variable b
3. Pengisian variable c
4. Cetak variable a dengan %i
5. Cetak variable b dengan %f dan variable c dengan %s

OUTPUT (MEMFORMAT STRING MENGGUNAKAN %)

- Jika anda menggunakan tipe konversi seadanya (%i, %d, %f atau %s) saja, maka panjang data yang ditampilkan akan apa adanya, sehingga ketika anda menampilkan data dengan banyak baris maka tampilan seakan-akan tidak beraturan.
- Proses mengisi sebuah string dengan karakter tertentu agar selalu mendapatkan panjang yang konsisten disebut sebagai padding.
- Karakter penanda yang dapat digunakan untuk melakukan padding adalah :
  - 0 digunakan kalau ingin padding (mengisi ruang kosong agar mencapai panjang tertentu) pada sebelah kiri dengan karakter 0.
  - - (tanda minus) digunakan untuk melakukan padding di sebelah kiri dengan karakter 0. Hal ini akan mengakibatkan string yang ditampilkan menjadi right-justify (rata kanan)
  - + (tanda plus) digunakan untuk memastikan tanda + atau – selalu ditampilkan
- Ketika anda menggunakan padding, maka setelah karakter penanda padding, anda harus menyertakan berapa spasi ruang untuk variable tersebut.
- Untuk %f angka pengatur padding terdiri dari 2, yaitu banyaknya spasi ruang dan berapa digit pecahan yang akan ditampilkan. Kedua angka ini dipisahkan dengan tanda koma.

OUTPUT (MEMFORMAT STRING MENGGUNAKAN %)

- Contoh tidak menggunakan padding

```
>>> a = "Budi"
>>> b = "Kurniawan"
>>> c = "Ai"
>>> print("[%s]" % a)
[Budi]
>>> print("[%s]" % b)
[Kurniawan]
>>> print("[%s]" % c)
[Ai]
```

Keterangan :

1. Pengisian variable a
2. Pengisian variable b
3. Pengisian variable c
4. Cetak variable a tanpa padding (menampilkan string a apa adanya)
5. Cetak variable b tanpa padding (menampilkan string b apa adanya)
6. Cetak variable c tanpa padding (menampilkan string c apa adanya)

OUTPUT
(MEMFORMAT STRING MENGGUNAKAN %)

- Contoh menggunakan padding (15 spasi)

```
>>> a = "Budi"
>>> b = "Kurniawan"
>>> c = "Ai"
>>> print("[%15s]" % a)
[          Budi]
>>> print("[%15s]" % b)
[      Kurniawan]
>>> print("[%15s]" % c)
[             Ai]
>>> print("[%-15s]" % a)
[Budi          ]
>>> print("[%-15s]" % b)
[Kurniawan     ]
>>> print("[%-15s]" % c)
[Ai            ]
```

Keterangan :

1. Pengisian variable a
2. Pengisian variable b
3. Pengisian variable c
4. Cetak variable a dengan padding kiri (menghasilkan rata kanan)
5. Cetak variable b dengan padding kiri (menghasilkan rata kanan)
6. Cetak variable c dengan padding kiri (menghasilkan rata kanan)
7. Cetak variable a dengan padding kanan (menghasilkan rata kiri)
8. Cetak variable b dengan padding kanan (menghasilkan rata kiri)
9. Cetak variable c dengan padding kanan (menghasilkan rata kiri)

OUTPUT (MEMFORMAT STRING MENGGUNAKAN %)

- Contoh menggunakan padding (10 spasi) pada data integer

```
>>> a = 12500
>>> b = 1000
>>> c = 250000
>>> d = 2500000
>>> print("Rp. %10i" % a);
Rp.      12500
>>> print("Rp. %10i" % b);
Rp.       1000
>>> print("Rp. %10i" % c);
Rp.     250000
>>> print("Rp. %10i" % d);
Rp.    2500000
>>> print("Rp. %010i" % a);
Rp. 0000012500
>>> print("Rp. %+10i" % a);
Rp.    +12500
```

Keterangan :

1. Pengisian variable a
2. Pengisian variable b
3. Pengisian variable c
4. Pengisian variable d
5. Cetak variable a dengan padding kiri (menghasilkan rata kanan)
6. Cetak variable b dengan padding kiri (menghasilkan rata kanan)
7. Cetak variable c dengan padding kiri (menghasilkan rata kanan)
8. Cetak variable d dengan padding kiri (menghasilkan rata kanan)
9. Cetak variable a dengan padding kiri menggunakan karakter 0.
10. Cetak variable a dengan + atau – ditampilkan.

OUTPUT
(MEMFORMAT STRING MENGGUNAKAN %)

- Contoh menggunakan padding (15 spasi) pada data float

```
>>> a = 12345.67890
>>> print("a:%f" % a)
a:12345.678900
>>> print("a:%15f" % a)
a:    12345.678900
>>> print("a:%15.2f" % a)
a:          12345.68
>>> print("a:%15.3f" % a)
a:         12345.679
>>> print("a:%+15.3f" % a)
a:        +12345.679
```

Keterangan :

1. Pengisian variable a
2. Cetak variable a tanpa padding
3. Cetak variable a dengan padding kiri (menghasilkan rata kanan) tanpa pengaturan digit pecahan.
4. Cetak variable a dengan padding kiri (menghasilkan rata kanan) dengan hanya menampilkan 2 digit pecahan.
5. Cetak variable a dengan padding kiri (menghasilkan rata kanan) dengan hanya menampilkan 3 digit pecahan.
6. Cetak variable a dengan padding kiri (menghasilkan rata kanan) dengan menampilkan 3 digit pecahan dan simbol + atau – ditampilkan.

OUTPUT
(MEMFORMAT STRING MENGGUNAKAN STR.FORMAT)

- Operasi ini menggunakan method format dari class string.
- Sintak Dasarnya :

```
str.format(objects)
```

- Keterangan :
  - Str adalah objek string yang terformat. Suatu field (data) yang akan ditampilkan dalam format harus diapit kurung kurawal {}. Tanda {} menyatakan field.
  - Objects adalah objek-objek yang akan diformat.
- Anda tidak perlu memikirkan tipe data dari objek yang akan ditulis.

Referensi Lanjut :

1. https://docs.python.org/id/3.8/library/string.html#format-examples

OUTPUT
(MEMFORMAT STRING MENGGUNAKAN %)

- Contoh

```
>>> nama = "Adi"
>>> kota = "Bandung"
>>> umur = 30
>>> print("Nama saya {}, tinggal di {}. Umur saya {}".format(nama, kota,umur))
Nama saya Adi, tinggal di Bandung. Umur saya 30
```

Jika di dalam {} tidak ada apa pun, maka data yang akan ditampilkan sesuai dengan urutan objek-objek

Keterangan :

1. Pengisian variable nama
2. Pengisian variable kota
3. Pengisian variable umur
4. Cetak string yang terformat (yang merupakan gabungan nama, kota dan umur)

OUTPUT
(MEMFORMAT STRING MENGGUNAKAN %)

- Contoh mengakses menggunakan index

```
>>> nama = "Adi"
>>> kota = "Bandung"
>>> umur = 30
>>> print("Nama saya {0}, tinggal di {1}. Umur saya {2}".format(nama, kota,umur))
Nama saya Adi, tinggal di Bandung. Umur saya 30
>>> print("Nama saya {2}, tinggal di {0}. Umur saya {1}".format(nama, kota,umur))
Nama saya 30, tinggal di Adi. Umur saya Bandung
```

Di dalam kurung kurawal boleh diisi index data objek. Index objek dimulai dari index 0, 1 dst.

Keterangan :

1. Pengisian variable nama
2. Pengisian variable kota
3. Pengisian variable umur
4. Cetak string yang terformat (yang merupakan gabungan nama, kota dan umur) sesuai urutan objek
5. Cetak string yang terformat (yang merupakan gabungan nama, kota dan umur) sesuai urutan objek yang tidak berurutan

OUTPUT
(MEMFORMAT STRING MENGGUNAKAN %)

- Contoh mengakses menggunakan nama

```
>>> harga_barang_per_item = 5000
>>> print("Harga : Rp. {hb}".format(hb=harga_barang_per_item))
Harga : Rp. 5000
```

Keterangan :

1. Pengisian variable harga_barang_per_item
2. Mencetak harga dimana datanya diambil dari hb yang diisi dari harga_barang_per_item

OUTPUT
(MEMFORMAT STRING MENGGUNAKAN %)

- Contoh str.format dengan pengaturan alignment
- Gunakan tanda titik 2 dan angka maksimum karakter yang akan ditampilkan.
- Tambahkan tanda lebih kecil (<) sebelum angka (atau tidak ditulis sama sekali) untuk mendapatkan rata kiri
- Tambahkan tanda lebih besar (>) sebelum angka untuk mendapatkan rata kanan
- Tambahkan tanda caret (^) sebelum angka untuk mendapatkan rata tengah.

```
>>> nama = "Alica Rahmanita"
>>> print("[{:30}]".format(nama))
[Alica Rahmanita                ]
>>> print("[{:<30}]".format(nama))
[Alica Rahmanita                ]
>>> print("[{:>30}]".format(nama))
[                Alica Rahmanita]
>>> print("[{:^30}]".format(nama))
[        Alica Rahmanita        ]
```

Keterangan :
1. Pengisian variable nama
2. Format 30 karakter rata kiri
3. Format 30 karakter rata kiri
4. Format 30 karakter rata kanan
5. Format 30 karakter rata tengah

OUTPUT
(MEMFORMAT STRING DENGAN F-STRING)

- F-string memperbolehkan anda menyertakan nilai dari ekspresi python ke dalam suatu string dengan menuliskan string yang diawali dengan f atau F.
- Data yang akan ditulis diwakilkan kepada field yang ditulis di dalam kurung kurawal yang isinya adalah nama variable/objek yang akan ditampilkan/diformat.
- Sintak Dasarnya :

```
F"format"
```

- Keterangan :
  - Huruf F atau f di awal string format wajib ditulis sebagai penanda f-string.
  - Data yang akan diformat ditulis di dalam format dengan diapit kurung kurawal.

Referensi Lanjut :

1. https://docs.python.org/3/tutorial/inputoutput.html#formatted-string-literals

OUTPUT
(MEMFORMAT STRING DENGAN F-STRING)

- Contoh

```
>>> nama = "Idris"
>>> kota = "Bandung"
>>> umur = 5
>>> print(f"Nama saya {nama}, tinggal di {kota}. Saya berumur {umur} tahun")
Nama saya Idris, tinggal di Bandung. Saya berumur 5 tahun
```

Keterangan :

1. Pengisian variable nama
2. Pengisian variable kota
3. Pengisian variable umur
4. Mencetak string dengan menggunakan f-string yang menampilkan 3 buah variable

OUTPUT
(MEMFORMAT STRING DENGAN F-STRING)

- F-string juga memperbolehkan menggunakan format tambahan (dipisahkan dengan lambang : setelah nama variable) seperti metode-metode format string sebelumnya seperti:
  - Lambang <. >, ^ untuk mengatur alignment.
  - Format “N.Mf” untuk pengaturan padding (N) dan banyak digit pecahan (M) pada tipe data float
  - memperbolehkan anda menyertakan nilai dari ekspresi python ke dalam suatu string dengan menuliskan string yang diawali dengan f atau F.

OUTPUT
(MEMFORMAT STRING DENGAN F-STRING)

- Contoh

```
>>> teks = "ABCDEFGH"
>>> print(f"[{teks:<20}]")
[ABCDEFGH            ]
>>> print(f"[{teks:>20}]")
[            ABCDEFGH]
>>> print(f"[{teks:^20}]")
[      ABCDEFGH      ]
>>> a = 12345678
>>> print(f"A:{a:15}")
A:       12345678
>>> b = 123456789.012345
>>> print(f"B:{b:15.2f}")
B:    123456789.01
```

Keterangan :

1. Pengisian variable teks
2. Mencetak teks rata kiri 20 spasi
3. Mencetak teks rata kanan 20 spasi
4. Mencetak teks rata tengah 20 spasi
5. Pengisian variable a bertipe int
6. Menampilkan variable a rata kanan 15 spasi
7. Pengisian variable b bertipe float
8. Menampilkan variable b rata kanan 15 spasi dengan 2 angka pecahan.

Sekian
&
Terima Kasih

FORUM DISKUSI

LMS UNIKOM
https://lms.unikom.ac.id

Group Whatsapp Perkuliahan

Youtube Playlist
https://unikom.id/YT-ASD1