DASAR PYTHON

BAB 3 : STRING

3.1 Tujuan

Mahasiswa mengenal String pada bahasa pemrograman python

3.2 Ulasan Materi

A. String

1. String

String dalam python dikelilingi oleh tanda kutip tunggal, atau tanda kutip ganda. 'hello' sama dengan “hello”.

Anda dapat menampilkan string literal dengan fungsi print(). Contoh :

```
print("Hello")
print('Hello')
```

Output :

```
Hello
Hello
```

2. Tetapkan String ke Variabel

Menetapkan string ke variabel dilakukan dengan nama variabel diikuti dengan tanda sama dengan dan string. Contoh :

```
a = "Hello"
print(a)
```

Output :

```
Hello
```

3. Multiline String

Anda dapat menetapkan string multiline ke variabel dengan menggunakan tiga tanda kutip. Contoh :

```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
```

Output :

```
Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.
```

Atau tiga kutipan tunggal :

```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
```

Output :

```
Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.
```

4. **String adalah Array**

Seperti banyak bahasa pemrograman populer lainnya, string dalam Python adalah array byte yang mewakili karakter unicode. Namun, Python tidak memiliki tipe data karakter, satu karakter hanyalah string dengan panjang 1. Tanda kurung siku

dapat digunakan untuk mengakses elemen string. Contoh dapatkan karakter di posisi 1 (ingat bahwa karakter pertama memiliki posisi 0) :

```
a = "Hello, World!"
print(a[1])
```

Output :

```
e
```

5. **Looping Melalui String**

Karena string adalah array, sehingga dapat mengulang karakter dalam string, dengan for loop. Contoh huruf-huruf dalam kata "pisang" :

```
for x in "banana":
  print(x)
```

Output :

```
b
a
n
a
n
a
```

6. **String Length (Panjang String)**

Untuk mendapatkan panjang string, gunakan fungsi len().Contoh Fungsi len() mengembalikan panjang string :

```
a = "Hello, World!"
print(len(a))
```

Output :

```
13
```

7. **Check String**

   Untuk memeriksa apakah ada frase atau karakter tertentu dalam sebuah string, kita dapat menggunakan kata kunci in. Contoh, periksa apakah "free" ada dalam teks berikut :

```
txt = "The best things in life are free!"
print("free" in txt)
```

Output :

```
True
```

Gunakan dalam statement if. Contoh print hanya jika ada "free" :

```
txt = "The best things in life are free!"
if "free" in txt:
  print("Yes, 'free' is present.")
```

Output :

```
Yes, 'free' is present.
```

8. **Check apakah NOT**

   Untuk memeriksa apakah frasa atau karakter tertentu NOT(tidak) ada dalam string, kita dapat menggunakan kata kunci not in. Contoh periksa apakah " expensive" TIDAK ada dalam teks berikut :

```
txt = "The best things in life are free!"
print("expensive" not in txt)
```

Output :

```
True
```

Gunakan dalam statement if. Contoh, print hanya jika " expensive" TIDAK ada :

```
txt = "The best things in life are free!"
if "expensive" not in txt:
  print("No, 'expensive' is NOT present.")
```

Output :

```
No, 'expensive' is NOT present
```

9. Slicing String

Anda dapat mengembalikan rentang karakter dengan menggunakan sintaks slice. Tentukan indeks awal dan indeks akhir, dipisahkan oleh titik dua, untuk mengembalikan bagian dari string. Contoh, dapatkan karakter dari posisi 2 ke posisi 5 (tidak termasuk):

```
b = "Hello, World!"
print(b[2:5])
```

Output :

```
llo
```

10. Slicing dari awal

Dengan mengabaikan indeks awal, rentang akan dimulai pada karakter pertama. Contoh, dapatkan karakter dari awal ke posisi 5 (tidak termasuk) :

```
b = "Hello, World!"
print(b[:5])
```

Output :

```
Hello
```

11. Slicing sampai akhir

Dengan mengabaikan indeks akhir, rentang akan berakhir. Contoh, dapatkan karakter dari posisi 2, dan sampai akhir :

```
b = "Hello, World!"
print(b[2:])
```

Output :

```
llo, World!
```

12. Indeks Negatif

Gunakan indeks negatif untuk memulai irisan dari akhir string. Contoh :

Dapatkan karakter :

Dari : "o" di " World!" (posisi -5)

Ke, tetapi tidak termasuk: "d" di " World!" (posisi -2):

```
b = "Hello, World!"
print(b[-5:-2])
```

Output :

```
Orl
```

13. Modify Strings

Python memiliki seperangkat metode bawaan yang dapat digunakan pada string.

14. Upper Case

Metode upper() mengembalikan string dalam huruf besar. Contoh :

```
a = "Hello, World!"
print(a.upper())
```

```

```

Output :

```
HELLO, WORLD!
```

15. Lower Case

Metode lower() mengembalikan string dalam huruf kecil. Contoh :

```
a = "Hello, World!"
print(a.lower())
```

Output :

```
hello, world!
```

16. Menghapus Spasi (*Remove Whitespace*)

Whitespace adalah spasi sebelum dan/atau setelah teks yang sebenarnya, dan sangat sering Anda ingin menghapus spasi ini.

Metode strip() menghapus spasi apa pun dari awal atau akhir. Contoh :

```
a = " Hello, World! "
print(a.strip())
```

Output :

```
Hello, World!
```

17. Replace String

Metode replace() menggantikan string dengan string lain. Contoh :

```
a = "Hello, World!"
print(a.replace("H", "J"))
```

Output :

```

```

```
Jello, World!
```

18. Split String

Metode split() mengembalikan daftar tempat teks di antara pemisah yang ditentukan menjadi item daftar.

Metode split() membagi string menjadi substring jika menemukan contoh pemisah.

Contoh :

```
a = "Hello, World!"
b = a.split(",")
print(b)
```

Output :

```
['Hello', ' World!']
```

19. Penggabungan String (*String Concatenation*)

Untuk menggabungkan, atau menggabungkan, dua string dapat menggunakan operator +.

Gabungkan variabel a dengan variabel b menjadi variabel c. Contoh :

```
a = "Hello"
b = "World"
c = a + b
print(c)
```

Output :

```
HelloWorld
```

Untuk menambahkan spasi di antara keduanya, tambahkan " ":

```
a = "Hello"
b = "World"
c = a + " " + b
```

```
print(c)
```

Output :

```
Hello World
```

20. Format String

Seperti yang dipelajari pada bab Variabel Python, kita tidak dapat menggabungkan string dan angka seperti ini :

```
age = 36
txt = "My name is John, I am " + age
print(txt)
```

Output :

```
Traceback (most recent call last):
  File "demo_string_format_error.py", line 2, in <module>
    txt = "My name is John, I am " + age
TypeError: must be str, not int
```

Tetapi kita dapat menggabungkan string dan angka dengan menggunakan method format()!. Method format() mengambil argumen yang diteruskan, memformatnya, dan menempatkannya dalam string tempat placeholder {} berada.

Gunakan metode format() untuk memasukkan angka ke dalam string. Contoh :

```
age = 36
txt = "My name is John, and I am {}"
print(txt.format(age))
```

Output :

```
My name is John, and I am 36
```

Method format() mengambil argumen dalam jumlah tak terbatas, dan ditempatkan ke masing-masing placeholder. Contoh :

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))
```

Output :

```
I want 3 pieces of item 567 for 49.95 dollars.
```

Anda dapat menggunakan nomor indeks {0} untuk memastikan argumen ditempatkan di tempat penampung yang benar. Contoh :

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want to pay {2} dollars for {0} pieces of item
{1}."
print(myorder.format(quantity, itemno, price))
```

Output :

```
I want to pay 49.95 dollars for 3 pieces of item 567
```

21. Karakter Escape (*Escape Characters*)

Untuk menyisipkan karakter yang ilegal dalam string, gunakan karakter escape.

Karakter escape adalah garis miring terbalik \ diikuti oleh karakter yang ingin Anda sisipkan.

Contoh karakter ilegal adalah tanda kutip ganda di dalam string yang dikelilingi oleh tanda kutip ganda:

```
txt = "We are the so-called "Vikings" from the north."
```

```
#You will get an error if you use double quotes inside a string
that are surrounded by double quotes:
```

Output :

```
  File "demo_string_escape_error.py", line 1
    txt = "We are the so-called "Vikings" from the north."
                                      ^
SyntaxError: invalid syntax
```

Untuk memperbaiki masalah ini, gunakan karakter escape \". Karakter escape memungkinkan Anda menggunakan tanda kutip ganda ketika Anda biasanya tidak diizinkan:

```
txt = "We are the so-called \"Vikings\" from the north."
print(txt)
```

Output :

```
We are the so-called "Vikings" from the north.
```

22. Escape Characters

Karakter pelarian lain yang digunakan dalam Python :

a. \' = Single Quote

```
txt = 'It\'s alright.'
print(txt)
```

Output :

```
It's alright.
```

b. \\ = Backslash

```
txt = "This will insert one \\ (backslash)."
print(txt)
```

Output :

```
This will insert one \ (backslash).
```

c. \n = New Line

```
txt = "Hello\nWorld!"
print(txt)
```

Output :

```
Hello
World!
```

d. \r = Carriage Return

```
txt = "Hello\rWorld!"
print(txt)
```

Output :

```
Hello
World!
```

e. \t = Tab

```
txt = "Hello\tWorld!"
print(txt)
```

Output :

```
```

```
Hello   World!
```

f. \b = Backspace

```
#This example erases one character (backspace):

txt = "Hello \bWorld!"
print(txt)
```

Output :

```
HelloWorld!
```

g. \f = Form Feed

h. \ooo = Octal value

```
##A backslash followed by three integers will result in a
octal value:

txt = "\110\145\154\154\157"
print(txt)
```

Output :

```
Hello
```

i. \xhh = Hex value

```
#A backslash followed by an 'x' and a hex number represents
a hex value:

txt = "\x48\x65\x6c\x6c\x6f"
print(txt)
```

Output :

```
Hello
```

23. String Methods

Python memiliki seperangkat methods bawaan yang dapat digunakan pada string.

Catatan: Semua metode string mengembalikan nilai baru. Mereka tidak mengubah string asli

a. capitalize()

   Mengonversi karakter pertama menjadi huruf besar

b. casefold()

   Mengubah string menjadi huruf kecil

c. center()

   Mengembalikan string terpusat

d. count()

   Mengembalikan berapa kali nilai tertentu muncul dalam string

e. encode()

   Mengembalikan versi string yang disandikan

f. endswith()

   Mengembalikan nilai true jika string diakhiri dengan nilai yang ditentukan

g. expandtabs()

   Menyetel ukuran tab string

h. find()

   Mencari string untuk nilai yang ditentukan dan mengembalikan posisi di mana ia ditemukan

i. format()

   Memformat nilai yang ditentukan dalam string

j. format_map()

   Memformat nilai yang ditentukan dalam string

k. index()

   Mencari string untuk nilai tertentu dan mengembalikan posisi di mana ia ditemukan

l. isalnum()

Mengembalikan True jika semua karakter dalam string adalah alfanumerik

m. isalpha()

Mengembalikan True jika semua karakter dalam string ada dalam alfabet

n. isdecimal()

Mengembalikan True jika semua karakter dalam string adalah desimal

o. isdigit()

Mengembalikan True jika semua karakter dalam string adalah angka

p. isidentifier()

Mengembalikan True jika string adalah pengidentifikasi

q. islower()

Mengembalikan True jika semua karakter dalam string adalah huruf kecil

r. isnumeric()

Mengembalikan True jika semua karakter dalam string adalah numerik

s. isprintable()

Mengembalikan True jika semua karakter dalam string dapat dicetak

t. isspace()

Mengembalikan True jika semua karakter dalam string adalah spasi putih

u. istitle()

Mengembalikan True jika string mengikuti aturan judul

v. isupper()

Mengembalikan True jika semua karakter dalam string adalah huruf besar

w. join()

Menggabungkan elemen-elemen dari iterable ke akhir string

x. ljust()

Mengembalikan versi kanan kiri dari string

y. lower()

Mengubah string menjadi huruf kecil

z. lstrip()

Mengembalikan versi trim kiri dari string

aa. maketrans()

Mengembalikan tabel terjemahan untuk digunakan dalam terjemahan

bb. partition()

Mengembalikan tuple di mana string dipecah menjadi tiga bagian

cc. replace()

Mengembalikan string di mana nilai tertentu diganti dengan nilai tertentu

dd. rfind()

Mencari string untuk nilai yang ditentukan dan mengembalikan posisi terakhir di mana ia ditemukan

ee. rindex()

Mencari string untuk nilai tertentu dan mengembalikan posisi terakhir di mana ia ditemukan

ff. rjust()

Mengembalikan versi string yang dibenarkan dengan benar

gg. rpartition()

Mengembalikan tuple di mana string dibagi menjadi tiga bagian

hh. rsplit()

Membagi string pada pemisah yang ditentukan, dan mengembalikan daftar

ii. rstrip()

Mengembalikan versi trim kanan dari string

jj. split()

Membagi string pada pemisah yang ditentukan, dan mengembalikan daftar

kk. splitlines()

Membagi string pada jeda baris dan mengembalikan daftar

ll. startswith()

Mengembalikan nilai true jika string dimulai dengan nilai yang ditentukan

mm. strip()

Mengembalikan versi string yang dipangkas

nn. swapcase()

Tukar kasus, huruf kecil menjadi huruf besar dan sebaliknya

oo. title()

Mengonversi karakter pertama setiap kata menjadi huruf besar

pp. translate()

Mengembalikan string yang diterjemahkan

qq. upper()

Mengubah string menjadi huruf besar

rr. zfill()

Mengisi string dengan sejumlah nilai 0 yang ditentukan di awal

3.3 Praktikum

Contoh : Menghitung panjang string menggunakan fungsi len()

Pada contoh ini kita akan membuat variabel dengan nama a dan bernilai string "Hello, World!". Selanjutnya kita akan menghitung panjang string dari variabel a menggunakan fungsi len(). Berikut ini adalah langkah-langkahnya :

1. Tuliskan kode yang mendefinisikan variabel dengan nama a dengan nilai string “Hello, World!”

```
a = "Hello, World!"
```

2. Buat fungsi dengan nama ***panjang*** yang memiliki parameter variabel a

```
def panjang(a):
```

3. Tuliskan perintah return di dalam fungsi panjang dan berikan fungsi len() yang memiliki variabel a untuk dihitung panjang dari string tersebut.

```
    return len(a)
```

4. Tampilkan hasil menggunakan perintah print() dan memanggil fungsi panjang yang memiliki parameter variabel a

```
print(panjang(a))
```

5. Berikut full source code dari contoh menghitung panjang string

```
a = "Hello, World!"

def panjang(a):
   return len(a)

print(panjang(a))
```

Percobaan 1 : Menghitung panjang string menggunakan fungsi len()

Sekarang kerjakan percobaan 1 dengan mengikuti langkah-langkah dibawah ini !

1. Buatlah variabel ***tulis*** yang memiliki nilai “Belajar Python”

2. Buat fungsi ***panjang*** yang memiliki parameter variabel tulis
3. Berikan sintaks untuk menghitung string tersebut
4. Tampilkan hasil panjang nilai dari variabel tulis menggunakan perintah print()
5. Jalankan kode program percobaan 1 diatas dengan menekan tombol Check Code Validity