

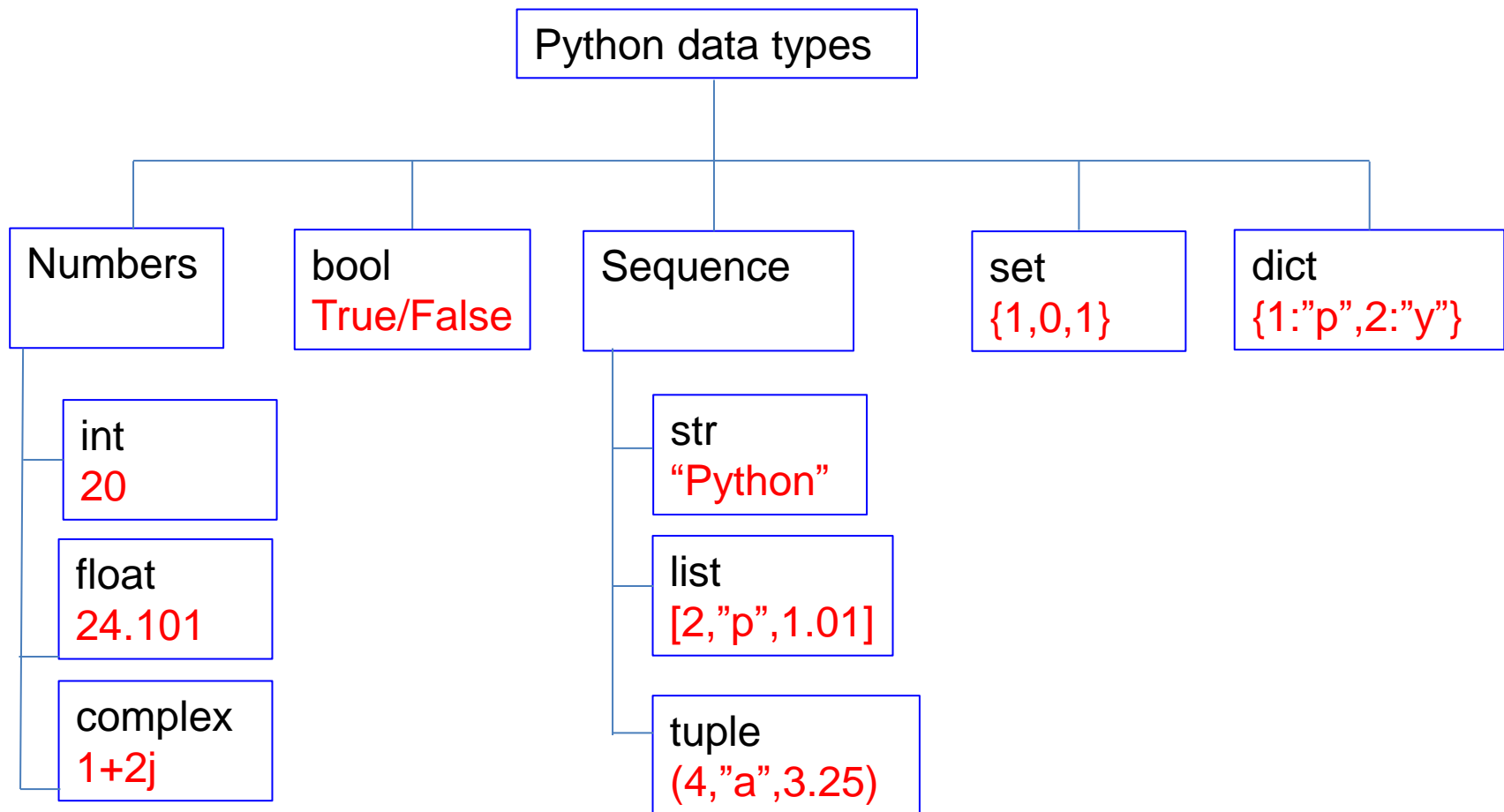
String

สตริง หรือ สายอักขระ

1. แนะนำสตริง
2. การทำงานและเข้าถึงอักขระ/ข้อความย่อยในสตริง
3. ฟังก์ชันที่ใช้ทำงานกับสตริง (string manipulation)

1. แนะนำสตริง

ชนิดข้อมูล (data type)



ชนิดข้อมูลในภาษา python มีหลายชนิด :ที่เราเรียน int,float,bool,str,list หากสนใจสามารถศึกษาเพิ่มเติมได้

1. แนะนำสตริง

สตริงหรือสายอักขระ คือข้อความ /อักขระที่เรียงต่อกันในเครื่องหมายคำพูด ภาษาไพทอนใช้ได้ทั้ง

' (single)

" (double)

''' or '''' (triple)

อักขระที่อยู่ในเครื่องหมายคำพูด เป็นได้ทั้ง ตัวอักษร ตัวเลข อักขระพิเศษ ตัวอย่างสตริง

```
word = 'python 3.8'
```

```
sentence = "I love python."
```

ใช้ triple (''' or ''''') เพื่อกำหนดข้อความที่มีหลายบรรทัดได้ เช่น

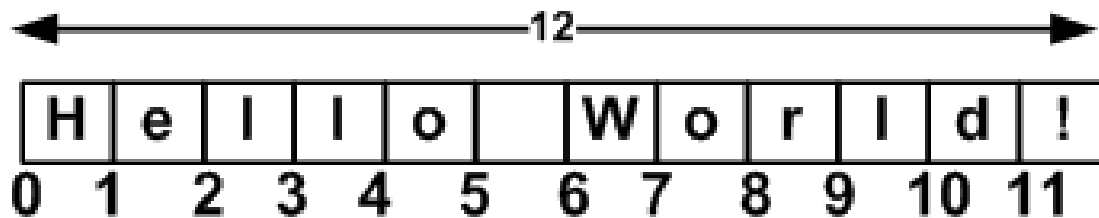
```
paragraph = """"This is a paragraph. It is  
made up of multiple lines and sentences."""
```

2. การทำงานและเข้าถึงอักขระ/ข้อความย่อยในสตริง

- ใช้เครื่องหมาย `[]` หรือ `[:]` เพื่อทำงานกับอักขระ/ข้อความย่อยในสตริง โดยระบุตำแหน่งการจัดเก็บอักขระแต่ละตัว ด้วยเลขจำนวนเต็ม เริ่มที่ 0 จบที่ `end-1`

การจัดเก็บ สตริง อธิบายได้ดังนี้

ตัวอย่างการจัดเก็บข้อความ Hello World!



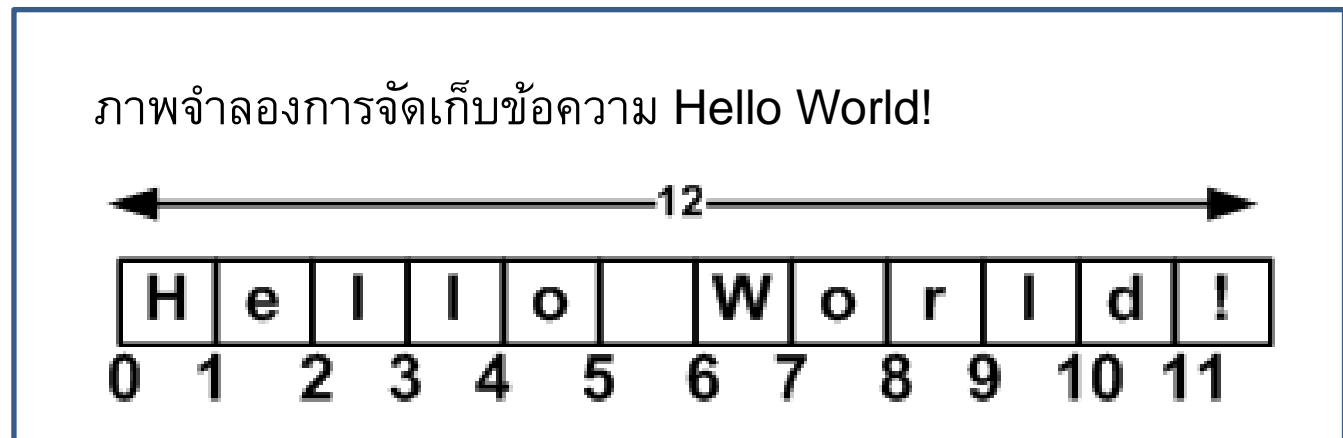
จากตัวอย่างข้อความ Hello world! มีความยาวทั้งหมด 12 ตัวอักษร ตำแหน่งการจัดเก็บตัวอักษร เริ่มที่ 0 เก็บตัวอักษร H ตัวสุดท้ายอยู่ที่ตำแหน่ง 11 คือ $end-1 = 12-1$ ซึ่งเก็บตัวอักษร !

ตัวอย่างการทำงานกับสตริง

```
str = 'Hello World!'
print (str)      # Prints complete string
print (str[0])   # Prints first character of the string
print (str[2:5]) # Prints characters starting from 3rd to 5th (end-1 คือ 5-1=4)
print (str[2:])  # Prints string starting from 3rd character
```

This will produce the following result:

```
Hello World!
H
llo
llo World!
```



การทำงานกับสตริง

นอกจากนี้ การจัดเก็บข้อความ ตำแหน่งที่จัดเก็บสามารถไล่ตำแหน่งที่เก็บจากขวาไปซ้าย ตำแหน่งเริ่มที่ -1 อธิบายได้ดังนี้

```
+----+----+----+----+----+----+
| P | y | t | h | o | n |
+----+----+----+----+----+----+
    0   1   2   3   4   5
   -6  -5  -4  -3  -2  -1
```

ปกติตำแหน่งที่เก็บ หากเริ่มจากด้านซ้ายไปขวา จะเริ่มที่ 0 จบที่ **end-1**

แต่หากขวาไปซ้าย **Negative number : start** จากด้านขวา เริ่มที่ **-1**

```
my_string = "Python"
print(my_string[0])    #P
print(my_string[-1])  #n
```

ตัวอย่างการทำงานกับสตริง

การทำงานข้อความย่อยในสตริง ใช้เครื่องหมาย [] หรือ [:] หรือ [::]

ตัวอย่าง

```
my_string = "Python"
print(my_string[0])      #P
print(my_string[-1])    #n
print(my_string[0:4])   #Pyth
print(my_string[-1:-3:-1]) #no
my_string_reversed = my_string[::-1]
print(my_string)        #Python
print(my_string_reversed) #nohtyP
```

โดย default คือ 1 แต่หากต้องการ
ไล่ตำแหน่งการจัดเก็บจากขวาไป
ซ้าย ก็ใช้เลข -1 ตัวอย่าง [::-1]

ตัวอย่างการทำงานกับสตริง

นอกจากนี้การทำงานกับสตริงเราสามารถใชั ตัวดำเนินการ + *

-เครื่องหมาย + นำสตริงหรือข้อความมาต่อกัน

-เครื่องหมาย * ทำซ้ำ

ตัวอย่าง

```
str = 'Hello World!'
print (str + "TEST ")      # Prints concatenated string
print (str * 2)           # Prints string two times
print(“+”*20)             # Print + 20 times
```

[This will produce the following result:](#)

```
Hello World!TEST
Hello World!Hello World!
+++++
```


3.String manipulation

ฟังก์ชัน ที่ใช้ในการทำงานกับสตริง

- หาข้อมูลเพิ่มเติมได้จาก

<https://docs.python.org/3/library/stdtypes.html#string-methods>

- หรือจากซอฟต์แวร์pythonที่เราติดตั้ง

เมนู Help > Python Docs

ขั้นตอนการทำคือ

Help> Python Docs(F1)

ที่ Index>พิมพ์string>เลือก methods>Display

จะได้ดังรูป

4.7.1. String Methods

Strings implement all of the *common* sequence operations, along with the additional methods

Strings also support two styles of string formatting, one providing a large degree of flexibility and handles a narrower range of types and is slightly harder to use correctly, but is often faster

The *Text Processing Services* section of the standard library covers a number of other methods

str.**capitalize()**

Return a copy of the string with its first character capitalized and the rest lowercased.

str.**casefold()**

Return a casefolded copy of the string. Casefolded strings may be used for caseless matching

Casefolding is similar to lowercasing but more aggressive because it is intended to remove all distinctions between characters. For example, `lower()` would do nothing to 'ß'; `casefold()` converts it to "ss".

The casefolding algorithm is described in section 3.13 of the Unicode Standard.

New in version 3.3.

str.**center(width[, fillchar])**

Return centered in a string of length *width*. Padding is done using the specified *fillchar*

String manipulation

- ทบทวน ฟังก์ชันที่ใช้ทำงานกับสตริง ที่เรียนมาแล้ว
 - count() – นับจำนวน ตัวอักษร/ข้อความ
 - find() – ค้นหาตำแหน่งของ ตัวอักษร/ข้อความ
 - isalpha() – ข้อความนั้นเป็นตัวอักษรทั้งหมดหรือไม่ (True/False)
 - isdigit() – ข้อความนั้นเป็นตัวเลขทั้งหมดหรือไม่ (True/False)
 - upper() - เปลี่ยนเป็นตัวพิมพ์ใหญ่
 - lower() - เปลี่ยนเป็นตัวพิมพ์ใหญ่
 - ljust() - จัดข้อความชิดซ้าย
 - rjust() - จัดข้อความชิดขวา
 - center() - จัดข้อความกึ่งกลาง

ฟังก์ชัน len(s) นับความยาวของสตริง

ตัวอย่าง

เรียนรู้ฟังก์ชันอื่นๆ เพื่อทำงานกับสตริง เพิ่มเติม

```
s='AbCd';  
print("s start=",s)  
print("islower ",s.islower())  
print("isupper ",s.isupper())  
print("isdigit ",s.isdigit())  
print("isalpha ",s.isalpha())  
print("isalphanumeric ",s.isalnum())  
print("isspace ",s.isspace())  
print("toUpper ",s.upper())  
print("toLower ",s.lower())  
print("swapcase ",s.swapcase())  
print("s end=",s)
```

```
s start= AbCd  
islower False  
isupper False  
isdigit False  
isalpha True  
isalphanumeric True  
isspace False  
toUpper ABCD  
toLower abcd  
swapcase aBcD  
s end= AbCd
```

String manipulation

strip()

lstrip()

rstrip()

replace()

join()

split()

str.strip() จะตัดช่องว่าง หน้า หลัง ออก

str.lstrip() จะตัดช่องว่าง ด้านหน้า(ซ้าย) ออก

str.rstrip() จะตัดช่องว่าง ด้านหลัง(ขวา) ออก

```
s=" i love python "
```

```
s1=s.strip()
```

```
s2=s.lstrip()
```

```
s3=s.rstrip()
```

```
print("s =",s)
```

```
print("s1 =",s1)
```

```
print("s2 =",s2)
```

```
print("s3 =",s3)
```

s 

s1 

s2 

s3 

String manipulation

strip()
rstrip()
lstrip()
rstrip()
replace()
join()
split()

str.replace(old,new) จะแทนตัวอักษร argument ตัวที่ 1 ด้วย
argument ตัวที่ 2 คือแทน old ด้วย new

เช่น

```
s="PythonP"  
s1=s.replace("P","Q")  
print(s1) #QythonQ
```

String manipulation

strip()
lstrip()
rstrip()
replace()
split()
join()

`str.join(iterable)` นำสตริงมาต่อกัน

Return a string which is the concatenation of the strings in *iterable*.

`str.split()` แยกสตริงเป็นคำ (word) ด้วยอักขระที่กำหนด

Return a list of the words in the string, using *sep* as the delimiter string

```
s="A,B,C,D"  
s1=s.split(",")  
s2="+".join(s1)
```

```
print(s1) #['A', 'B', 'C', 'D']  
print(s2) #A+B+C+D
```

	s[0]	s[1]	s[2]	s[3]
s1	A	B	C	D

ตัวอย่าง join() , split()

```
1 def getMarried(bridename, groomName):
2     brideName[1] = groomName[1]
3
4 # Input name in the form 'Firstname Lastname'
5 # What we get from split(' ') is a list in the form ['firstname', 'lastname']
6 brideName = input("What's the name of the bride: ").split(' ')
7 groomName = input("What's the name of the groom: ").split(' ')
8
9 getMarried(bridename, groomName)
10
11 # ' '.join(bridename) means joining elements of brideName with ' ' (space)
12 print("Congratulations ", ' '.join(bridename), " and ", ' '.join(groomName))
```

	0	1
bridename	"JK"	"Rolling"
groomName	"QR"	"Code"

ตัวอย่าง

```
print('Happy New Year'.find('ew'))  
print('Happy New Year'.count('ew'))  
print('Happy New Year'.replace('Happy','Brilliant'))
```

```
my_string = 'dollar'  
print('$'.join(my_string))
```

```
my_string = "ferrari"  
my_string_reversed = my_string[::-1]  
print(my_string)  
print(my_string_reversed)
```

7

1

Brilliant New Year

d\$o\$I\$I\$a\$r

ferrari

irarref

ตัวอย่าง

การคำนวณ : รับตัวเลขแล้วนำไปคำนวณ

```
s=input("Enter n1 and n2:separete with space: ")
s=s.split(" ")
n1=int(s[0])
n2=int(s[1])
cal=n1+n2
print(cal)
```

Enter n1 and n2:separete with space: **25 50**
75

การวนรอบเพื่อทำงานกับตัวอักษรในสตริง

```
s=input("Enter string ") #Python
for i in range(len(s)):
    print(s[i])
```

P
y
t
h
o
n

ตัวอย่าง การวนรอบเพื่อทำงานกับตัวอักษรในสตริง

```
# Example 1
test_str = "Canada"
for i in range(len(test_str)):
    print(i, test_str[i])

print("-----")
# Example 2
i = 0
while i < len(test_str):
    print(i, test_str[i])
    i += 1

print("-----")
# Example 3
for char in test_str:
    print(char)
```

```
0 C
1 a
2 n
3 a
4 d
5 a
-----
0 C
1 a
2 n
3 a
4 d
5 a
-----
C
a
n
a
d
a
```

แบบฝึกหัด

โจทย์ รับชื่อต้น ชื่อกลาง นามสกุล แล้วแสดงผลตามรูปแบบนี้

เช่น Enter : James Smith Borton

Result = Borton S, James

Reference

- Deitel , "Python How to program" , "Prentice-Hall,Inc." , 2002.
- Matt Telles , "Python Power !" , "Thomson Course Teachnology",2008.
- *python 3.8.5 help documentation*
- PYTHON TUTORIAL *Simply Easy Learning by tutorialspoint.com*