Tuple

- Tuple: คือ list ที่ แก้ไข ของข้างในไม่ได้ (immutable)
  - update ค่า, แก้ไข ค่าในไม่ได้
- Syntax: tuple_name = (item1, item2, item3)
  - กรณี ตัวเดียว (item1,)

- สิ่งที่ทำได้
  - indexing
  - len(), min(), max(), sum()
  - +, *, in
  - list comprehension
  - loop ได้
- สิ่งที่ทำไม่ได้
  - append
  - remove
  - insert
  - sort
  - update ตรง index

EX1

```
my_tuple = (10, 20, 30)
print(my_tuple[1])      → 20 ✓
print(len(my_tuple))    → 3  ✓
my_tuple.append(40)     → error! ✗
my_tuple[0] = 1         → error! ✗
```

(*) เอาความสามารถ สร้าง ตัวแปร หลาย ตัว มารับ ค่า ใน tuple ได้

EX

```
a, b, c = (20, 30, 40)
x, y, z = 20, 30, 40
```

* ใช้ใน function ที่ return หลายค่า

Conversion

- list() : แปลง tuple เป็น list

- list() : เปลี่ยน tuple เป็น list
- tuple() : เปลี่ยน list เป็น tuple

↳ ทั้งสอง functions ใช้ กับ range () ได้

EX.

```
my_list = list(range(5))          [0,1,2,3,4]
print(my_list)  →  [0, 1, 2, 3, 4]
my_tuple = tuple(my_list)
print(my_tuple)  →  (0, 1, 2, 3, 4)
```

String

↳ string (ข้อความ) คือ sequence ของ ตัวอักษร/character

↳ คล้าย tuple แต่ของ ข้างใน เป็น ตัวอักษร

↳ string เราไม่สามารถ update/เปลี่ยน ตัวอักษรได้
  ↘ is immutable!

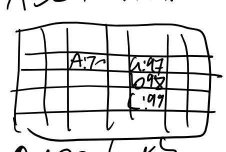
↳ เรา เข้า index ได้ และ วน ลูป ได้

```
my_string = "ABCDEF"
          -6   -5   -4   -3   -2   -1
        ('A', 'B', 'C', 'D', 'E', 'F')
index :   0    1    2    3    4    5

print(my_string[0])   →  A
print(my_string[-1])  →  F
my_string[1] = '2'  ← error! ✗ (immutable!)
```

* Aside

character มันมี ตัวเลข/id ของแต่ละตัว ASCII number



1 บอกว่าตัวไหน มาก่อนแล้วมีได้

แล้ว เปรียบ เทียบ 7,2

For loop

For loop

①

```python
name = "Juliet"
for char in name:
    print(char)
```

Output:
J
u
l
i
e
t

②

```python
name = "Juliet"
for i in range(len(name)):
    print(name[i])
```

ใช้กับ string ได้!

Output:
J
u
l
i
e
t

EX นับ ตัว อักษร ใน string

```python
target = 'T'
my_string = "Hello There TT"
count=0
for ch in my_string:
    if ch == target:
        count += 1
print(count)
```

+1 +1 +1

→ 3

Slicing

↳ การตัด string หรือ list/tuple ตาม index ที่ใช้ระบุ (เป็น range)

↳ จะได้ string / list ใหม่

↳ ใช้ได้ทั้ง string, list, tuple !

Syntax: sequences[start : stop : step]

index range

↑ เหมือน range!

Syntax: sequences [start : stop : step]

└ เหมือน range!
คือ ต้องลบก่อน stop!

① หา range ของ index ก่อน

② เอาไปหาค่าตาม index range ที่ได้มา

EX 1

```
my_string = "Hello"
             01234
new_string = my_string[0:4:1]
print(new_string) → Hell
```

① [0, 1, 2, 3]
② "H e l l"

EX 2

```
x = "Hello World"
     0 1234 5678910
y = x[2:8:2]
print(y) → low
```

① [2, 4, 6]
② "l o W"

→ Default values

↳ ถ้าไม่มี step → default เป็น 1.

```
str = "HelloX"
       012345
```

① str[0:2:] → "He"
  ↳ 0:2:1

② str[0:2] → "He"
  ↳ 0:2:1

↳ ถ้าไม่มี end → เอาจนถึงตัวสุดท้าย (รวมตัวสุดท้ายด้วย!)

str[1::1] → "elloX"
  ↳ [1:6:1]

↳ ถ้าไม่มี start → เริ่มตัวแรกสุด

str[:3:1] → "Hel"
  ↳ [0:3:1]

EX 1

str[::] → "HelloX"
  ↳ [0:6:1]

EX1 str[0:6:1] → "Xolleh" (reverse)

str[::-1] → "Xolleh" (reverse)

ถอยหลัง start = -1 [-1:-7:-1]

[-1, -2, -3, -4, -5, -6]

EX2 full_name = "Patty_Lynn_Smith"
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

P_name = fullname[0:5] (fullname[:5])

L_name = fullname[6:10]

S_name = fullname[11:16] (fullname[11:])

↓ ↑ สามารถ ตัวแปรเครื่องหมายลบได้

fullname[11: len(fullname)]

String Operators

- + = รวม string มาต่อกัน (concatenation) ผลลัพธ์ string ใหม่ออกมา

e.g.
```
new_string = "AAA" + "BBB" + "CC"
print(new_string) → AAABBBCC
```

- * = เอา string นั้นมาต่อตัวเอง n ครั้ง

e.g.
```
my_str = "ABC" * 3 → "ABCABCABC"
print(my_str) → ABCABCABC
```

- in = ใช้เช็คว่า string อยู่ในอีก string ไหม? → True / False

"A" in "ABC" → True

"ab" in "ABC" → False

"a" in "ABC" → False

String methods

String . method_name ()

* string methods จะไม่เปลี่ยน string จะส่งค่าใหม่ออกมาแทน !!

(1) String methods สำหรับเช็ค characteristic ของ string

- .isalnum() : string ทั้งก้อน เป็น ตัวอักษรหรือตัวเลข ไหม?
- .isalpha() : string ทั้งก้อน เป็น ตัวอักษร หมดไหม?
- .isdigit() : string ทั้งก้อน เป็น ตัวเลข หมดไหม?
- .islower() : string ทั้งก้อน เป็น ตัวเล็ก หมดไหม?
- .isupper() : string ทั้งก้อน เป็น ตัวใหญ่ หมดไหม? (upper case)
- .isspace() : string ทั้งก้อน เป็น ช่องว่าง ไหม?
  - '\n' = ขึ้นบรรทัดใหม่
  - '\t' = tab

จะส่ง True หรือ False ออกมา

ex

```
str1 = "ABC12"
print( str1.isalnum() ) → True
print( str1.isdigit() ) → False
print( "123".isdigit() ) → True
print( str1[-1].isdigit() )
```

"2".isdigit() → True

```
print( str[3:].isdigit() ) → True
```

3:5:1

"12".isdigit() ✓

String methods เพื่อสร้าง string ใหม่ออกมา

String methods

(1) .lower() → สร้าง string ใหม่ ที่เป็น lower case ทั้งหมด

.upper() → สร้าง string ใหม่ ที่เป็น upper case ทั้งหมด

```
my_str = "AaB"
new_str = my_str.lower() → "aab"
print(my_str, new_str) → AaB aab
                "AaB"  "aab"
```

(2) string methods สร้าง ใหม่ด้วยการลบของเก่าหน้า/ท้ายออก

(2.1) ลบช่องว่างออกจากหน้า/ท้าย (ซ้าย/ขวา) ของ string

.lstrip() = ส่ง string ใหม่ที่ลบช่องว่างด้านซ้าย/หน้าออกทั้งหมด

.rstrip() = " ขวา/ท้ายออกทั้งหมด

.strip() = " ลบทั้งหน้าและท้าย "

Ex

```
my_str = "  abc
print(my_str.lstrip()) → abc
print(my_str.rstrip()) →   abc
print(my_str.strip()) → abc
print("abc d".rstrip()) → abc d
```

* `print(my_str.lstrip().upper())`

"abc ".upper()

"ABC " → ABC

(*) ทำจากซ้ายไปขวา

(*) ทำจากซ้ายไปขวา

2.2 ลบตัวอักษร หน้า หลัง ได้

. lstrip (char) → ลบ 'char' ที่อยู่ด้านหน้าออกจนหมด

. rstrip (char) → ---------- หลัง ----------

. strip (char) → ---------- หน้า/หลัง ----------

Ex my_str = "aaabbcacddd"

print (my_str.lstrip('a')) → bbcacddd

print (my_str.rstrip('d').strip('a'))

"aaabbcac".strip('a')

"bbcac"

String methods สำหรับค้นหา

1) .find (substring) ← ctrl+f

↳ จะ index แรก ของ substring ที่ได้มา

↳ ถ้าไม่เจอ return -1

ex "AA Hello BB".find("Hello") → 3

0 1 2 3

"ABC".find("a") → -1

2) .replace (old_str, new_str) ← ctrl+f + replace

↳ แทน old_str ทั้งหมด ด้วย new_str แล้วสร้าง str ขึ้นมาใหม่ออกมา

Ex my_str = "Hello Hello Hello ABC"

new_str = my_str.replace("Hell", "X")

"X o X o ABC"

new_str = my_str.rep...

"Xo Xo Xo ABC"

* สำคัญมากที่สุด *

* .split(splitter) : ตัด string ที่ splitter ออกมาเป็น list

string "ABC=DEF=GHI".split("=")

↓

list ของ string ["ABC", "DEF", "GHI"]

Ex: "This is my cat".split(" ")

.split() ← default ช่องว่าง

["This", "is", "my", "cat"]

Ex: my_str = "101//202//303"

new_lst = my_str.split("//") → ['101', '202', '303']

print(new_list) → ['101', '202', '303']

* รับ 1 string จาก user แล้วมาเปลี่ยนเป็น list ของตัวเลข

user_input = input("Enter numbers: ")

สมมติ "10 20 30 40"

string ของตัวเลขที่คั่นด้วยช่องว่าง

↓

เป้าหมาย = [10,20,30,40]

list of str = user_input.split(" ") ↑?

["10", "20", "30", "40"]

listofnum = []

for str in listofstr:

listofnum = []

```
for str in listofstr:
    num = int(str)
    listofnum.append(num)
print(listofnum)
```

→ [10, 20, 30, 40]

---

* .join(list) → ใช้ต่อ list ของ str เป็น str

"=".join(['a', 'b', 'c'])

↓

"a=b=c"