

Advanced Python

Python for AI

CONTENTS

- Data type
- List
- String
- Tuple
- Dictionary
- Set
- File
- Important Packages

CONTENTS

- **Data type**
 - List
 - String
 - Tuple
 - Dictionary
 - Set
 - File
 - Important Packages

Basic data types

Integer	1, 2, 3, 0, -1, -2
Float	1.5, 0.5, -3.21, 1.0
String	'Joe', 'Schmoe', "Joe", "Schmoe"
Boolean	True, False

Abstract data types

List	[1, 2, 3, 4, 5]
Tuple	(1, 2, 3, 4, 5)
Dictionary	{'lr': 0.1, 'optimizer': 'Adam',}
Class	Student, Employer

Mutable and Immutable

Immutable

Integer, Float

String

Boolean

Tuple

Mutable

List

Dictionary

Set

Class

Mutable and Immutable

Immutable

Integer, Float

String

Boolean

Tuple

a_number

```
1 a_number = 5
2 print(a_number, ' --- ', id(a_number))
3
4 a_number = 6
5 print(a_number, ' --- ', id(a_number))

5 --- 140710083400640
6 --- 140710083400672
```

Integer Object

5

140710083400640

Mutable and Immutable

Immutable

Integer, Float

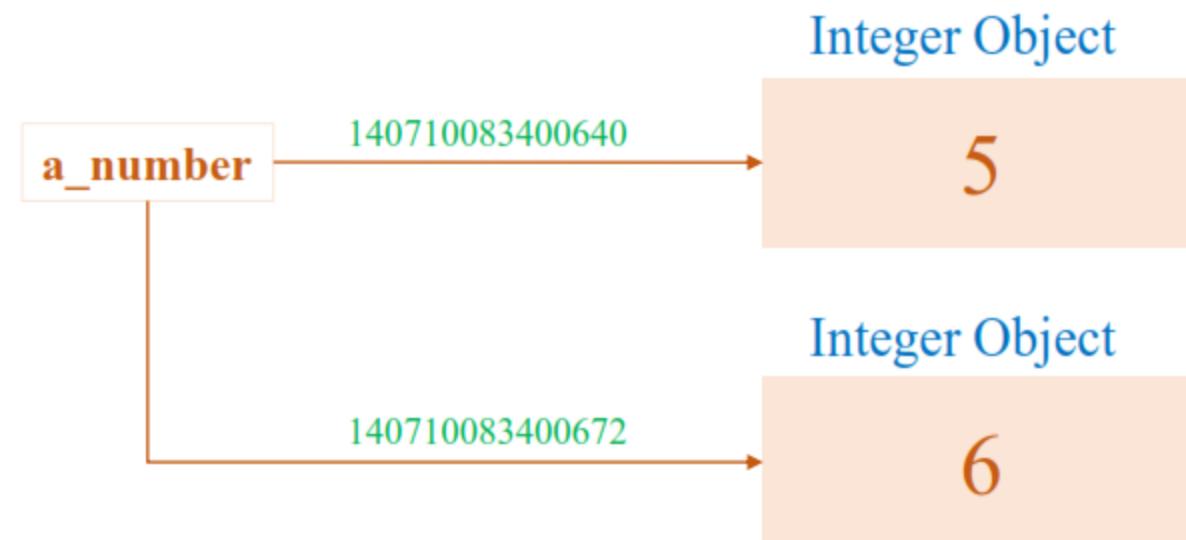
String

Boolean

Tuple

```
1 a_number = 5  
2 print(a_number, ' --- ', id(a_number))  
3  
4 a_number = 6  
5 print(a_number, ' --- ', id(a_number))
```

5 --- 140710083400640
6 --- 140710083400672



Mutable and Immutable

Immutable

Integer, Float

String

Boolean

Tuple

```
1 a_number = 5
2 print(a_number, ' --- ', id(a_number))
3
4 a_number = 6
5 print(a_number, ' --- ', id(a_number))

5 --- 140710083400640
6 --- 140710083400672
```

Integer Object

5

Integer Object

6

a_number

140710083400672

Mutable and Immutable

Immutable

Integer, Float

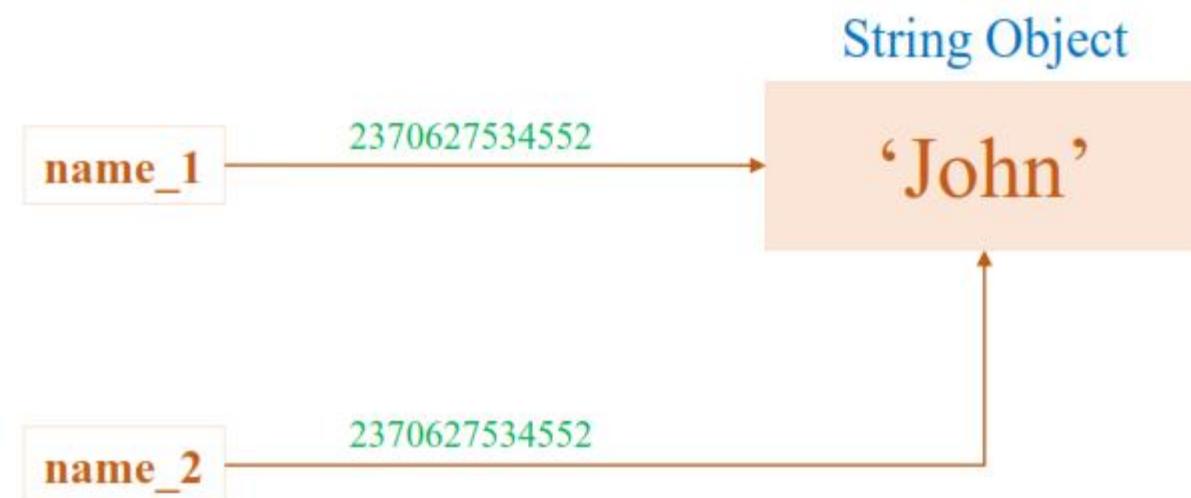
String

Boolean

Tuple

```
1 name_1 = 'John'
2 print(name_1, ' --- ', id(name_1))
3
4 name_2 = name_1
5 print(name_2, ' --- ', id(name_2))
6
7 name_2 = 'Jose'
8 print(name_2, ' --- ', id(name_2))
```

John --- 2370627534552
John --- 2370627534552
Jose --- 2370627692616



Mutable and Immutable

Immutable

Integer, Float

String

Boolean

Tuple

```
1 name_1 = 'John'
2 print(name_1, ' --- ', id(name_1))
3
4 name_2 = name_1
5 print(name_2, ' --- ', id(name_2))
6
7 name_2 = 'Jose'
8 print(name_2, ' --- ', id(name_2))
```

John --- 2370627534552
John --- 2370627534552
Jose --- 2370627692616

String Object



String Object



Mutable and Immutable

Immutable

Integer, Float

String

Boolean

Tuple

However

```
1 number_1 = 5
2 number_2 = 5
3
4 print(id(number_1))
5 print(id(number_2))
```

140710083400640
140710083400640

```
1 name_1 = 'A'
2 print(name_1, ' --- ', id(name_1))
3
4 name_2 = 'A'
5 print(name_2, ' --- ', id(name_2))
```

A --- 2370587045368
A --- 2370587045368

Mutable and Immutable

Mutable

List

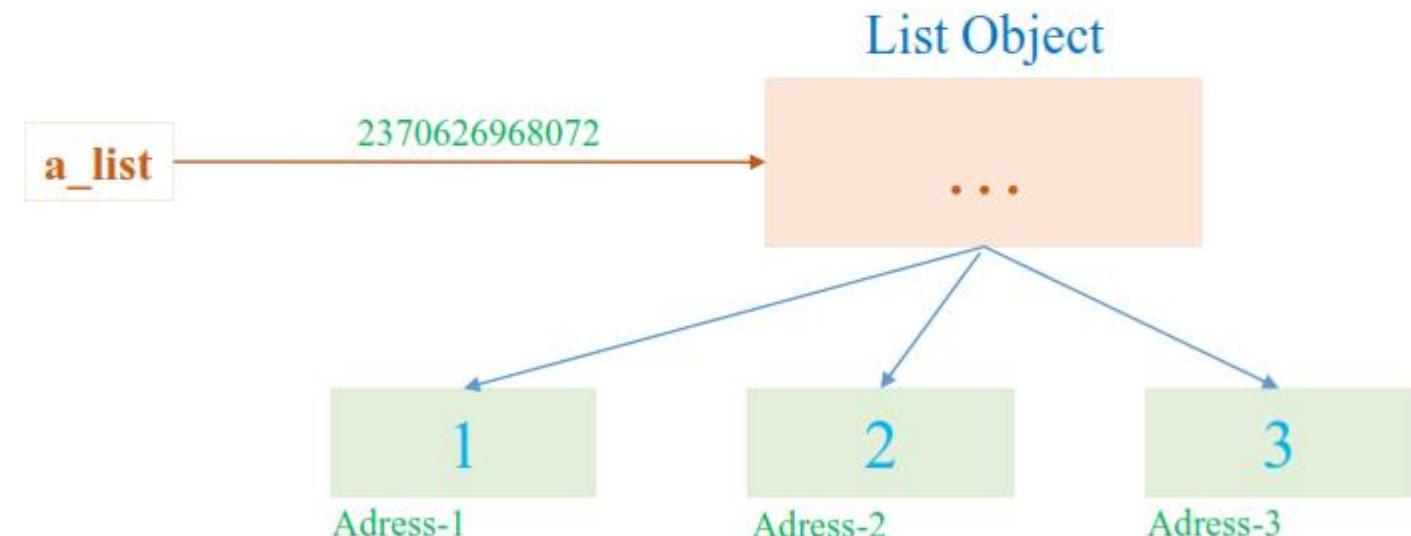
Dictionary

Set

Class

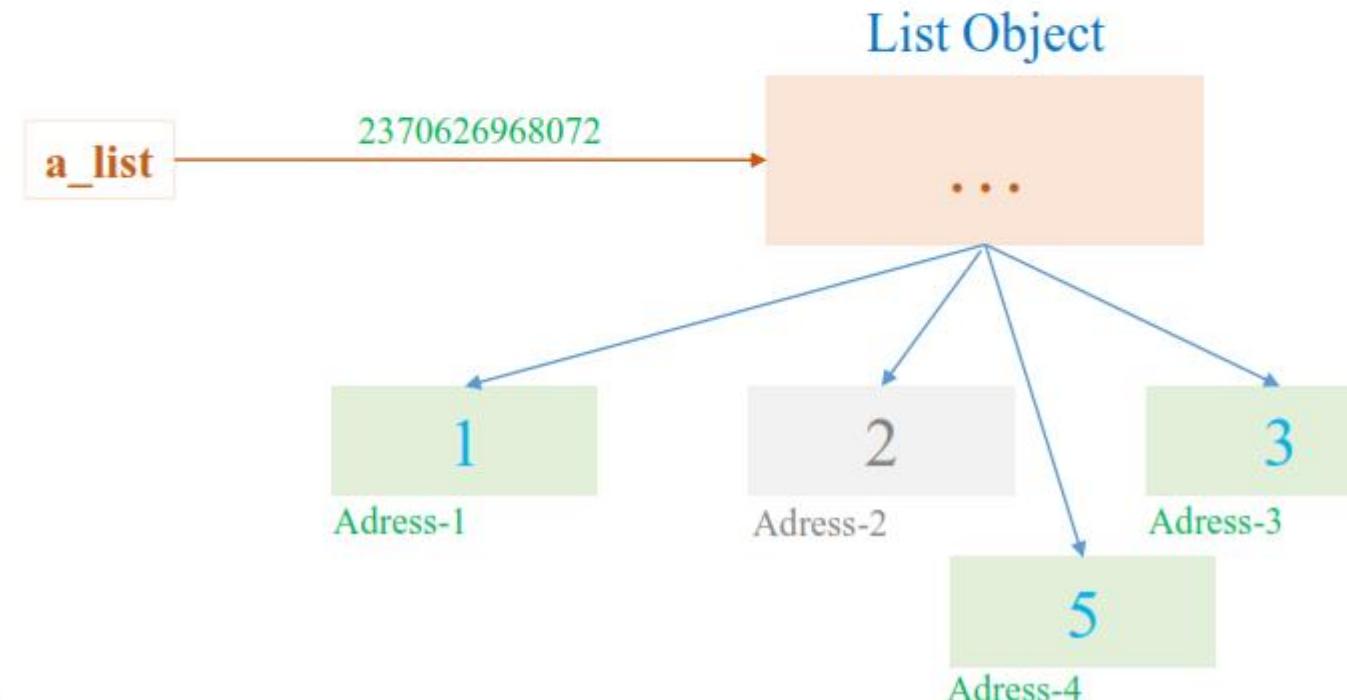
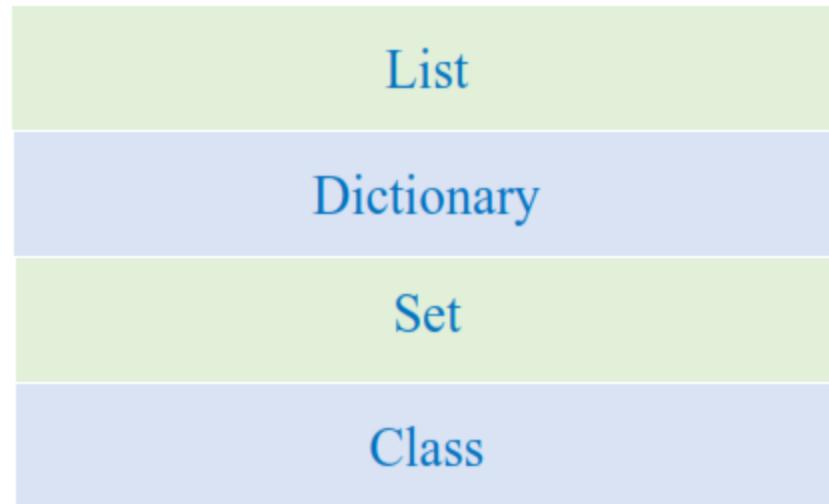
```
1 a_list = [1,2,3]
2 print(a_list, id(a_list))
3
4 a_list[1] = 5
5 print(a_list, id(a_list))
```

```
[1, 2, 3] 2370626968072
[1, 5, 3] 2370626968072
```



Mutable and Immutable

Mutable



Mutable and Immutable

Mutable

List

Dictionary

Set

Class

Shadow or Deep

```
1 a_list = [1,[1,2],3]
2 print(a_list, id(a_list))
3 print(id(a_list[0]))
4 print(id(a_list[1]))
5 print(id(a_list[2]))
6
7 print('\n')
8 a_list[1][0] = 5
9 a_list[0] = 8
10
11 print(a_list, id(a_list))
12 print(id(a_list[0]))
13 print(id(a_list[1]))
14 print(id(a_list[2]))
```

```
[1, [1, 2], 3] 2370627741832
140710083400512
2370627742280
140710083400576
```

```
[8, [5, 2], 3] 2370627741832
140710083400736
2370627742280
140710083400576
```

CONTENTS

- Data type
- **List**
- String
- Tuple
- Dictionary
- Set
- File
- Important Packages

- A container that can contain elements

list_name = [element-1, ..., element-n]

```
1. # danh sách trống
2. empty_list = []
3.
4. # danh sách số tự nhiên nhỏ hơn 10
5. my_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
6.
7. # danh sách kết hợp nhiều kiểu dữ liệu
8. mixedList = [True, 5, 'some string', 123.45]
9. n_list = ["Happy", [2,0,1,5]]
10.
11. #danh sách các loại hoa quả
12. shoppingList = ['táo', 'chuối', 'cherries', 'dâu', 'mận']
```

❖ Index

`data = [4, 5, 6, 7, 8, 9]`

Forward index



Backward index



`data[0]`



`data[3]`



`data[-1]`



`data[-3]`



❖ Slicing

`data = [4, 5, 6, 7, 8, 9]`

Forward index



`data[:3]`



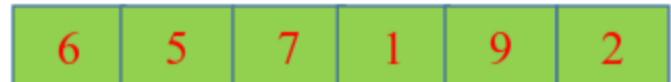
`data[2:4]`



`data[3:]`

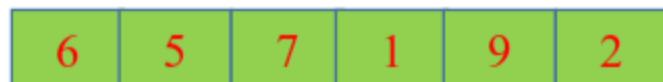


❖ Add an element

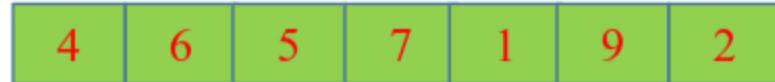
data = 

data.append(4) # thêm 4 vào vị trí cuối list

data = 

data = 

data.insert(0, 4) # thêm 4 vào vị trí có
index = 0

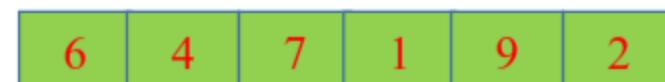
data = 

❖ Updating an element

data = 

thay đổi phần tử thứ 1

data[1] = 4

data = 

❖ + and * operators

```
data1 = [6 5 7]
```

```
data2 = [1 9 2]
```

nối 2 list

```
data = data1 + data2
```

```
data = [6 5 7 1 9 2]
```

```
data = [6 5]
```

nhân list với một số nguyên

```
data_m = data * 3
```

```
data_m = [6 5 6 5 6 5]
```

❖ sort() – Sắp xếp các phần tử

```
data = [6 5 7 1 9 2]
```

```
data.sort()
```

```
data = [1 2 5 6 7 9]
```

```
data = [6 5 7 1 9 2]
```

```
data.sort(reverse = True)
```

```
data = [9 7 6 5 2 1]
```

❖ Delete an element

`data = [6, 5, 7, 1, 9, 2]`

`data.pop(2)` # tại vị trí index = 2

`data = [6, 5, 1, 9, 2]`

`data = [6, 5, 7, 1, 9, 2]`

`data.remove(5)` # xóa phần tử đầu tiên
có giá trị là 5

`data = [6, 7, 1, 9, 2]`

❖ Delete elements

`data = [6, 5, 7, 1, 9, 2]`

xóa phần tử thứ 1 và 2
`del data[1:3]`

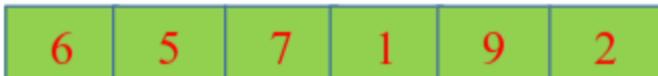
`data = [6, 1, 9, 2]`

`data = [6, 5, 7, 1, 9, 2]`

`data.clear()`

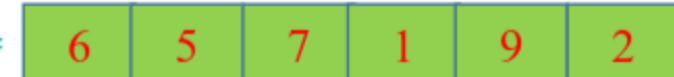
`data = []`

index() – Trả về vị trí đầu tiên

data = 

trả về vị trí của phần tử đầu tiên có giá trị là 9
data.index(9) = 4

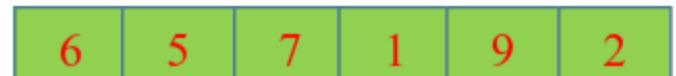
reverse() – Đảo ngược vị trí các phần tử

data = 

data.reverse()

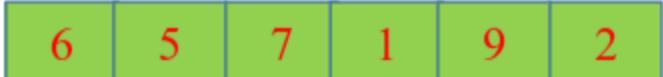
data = 

count() – Trả về số lần xuất hiện của một phần tử

data = 

trả về số lần phần tử 7 xuất hiện trong list
data.count(7) = 1

copy() – copy một list

data = 

data_copy = data.copy()

data_copy = 

List Comprehension

list_name = [expression for element in iterable]

```
1 # create a list
2 list_1 = []
3 for c in 'Hello':
4     list_1.append(c)
5
6 print(list_1)
```

```
['H', 'e', 'l', 'l', 'o']
```

```
1 # list comprehension
2 list_2 = [c for c in 'Hello']
3
4 print(list_2)
```

```
['H', 'e', 'l', 'l', 'o']
```

```
1 # create a list
2 list_1 = []
3 for i in range(10):
4     if i%2 == 0:
5         list_1.append(i)
6
7 print(list_1)
```

```
[0, 2, 4, 6, 8]
```

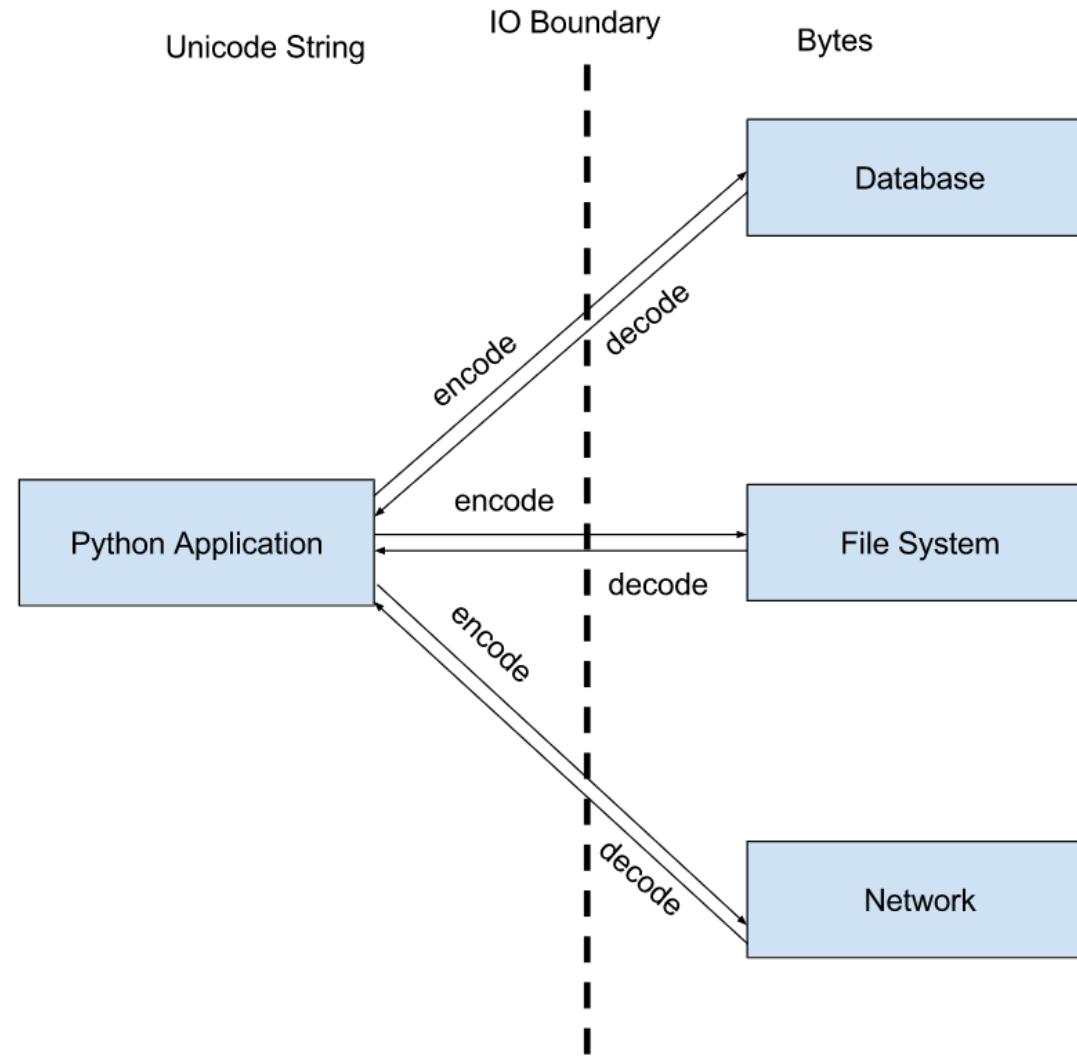
```
1 # list comprehension
2 list_2 = [i for i in range(10) if i%2 == 0]
3
4 print(list_2)
```

```
[0, 2, 4, 6, 8]
```

CONTENTS

- Data type
- List
- **String**
- Tuple
- Dictionary
- Set
- File
- Important Packages

String representation



American Standard Code for Information Interchange

ASCII represents 128 characters

ASCII is stored as 8-bit byte

ASCII is not standardized

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]
19	13	10011	23	[DEVICE CONTROL 3]
20	14	10100	24	[DEVICE CONTROL 4]
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]
22	16	10110	26	[SYNCHRONOUS IDLE]
23	17	10111	27	[ENG OF TRANS. BLOCK]
24	18	11000	30	[CANCEL]
25	19	11001	31	[END OF MEDIUM]
26	1A	11010	32	[SUBSTITUTE]
27	1B	11011	33	[ESCAPE]
28	1C	11100	34	[FILE SEPARATOR]
29	1D	11101	35	[GROUP SEPARATOR]
30	1E	11110	36	[RECORD SEPARATOR]
31	1F	11111	37	[UNIT SEPARATOR]
32	20	100000	40	[SPACE]

Returning an integer representing the ASCII (Unicode) character

```
1 ascii_value = ord('A')
2 print(ascii_value)
```

65

Returning the ASCII (Unicode) character

```
1 a_character = chr(65)
2 print(a_character)
```

A

Decimal	Hexadecimal	Binary	Octal	Char
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P

```
1. text1 = 'Tôi yêu AI VIET NAM'  
2. text2 = "Tôi yêu AI VIET NAM"  
3. text3 = '''Tôi yêu AI VIET NAM'''  
4. text4 = """Tôi yêu AI VIET NAM"""
```

```
1 text1  
'Tôi yêu AI VIET NAM '
```

```
1 text2  
'Tôi yêu AI VIET NAM '
```

```
1 text3  
'Tôi yêu AI VIET NAM'
```

```
1 text4  
'Tôi yêu AI VIET NAM'
```

```
1. text1 = "Tôi yêu 'AI VIET NAM'"  
2. text2 = 'Tôi yêu "AI VIET NAM"'  
3. text3 = """Chuỗi có kí tự đặc biệt " và ' bên trong""""
```

```
1 text1  
"Tôi yêu 'AI VIET NAM'"
```

```
1 text2  
'Tôi yêu "AI VIET NAM'"
```

```
1 text3  
'Chuỗi có kí tự đặc biệt " và \' bên trong'
```

```
1 text = "Chuỗi có kí tự đặc biệt \", \' bên trong"  
2 print(text)
```

Chuỗi có kí tự đặc biệt ", ' bên trong

Insert into a string

```
1 value1 = "AI VIETNAM"
2 print("Hello %s. Have a nice day!" % value1)
3 print("Hello %s. Have a nice day!" % (value1))
4
5 value2 = "Hi"
6 print("****%s %s****" % (value2, value1))
```

Hello AI VIETNAM. Have a nice day!
Hello AI VIETNAM. Have a nice day!
Hi AI VIETNAM

```
1 i = 5
2 value = 48.2
3
4 print("Error rate at the %dth iteration is %f" % (i, value))
```

Error rate at the 5th iteration is 48.200000

```
1 value1 = "hello"
2 value2 = "AIVIETNAM"
3
4 print("%s, %s!" % (value1, value2))
5 print("%s, %s!" % (value2, value1))
```

hello, AIVIETNAM!
AIVIETNAM, hello!

```
1 value1 = "hello"
2 value2 = "AIVIETNAM"
3
4 s1 = "{n1}, {n2}!".format(n1=value1, n2=value2)
5 s2 = "{n1}, {n2}!".format(n2=value2, n1=value1)
6
7 print(s1)
8 print(s2)
```

hello, AIVIETNAM!
hello, AIVIETNAM!

❖ + and * operators

```
1 s1 = 'Tôi thích '
2 s2 = 'AI!'
3
4 s3 = s1 + s2
5 s4 = s3*2
6
7 print(s3)
8 print(s4)
```

Tôi thích AI!
Tôi thích AI!Tôi thích AI!

❖ Logic operators

```
1 s1 = 'a'
2 s2 = 'b'
3
4 print(s1 == s1)
5 print(s1 == s2)
6
7 print(s1 != s2)
8 print(s1 < s2)
9 print(s1 > s2)
```

True
False
True
True
False

isdigit(): Kiểm tra xem string gồm các kí tự số

```

1 # Kiểm tra xem string gồm các chữ số
2 print("10".isdigit())
3 print("abc".isdigit())

```

True
False

isalpha(): Kiểm tra xem string chỉ được tạo từ các kí tự chữ cái

```

1 # Kiểm tra xem string chỉ được tạo từ các kí tự chữ cái
2 print("10".isalpha())
3 print("abc".isalpha())

```

False
True

islower(): Kiểm tra xem string với tất cả các kí tự ở dạng chữ thường

```

1 # Kiểm tra xem string với tất cả các kí tự ở dạng chữ thường
2 print("ab".islower())
3 print("Ab".islower())

```

True
False

isupper(): Kiểm tra xem string với tất cả các kí tự ở dạng chữ hoa

```

1 # Kiểm tra xem string với tất cả các kí tự ở dạng chữ hoa
2 print("Ab".isupper())
3 print("AB".isupper())

```

False
True

istitle(): Kiểm tra xem string có bắt đầu bằng chữ in hoa

```

1 # Kiểm tra xem string có bắt đầu bằng chữ in hoa
2 print("Hello".istitle())
3 print("HELLO".istitle())
4 print("hello".istitle())

```

True
False
False

isspace(): Kiểm tra xem string chỉ là khoảng trắng

```

1 # Kiểm tra xem string chỉ là khoảng trắng
2 print("").isspace()
3 print(" ".isspace())
4 print("  ".isspace())
5 print("abc ".isspace())

```

False
True
True
False

count() để đếm số kí tự xuất hiện trong một string,
len() để tính chiều dài của một string

```

1 s = "AI VIET NAM"
2
3 print(s.count('A'))
4 print(s.count('a'))
5 print(len(mystr))

```

2
0
11

title(): Chuyển đổi kí tự đầu từng từ (word) trong một chuỗi thành kí tự hoa

```
1 # Chuyển đổi kí tự đầu từng từ (word) trong một chuỗi thành kí tự hoa
2 mystr = "Đây là bài học của AI VIET NAM"
3 mystr.title()
```

'ĐÂY LÀ BÀI HỌC CỦA AI VIET NAM'

swapcase(): Chuyển đổi các kí tự từ chữ thường sang chữ hoa và ngược lại

```
1 # Chuyển đổi các kí tự từ chữ thường sang chữ hoa và ngược lại
2 mystr = "Đây là bài học của AI VIET NAM"
3 mystr.swapcase()
```

'đÂY LÀ BÀI HỌC CỦA AI VIET NAM'

capitalize(): Chuyển đổi chữ cái đầu tiên của một chuỗi thành chữ hoa và các kí tự sau thành chữ thường

```
1 # Chuyển đổi chữ cái đầu tiên của một chuỗi thành chữ hoa
2 # và các kí tự sau thành chữ thường
3 mystr = "Đây là bài học của AI VIET NAM"
4 mystr.capitalize()
```

'ĐÂY LÀ BÀI HỌC CỦA AI VIET NAM'

upper(): Chuyển đổi tất cả các kí tự trong chuỗi thành chữ hoa

```
1 # Chuyển đổi tất cả các kí tự trong chuỗi thành chữ hoa  
2 mystr = "Đây là bài học của AI VIET NAM"  
3 mystr.upper()
```

'ĐÂY LÀ BÀI HỌC CỦA AI VIET NAM'

lower(): Chuyển đổi các kí tự sang chữ thường

```
1 # Chuyển đổi các kí tự sang chữ thường  
2 mystr = "Đây là bài học của AI VIET NAM"  
3 mystr.lower()
```

'đây là bài học của ai viet nam'

center(): Chính chuỗi ở trung tâm, và chiều dài của chuỗi là 40

```
1 # Chính chuỗi ở trung tâm, và chiều dài của chuỗi là 40  
2 mystr = "Đây là bài học của AI VIET NAM"  
3 mystr.center(40, '-')
```

'-----Đây là bài học của AI VIET NAM-----'

strip(): Loại bỏ khoảng trắng ở cả hai đầu của chuỗi

```
1 # Loại bỏ khoảng trắng ở cả hai đầu của chuỗi  
2 mystr = " Đây là bài học của AI VIET NAM "  
3 mystr.strip()
```

'Đây là bài học của AI VIET NAM'

replace(): Thay thế chuỗi

```
1 # Thay thế chuỗi  
2 mystr = "Đây là bài học của AI VIET NAM"  
3 mystr.replace('AI VIET NAM', 'AIVIETNAM')
```

'Đây là bài học của AIVIETNAM'

partition(): Tách chuỗi

```
1 # Tách chuỗi  
2 mystr = "Đây là bài học của AI VIET NAM"  
3 mystr.partition('của')
```

('Đây là bài học ', 'của', ' AI VIET NAM')

endswith(): Kiểm tra phần kết thúc của một string.

```
1 # Kiểm tra phần kết thúc của một string  
2 mystr = "Đây là bài học của AI VIET NAM"  
3 print(mystr.endswith('VIET NAM'))  
4 print(mystr.endswith('bài học'))
```

True
False

startswith(): Kiểm tra phần đầu của một string.

```
1 # Kiểm tra phần đầu của một string  
2 mystr = "Đây là bài học của AI VIET NAM"  
3 print(mystr.startswith('Đây'))  
4 print(mystr.startswith('AI'))
```

True
False

find(): Tìm vị trí xuất hiện của string s1 trong string s2. Giá trị -1 được trả về trong trường hợp s1 không được tìm thấy trong s2.

```
1 # Tìm vị trí của một string trong một string khác  
2 # trả về -1 nếu không tìm thấy  
3 mystr = "Đây là bài học của AI VIET NAM"  
4 print(mystr.find('học'))  
5 print(mystr.find('hello'))
```

11
-1

CONTENTS

- Data type
- List
- String
- **Tuple**
- Dictionary
- Set
- File
- Important Packages

❖ Structure

tuple_name = (element-1, ..., element-n)

Create a tuple

```
1. t = (1, 2, 3)
2.
3. print(t[0])
4. print(t[1])
5. print(t[2])
```

1
2
3

Tuple unpacking

```
1 x1,y1,z1 = ('a','b','c')
2 (x2,y2,z2) = ('a','b','c')
3 print(x1)
4 print(x2)
```

a
a

❖ Structure

tuple_name = (element-1, ..., element-n)

() can be removed

```
1. t = 1, 2  
2. print(t)
```

```
(1, 2)
```

Tuple with one element

```
1 var1 = (1 + 2) * 5  
2 print(type(var1), ' ', var1)  
3  
4 var2 = (1)  
5 print(type(var2), ' ', var2)  
6  
7 var3 = (1,)  
8 print(type(var3), ' ', var3)
```

```
<class 'int'>    15  
<class 'int'>    1  
<class 'tuple'>   (1,)
```

❖ + and * operators

```
1. t1 = (1, 0)
2. print(t1)
3.
4. t1 += (2,)
5. print(t1)
```

```
(1, 0)
(1, 0, 2)
```

```
1. t = (1,) * 5
```

```
(1, 1, 1, 1, 1)
```

index() - tìm sự xuất hiện của một giá trị
count() - đếm số lần xuất hiện của một giá trị

```
1. t = (1,2,3,1)
2. count = t.count(1)
3. index = t.index(2)
4.
5. print(count)
6. print(index)
```

```
2
1
```

Dùng hàm zip() cho tuple

```

1. t1 = (1, 2, 3, 4, 5)
2. t2 = ('a', 'b', 'c', 'd', 'e')
3.
4. print(t1)
5. print(t2)
6.
7. t3 = zip(t1, t2)
8. for x,y in t3:
9.     print(x, y)

```

```

(1, 2, 3, 4, 5)
('a', 'b', 'c', 'd', 'e')
1 a
2 b
3 c
4 d
5 e

```

Sắp xếp các giá trị trong một tuple

```

1. t = (4, 7, 3, 9, 6)
2. t_s = sorted(t)
3. print(t_s)

```

```
[3, 4, 6, 7, 9]
```

...Tuple

len() - Tìm chiều dài của một tuple

```

1. t = (1, 2, 3, 4)
2. len(t)

```

```
4
```

Lấy giá trị min và max của một tuple

```

1. t = (1, 2, 3, 4, 5)
2.
3. print(min(t))
4. print(max(t))
5. print(sum(t))

```

```
1
5
15
```

❖ Immutable

```
1. t = (1, 2, 3, 4, 5)
2. t[2] = 9
3. print(t)
```

```
-----  
TypeError                                     Traceback (most recent call last)  
<ipython-input-30-b1f85e7d332a> in <module>  
      1 t = (1, 2, 3, 4, 5)  
----> 2 t[2] = 9  
      3 print(t)  
  
TypeError: 'tuple' object does not support item assignment
```

however

```
1. t = (1, 2, [3, 10])
2. t[2][1] = 4
3. print(t)
```

```
(1, 2, [3, 4])
```

❖ Example: Solve quadratic equation

```

1 import math
2
3 def quadratic_equation(a, b, c):
4     """
5         This function aims at solving the quadratic equation
6         a, b, c --- three parameters and a != 0
7     """
8
9     # compute delta
10    delta = b*b - 4*a*c
11
12    if delta < 0:
13        return ()
14    elif delta == 0:
15        x = -b/2*a
16        return (x,)
17    else:
18        x1 = (-b+math.sqrt(delta))/(2*a)
19        x2 = (-b-math.sqrt(delta))/(2*a)
20        return (x1,x2)
21

```

Case 1: delta<0

```

1 result = quadratic_equation(a=5, b=0, c=1)
2 print(type(result))
3 print(len(result))
4 print(result)

```

```

<class 'tuple'>
0
()
```

Case 2: delta>0

```

1 result = quadratic_equation(a=5, b=5, c=1)
2 print(type(result))
3 print(len(result))
4 print(result)

```

```

<class 'tuple'>
2
(-0.276393202250021, -0.7236067977499789)
```

Case 3: delta=0

```

1 result = quadratic_equation(a=4, b=4, c=1)
2 print(type(result))
3 print(len(result))
4 print(result)

```

```

<class 'tuple'>
1
(-8.0,)
```

Data is protected

```

1 result = quadratic_equation(a=4, b=4, c=1)
2 result[0] = 1

```

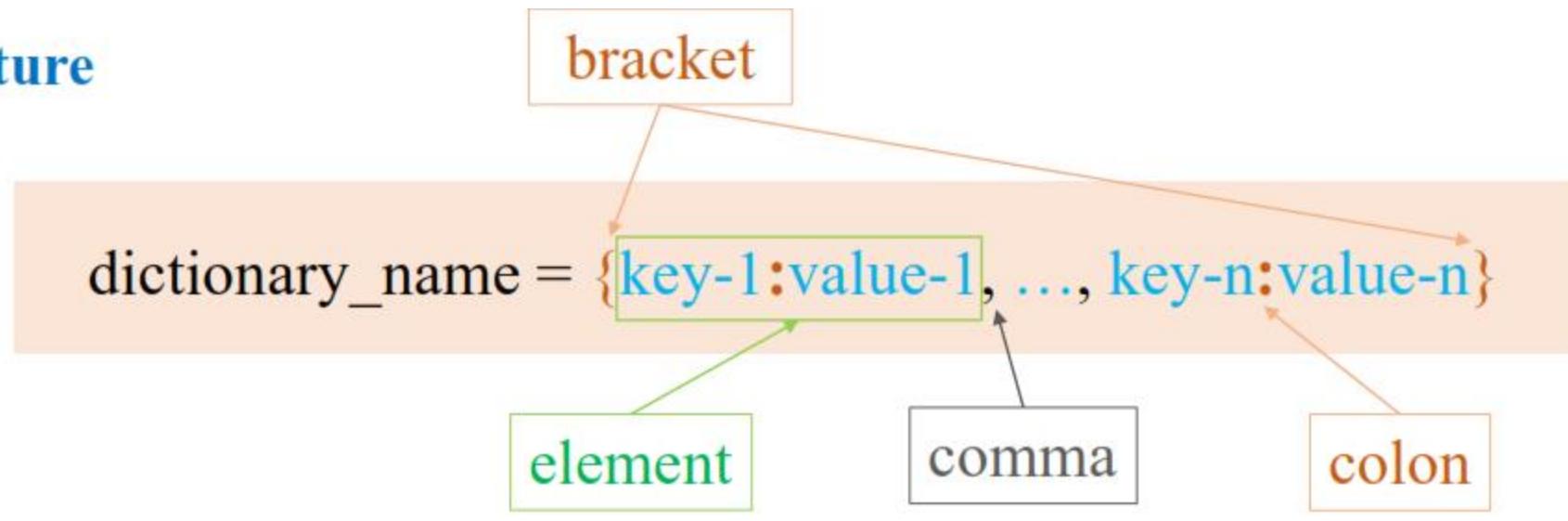
TypeError Traceback (most recent call last):
 <ipython-input-21-55f394e5ddc8> in <module>
 1 result = quadratic_equation(a=4, b=4, c=1)
----> 2 result[0] = 1

TypeError: 'tuple' object does not support item assignment

CONTENTS

- Data type
- List
- String
- Tuple
- **Dictionary**
- Set
- File
- Important Packages

Structure



Create a dictionary

```
1 parameters = {'learning_rate': 0.1,  
2                 'optimizer': 'Adam',  
3                 'metric': 'Accuracy'}  
4  
5 print(parameters)  
6 print(type(parameters))
```

```
{'learning_rate': 0.1, 'optimizer': 'Adam', 'metric': 'Accuracy'}  
<class 'dict'>
```

❖ Update a value

```
1 parameters = {'learning_rate': 0.1,  
2                 'metric': 'Accuracy'}  
3  
4 parameters['learning_rate'] = 0.2  
5 print(parameters)  
  
{'learning_rate': 0.2, 'metric': 'Accuracy'}
```

❖ Copy a dictionary

```
1 parameters = {'learning_rate': 0.1,  
2                 'metric': 'Accuracy'}  
3  
4 a_copy = parameters.copy()  
5 a_copy['learning_rate'] = 0.2  
6  
7 print(parameters)  
8 print(a_copy)  
  
{'learning_rate': 0.1, 'metric': 'Accuracy'}  
{'learning_rate': 0.2, 'metric': 'Accuracy'}
```

❖ Hàm copy() chỉ sao chép kiểu shallow

```
1. d1 = {'a': [1,2], 'b': 5}
2. d2 = d1.copy()
3.
4. # thay đổi giá trị d2 sẽ ảnh hưởng đến d1
5. d2['a'][0] = 3
6. d2['a'][1] = 4
7.
8. print('d1:', d1)
9. print('d2:', d2)
```

```
d1: {'a': [3, 4], 'b': 5}
d2: {'a': [3, 4], 'b': 5}
```

❖ Sử dụng hàm deepcopy() trong module copy

```
1. import copy
2.
3. d1 = {'a': [1,2], 'b': 5}
4. d2 = copy.deepcopy(d1)
5.
6. # thay đổi giá trị d2
7. d2['a'][0] = 3
8. d2['a'][1] = 4
9.
10. print('d1:', d1)
11. print('d2:', d2)
```

```
d1: {'a': [1, 2], 'b': 5}
d2: {'a': [3, 4], 'b': 5}
```

❖ Get keys and values

Get keys

```
1 keys = parameters.keys()  
2 for key in keys:  
3     print(key)
```

learning_rate
optimizer
metric

Get values

```
1 values = parameters.values()  
2 for value in values:  
3     print(value)
```

0.1
Adam
Accuracy

Get keys

```
1 parameters = {'learning_rate': 0.1,  
2                      'optimizer': 'Adam',  
3                      'metric': 'Accuracy'}  
4  
5 for key in parameters:  
6     print(key)
```

learning_rate
optimizer
metric

Get keys and values

```
1 parameters = {'learning_rate': 0.1,  
2                      'optimizer': 'Adam',  
3                      'metric': 'Accuracy'}  
4  
5 items = parameters.items()  
6 for key, value in items:  
7     print(key, value)
```

learning_rate 0.1
optimizer Adam
metric Accuracy

❖ Get a value by a key

Get value using get() function

```
1 parameters = {'learning_rate': 0.1,  
2                 'optimizer': 'Adam',  
3                 'metric': 'Accuracy'}  
4  
5 value = parameters.get('learning_rate')  
6 print(value)  
7  
8 print('\nAfter using get() function')  
9 print(parameters)
```

0.1

After using get() function
{'learning_rate': 0.1, 'optimizer': 'Adam', 'metric': 'Accuracy'}

Get value and delete the corresponding item

```
1 parameters = {'learning_rate': 0.1,  
2                 'optimizer': 'Adam',  
3                 'metric': 'Accuracy'}  
4  
5 value = parameters.pop('learning_rate')  
6 print(value)  
7  
8 print('\nAfter using pop() function')  
9 print(parameters)
```

0.1

After using pop() function
{'optimizer': 'Adam', 'metric': 'Accuracy'}

popitem() - lấy ra một phần tử ở cuối dictionary

```

1 parameters = {'learning_rate': 0.1,
2                 'optimizer': 'Adam',
3                 'metric': 'Accuracy'}
4
5 item = parameters.popitem()
6
7 print(item)
8 print(parameters)

('metric', 'Accuracy')
{'learning_rate': 0.1, 'optimizer': 'Adam'}
```

clear() - xóa tất cả các phần tử của một dictionary

```

1 parameters = {'learning_rate': 0.1,
2                 'metric': 'Accuracy'}
3
4 print('Before using clear() function')
5 print(parameters)
6
7 parameters.clear()
8
9 print('\nAfter using clear() function')
10 print(parameters)
```

Before using clear() function
 {'learning_rate': 0.1, 'metric': 'Accuracy'}

After using clear() function
 {}

Use del keyword to delete an item

```

1 parameters = {'learning_rate': 0.1,
2                 'metric': 'Accuracy'}
3 print(parameters)
4
5 del parameters['metric']
6 print(parameters)

{'learning_rate': 0.1, 'metric': 'Accuracy'}
{'learning_rate': 0.1}
```

❖ Key that does not exist

Try to delete a non-existing item

```
1 parameters = {'learning_rate': 0.1,  
2                 'metric': 'Accuracy'}  
3  
4 del parameters['algorithm']
```

```
-----  
KeyError Traceback  
<ipython-input-29-e759e1753a28> in <module>  
      2                 'metric': 'Accuracy')  
      3  
----> 4 del parameters['algorithm']  
  
KeyError: 'algorithm'
```

Try to get an item by a non-existing key

```
1 parameters = {'learning_rate': 0.1,  
2                 'optimizer': 'Adam',  
3                 'metric': 'Accuracy'}  
4  
5 value = parameters.pop('algorithm')
```

```
-----  
KeyError Traceback  
<ipython-input-30-8310550c04f4> in <module>  
      3                 'metric': 'Accuracy')  
      4  
----> 5 value = parameters.pop('algorithm')  
  
KeyError: 'algorithm'
```

CONTENTS

- Data type
- List
- String
- Tuple
- Dictionary
- **Set**
- File
- Important Packages

❖ Structure

set_name = {element-1, ..., element-n}

Create a set

```
1 a_set = {1, 2, 3, 4, 5}
2 print(a_set)
3 print(type(thisset))
```

```
{1, 2, 3, 4, 5}
<class 'set'>
```

❖ Structure

Method	Description
<u>add()</u>	Adds an element to the set
<u>clear()</u>	Removes all the elements from the set
<u>copy()</u>	Returns a copy of the set
<u>difference()</u>	Returns a set containing the difference between two or more sets
<u>difference_update()</u>	Removes the items in this set that are also included in another, specified set
<u>discard()</u>	Remove the specified item
<u>intersection()</u>	Returns a set, that is the intersection of two other sets
<u>intersection_update()</u>	Removes the items in this set that are not present in other, specified set(s)

❖ Add() function

```
1 a_set = {1, 2, 3, 4, 5}  
2 print(a_set)  
3  
4 a_set.add(6)  
5 print(a_set)
```

```
{1, 2, 3, 4, 5}  
{1, 2, 3, 4, 5, 6}
```

```
1 a_set = {1, 2, 3, 4, 5}  
2 print(a_set)  
3  
4 a_set.add(2)  
5 print(a_set)
```

```
{1, 2, 3, 4, 5}  
{1, 2, 3, 4, 5}
```

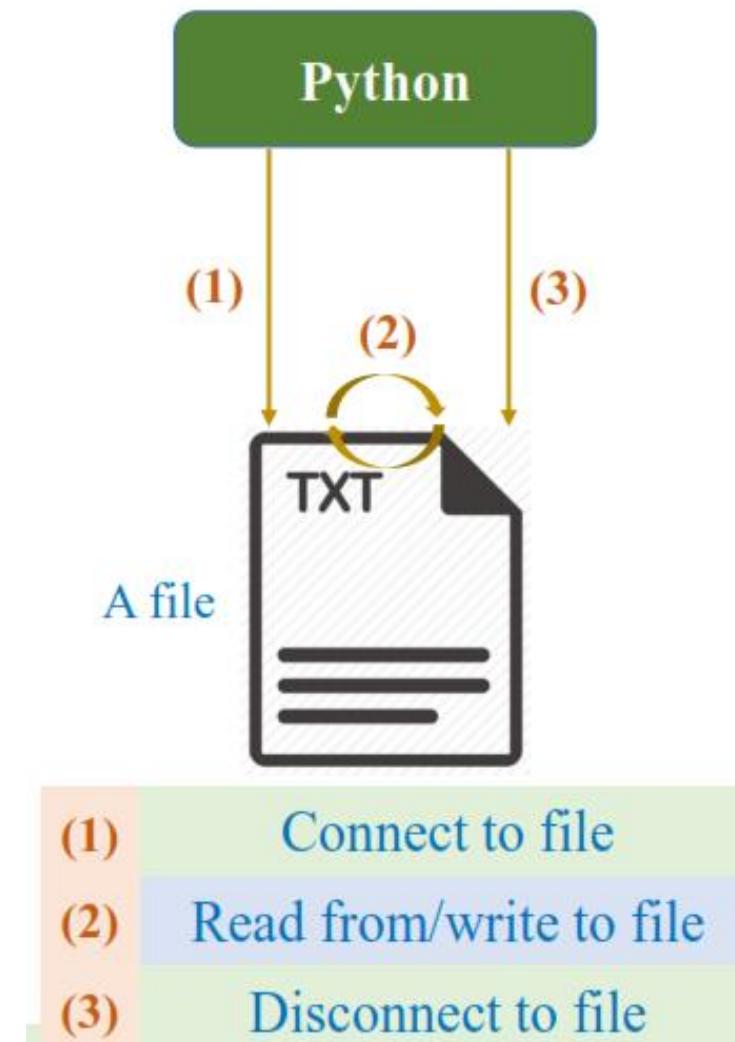
❖ Structure

Method	Description
<u>isdisjoint()</u>	Returns whether two sets have a intersection or not
<u>issubset()</u>	Returns whether another set contains this set or not
<u>issuperset()</u>	Returns whether this set contains another set or not
<u>pop()</u>	Removes an element from the set
<u>remove()</u>	Removes the specified element
<u>symmetric_difference()</u>	Returns a set with the symmetric differences of two sets
<u>symmetric_difference_update()</u>	inserts the symmetric differences from this set and another
<u>union()</u>	Return a set containing the union of sets
<u>update()</u>	Update the set with the union of this set and others

CONTENTS

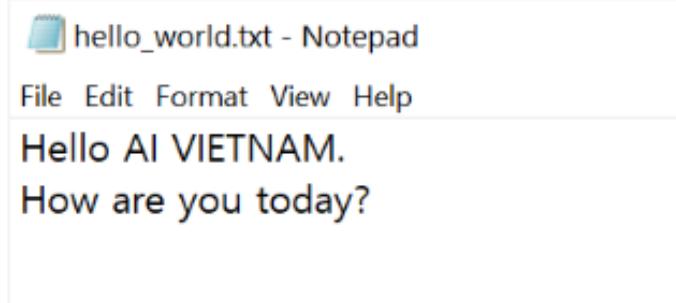
- Data type
- List
- String
- Tuple
- Dictionary
- Set
- **File**
- Important Packages

❖ Typical procedure



Read from a file (already exist)

(1)	open(file_path, 'r')
(2)	read()
(3)	close()

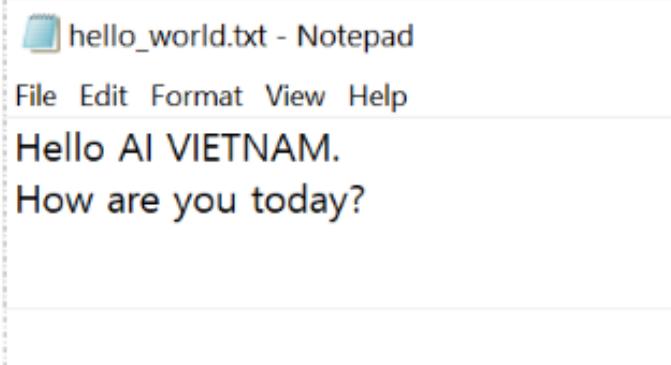


```
1 # kết nối với file
2 a_file = open('hello_world.txt','r')
3
4 # read content as string
5 data = a_file.read()
6
7 print(type(data))
8 print(data)
9
10 # Đóng kết nối với file
11 a_file.close()
```

```
<class 'str'>
Hello AI VIETNAM.
How are you today?
```

Read content from a file as lines

```
(1) open(file_path, 'r')
(2)     readlines()
(3)         close()
```



```
1 # kết nối với file
2 a_file = open('hello_world.txt', 'r')
3
4 # read content as string
5 lines = a_file.readlines()
6 for line in lines:
7     print(line)
8
9 # Đóng kết nối với file
10 a_file.close()
```

Hello AI VIETNAM.

How are you today?

Write to a file (not exist)

- (1) `open(file_path, 'w')`
- (2) `write()`
- (3) `close()`

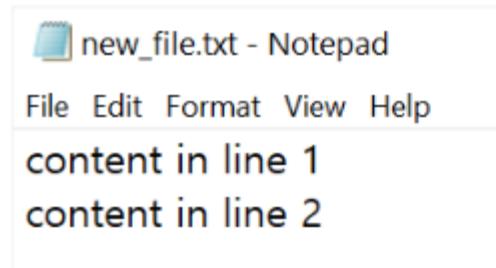
```
1 # kết nối với file
2 a_file = open('new_file.txt', 'w')
3
4 text1 = 'content in line 1 \n'
5 a_file.write(text1)
6
7 text2 = 'content in line 2 \n'
8 a_file.write(text2)
9
10 # Đóng kết nối với file
11 a_file.close()
```

 new_file.txt - Notepad
File Edit Format View Help
content in line 1
content in line 2

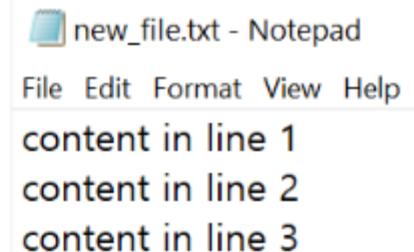
Write to a file (appending content if the file already exists)

- (1) open(file_path, 'a')
- (2) write()
- (3) close()

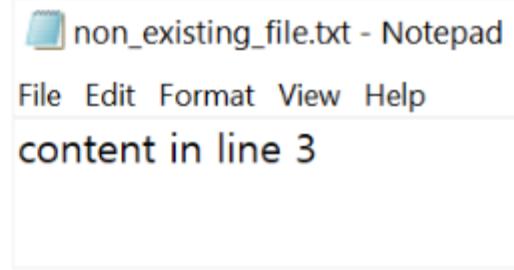
...File



```
1 # kết nối với file
2 a_file = open('new_file.txt', 'a')
3
4 text3 = 'content in line 3 \n'
5 a_file.write(text3)
6
7 # Đóng kết nối với file
8 a_file.close()
```



```
1 # kết nối với file
2 a_file = open('non_existing_file.txt', 'a')
3
4 text3 = 'content in line 3 \n'
5 a_file.write(text3)
6
7 # Đóng kết nối với file
8 a_file.close()
```



❖ Useful functions

Check if a file exists

```

1 import os
2
3 file_path1 = 'my_file.txt'
4 check1 = os.path.exists(file_path1)
5 print('my_file.txt có tồn tại không?', check1)
6
7 file_path2 = 'non_existence_file.txt'
8 check2 = os.path.exists(file_path2)
9 print('non_existence_file.txt có tồn tại không?', check2)

my_file.txt có tồn tại không? True
non_existence_file.txt có tồn tại không? False

```

String splitting

```

1 string1 = '001,john,12-06-1999'
2 tokens = string1.split(',')
3
4 for token in tokens:
5     print(token)

```

001
john
12-06-1999

String joining

```

1 student_info = ['001', 'john', '12-06-1999']
2 string_joined = (',').join(student_info)
3
4 print(string_joined)

```

001,john,12-06-1999

❖ Example: data.csv

area	price
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7

data.csv - Notepad

File Edit Format View Help

```
area,price
6.7,9.1
4.6,5.9
3.5,4.6
5.5,6.7
```

```
1 # kết nối với file
2 file = open('data.csv', 'r')
3
4 # read lines
5 lines = file.readlines()
6
7 # in các dòng
8 for line in lines:
9     print(line)
10
11 # Đóng kết nối với file
12 file.close()
```

```
area,price
6.7,9.1
4.6,5.9
3.5,4.6
5.5,6.7
```

❖ Example

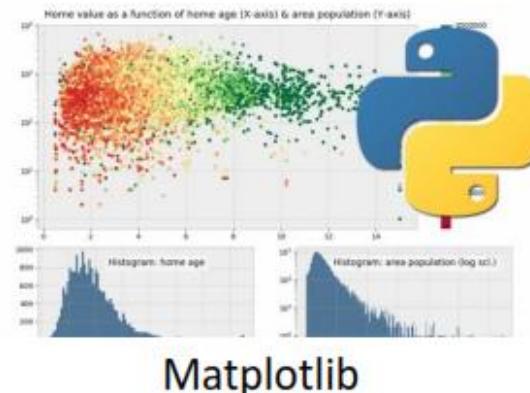


The screenshot shows a Notepad window titled "iris_demo.csv - Notepad". The menu bar includes File, Edit, Format, View, and Help. The content of the file is as follows:

Petal_Length	Petal_Width	Label
1.4	0.2	0.0
1.5	0.2	0.0
3.0	1.1	1.0
4.1	1.3	1.0

```
1 # kết nối với file
2 file = open('iris_demo.csv', 'r')
3
4 # read lines
5 lines = file.readlines()
6
7 # in các dòng
8 for line in lines:
9     print(line)
10
11 # Đóng kết nối với file
12 file.close()
```

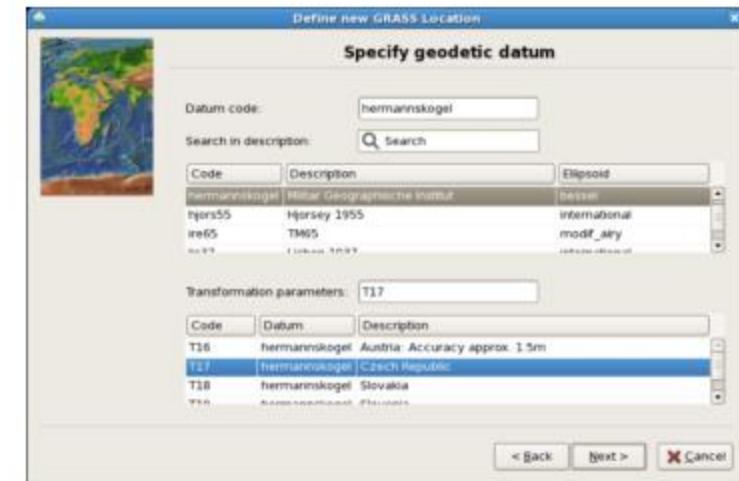
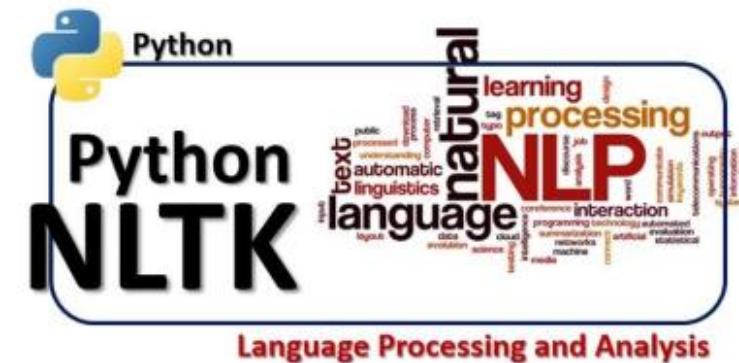
```
Petal_Length, Petal_Width, Label
1.4, 0.2, 0.0
1.5, 0.2, 0.0
3.0, 1.1, 1.0
4.1, 1.3, 1.0
```



SQLAlchemy



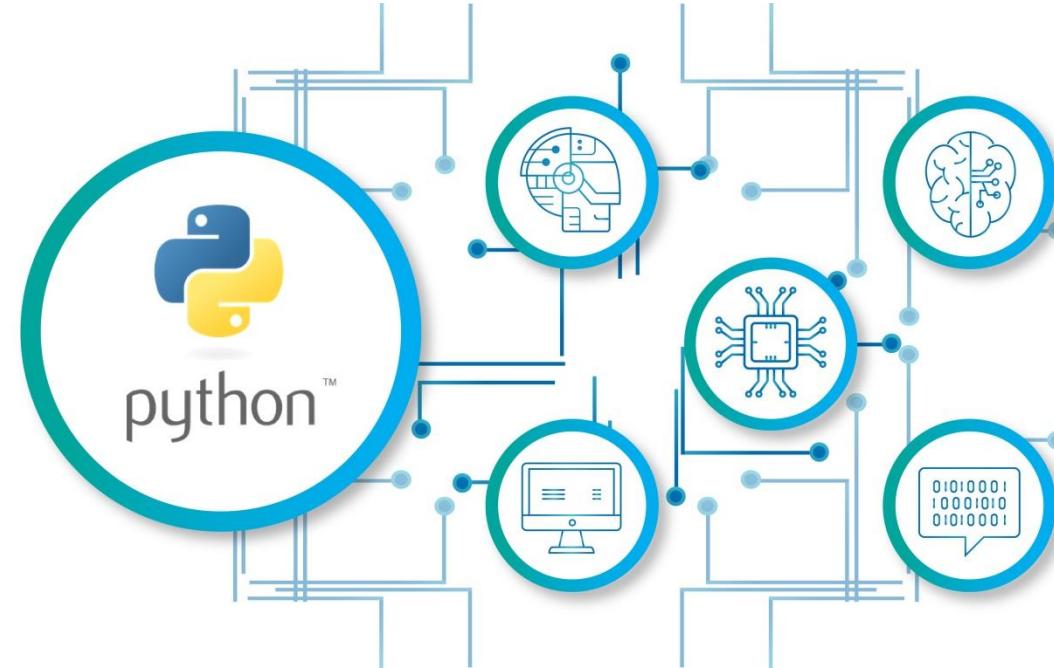
Important Packages



wxPython



Python for AI



Thank You...!