ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA

LẬP TRÌNH PYTHON

Khoa Công nghệ thông tin
ThS. Nguyễn Thị Lệ Quyên

Nội dung môn học

- Cơ bản ngôn ngữ lập trình Python
- Numpy và Machine Learning
- Tương tác Cơ sở dữ liệu
- Cơ bản HTML, CSS, JavaScript
- Flask framework
- Kiến thức bổ sung

CƠ BẢN

NGÔN NGỮ LẬP TRÌNH PYTHON

Nội dung

1. Tổng quan và hướng dẫn cài đặt
2. Cơ bản Python
   - Lệnh nhập/ xuất
   - Kiểu dữ liệu: Chuỗi, List, Tuple, Set, Dict
   - Điều kiện, vòng lặp
   - Hàm
3. Hướng đối tượng Python
4. Exception
5. Regular Expressions
6. Bài tập

Các editor của Python

- PyCharm
- Jupiter Notebook
- Atom
- Sublime Text
- Visual Studio Code
- Google colab

Tham khảo

- The Python Tutorial — Python 3.10.2 documentation
- Python Tutorial (w3schools.com)

Ví dụ

```c
#include <stdio.h>
int main(){
    for (int i=0; i<100; i++)
        printf("I love you");
    return 0;
}
```

```python
print("I love you" * 100)
```

Giới thiệu

- Ngôn ngữ kịch bản và thông dịch
- Trong sáng, gần gũi và dễ đọc
  - Tăng cường sử dụng từ khóa tiếng Anh, hạn chế các kí hiệu
  - Hướng tới sự đơn giản
    - Có while bỏ do...while
    - Có elif bỏ switch...case
  - Đa năng
    - Lập trình web
    - Ứng dụng desktop, đồ họa, game
    - Lập trình cho điện thoại
    - Đặc biệt hiệu quả trong lập trình tính toán khoa học

Lệnh nhập/xuất

- Lệnh xuất
  - Cú pháp: **`print(nội_dung, [sep="", end="\n"])`**
  - Hàm có thể in ra 1 chuỗi UTF-8 bất kỳ
  - Chuỗi dùng ký tự nháy đơn `'` hoặc nháy kép `"`
  - Ví dụ:

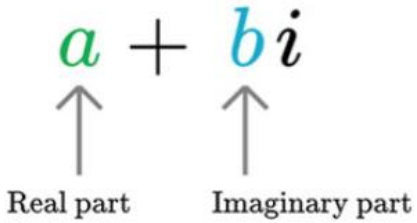
```
print("Hello world!")
print("Tôi học Lập trình Python")
```

Kiểu dữ liệu

Category	Name	Description
None	None	The null object
Numeric	int	Integer
	float	Floating point number
	complex	Complex number
	bool	Boolean (True, False)
Sequences	str	String of characters
	list	List of arbitrary objects
	tuple	Group of arbitrary items
	range	Creates a range of intergers
Mapping	dict	Dictionary of key-value pairs
	set	Mutable, unordered collection of unique items
	frozenset	Immutable set

Kiểu dữ liệu: Numbers, Boolean, None

- Có 3 kiểu đại diện cho số trong Python:
  - Số nguyên (integers)
  - Số thực (floating point numbers)
  - Số phức (complex numbers)
    - VD: `c = 1 + 2j`



- Thử đoạn code sau:

```
x = 2.01
print(x*100)
```

Kiểu dữ liệu: Numbers, Boolean, None

- Kiểu Boolean
  - True
  - False
- Kiểu None (NoneType)
  - null

```
winner = None
print(winner is None)
# Hoắc so sánh khác sử dụng is not
```

Phép toán

Operator	Description	Example
+	Add the left and right values together	1 + 2
−	Subtract the right value from the left value	3 − 2
*	Multiple the left and right values	3 * 4
/	Divide the left value by the right value	12/3
//	Integer division (ignore any remainder)	12//3
%	Modulus (aka the remainder operator)—only return any remainder	13%3
**	Exponent (or power of) operator—with the left value raised to the power of the right	3 ** 4

Bài tập 1

Kết quả của đoạn code sau là gì?

```
print('True division 3/2:', 3 / 2)
print('True division -3/2:', -3 / 2)
print('Integer division 3//2:', 3 // 2)
print('Integer division -3//2:', -3 // 2)
```

Biến

- Không khai báo mà chỉ gán giá trị để Python tự động nhận kiểu dữ liệu
- Kiểu động (khác với kiểu tĩnh của Java hoặc C/C++)

```
my_variable = 'Quyen'
print(my_variable)
my_variable = 42
print(my_variable)
my_variable = True
print(my_variable)
```

```
# Kết quả
Quyen
42
True
```

Lệnh nhập xuất

- Nhập dữ liệu
  - Nhập 1 chuỗi
    - Cú pháp: **`input()`**
  - Nhập 1 số:
    - **`int(input())`**
  - Nhập nhiều số trên 1 hàng:
    - **`x, y, z = map(int, input.split())`**

Lệnh nhập xuất

- Ép kiểu
  - Chương trình sau sẽ báo lỗi vì khác kiểu

```
age = 18
print("Tôi" + age + "tuổi")

TypeError: can only concatenate str (not "int") to str
```

  - Có 3 hàm ép kiểu
    - str()
    - int()
    - float()
    - bool()

```
print("Tôi" + str(age) + "tuổi")
```

LỆNH ĐIỀU KIỆN

Lệnh if

- Cấu trúc
  - if *điều kiện* :

    <tab>Lệnh 1

    <tab>Lệnh 2
- Chú ý sử dụng ***Tab*** để biết các lệnh bên trong điều kiện if
- Giá trị 0, chuỗi rỗng “”, None đều bằng False; ngược lại là True

If - else

- Cấu trúc
  - if *điều kiện* :

    <tab>Lệnh 1

    else :

    <tab>Lệnh 2

elif

- Cấu trúc

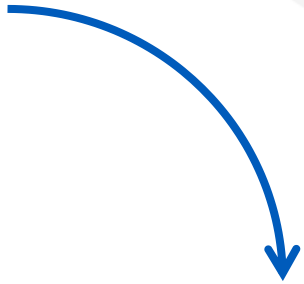
```
if điều kiện :
    <tab>Lệnh 1
elif điều kiện :
    <tab>Lệnh 2
elif điều kiện :
    <tab>Lệnh 3
else :
    <tab>Lệnh 4
```

If expression

- Cú pháp:

```
<result1> if <condition-is-met> else <result2>
```

```
age = 15
status = None
if (age > 12) and age < 20:
    status = 'teenager'
else:
    status = 'not teenager'
print(status)
```



```
status = ('teenager' if age > 12 and age < 20 else 'not teenager')
print(status)
```

Toán tử so sánh

- a == b
- a != b
- a > b
- a >= b
- a < b
- a <= b

And/Or/Not

- *ĐK1* and *ĐK2*
- *ĐK1* or *ĐK2*
- *not ĐK*

- *Ví dụ*
  - *print(n_1 > 8 and n_1 < 14)*

Bài tập ví dụ

- Viết chương trình nhập vào một số nguyên dương x. In ra chữ “YES” nếu x là số chẵn, ngược lại in số chữ “NO”
- Dữ liệu vào
  - Một số nguyên dương x
- Kết quả
  - In ra chữ “YES” nếu x là số chẵn, ngược lại in số chữ “NO”.

Gợi ý giải

- Cách 1:
  - Xét tính chia hết cho 2,
    - Nếu chia 2 dư 1 thì là lẽ
    - Ngược lại thì là chẵn
- Cách 2:
  - Xét bít cuối cùng
    - x & 1
- Cách 3:
  - Xét chữ số cuối cùng của số biểu diễn dưới dạng xâu

LỆNH VÒNG LẶP

Lệnh while

- **while** điều kiện:

  <tab> Xử lý 1

  <tab> Xử lý 2

- Ví dụ

```
count = 0
print('Starting')
while count < 10:
    print(count, ' ', end='')    # Thuộc vòng lặp while
    count += 1                   # Cũng thuộc vòng while
print()                          # KO thuộc vòng while
print('Done')
```

```
# Kết quả
Starting
0 1 2 3 4 5 6 7 8 9
Done
```

Lênh for

- Cấu trúc
  - **for** biến **in** range(...):
    <tab> xử lý
- Ví du:

```
print('Print out values in a range')
for i in range(0, 10):
    print(i, ' ', end='')
print()
print('Done’)

# Kết quả
Print out values in a range
0 1 2 3 4 5 6 7 8 9
Done
```

Lệnh for

- Xét ví dụ:

```
for _ in range(0,10,2):
    print('.', end='')
print()
```

- for –else ?

```
num = int(input('Enter a number to check for: '))
for i in range(0, 6):
    if i == num:
        break
    print(i, ' ', end='')
else:
    print()
    print('All iterations successful')
```

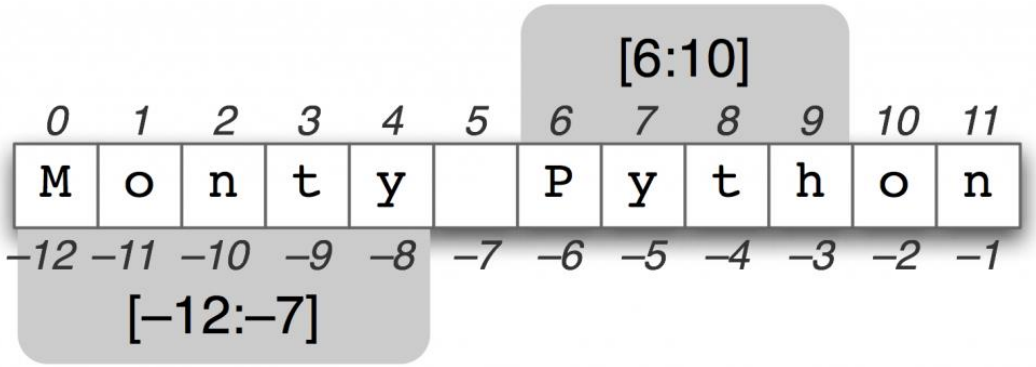
Các từ khóa trong vòng lặp

- continue
- break
- enumerate()

```
list = ["a", "b"]
for index, value in enumerate(list): # lấy index của list
    print(index, value)
```

Kiểu dữ liệu: str

- Bất biến



Kiểu dữ liệu: str

- Có thể sử dụng nháy đơn hoặc nháy kép
- Sử dụng ba dấu nháy đơn hoặc ba dấu nháy kép để biểu thị một chuỗi trên nhiều dòng

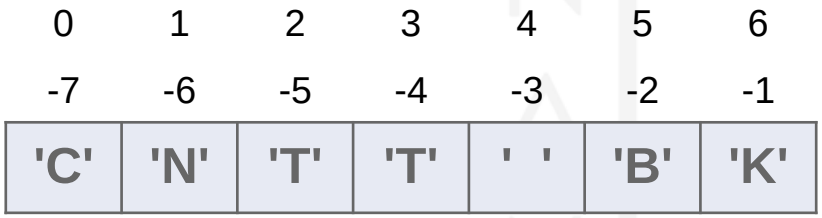
```
print("""Tháp mười đẹp nhất bông sen
Việt Nam đẹp nhất có tên Bác Hồ""")
```

- Sử dụng toán tử + để nối chuỗi
- Sử dụng toán tử * để lặp lại chuỗi

```
print("Hello" + "Mary")
print("Hello Peter " * 5)
```

Kiểu dữ liệu: str

- Các ký tự của chuỗi có thể được truy cập bằng chỉ mục tiến hoặc lùi
  - Chỉ mục tiến bắt đầu từ 0, 1, 2, ...
  - Chỉ mục ngược bắt đầu từ -1, -2, ... Trong đó -1 là phần tử cuối cùng



Kiểu dữ liệu: str

- Chuỗi con
  - Có thể tạo các chuỗi con có độ dài nhỏ hơn

```
str1 = "Python"
print(str1[:3]) # lấy từ vị trí đầu tiên đến trước kí tự thứ 3
print(str1[1:5]) # lấy từ vị trí 1 đến vị trí thứ 4
print(str1[-5:]) # # lấy từ vị trí -5 đến hết chuỗi
```

```
Pyt
ytho
ython
```

Kiểu dữ liệu: str

- Chuỗi trong Python là bất biến → không thể gán giá trị của các kí tự trong chuỗi thành 1 ki tự khác

```
str1 = "Python"
str1[0] = "Q“

TypeError: 'str' object does not support item assignment
```

- → Tạo một chuỗi khác

```
str1 = "Python"
str1 = "Q" + str1[1:]
print(str1)
```

Kiểu dữ liệu: str

- Định dạng chuỗi:

```
name = "Trà"
age = str(18)
print(name + " năm nay " + age + " tuổi. Hỏi bạn trai của "
    + name + " bao nhiêu tuổi?")
```

- Định dạng chuỗi với f-string

```
name = "Trà"
age = 18
print(f"{name} năm nay {age} tuổi. Hỏi bạn trai của {name}
        bao nhiêu tuổi?")
```

- Định dạng chuỗi với format

```
name = "Trà"
age = 18
print("{0} năm nay {1} tuổi. Hỏi bạn trai của {0} bao nhiêu
        tuổi?".format(name, age))
```

Bài tập 2

Nhập vào 1 số thực, in ra số thực đó với 2 chữ số thập phân

```
print('%.2f' % x)
```

```
x = 1.333355453
format_float = "{:.2f}".format(x)
print(format_float)
```

Tham khảo thêm: *https://docs.python.org/3/library/string.html#format-specification-mini-language*

Kiểu dữ liệu: str

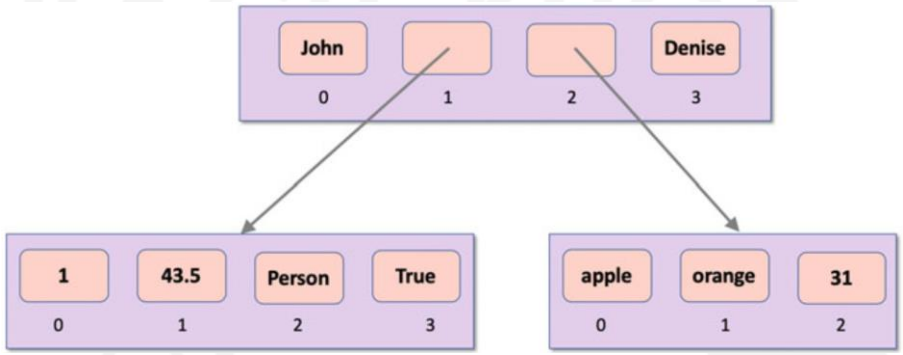
- Một số hàm thường dùng
  - `len()`
  - `.strip()`: loại bỏ tất cả khoảng trắng ở đầu và cuối của chuỗi
  - `.upper()` và `.lower()`
  - `.replace(s1, s2)`
  - `.split()`
  - `.count(s)`: trả về số lần xuất hiện một chuỗi con s trong chuỗi
  - `.find(string_to_find)`
  - `ord(s)` và `chr(s)`

Kiểu dữ liệu

Category	Name	Description
None	None	The null object
Numeric	int	Integer
	float	Floating point number
	complex	Complex number
	bool	Boolean (True, False)
Sequences	str	String of characters
	list	List of arbitrary objects
	tuple	Group of arbitrary items
	range	Creates a range of intergers
Mapping	dict	Dictionary of key-value pairs
	set	Mutable, unordered collection of unique items
	frozenset	Immutable set

Kiểu dữ liệu: list

- Kiểu list
  - Cấu trúc: [phần tử 1, phần tử 2, ...]
  - Ví dụ:
    - [“Táo”, “Lê”, “Dưa hấu”]
    - [1, 5, 7, 9]
    - [“Táo”, 3, “Xe”]
  - Lưu ý:
    - Các phần tử trong list không cần cùng kiểu
    - Có thể lồng list trong list, ví dụ: [[“Cam”, 5], [“Chanh”, 2], 4]



Kiểu dữ liệu: list

- Kiểu list
  - Truy cập phần tử trong list
    - Cách thức: `tên_list[index]`
    - Index được đánh số từ 0,1,2, .. từ trái qua, hoặc đánh số từ -1, -2, .. từ bên phải qua
  - Truy cập list trong list
    - tên_list[a:b]
      - Truy cập từ phần tử thứ a đến b-1
    - tên_list[:b]
      - Truy cập từ phần tử thứ 0 đến b-1
    - tên_list[a:]
      - Truy cập từ phần tử thứ a đến phần tử cuối

Kiểu dữ liệu: list

- Tạo list
  - Từ chuỗi

```
s1 = "Quyen"
print(list(s1))

# ['Q', 'u', 'y', 'e', 'n']
```

  - Biến đổi từ các kiểu dữ liệu iterable khác (tuple, set, dictionary)

```
list(iterable)
```

```
vowelTuple = ('a', 'e', 'i', 'o', 'u')
print(list(vowelTuple))

#Kết quả
['a', 'e', 'i', 'o', 'u']
```

Kiểu dữ liệu: list

- Chỉnh sửa list
  - Thay đổi 1 phần tử trong list
    - tên_list[index] = giá trị mới
    - tên_list[a:b] = [các phần tử mới]
  - Thêm 1 phần tử vào cuối list
    - tên_list.append(1 phần tử mới)
  - Thêm list A vào cuối list B
    - tên_listB += tên_listA
- Xóa phần tử trong list
  - del tên_list[index]
  - del tên_list[a:b]

Kiểu dữ liệu: list

- Copy list thông thường

```
fruits = ["Táo", "Nho", "Lê"]
fruits_copy = fruits
fruits_copy[0] = "Chôm chôm"
print(fruits)           # ['Chôm chôm', 'Nho', 'Lê']
print(fruits_copy)      # ['Chôm chôm', 'Nho', 'Lê']
```

- Copy có nhân bản

```
fruits_copy = fruits[:]
```

Kiểu dữ liệu: list

- 1 vài phương thức trong list

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

List comprehension

- Cú pháp:

```
newList = [ expression(element) for element in oldList if condition ]
```

- Ví dụ:

```
odd_square = [x ** 2 for x in range(1, 11) if x % 2 == 1]
print (odd_square)
# [1, 9, 25, 49, 81]
```

```
odd_square = []

for x in range(1, 11):
    if x % 2 == 1:
        odd_square.append(x**2)

print (odd_square)
```

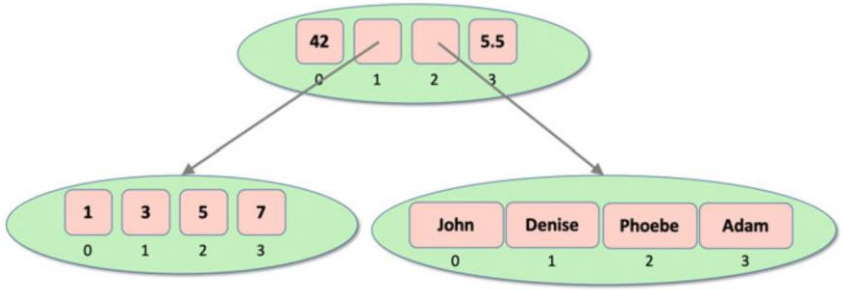
List comprehension

- Ví dụ 2

```
matrix = [[i for i in range(5)] for _ in range(6)]
print(matrix)
#[[0, 1, 2, 3, 4],
  [0, 1, 2, 3, 4],
  [0, 1, 2, 3, 4],
  [0, 1, 2, 3, 4],
  [0, 1, 2, 3, 4],
  [0, 1, 2, 3, 4]]
```

Kiểu dữ liệu: tuple

- Kiểu list
  - Cấu trúc: (phần tử 1, phần tử 2, ...)
  - Ví dụ:
    - (1, 5, 7, 9)
    - ("Táo", 3, "Xe")
  - Truy cập phần tử trong tuple:
    - Tên tuple[index]
  - Lưu ý:
    - Các phần tử trong tuple không cần cùng kiểu
    - Có thể lồng tuple trong tuple



Kiểu dữ liệu: tuple

- Tạo tuple

```
tup1 = (1, 3, 5, 7)
```

- Tạo tuple từ các kiểu dữ liệu iterable khác (Set, List, Dictionary)

```
tuple(iterable)
```

- Ví dụ:

```
list1 = [1, 2, 3]
t1 = tuple(list1)
print(t1)
# (1, 2, 3)
```

Kiểu dữ liệu: tuple

- Ví dụ:

```
tup1 = (1, 'Quyen', Person('Phoebe', 21), True, -23.45)
print(tup1)
```

- In các phần tử của tuple

```
tup2 = ('apple', 'pear', 'orange', 'plum', 'apple')
for x in tup2:
    print(x)
```

- Kiểm tra phần tử nào có nằm trong tuple

```
if 'orange' in tup2:
    print('orange is in the Tuple')
```

Hàm

- Cách thức xây dựng hàm
  - def tên hàm (danh sách tham số):

    <tab> xử lý
- Ví dụ

```
def print_msg():
    print('Hello World!')

print_msg()
```

```
def square(n):
    return n * n
```

```
def swap(a, b):
    return b, a

a = 2
b = 3
x, y = swap(a, b)
print(x, y)

z = swap(a, b)
print(z)
```

Hàm

```
def greeter(name,
            title = 'Dr',
            prompt = 'Welcome',
            message = 'Live Long and Prosper'):
    """
    This function takes defines 4 parameters;
    name, title, prompt and message
    """
    print(prompt, title, name, '-', message)

greeter("Quyen")
greeter(message = 'We like Python', name = 'Lisa')
greeter('Rossie', message = 'We like Python')
```

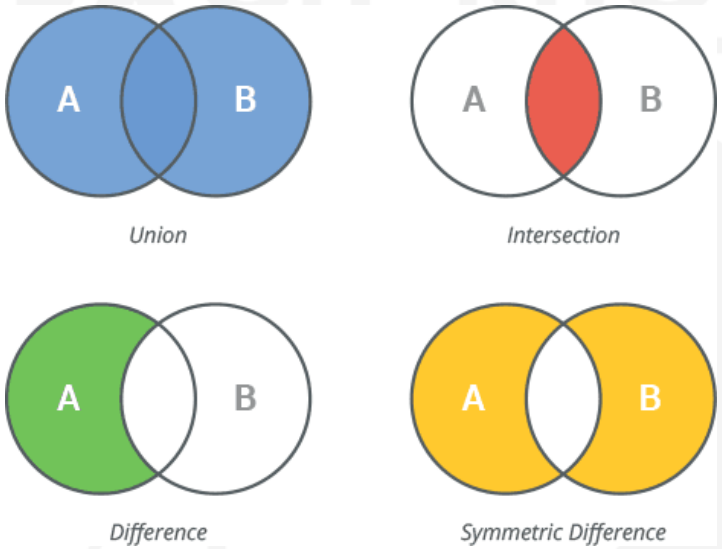
Biến global, local, nonlocal

```
max = 100
def print_max():
    global max
    max = max + 1
    print(max)

print_max()
print(max)
```

```
def outer():
    title = 'original title'
    def inner():
        nonlocal title
        title = 'another title'
        print('inner:', title)
    inner()
    print('outer:', title)

outer()
```

Kiểu dữ liệu: set

- Kiểu set là một cấu trúc dữ liệu để lưu một danh sách các phần tử, trong đó các phần tử không được trùng nhau. Kiểu tập hợp có các phương thức chủ yếu sau:
  - Phép gán
  - Phép hợp (union)
  - Phép giao (intersection)
  - Phép hiệu (difference)
  - Phép hiệu đối xứng (symmetric diference)
  - Phép thử



Kiểu dữ liệu: set

- Lưu ý:
  - Kiểu tập hợp không cho phép các phần tử trùng nhau

```
S = {'red', 'green', 'blue', 'red'}
print(S)
# {'red', 'blue', 'green'}
```

  - Kiểu tập hợp chỉ cho phép các phần tử bất biến

```
S = {1, 'red', ('a', 'b'), True}
print(S)
# {1, ('a', 'b'), 'red'}
```

Kiểu dữ liệu: set

- Biến đổi từ các kiểu iterable khác sang set

```
set(iterable)
```

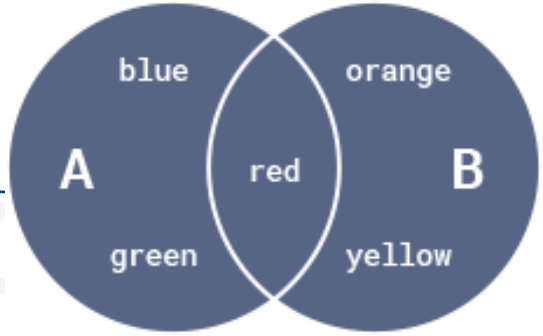
```
S = set('abc')
print(S)
# {'c', 'a', 'b'}

S = set(range(0, 4))
print(S)
# {0, 1, 2, 3}

S = set([1, 2, 3])
print(S)
# {1, 2, 3}
```

Kiểu dữ liệu: set

Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two other sets
intersection_update()	Removes the items in this set that are not present in other, specified set(s)
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not
pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others

Kiểu dữ liệu: set

- Phép hợp
  - Sử dụng phương thức union() hoặc toán tử '|':

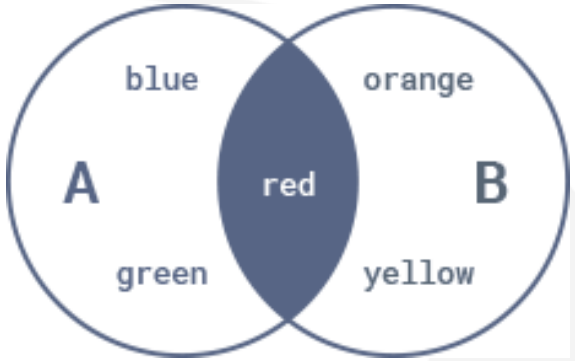



```
A = {'red', 'green', 'blue'}
B = {'yellow', 'red', 'orange'}
# by operator
print(A | B)
# {'green', 'red', 'yellow', 'blue', 'orange'}

# by method
print(A.union(B))
# {'green', 'red', 'yellow', 'blue', 'orange'}
```

Kiểu dữ liệu: set

- Phép giao
  - Sử dụng phương thức intersection() hoặc toán tử '&':

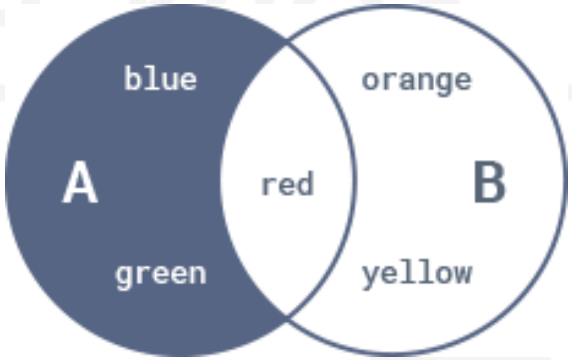
```
A = {'red', 'green', 'blue'}
B = {'yellow', 'red', 'orange'}
# by operator
print(A & B)
# {'red'}

# by method
print(A.intersection(B))
# {'red'}
```



Kiểu dữ liệu: set

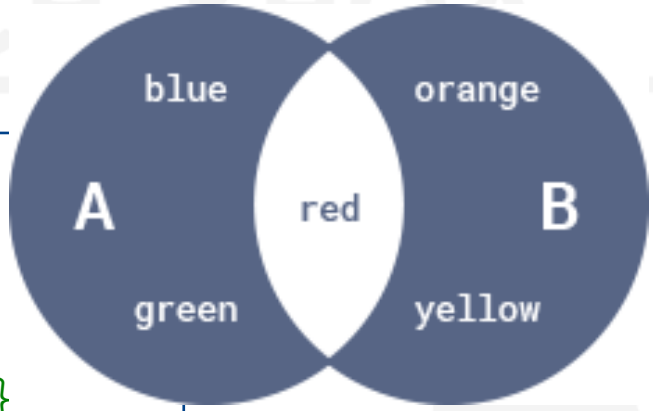
- Phép hiệu
  - Sử dụng phương thức difference() hoặc toán tử '-':

```
A = {'red', 'green', 'blue'}
B = {'yellow', 'red', 'orange'}
# by operator
print(A - B)
# {'blue', 'green'}

# by method
print(A.difference(B))
# {'blue', 'green'}
```



Kiểu dữ liệu: set

- Phép hiệu đối xứng
  - Sử dụng phương thức symmetric_difference() hoặc toán tử '^':

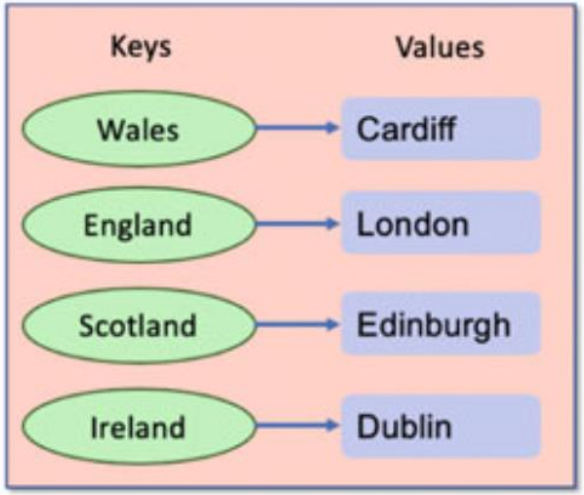



```
A = {'red', 'green', 'blue'}
B = {'yellow', 'red', 'orange'}
# by operator
print(A ^ B)
# {'orange', 'green', 'blue', 'yellow'}

# by method
print(A.symmetric_difference(B))
# {'orange', 'green', 'blue', 'yellow'}
```

Kiểu dữ liệu: Từ điển (Dictionaries)

- Kiểu từ điển trong Python là một cấu trúc dữ liệu dạng bảng băm, giúp cho chúng ta có thể tạo và truy cập phần tử 1 cách nhanh chóng

```
cities = {   'Wales': 'Cardiff',
             'England': 'London',
             'Scotland': 'Edinburgh',
             'Northern Ireland': 'Belfast',
             'Ireland': 'Dublin'}
print(cities)
```



Kiểu dữ liệu: Từ điển

- Cấu trúc:
  - `{key1: value1, key2: value2, ...}`
- Truy cập phần tử trong từ điển
  - **Cách 1:** `Tên từ điển["key"]`
    - Ví dụ:

```
dic = {"Japan": "Tokyo", "Korea": "Seoul"}
print(dic["Japan"]) # Tokyo
```

  - **Cách 2:** `Tên từ điển.get("key")`
    - Nếu không có trong từ điển sẽ trả về None (cách 1 báo lỗi)
    - Nên sử dụng cách này trước khi thay đổi giá trị

Kiểu dữ liệu: Từ điển

- Tạo từ điển

```
# note keys are not strings
dict1 = dict(uk='London', ireland='Dublin', france='Paris')
print('dict1:', dict1)

# key value pairs are tuples
dict2 = dict([('uk', 'London'), ('ireland', 'Dublin'),
              ('france', 'Paris')])
print('dict2:', dict2)

# key value pairs are lists
dict3 = dict((['uk', 'London'], ['ireland', 'Dublin'],
              ['france', 'Paris']))
print('dict3:', dict3)
```

Kiểu dữ liệu: Từ điển

- Thêm hoặc sửa phần tử trong từ điển
  - Tên từ điển[“key”] = giá trị mới
  - Ví dụ:

```
dic ={"Japan": "Osaka", "Korea": "Seoul"}
dic["Vietnam"] = "Hanoi" # Thêm phần tử
print(dic)
# {'Japan': 'Osaka', 'Korea': 'Seoul', 'Vietnam': 'Hanoi'}
```

- Xóa phần tử
  - del Tên từ điển[“key”]
  - Ví dụ:

```
del dic["Korea"]
```

Kiểu dữ liệu: Từ điển

- for cho từ điển
  - Sử dụng phương thức `items()`

```
fruits = {"strawberry": "red", "peach": "pink", "banana": "yellow"}
for fruit, color in fruits.items():
    print(fruit + " is " + color)
```

  - Sử dụng phương thức `keys()`

```
fruits = {"strawberry": "red", "peach": "pink", "banana": "yellow"}
for fruit in fruits.keys(): # fruit là key
    print(fruit + "is" + fruits.get(fruit))
```

Kiểu dữ liệu: Từ điển

- Kiểm tra key có trong từ điển hay không

```
print('Wales' in cities)
print('France' not in cities)
```

Kiểu dữ liệu: Từ điển

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and values
get()	Returns the value of the specified key
items()	Returns a list containing the tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

ThS. Nguyễn Thị Lệ Quyên

HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

Lớp (Classes)

- Trong Python mọi thứ đều là 1 đối tượng
  - VD: số nguyên là ví dụ của lớp `int`, số thực là ví dụ của lớp `float`...

```
print(type(4))                  # <class 'int'>
print(type(5.6))                # <class 'float'>
print(type(True))               # <class 'bool'>
print(type('Quyên'))            # <class 'str'>
print(type([1, 2, 3, 4]))       # <class 'list'>
```

Định nghĩa Class

- 1 lớp được định nghĩa theo định dạng sau

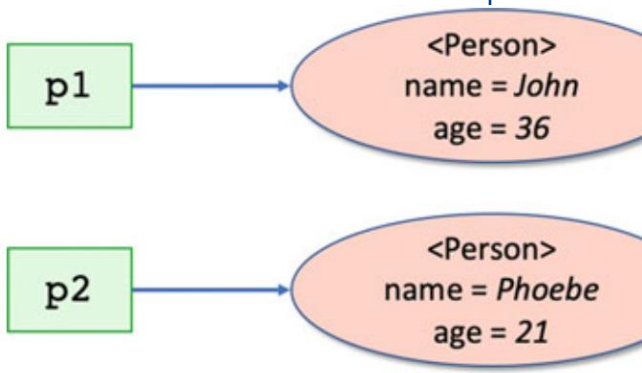
```
class nameOfClass(SuperClass):
    __init__
    attributes
    methods
```

- Ví dụ

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

Tạo đối tượng (instance/object/example) cho lớp Person

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

p1 = Person('John', 36)
p2 = Person('Phoebe', 21)
```



- Mỗi đối tượng p1, p2 có id riêng

```
print('id(p1):', id(p1))    # id(p1): 2547808209936
print('id(p2):', id(p2))    # id(p2): 2547808264544
```

Truy xuất thuộc tính của đối tượng

- Truy xuất thuộc tính hoặc cập nhật giá trị mới cho thuộc tính
  - Tên đối tượng.thuộc tính

```
print(p1.name, 'is', p1.age) # John is 36
p1.name = 'Bob'
print(p1.name, 'is', p1.age) # Bob is 36
```

Xuất đối tượng

- Sử dụng print

```
print(p1) # <__main__.Person object at 0x000001198750C10>
print(p2) # <__main__.Person object at 0x000001198750E160>
```

- Thay đổi chuỗi xuất mặc định

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def __str__(self):
        return self.name + ' is ' + str(self.age)
```

Thêm phương thức, vd1

```
class Person:
    """ An example class to hold a persons name and age"""

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __str__(self):
        return self.name + ' is ' + str(self.age)

    def birthday(self):
        print ('Happy birthday you were', self.age)
        self.age += 1
        print('You are now', self.age)
```

Thêm phương thức, vd2

```
class Person:
    # ...
    def calculate_pay(self, hours_worked):
        rate_of_pay = 7.50
        if self.age >= 21:
            rate_of_pay += 2.50
        return hours_worked * rate_of_pay

p1 = Person('John', 36)
p2 = Person('Adam', 19)

pay = p1.calculate_pay(40)
print('Pay', p1.name, pay)  # Pay John 400.0
pay = p2.calculate_pay(40)
print('Pay', p2.name, pay)  # Pay John 300.0
```

Từ khóa del

- Sử dụng khi muốn xóa đối tượng
  - Cú pháp: del đối_tượng_cần_xóa
- ***Dùng del và gán đối tượng về None khác gì nhau?***

Thuộc tính nội tại (Intrinsic Attibutes)

- Lớp(class)
  - `__name__` the name of the class
  - `__module__` the module (or library) from which it was loaded
  - `__bases__` a collection of its base classes
  - `__dict__` a dictionary (a set of key-value pairs) containing all the attributes (including methods)
  - `__doc__` the documentation string.
- Đối tượng (object)
  - `__class__` the name of the class of the object
  - `__dict__` a dictionary containing all the object's attributes.

```
print('Class attributes')
print(Person.__name__)
print(Person.__module__)
print(Person.__doc__)
print(Person.__dict__)
print('Object attributes')
print(p1.__class__)
print(p1.__dict__)
```

Biến/Thuộc tính của lớp

```
class Person:
    """ An example class to hold a persons name and age"""

    instance_count = 0

    def __init__(self, name, age):
        Person.instance_count += 1
        self.name = name
        self.age = age

p1 = Person('Jason', 36)
p2 = Person('Carol', 21)
p3 = Person('James', 19)
p4 = Person('Tom', 31)
print(Person.instance_count) # 4
```

Phương thức của lớp

```
class Person:
    """ An example class to hold a persons name and age"""

    instance_count = 0

    @classmethod
    def increment_instance_count(cls):
        cls.instance_count += 1

    def __init__(self, name, age):
        Person.increment_instance_count()
        self.name = name
        self.age = age
```

Phương thức tĩnh (static methods)

```
class Person:

    @staticmethod
    def static_function():
        print('Static method')

Person.static_function()
```

Class method vs Static method

```
class Apple:

    _counter = 0

    @staticmethod
    def about_apple():
        print('Apple is good for you.')

        # @staticmethod
        #    print('Number of apples have been juiced: %s' % Apple._counter)
        #
        # @classmethod
        #    print('Number of apples have been juiced: %s' % cls._counter)

    @classmethod
    def make_apple_juice(cls, number_of_apples):
        print('Make juice:')
        for i in range(number_of_apples):
            cls._juice_this(i)

    @classmethod
    def _juice_this(cls, apple):
        print('Juicing %d...' % apple)
        cls._counter += 1
```

Kế thừa

- Cú pháp:

```
class SubClassName(BaseClassName):
    class-body
```

- Ví dụ:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def birthday(self):
        print('Happy birthday you were', self.age)
        self.age += 1
        print('You are now', self.age)

class Employee(Person):
    def __init__(self, name, age, id):
        super().__init__(name, age)
        self.id = id

    def calculate_pay(self, hours_worked):
        rate_of_pay = 7.50
        if self.age >= 21:
            rate_of_pay += 2.50
        return hours_worked * rate_of_pay
```

Override phương thức

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def __str__(self):
        return self.name + ' is ' + str(self.age)

class Employee(Person):
    def __init__(self, name, age, id):
        super().__init__(name, age)
        self.id = id
    def __str__(self):
        return self.name + ' is ' + str(self.age)
                + ' - id(' + str(self.id) + ')'
```

Mở rộng

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def __str__(self):
        return self.name + ' is ' + str(self.age)

class Employee(Person):
    def __init__(self, name, age, id):
        super().__init__(name, age)
        self.id = id
    def __str__(self):
        return super().__str__()
                + ' - id(' + str(self.id) + ')'
```

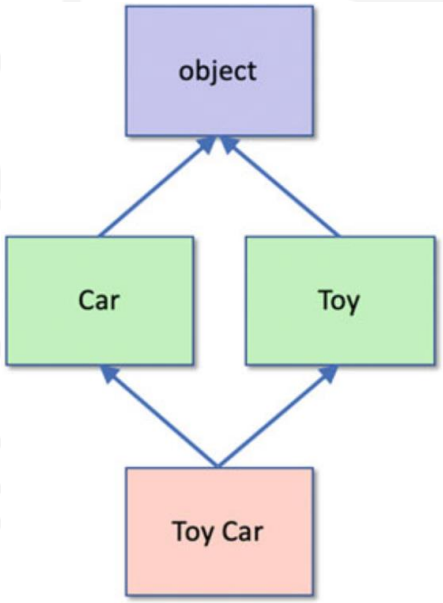
Đa kế thừa

- Cú pháp

```
class SubClassName(BaseClassName1, …, BaseClassNameN):
    class-body
```

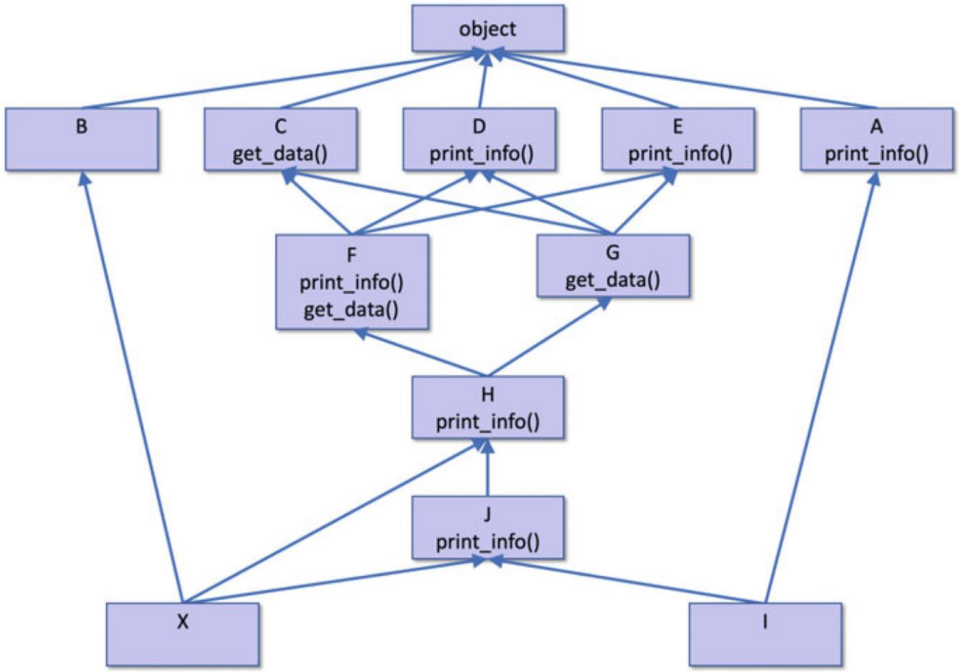
- Ví dụ

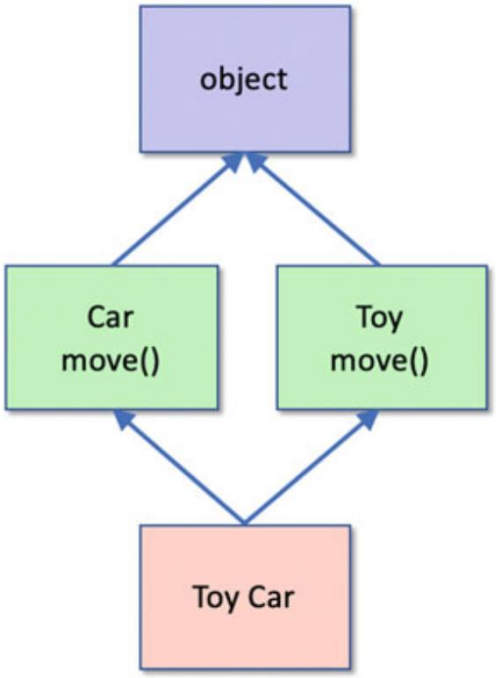
```
class Car:
    """ Car """

class Toy:
    """ Toy """

class ToyCar(Car, Toy):
    """ A Toy Car """
```



Đa kế thừa

- Chọn move() nào?
  - BFS





Ref: https://en.wikipedia.org/wiki/Multiple_inheritance

Overloading

```
class Quantity:
    def __init__(self, value=0):
        self.value = value

    def __add__(self, other):
        new_value = self.value + other.value
        return Quantity(new_value)

    def __sub__(self, other):
        new_value = self.value - other.value
        return Quantity(new_value)

    def __str__(self):
        return 'Quantity[' + str(self.value) + ']'

q1 = Quantity(5)
q2 = Quantity(10)
print('q1 =', q1, ', q2 =', q2) # q1 = Quantity[5] , q2 = Quantity[10]
q3 = q1 + q2
print('q3 =', q3)               # q3 = Quantity[15]
```

Phép toán

Operator	Expression	Method
Addition	`q1 + q2`	`__add__(self, q2)`
Subtraction	`q1 – q2`	`__sub__(self, q2)`
Multiplication	`q1 * q2`	`__mul__(self, q2)`
Power	`q1 ** q2`	`__pow__(self, q2)`
Division	`q1 / q2`	`__truediv__(self, q2)`
Floor Division	`q1 // q2`	`__floordiv__(self, q2)`
Modulo (Remainder)	`q1 % q2`	`__mod__(self, q2)`
Bitwise Left Shift	`q1 ≪ q2`	`__lshift__(self, q2)`
Bitwise Right Shift	`q1 ≫ q2`	`__rshift__(self, q2)`

q1 / 2 ?

```
class Quantity:
    # ...
    def __mul__(self, other):
        if isinstance(other, int):
            new_value = self.value * other
        else:
            new_value = self.value * other.value
        return Quantity(new_value)

    def __truediv__(self, other):
        if isinstance(other, int):
            new_value = self.value / other
        else:
            new_value = self.value / other.value
        return Quantity(new_value)
```

Phép so sánh

Operator	Expression	Method
Less than	`q1 < q2`	`__lt__(q1, q2)`
Less than or equal to	`q1 <= q2`	`__le__(q1, q2)`
Equal to	`q1 == q2`	`__eq__(q1, q2)`
Not Equal to	`q1 != q2`	`__ne__(q1, q2)`
Greater than	`q1 > q2`	`__gt__(q1, q2)`
Greater than or equal to	`q1 >= q2`	`__ge__(q1, q2)`

- Ví dụ

```
class Quantity:
    # ...
    def __eq__(self, other):
        return self.value == other.value
    def __ne__(self, other):
        return self.value != other.value
    # ...
```

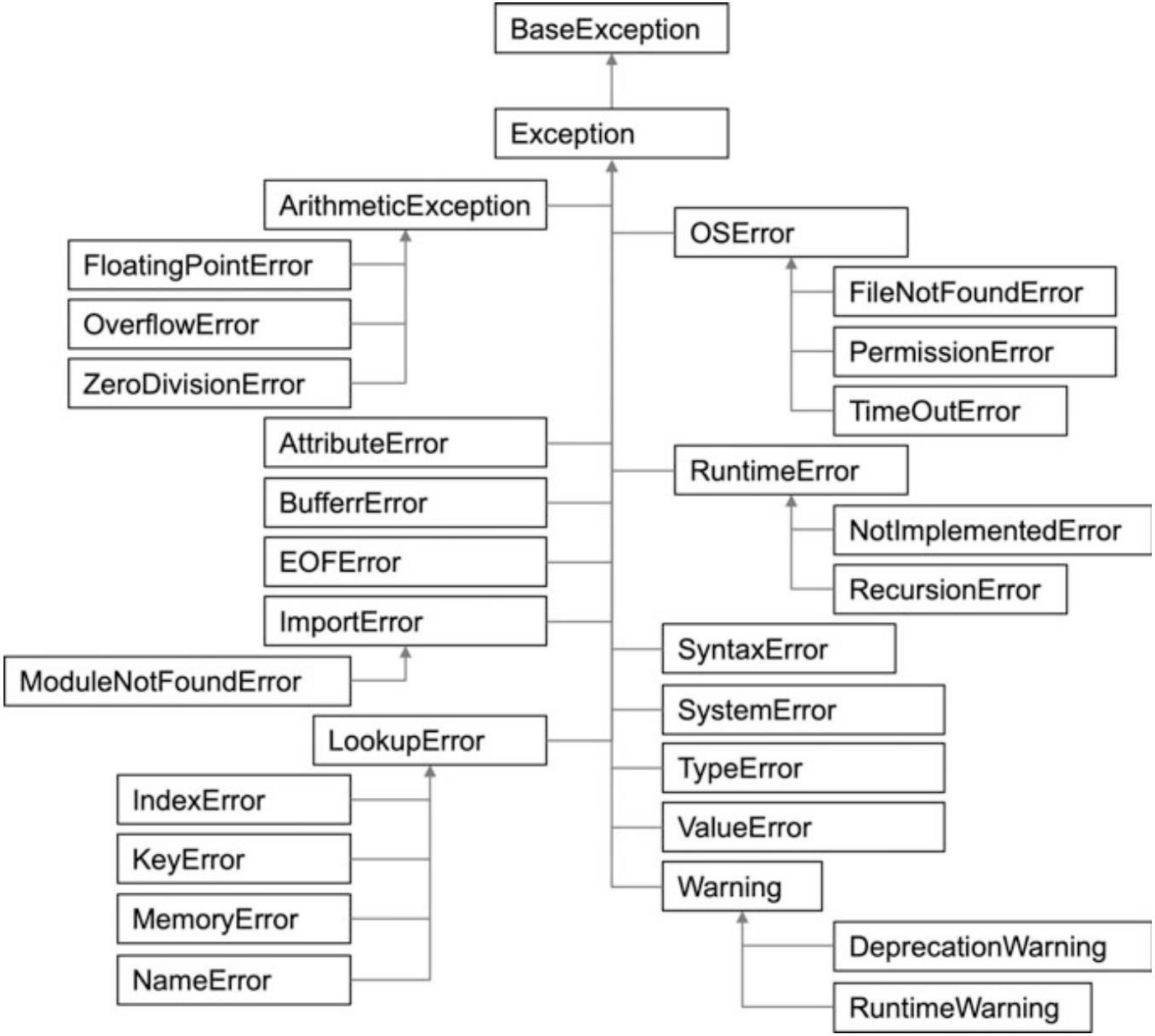
Phép toán logic

Operator	Expression	Method
AND	`q1 & q2`	`__and__(q1, q2)`
OR	`q1 | q2`	`__or__(q1, q2)`
XOR	`q1 ^ q2`	`__xor__(q1, q2)`
NOT	`~q1`	`__invert__()`

Nội dung

1. Tổng quan và hướng dẫn cài đặt
2. Cơ bản Python
   - Lệnh nhập/ xuất
   - Kiểu dữ liệu: Chuỗi, List, Tuple, Set, Dict
   - Điều kiện, vòng lặp
   - Hàm
3. Hướng đối tượng Python
4. Exception
5. Regular Expressions
6. Bài tập

ERROR AND EXCEPTION HANDLING


BaseException
Exception
ArithmeticException
FloatingPointError
OverflowError
ZeroDivisionError
AttributeError
BufferrError
EOFError
ImportError
ModuleNotFoundError
LookupError
IndexError
KeyError
MemoryError
NameError
OSError
FileNotFoundError
PermissionError
TimeOutError
RuntimeError
NotImplementedError
RecursionError
SyntaxError
SystemError
TypeError
ValueError
Warning
DeprecationWarning
RuntimeWarning

Lỗi và ngoại lệ

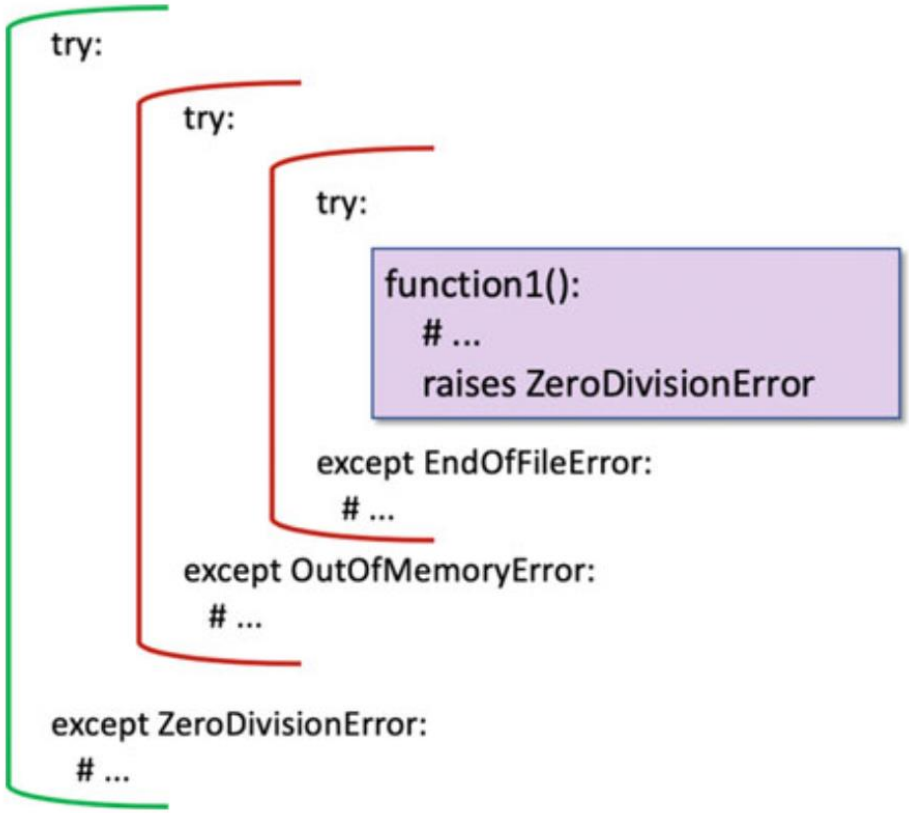
- Cú pháp

```
try:
    <code to monitor>
except <type of exception to monitor for>:
    <code to call if exception is found>
```

- Ví dụ:

```
def runcalc(x):
    x / 0

try:
    runcalc(6)
except ZeroDivisionError:
    print('oops')
```

Lỗi và ngoại lệ



Lỗi và ngoại lệ

- Bắt mọi trường hợp ngoại lệ

```
try:
    my_function(6, 0)
except ZeroDivisionError:
    print('oops')
except IndexError as e:
    print(e)
except:
    print('Something went wrong')
```

Lỗi và ngoại lệ

- Else và finally trong try except

```
try:
    my_function(6, 2)
except ZeroDivisionError as e:
    print(e)
else:
    print('Everything worked OK')
finally:
    print('Always runs')
```

Ném ngoại lệ

- Cú pháp:

```
raise <Exception/Error type to raise>()
```

- Ví dụ:

```
def function_bang():
    print('function_bang in')
    raise ValueError('Bang!')
    print('function_bang')

try:
    function_bang()          # function_bang in
except ValueError as ve:
    print(ve)                # Bang!
```

Ném ngoại lệ

- Có thể ném ngoại lệ không cần tham số

```
raise ValueError    # short hand for raise ValueError()
```

- Có thể ném lại ngoại lệ để xử lý sau đó

```
try:
    function_bang()
except ValueError:
    print('oops')
raise
```

Định nghĩa 1 ngoại lệ tùy chỉnh (custom exception)

- Ví dụ:

```
class InvalidAgeException(Exception):
    """ Valid Ages must be between 0 and 120 """

    def __init__(self, value):
        self.value = value

    def __str__(self):
        return 'InvalidAgeException(' + str(self.value) + ')'
```

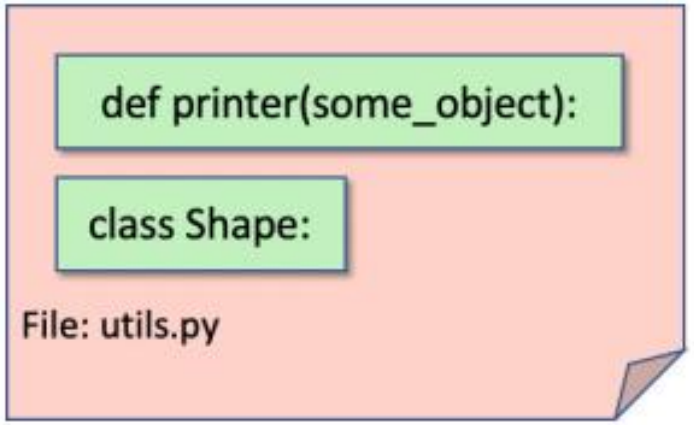
```
raise InvalidAgeException(value)
```

PYTHON MODULES

VÀ PACKAGES

Python modules

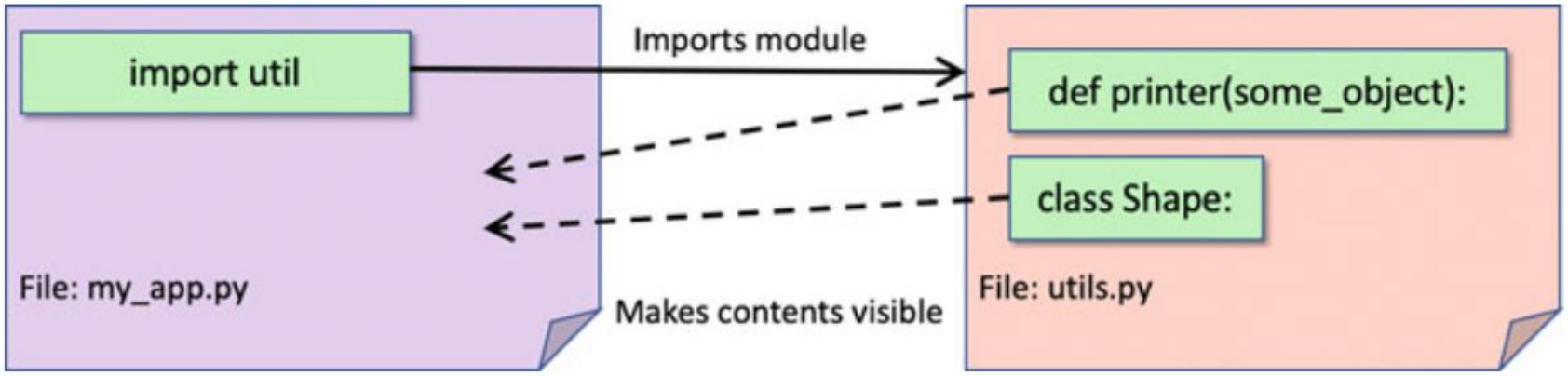
- Trong Python, 1 module tương đương với 1 tệp chứa mã code Python
- 1 module có thể chứa:
  - Hàm
  - Lớp
  - Biến
  - Mã thực thi
  - Các thuộc tính liên kết với module như tên, ...
- Tên module là tên file (không có hậu tố .py)




```
import utils
```

Import module

```
import utils

utils.printer(utils.default_shape)
shape = utils.Shape('circle')
utils.printer(shape)
```



Import từ 1 module

- Cú pháp:

```
from <module name> import *
```

- Ý nghĩa: Từ module <name> import tất cả mọi thứ vào và sử dụng trực tiếp

```
import utils

utils.printer(utils.default_shape)
shape = utils.Shape('circle')
utils.printer(shape)
```

```
from utils import *

printer(default_shape)
shape = Shape('circle')
printer(shape)
```

- Muốn import 1 phần

```
from utils import Shape

s = Shape('rectangle')
print(s)
```

Import từ 1 module

- Có thể đặt tên khác (alias) cho module import vào
  - Cú pháp:

```
import <module_name> as <alternative_module_name>
from <module_name> import <element> as <alias>
```

  - Ví dụ

```
import utils as utilities
utilities.printer(utilities.default_shape)
```

```
from utils import Shape, printer as myfunc
s = Shape('oval')
myfunc(s)
```

Python modules

- Có thể import module trong một hàm như là 1 module cục bộ, chỉ sử dụng trong hàm

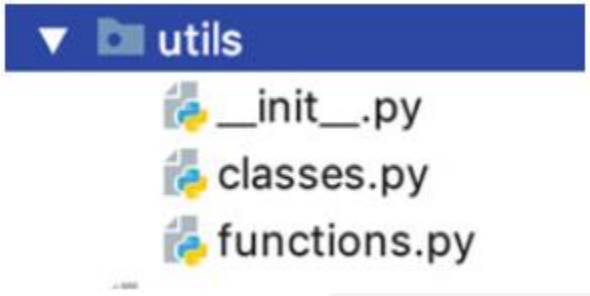
```
def my_func():
    from utils import Shape

    s = Shape('line')
```

- Tham khảo thêm các module chuẩn của Python
  - https://docs.python.org/3/library/

Python packages

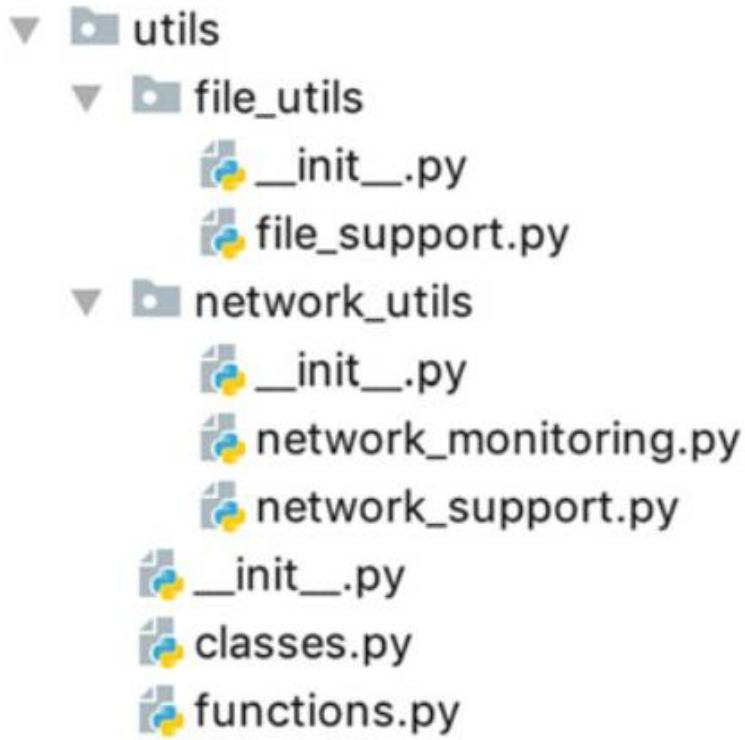
- Python cho phép tổ chức các module lại với nhau trong 1 package có cấu trúc phân tầng dựa vào đường dẫn
- 1 package được định nghĩa gồm:
  - Đường dẫn chứa 1 hoặc nhiều file mã nguồn Python
  - (Optional) file __init__.py, file này chứa code sẽ được thực thi khi module được import từ package

```
from utils.functions import *
from utils.classes import *
```



Python subpackage

```
import utils.file_utils.file_support
from utils.file_utils.file_support import file_logger
```



ThS. Nguyễn Thị Lệ Quyên

REGULAR EXPRESSIONS IN PYTHON

Regex (biểu thức chính quy)

- Regular expression (còn được biết đến là regex hoặc re) là đoạn các ký tự (chữ cái, chữ số và ký tự đặc biệt) dùng để so khớp các chuỗi hoặc một tập các chuỗi
- Sử dụng module `re`
- Regex được sử dụng rộng rãi để tìm kiếm thông tin trong file, ví dụ:
  - Tìm tất cả các dòng trong file log tương ứng với người dùng cụ thể hoặc hoạt động cụ thể nào đó
  - Xác thực đầu vào, chẳng hạn kiểm tra xem một chuỗi có phải là địa chỉ email hợp lệ hoặc mã bưu điện hợp lệ, ...

Mẫu

Character	Description	Example
[]	A set of characters	[a-d] characters in the sequence ‘a’ to ‘d’
\	Indicates a special sequence (can also be used to escape special characters)	‘\d’ indicates the character should be an integer
.	Any character with the exception of the newline character	‘J.hn’ indicates that there can be any character after the ‘J’ and before the ‘h’
^	Indicates a string must start with the following pattern	“^hello” indicates the string must start with ‘hello’
$	Indicates a string must end with the preceding pattern	“world$” indicates the string must end with ‘world’
*	Zero or more occurrences of the preceding pattern	“Python*” indicates we are looking for zero or more times Python is in a string
+	One or more occurrences of preceding pattern	“info+” indicates that we must find info in the string at least once
?	Indicates zero or 1 occurrences of the preceding pattern	“john?” indicates zero or one instances of the ‘John’
{}	Exactly the specified number of occurrences	“John{3}” this indicates we expect to see the ‘John’ in the string three times. “X{1,2}” indicates that there can be one or two Xs next to each other in the string
|	Either or	“True|OK” indicates we are looking for either True or OK
()	Groups together a regular expression; you can then apply another operator to the whole group	“(abc|xyz){2}” indicates that we are looking for the string abc or xyz repeated twice

Sequence	Description	Example
\A	Returns a match if the following characters are at the beginning of the string	"\AThe" must start with 'The'
\b	Returns a match where the specified characters are at the beginning or at the end of a word	"\bon" or "on\b" indicates a string must start or end with 'on'
\B	Indicates that the following characters must be present in a string but not at the start (or at the end) of a word	r"\Bon" or r"on\B" must not start or end with 'on'
\d	Returns a match where the string contains digits (numbers from 0–9)	"\d"
\D	Returns a match where the string DOES NOT contain digits	"\D"
\s	Returns a match where the string contains a white space character	"\s ~ "
\S	Returns a match where the string DOES NOT contain a white space character	"\S"
\w	Returns a match where the string contains any word characters (characters from a to Z, digits from 0–9, and the underscore _ character)	"\w"
\W	Returns a match where the string DOES NOT contain any word characters	"\W"
\Z	Returns a match if the following characters are present at the end of the string	"Hunt\Z"

Set

Set	Description
[jeh]	Returns a match where one of the specified characters (j, e or h) are present
[a–x]	Returns a match for any lower-case character, alphabetically between a and x
[^zxc]	Returns a match for any character EXCEPT z, x and c
[0123]	Returns a match where any of the specified digits (0, 1, 2, or 3) are present
[0–9]	Returns a match for any digit between 0 and 9
[0–9][0–9]	Returns a match for any two-digit numbers from 00 and 99
[a–zA–Z]	Returns a match for any character alphabetically between a and z or A and Z

Lookaround

- Positive lookahead `(?=ABC)`
- Negative lookahead `(?!ABC)`
- Positive lookbehind `(?<=ABC)`
- Negative lookbehind `(?<!ABC)`

```
import re

pattern = "\d+(?=px)"
str = "1pt 12px 13pt 1555px px"

results = re.findall(pattern, str)
print(results)      # ['12', '1555']
```

Ref: https://regexr.com/

Python Regex

- Sử dụng chuỗi thô (raw string)

```
s = r'Hello \n world'
print(s)    # Hello \n world
```

- Ví dụ:

```
import re

text1 = 'john williams'
pattern = r'[Jj]ohn'

if re.search(pattern, text1):
    print('Match has been found')
```

Đối tượng Match

- Được trả về bởi hàm search() hoặc match()
  - Kết quả là None nếu không tìm thấy (no match)
  - Kết quả luôn có giá trị bool là True nếu tìm thấy
- Có các phương thức và thuộc tính sau:
  - `match.re`: đối tượng regex được trả về bởi phương thức match() hoặc search()
  - `match.string`: Chuỗi truyền vào hàm match() hoặc search()
  - `match.start([group]) / match.end([group])`: Trả về chỉ số phần bắt đầu và kết thúc của chuỗi con được so khớp theo group
  - `match.group()`: trả về phần của chuỗi mà tại đó khớp

Hàm search()

- Hàm search() tìm kiếm trong một chuỗi để tìm kết quả phù hợp và trả về một đối tượng Match nếu có kết quả phù hợp
- Cú pháp:

```
re.search(pattern, string, flags=0)
```

Flags

Flag	Description
`re.IGNORECASE`	Performs case-insensitive matching
`re.LOCALE`	Interprets words according to the current locale. This interpretation affects the alphabetic group (\w and \W), as well as word boundary behavior(\b and \B)
`re.MULTILINE`	Makes $ match the end of a line (not just the end of the string) and makes ^ match the start of any line (not just the start of the string)
`re.DOTALL`	Makes a period (dot) match any character, including a newline
`re.UNICODE`	Interprets letters according to the Unicode character set. This flag affects the behavior of \w, \W, \b, \B
`re.VERBOSE`	Ignores whitespace within the pattern (except inside a set `[]` or when escaped by a backslash) and treats unescaped # as a comment marker

Ví dụ 1

```
import re

line1 = 'The price is 23.55'
containsIntegers = r'\d+'

if re.search(containsIntegers, line1):
    print('Line 1 contains an integer')
else:
    print('Line 1 does not contain an integer’)

# Line 1 contains an integer
```

Ví dụ 2

```
import re

music = r'Beatles|Adele|Gorillaz'
request = 'Play some Adele'

if re.search(music, request):
    print('Set Fire to the Rain')
else:
    print('No Adele Available')

# Set Fire to the Rain
```

Hàm match()

- Hàm so khớp một mẫu biểu thức chính quy với đầu một chuỗi và và trả về một đối tượng Match nếu có kết quả phù hợp
- Cú pháp:

```
re.match(pattern, string, flags=0)
```

- match() vs search()
  - match() chỉ so khớp với đầu một chuỗi
  - search() kiểm tra trùng khớp ở bất kỳ đâu trong chuỗi

Phương thức fullmatch()

- Trả về đối tượng match nếu toàn bộ chuỗi khớp với mẫu biểu thức chính quy, ngược lại sẽ trả về None
- Cú pháp:

```
re.fullmatch(pattern, string, flags=0)
```

- Ví dụ:

```
import re

string = "102190123"
pattern = "^102\d{6}"

print(re.fullmatch(pattern, string))
# <re.Match object; span=(0, 9), match='102190123'>
```

match() vs fullmatch()

```
import re

string = "102190123"
pattern = "102"

print(re.match(pattern, string))
# <re.Match object; span=(0, 3), match='102'>

print(re.fullmatch(pattern, string))
# None
```

Hàm findall()

- Hàm trả về một danh sách chứa tất cả trùng khớp
- Cú pháp:

```
re.findall(pattern, string, flags=0)
```

- Ví dụ:

```
import re

str = 'The rain in Spain stays mainly on the plain'
results = re.findall('[a-zA-Z]{2}ai.', str)

print(results)      # ['Spain', 'plain']
for s in results:
    print(s)        # Spain
                    # plain
```

Hàm finditer()

```
import re

str = 'The rain in Spain stays mainly on the plain'
results = re.finditer('[a-zA-Z]{2}ai.', str)

print(results)
# <callable_iterator object at 0x0000011788DD6FD0>

for s in results:
    print(s)

# <re.Match object; span=(12, 17), match='Spain'>
# <re.Match object; span=(38, 43), match='plain'>
```

Hàm split()

- Cú pháp:

```
re.split(pattern, string, maxsplit=0, flags=0)
```

- Ví dụ:

```
import re
str = 'It was a hot summer night'
x = re.split('\s', str)
print(x)
# ['It', 'was', 'a', 'hot', 'summer', 'night']
```

Hàm sub()

- Hàm thay thế số lần xuất hiện của mẫu biểu thức chính quy trong chuỗi ban đầu bằng chuỗi repl
- Cú pháp:

```
re.sub(pattern, repl, string, max=0)
```

- Ví dụ:

```
pattern = '(England|Wales|Scotland)'
input = 'England for football, Wales for Rugby and Scotla
nd for the Highland games'

print(re.sub(pattern, 'England', input ))
# England for football, England for Rugby and England for
 the Highland games
```

Hàm sub()

- Ví dụ 2

```
pattern = '(England|Wales|Scotland)'
input = 'England for football, Wales for Rugby and Scotl
and for the Highland games'

x = re.sub(pattern, 'Wales', input, 2)
print(x)
# Wales for football, Wales for Rugby and Scotland for t
he Highland games
```

- Có thể đếm có bao nhiêu lần thay thế bằng cách sử dụng hàm subn():

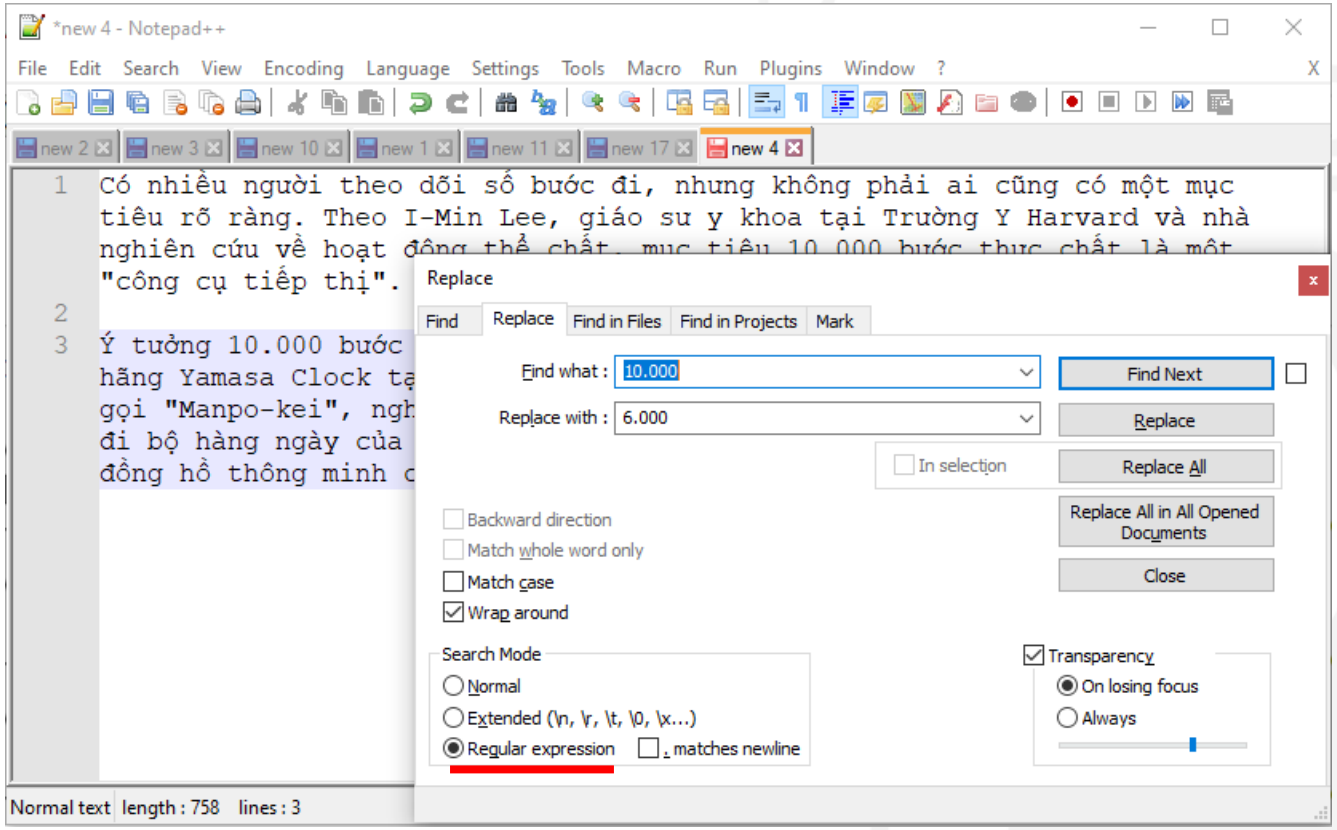
```
print(re.subn(pattern,'Scotland', input ))
```

Hàm compile()

```
import re

line1 = 'The price is 23.55'
containsIntegers = r'\d+'
rePattern = re.compile(containsIntegers)

matchLine1 = rePattern.search(line1)

if matchLine1:
    print('Line 1 contains a number')
else:
    print('Line 1 does not contain a number')
```

Ví dụ 1:

- Tìm kiếm và thay thế bằng một chuỗi khác trên 1 Text Editor



Ví dụ 2:

- Tìm kiếm files trong hệ điều hành
- Tìm kiếm text trong files

```
dir * /s/b | findstr \.r[0-9]+$

abc.txt.r12222
tjy.java.r9994
```

Bash

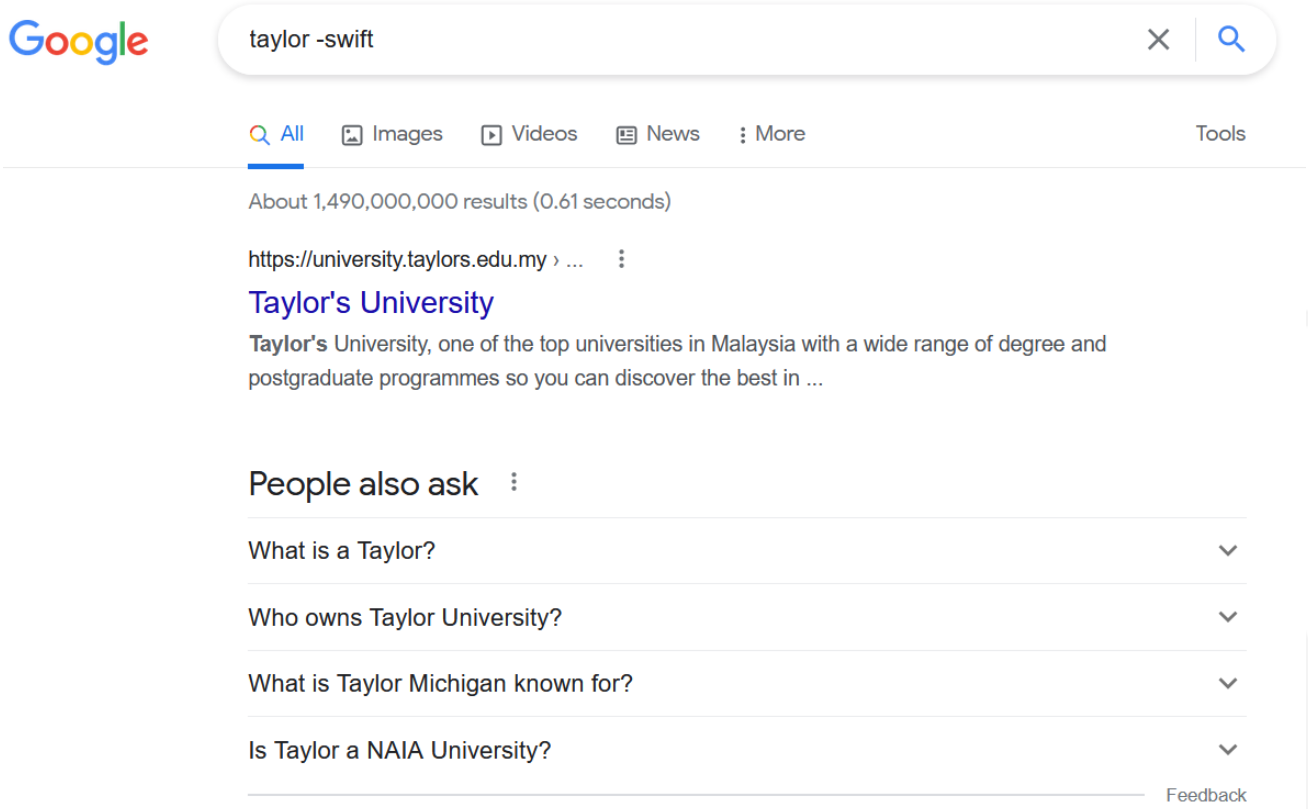
```
$ grep not haiku.txt
```

Output

```
Is not the true Tao, until
"My Thesis" not found
Today it is not working
```

Ví dụ 3:

- Search engines



Ref: http://www.googleguide.com/minus_operator.html

Ví dụ 4:

- Tìm/ Lấy email hoặc thông tin liên hệ khách hàng từ một email hoặc từ một văn bản nào đó.

```
re.findall(r"[\w.-]+@[\w.-]+", text)
```

Ví dụ 5:

- Parser (phân tích cú pháp)
- Syntax highlighting systems
- Lexical Analysis in a Compiler

```
import re

code = '''
def f1():
    return 1
def f2()
    return 2
'''

print(re.findall('def ([a-zA-Z0-9_]+)', code))
# ['f1', 'f2']
```

Ví dụ 6:

- Web scraping (Thu thập dữ liệu)

Ví dụ 7:

- Sử dụng regex để tiền xử lý (NLP – xử lý ngôn ngữ tự nhiên)
  - Loại bỏ các kí tự đặc biệt

Unicode emotions	☺, ♥...
Symbol icon	☏, ✉...
Currency symbol	€, £, $...

  - Loại bỏ stop words