# NGÔN NGỮ LẬP TRÌNH

python

Đỗ Thanh Nghị
dtnghi@cit.ctu.edu.vn

12-2019

# Nội dung



- Giới thiệu
- Lập trình căn bản
- Các kiểu dữ liệu phức
- Lập trình hướng đối tượng
- Vào/ra, ngoại lệ

# Nội dung



- **Giới thiệu**
- Lập trình căn bản
- Các kiểu dữ liệu phức
- Lập trình hướng đối tượng
- Vào/ra, ngoại lệ

# Giới thiệu Python



- Python là ngôn ngữ lập trình cấp cao, tương tác, thông dịch, hướng đối tượng
- Do Guido van Rossum đề xuất từ 1985 – 1990
- Mã nguồn Python: giấy phép GNU General Public License (GPL)
- Python sẵn dùng trên Unix, Linux, Mac OS, Windows
- Top 5 ngôn ngữ lập trình phổ biến
- Được sử dụng bởi các tổ chức tập đoàn như Wikipedia, Google, Yahoo!, CERN

# Giới thiệu Python



- Dễ học (Easy-to-learn)
- Dễ đọc (Easy-to-read)
- Dễ bảo trì (Easy-to-maintain)
- Thư viện chuẩn (standard library): tương thích UNIX, Linux, Windows và Mac
- Chế độ tương tác: thực thi, kiểm tra, gỡ rối
- Khả chuyển (Portable): phần cứng, hệ điều hành
- Khả năng mở rộng (Extendable): thêm mô-đun
- Kết nối với các hệ quản trị CSDL
- Lập trình giao diện đồ họa GUI
- Quy mô chương trình (Scalable)

# Giới thiệu Python



- Python được sử dụng trong giảng dạy,
- Tính toán khoa học,
- Công nghệ sinh-tin học,
- Phát triển ứng dụng Web,
- Lập trình ứng dụng mạng, nghiên cứu an ninh mạng,
- Kỹ thuật đồ họa, xử lý ảnh và thị giác máy tính,
- Máy học và khai thác dữ liệu, xử lý ngôn ngữ tự nhiên, trí tuệ nhân tạo
- Lập trình nhúng,
- Quản trị hệ thống,
- Lập trình trò chơi, v.v.


3
Python

# Nội dung



- Giới thiệu
- **Lập trình căn bản**
- Các kiểu dữ liệu phức
- Lập trình hướng đối tượng
- Vào/ra, ngoại lệ

# Trình thông dịch Python



```
[nghi@localhost python3]$ python3
Python 3.8.10 (default, May  4 2021, 00:00:00)
[GCC 10.2.1 20201125 (Red Hat 10.2.1-9)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello World!')
Hello World!
```

# Thực thi chương trình Python



- Soạn thảo chương trình: PyCharm, Geany, notepad++, Nano, Gedit, v.v.

- Chương trình **hello.py**

```
#!/usr/bin/python3
print('Hello World!')
```

- Thực thi **hello.py**

```
[nghi@localhost python3]$ python3 hello.py
Hello World!
```

# Cú pháp



- Phân biệt ký tự thường hoa
- Các từ khóa của Python được

```
and       exec      not
as        finally   or
assert    for       pass
break     from      print
class     global          raise
continue  if        return
def       import          try
del       in        while
elif      is        with
else      lambda    yield
except
```

# Cú pháp



- Sử dụng # để chú thích 1 dòng trong chương trình

```
# comment
```

- Sử dụng ''' để chú thích 1 đoạn

```
'''
print("We are in a comment")
print('We are still in a comment')
'''
```

- Dấu ' và "

```
word = 'word'
sentence = "This is a sentence."
message = """This message will
... span several lines."""
```

# Cú pháp



- Sử dụng canh lề (bắt buộc) để bao các khối lệnh của hàm, lớp hoặc luồng điều khiển
- Số khoảng trắng dùng để canh lề có thể nhiều ít tuỳ ý nhưng tất cả lệnh trong một khối phải được canh lề như nhau

```
if True:
    print("Answer")
    print("True")
else:
    print("False")
```

# Cú pháp



- Lệnh được viết trên nhiều dòng sử dụng ký tự ****

```
total = item_one + \
        item_two + \
        item_three
```

- Lệnh được bao bằng các cặp dấu ngoặc: **[]**, **{}**, **()** không cần phải sử dụng ký tự **** để tiếp tục dòng

```
days = ['Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday']
```

- Dấu **;** để cách nhiều lệnh trên dòng

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```

# Cú pháp



- Nhóm nhiều câu lệnh đơn tạo nên một khối lệnh và cũng được gọi là bộ (suites)
- Các lệnh phức như if, while, def và class cần một dòng header và một bộ
- Dòng header bắt đầu câu lệnh (bằng một từ khoá tương ứng ví dụ như if, def, …) và kết thúc bằng dấu hai chấm **:** theo sau là một suite

```
def hi(name):
    print('Hello ' + name)
    print('Have a good day!')

hi('nghi')
```

# Lệnh **print** trong Python



```
>>> print('hello')
hello
>>> print('hi','there')
hi there
>>> a = 7.0
>>> b = 2
>>> name = 'toto'
>>> print('%f/%d = %f' %(a, b, a/b))
7.000000/2 = 3.500000
>>> print('Hi %s' %name)
Hi toto
```

# Lệnh **input** trong Python



```
>>> name = input("Please enter your name: ")
Please enter your name: toto
>>> print('Hi %s' %name)
Hi toto
>>> a = int(input("a = "))
a = 5
>>> b = float(input("b = "))
b = 7.2
>>> a + b
12.2
```

# Biến, kiểu cơ bản, phép toán



- Tên: ký tự bắt đầu phải là **alphabet** hoặc **_**
- Không cần khai báo, chỉ gán giá trị (sử dụng dấu **=**)
- Được tạo ra trong lần đầu gán giá trị
- Phạm vi biến: cục bộ, toàn cục
- Tham khảo đến đối tượng
- Thông tin về kiểu gắn liền với đối tượng
- Kiểu cơ bản: int, float, complex, bool, string
- Các phép toán số học: +, -, *, /, %, **
- Phép toán so sánh: ==, !=, >, >=, <, <=
- Phép toán luận lý: and, or, not

# Biến, kiểu cơ bản, phép toán



```
>>> item_name = 'Computer'
>>> item_qty = 10
>>> item_value = 1000.23
>>> print(item_name, item_qty, item_value)
Computer 10 1000.23
>>> x = y = z = 1
>>> print(x, y, z)
1 1 1
>>> x,y,z = 1,2,'abcd'
>>> print(x, y, z)
1 2 abcd
>>> x, y = y, x
>>> print(x, y)
2 1
```

# Biến, kiểu cơ bản, phép toán



```
>>> var1 = "Python"
>>> def func1():
...     var1 = "PHP"
...     print("In side func1() var1 = ",var1)
...
>>> def func2():
...     global var1
...     print("In side func2() global var1 = ",var1)
...
>>> func1()
In side func1() var1 =  PHP
>>> func2()
In side func2() global var1 =  Python
```

# Biến, kiểu cơ bản, phép toán



```
>>> x = 7.2
>>> x
7.2
>>> type(x)
<type 'float'>
>>> x = 'hello'
>>> x
'hello'
>>> type(x)
<type 'str'>
>>> x = 123456789
>>> x ** 2
15241578750190521
>>> z = 3+2j
>>> t = 2-1j
>>> z+t
(5+1j)
```

# Biến, kiểu cơ bản, phép toán



```
>>> x = 7
>>> y = 2
>>> x%y
1
>>> x/y
3.5
>>> 1.0*x/y
3.5
>>> float(x)/y
3.5
>>> a = 7.2
>>> int(a)%y
1
>>> b = True
>>> type(b)
<class 'bool'>
```

# Biến, kiểu cơ bản, phép toán



```
>>> s = 'Hello World!'
>>> type(s)
<class 'str'>
>>> len(s)
12
>>> print(s)
Hello World!
>>> print(s[0])
H
>>> print(s[-1])
!
>>> print(s[2:5])
llo
>>> print(s[2:])
llo World!
>>> print(s * 2)
Hello World!Hello World!
>>> name = 'Ben ' + str(10)
>>> print(name)
Ben 10
```

# Cấu trúc điều khiển



- Cấu trúc rẽ nhánh **if**

```
if (cond1):
   ...
elif (cond2):
   ...
else:
   ...
```

```
#eq: ax + b = 0
a = float(input('a = '))
b = float(input('b = '))
if (a==0):
    if (b==0):
        print('Pt vo dinh')
    else:
        print('Pt vo nghiem')
else:
    x = -b/a
    print('x = ', x)
```

# Cấu trúc điều khiển



- Cấu trúc lặp **while**

```
while (cond):
    ...
    loop_body
    ...
```

```
#s = 1 + 2 + ... + n
n = int(input('n = '))
s = 0
i = 1
while (i<=n):
    s = s + i
    i = i + 1

print('s = 1 + 2 + ... + %d = %d' %(n, s))
```

# Cấu trúc điều khiển



- Cấu trúc lặp **for**

```
for iter_var in seqence:
    ...
    loop_body
    ...
```

```
#s = 1 + 2 + ... + n
n = int(input('n = '))
s = 0
for i in range(1, n+1):
    s = s + i

print('s = 1 + 2 + ... + %d = %d' %(n, s))
```

# Hàm



- Hàm xây dựng sẵn trong các mô-đun

```
>>> import os
>>> import random as rand
>>> from math import sqrt, cos, sin
>>> os.system('ls -l')
total 8
-rw-rw-r-- 1 nghi nghi 41 Apr 18 09:01 hello.py
-rw-rw-r-- 1 nghi nghi  5 Apr 18 10:05 toto.txt
0
>>> r = rand.random()
>>> print(r)
0.3512297233213161
>>> x = 64
>>> sqrt(x)
8.0
```

# Hàm



- Hàm do lập trình viên định nghĩa

```
def function_name ([parameters]):
    ...
    body_of_the_function
    ...
    return ...
```

- Tham số tùy chọn, có thể đặt giá trị mặc định
- Hàm có thể trả hoặc không trả về kết quả
- Đệ quy

# Hàm



```
>>> def hi():
...     print('Hi there!')
...
>>> hi()
Hi there!
>>>
>>> def invite(name = 'Toto'):
...     print('Dear', name)
...     print('It will be a great pleasure')
...     print('if you attend the birthday party of Tutu')
...
>>> invite()
Dear Toto
It will be a great pleasure
if you attend the birthday party of Tutu
>>> invite('Philippe')
Dear Philippe
It will be a great pleasure
if you attend the birthday party of Tutu
```

# Hàm



```
>>> def addTwo(a, b):
...     return a + b
...
>>> def divide(a, b):
...     return a/b, a%b
...
>>> x = 2
>>> y = 3
>>> z = addTwo(x, y)
>>> print(z)
5
>>> p, q = divide(x, y)
>>> print(p, q)
0.6666666666666666 2
```

# Hàm



```
>>> def gcd(m, n):
...     while (m != n):
...             if (m > n):
...                     m = m - n
...             else:
...                     n = n - m
...     return m
...
>>> def coPrime(a, b):
...     if (gcd(a, b) != 1):
...             return
...     else:
...             print("%d and %d are co-prime" %(a,b))
...
>>> x, y = 2, 5
>>> coPrime(x, y)
2 and 5 are co-prime
```

# Hàm



```
>>> def fact(n):
...     if (n == 1):
...             return 1
...     else:
...             return n*fact(n-1)
...
>>> n = 5
>>> print('%d! = %d' %(n,fact(n)))
5! = 120
```

# Nội dung



- Giới thiệu
- Lập trình căn bản
- **Các kiểu dữ liệu phức**
- Lập trình hướng đối tượng
- Vào/ra, ngoại lệ

# Kiểu String



```
>>> a = 'Hello world!'
>>> b = "Hello world!"
>>> a == b
True
>>> a = "Khang's lecture"
>>> print(a)
Khang's lecture
>>> a = "One line.\nAnother line."
>>> print(a)
One line.
Another line.
>>> b = """One line,
... another line,
... and some..."""
>>> print(b)
One line,
another line,
and some...
```

# Kiểu String



```
>>> a = "58"
>>> type(a)
<class 'str'>
>>> b=int(a)
>>> b
58
>>> type(b)
<class 'int'>
>>> f = float('1.2e-3')
>>> f
0.0012
>>> print(f)
0.0012
>>> 0.0012
0.0012
>>> eval('23-12')
11
```

# Kiểu String



```
>>> a = "Part 1"
>>> b = "and part 2"
>>> a + ' ' + b
'Part 1 and part 2'
>>> s = a * 2
>>> print(s)
Part 1Part 1
>>> s[0]
'P'
>>> s[0:4]
'Part'
>>> s[5:]
'1Part 1'
>>> s[-1]
'1'
>>> s[6:-1]
'Part '
```

# Kiểu String



```
>>> print(s)
Part 1Part 1
>>> len(s)
12
>>> 'p' in s
False
>>> 'P' in s
True
>>> 'Part' in s
True
>>> s[0] = 'B'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>> s = 'B' + s[1:]
>>> print(s)
Bart 1Part 1
```

# Kiểu String



```
>>> s = 'a string, with stuff'
>>> s.count('st')
2
>>> s.find('stu')
15
>>> s.split(' ')
['a', 'string,', 'with', 'stuff']
>>> three = '3'
>>> three.isdigit()
True
>>> supper = s.upper()
>>> supper
'A STRING, WITH STUFF'
>>> s.rjust(30)
'           a string, with stuff'
>>> "newlines\n\n\n".strip()
'newlines'
```

# Kiểu List
# Ordered collection of objects



```
>>> r
[1, 2.0, 3, 5]
>>> type(r)
<type 'list'>
>>> r[1]
2.0
>>> r[-1]
5
>>> r[1:3]
[2.0, 3]
>>> w = r + [10, 19]
>>> w
[1, 2.0, 3, 5, 10, 19]
>>> t = [0.0] * 10
>>> t
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

# Kiểu List



```
>>> r = [1, 2.0, 3, 5]
>>> r[3] = 'word'
>>> r
[1, 2.0, 3, 'word']
>>> r[0] = [9, 8]
>>> r
[[9, 8], 2.0, 3, 'word']
>>> r[0:3] = [1, 2, 5, 6]
>>> r
[1, 2, 5, 6, 'word']
>>> r[1:3] = []
>>> r
[1, 6, 'word']
>>> len(r)
3
>>> 6 in r
True
>>> r.index(6)
1
```

# Kiểu List



```
>>> r = [1, 2.0, 3, 5]
>>> r.append('thing')
>>> r
[1, 2.0, 3, 5, 'thing']
>>> r.append(['another', 'list'])
>>> r
[1, 2.0, 3, 5, 'thing', ['another', 'list']]
>>> r = [1, 2.0, 3, 5]
>>> r.extend(['item', 'another'])
>>> r
[1, 2.0, 3, 5, 'item', 'another']
>>> k = r.pop()
>>> k
'another'
>>> r
[1, 2.0, 3, 5, 'item']
```

# Kiểu List



```
>>> r = [2, 5, -1, 0, 20]
>>> r.sort()
>>> r
[-1, 0, 2, 5, 20]
>>> w = ['apa', '1', '2', '1234']
>>> w.sort()
>>> w
['1', '1234', '2', 'apa']
>>> w.reverse()
>>> w
['apa', '2', '1234', '1']
>>> v = w[:]
>>> v.reverse()
>>> v
['1', '1234', '2', 'apa']
>>> w
['apa', '2', '1234', '1']
```

# Kiểu List

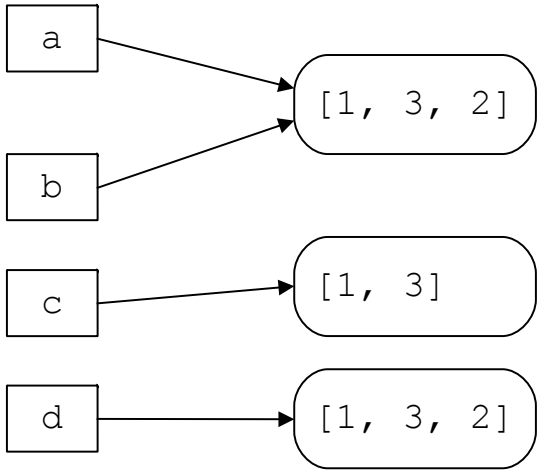


```
>>> s = 'biovitrum'
>>> w = list(s)
>>> w
['b', 'i', 'o', 'v', 'i', 't', 'r', 'u', 'm']
>>> w.reverse()
>>> w
['m', 'u', 'r', 't', 'i', 'v', 'o', 'i', 'b']
>>> r = ''.join(w)
>>> r
'murtivoib'
>>> d = '-'.join(w)
>>> d
'm-u-r-t-i-v-o-i-b'
>>> s = 'a few words'
>>> w = s.split()
>>> w
['a', 'few', 'words']
>>> ' | '.join(w)
'a | few | words'
```

# Kiểu List



```
>>> a = [1, 3, 2]
>>> b = a
>>> c = b[0:2]
>>> d = b[:]
>>> b.sort()
>>> a
[1, 2, 3]
```



# Kiểu Tuples (as List, except immutable)



```
>>> t = (1, 3, 2)
>>> t[1]
3
>>> (a, b, c) = t
>>> a
1
>>> b
3
>>> a, b, c
(1, 3, 2)
>>> a, b = b, a
>>> a, b
(3, 1)
>>> r = list(t)
>>> r
[1, 3, 2]
>>> tuple(r)
(1, 3, 2)
```

# Kiểu Dictionary
# An unordered collection of key/value pairs



```
>>> h = {'key': 12, 'nyckel': 'word'}
>>> h['key']
12
>>> h.has_key('nyckel')
True
>>> h['Per'] = 'Kraulis'
>>> h
{'nyckel': 'word', 'Per': 'Kraulis', 'key': 12}
>>> {'nyckel': 'word', 'Per': 'Kraulis', 'key': 12}
{'nyckel': 'word', 'key': 12, 'Per': 'Kraulis'}
>>> h['Per'] = 'Johansson'
>>> h
{'nyckel': 'word', 'Per': 'Johansson', 'key': 12}
>>> h = {'key': 12, 'nyckel': 'word'}
>>> del h['key']
>>> h
{'nyckel': 'word'}
```

# Kiểu Dictionary



```
>>> h = {'key': 12, 'nyckel': 'word'}
>>> 'Per' in h
False
>>> h['Per']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'Per'
>>> h.get('Per', 'unknown')
'unknown'
>>> h.get('key', 'unknown')
12
>>> h.keys()
['nyckel', 'key']
>>> h.values()
['word', 12]
>>> len(h)
2
```

# Kiểu Dictionary



```
>>> h = {'key': 12, 'nyckel': 'word'}
>>> g = h.copy()
>>> del h['key']
>>> h
{'nyckel': 'word'}
>>> g
{'nyckel': 'word', 'key': 12}
>>> h['Per'] = 'Johansson'
>>> h
{'nyckel': 'word', 'Per': 'Johansson'}
>>> h.update(g)
>>> h
{'nyckel': 'word', 'key': 12, 'Per': 'Johansson'}
```

# Nội dung



- Giới thiệu
- Lập trình căn bản
- Các kiểu dữ liệu phức
- **Lập trình hướng đối tượng**
- Vào/ra, ngoại lệ

# Lập trình hướng đối tượng



- Đối tượng (object)
- Lớp (class)
- Thực thể / thể hiện (instance)
- Trạng thái (state)
- Phương thức (method)
- Truyền thông điệp (message passing)
- Trừu tượng hoá (abstraction)
- Đóng gói (encapsulation)
- Kế thừa (inheritance)
- Đa hình (polymorphism)
- Tổng quát hoá (generalization)
- Cụ thể hoá (specialization)

# Lập trình hướng đối tượng



- Định nghĩa lớp

```
class ClassName:
    'Optional class documentation string'
    class_suite
```

- class_suite: các thuộc tính, phương thức (hàm)

- Tham số đầu tiên của phương thức thường được đặt tên là **self** để thỏa:

```
obj.meth(args) = class.meth(obj, args)
```

- **name**: public, **_name**: protected, **__name**: private

# Định nghĩa lớp



```
>>> class Employee:
...     'Common base class for all employees'
...     empCount = 0
...
...     def __init__(self, name, salary):
...             self.name = name
...             self.salary = salary
...             Employee.empCount += 1
...
...     def displayCount(self):
...             print("Total Employee %d" % Employee.empCount)
...
...     def displayEmployee(self):
...             print("Name: ", self.name,  ", Salary: ", self.salary)
...
>>>
```

# Tạo và sử dụng đối tượng



```
>>> emp1 = Employee("Toto", 2000)
>>> emp2 = Employee("Tutu", 5000)
>>> emp1.displayEmployee()
Name:  Toto , Salary:  2000
>>> emp2.displayEmployee()
Name:  Tutu , Salary:  5000
>>> print("Total Employee: %d" %Employee.empCount)
Total Employee: 2
>>> emp3 = emp2
>>> emp3.displayEmployee()
Name:  Tutu , Salary:  5000
>>> print("Total Employee: %d" %Employee.empCount)
Total Employee: 2
```

# Truy xuất thuộc tính



```
>>> emp1.age = 7
>>> emp1.age
7
>>> del emp1.age
>>> emp1.age
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Employee' object has no attribute 'age'
>>> hasattr(emp1, 'age')
False
>>> setattr(emp1, 'age', 8)
>>> getattr(emp1, 'age')
8
>>> delattr(emp1, 'age')
>>> emp1.age
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Employee' object has no attribute 'age'
```

# Thuộc tính định nghĩa sẵn



```
>>> print("Employee.__doc__:", Employee.__doc__)
Employee.__doc__: Common base class for all employees
>>> print("Employee.__name__:", Employee.__name__)
Employee.__name__: Employee
>>> print("Employee.__module__:", Employee.__module__)
Employee.__module__: __main__
>>> print("Employee.__bases__:", Employee.__bases__)
Employee.__bases__: (<class 'object'>,)
>>> print("Employee.__dict__:", Employee.__dict__)
Employee.__dict__: {'__module__': '__main__', '__doc__': 'Common base
class for all employees', 'empCount': 2, '__init__': <function Employe
e.__init__ at 0x7f220d64e3a0>, 'displayCount': <function Employee.disp
layCount at 0x7f220d64e430>, 'displayEmployee': <function Employee.dis
playEmployee at 0x7f220d64e4c0>, '__dict__': <attribute '__dict__' of
'Employee' objects>, '__weakref__': <attribute '__weakref__' of 'Emplo
yee' objects>}
```

# Xóa đối tượng



```
>>> class Point:
...     def __init( self, x=0, y=0):
...             self.x = x
...             self.y = y
...     def __del__(self):
...             class_name = self.__class__.__name__
...             print(class_name, "destroyed")
...
>>> pt1 = Point()
>>> pt2 = pt1
>>> pt3 = pt1
>>> print(id(pt1), id(pt2), id(pt3))
139784234776128 139784234776128 139784234776128
>>> del pt1
>>> del pt2
>>> del pt3
Point destroyed
```

# Kế thừa



- Định nghĩa lớp kế thừa

```
class SubClassName (ParentClass1[,
ParentClass2, ...]):
    'Optional class documentation string'
    class_suite
```

- Hàm issubclass(sub, sup), isinstance(obj, Class),
- Hàm super()

# Kế thừa (đa hình)



```
>>> class Animal:
...     'Common base class for animals'
...
...     def __init__(self, name='no-name'):
...             self.name = name
...
...     def say(self):
...             print("%s can't say" %self.name)
...
>>> class Bird(Animal):
...     def say(self):
...             print("%s twitter" %self.name)
...
>>> class Cat(Animal):
...     def say(self):
...             print("%s meow" %self.name)
...
>>>
```

# Kế thừa (đa hình)



```
>>> a = Animal()
>>> a.say()
no-name can't say
>>> a = Animal('Toto')
>>> a.say()
Toto can't say
>>> b = Bird('Flappy')
>>> b.say()
Flappy twitter
>>> c = Cat('Kitty')
>>> c.say()
Kitty meow
>>>
```

# Đa kế thừa



```
>>> class Owl(Bird,Cat):
...     pass
...
>>> o = Owl('Chic')
>>> o.say()
Chic twitter
>>> class Owlx(Bird,Cat):
...     "extra Owl"
...     def say(self):
...             print("owl...")
...             Cat.say(self)
...
>>> ox =Owlx('ChicX')
>>> ox.say()
owl...
ChicX meow
```

# Định nghĩa chồng phép toán



| OPERATOR | FUNCTION | METHOD DESCRIPTION |
|---|---|---|
| + | __add__(self, other) | Addition |
| * | __mul__(self, other) | Multiplication |
| - | __sub__(self, other) | Subtraction |
| % | __mod__(self, other) | Remainder |
| / | __truediv__(self, other) | Division |
| < | __lt__(self, other) | Less than |
| <= | __le__(self, other) | Less than or equal to |
| == | __eq__(self, other) | Equal to |
| != | __ne__(self, other) | Not equal to |
| > | __gt__(self, other) | Greater than |
| >= | __ge__(self, other) | Greater than or equal to |
| [index] | __getitem__(self, index) | Index operator |
| in | __contains__(self, value) | Check membership |
| len | __len__(self) | The number of elements |
| str | __str__(self) | The string representation |

# Định nghĩa chồng phép toán



```
>>> class Point2D:
...     def __init__(self, x=0, y=0):
...             self.x = x
...             self.y = y
...
...     def __str__(self):
...             return 'Point2D (%f, %f)' %(self.x, self.y)
...
...     def __add__(self,other):
...             return Point2D(self.x + other.x, self.y + other.y)
...
>>> p1 = Point2D(2,10)
>>> p2 = Point2D(5,-2)
>>> print(p1 + p2)
Point2D (7.000000, 8.000000)
```

# Thuộc tính có tên bắt đầu __ là thuộc tính ẩn bên trong đối tượng



```
>>> class JustCounter:
...     __secretCount = 0
...
...     def count(self):
...             self.__secretCount += 1
...             print(self.__secretCount)
...
>>> counter = JustCounter()
>>> counter.count()
1
>>> counter.count()
2
>>> print(counter.__secretCount)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'JustCounter' object has no attribute '__secretCount'
>>> print(counter._JustCounter__secretCount)
2
```

# Nội dung



- Giới thiệu
- Lập trình căn bản
- Các kiểu dữ liệu phức
- Lập trình hướng đối tượng
- **Vào/ra, ngoại lệ**

# Vào/ra



- Lệnh print, input

```
>>> print("Python is really a great language,", "isn't it?")
Python is really a great language, isn't it?
>>> istr = input("Enter your input: ")
Enter your input: My name is toto
>>> print("Received input is : ", istr)
Received input is :  My name is toto
>>> a = int(input("Enter an integer num: "))
Enter an integer num: 5
>>> b = float(input("Enter a float num: "))
Enter a float num: 7.2
>>> print(a, "+", b, "=", a + b)
5 + 7.2 = 12.2
>>> istr = input("Enter your input: ")
Enter your input: [x*5 for x in range(2,10,2)]
>>> print(eval(istr))
[10, 20, 30, 40]
```

# Vào/ra tập tin



- Hàm

```
file_obj = open(file_name [, access_mode][, buffering])
file_obj.close()
file_obj.write(string);
file_obj.read([count]);
file_obj.tell()
file_obj.seek(offset[, from])

import os
os.rename(current_file_name, new_file_name)
os.remove(file_name)
os.mkdir("newdir")
os.chdir("newdir")
os.getcwd()
os.rmdir('dirname')
```

# Vào/ra tập tin



```
>>> fo = open("foo.txt", "w")
>>> print("Name of the file: ", fo.name)
Name of the file:  foo.txt
>>> print("Closed or not : ", fo.closed)
Closed or not :  False
>>> print("Opening mode : ", fo.mode)
Opening mode :  w
>>> fo.write("Python is a great language.\nYeah its great!!\n")
45
>>> fo.close()
```

# Vào/ra tập tin



```
>>> fo = open("foo.txt", "r+")
>>> str = fo.read(10)
>>> print("Read String is : ", str)
Read String is :  Python is
>>> position = fo.tell();
>>> print("Current file position : ", position)
Current file position :  10
>>> position = fo.seek(0)
>>> str = fo.read()
>>> print("File content is : \n", str)
File content is :
 Python is a great language.
Yeah its great!!

>>> fo.close()
```

# Vào/ra tập tin



```
>>> import os
>>> os.rename("foo.txt", "foo1.txt")
>>> os.remove("foo1.txt")
>>> os.mkdir("/tmp/test")
>>> os.chdir("/home/nghi")
>>> os.getcwd()
'/home/nghi'
>>> os.rmdir("/tmp/test")
```

# Ngoại lệ



- Lệnh

```
assert Expression[, Arguments]

raise [Exception [, args [, traceback]]]

try:
   You do your operations here;
except ExceptionI:
   If there is ExceptionI, then execute this block.
except ExceptionII:
   If there is ExceptionII, then execute this block.
else:
   If there is no exception then execute this block.
finally:
   This would always be executed.
```

# Ngoại lệ



```
>>> def KelvinToFahrenheit(Temperature):
...     assert (Temperature >= 0),"Colder than absolute zero!"
...     return ((Temperature-273)*1.8)+32
...
>>> print(KelvinToFahrenheit(273))
32.0
>>> print(int(KelvinToFahrenheit(505.78)))
451
>>> print(KelvinToFahrenheit(-5))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in KelvinToFahrenheit
AssertionError: Colder than absolute zero!
```

# Ngoại lệ



```
>>> try:
...     fh = open("testfile", "r")
...     fh.write("This is my test file for exception handling!!")
... except IOError:
...     print("Error: can\'t find file or read data")
... else:
...     print("Written content in the file successfully")
... finally:
...     print("This would always be executed!!")
...
Error: can't find file or read data
This would always be executed!!
```

# Ngoại lệ



```
def functionName(level):
    if level < 1:
        raise Exception("Invalid level!", level)
        # The code below to this would not be executed
        # if we raise the exception

try:
    functionName(0)
except Exception:
    print("<!>: Invalid level")


class Networkerror(RuntimeError):
    def __init__(self, arg):
        self.args = arg

try:
    raise Networkerror("Bad hostname")
except Networkerror as e:
    print(e.args)
```

# Tài liệu tham khảo




- D. Beazley, B.K. Jones, "*Python Cookbook*", O'Reilly Media, 3rd ed., 2013
- Tutorialspoint, "*Python Tutorial*", 2019
- W3Schools, "Python Tutorial", 2019
- Phạm Thế Phi, Phạm Nguyên Khang, Đỗ Thanh Nghị. Lập trình mạng với Python. NXB Trường Đại học Cần Thơ, 2021.
- Python, https://www.python.org