LẬP TRÌNH PYTHON
LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

NGUYỄN HẢI TRIỀU[1]

[1]Bộ môn Kỹ thuật phần mềm,

Khoa Công nghệ thông tin, Trường ĐH Nha Trang

NhaTrang, February 2022

Nội dung

Nhắc lại các khái niệm

Class là một template cho một nhóm các đối tượng (objects), đối tượng là một thể hiện cụ thể của lớp. Ví dụ:

- Lớp: bản thiết kế của chiếc xe hơi.
- Đối tượng: chiếc xe hơi được tạo ra từ lớp ở trên.

Trong class có các thuộc tính (attribute) và phương thức (method).

- Mỗi lớp được xây dựng để thực hiện một nhóm chức năng đặc trưng riêng của lớp đó ⇒ Tính đóng gói–Encapsulation: cho phép dấu thông tin của đối tượng.
- Cho phép xây dựng một lớp mới dựa trên các định nghĩa của một lớp đã có ⇒ Tính kế thừa–inheritance: lớp con kế thừa tất cả các thành phần của lớp cha, có thể mở rộng, bổ sung thêm các thành phần mới.

Định nghĩa class

Một lớp dùng định nghĩa một kiểu dữ liệu mới. Để tạo một lớp, sử dụng từ khóa **class**.

Cú pháp 1.1

class *ClassName*:

    """Optional class documentation string"""
    class_suite

Trong đó, classdocs giới thiệu về class; class_suite: các thuộc tính, phương thức (hàm).

Ví dụ 1.1

Khai báo lớp tam giác

```
1 class Triangle(object):
2     '''khoi tao lop tam giac'''
3     #cac thuoc tinh va phuong thuc
```

Thuộc tính (attribute)

- Mô tả trạng thái của đối tượng
- Mỗi đối tượng đều có 1 bản sao của thuộc tính
- Được định nghĩa trong phương thức khởi tạo

Phương thức khởi tạo

__init__()

- Là phương thức (method) đặc biệt (special/magic method: *bắt đầu và kết thúc bằng 2 kí tự* _ _) mà Python sẽ gọi khi tạo một thực thể mới của class.
- Tham số đầu tiên của phương thức thường được đặt tên là **self**. Python tự động thêm tham số **self** vào phương thức.
- Khi gọi phương thức không cần phải truyền **self** vào

Phương thức khởi tạo

Ví dụ 1.2

Khai báo phương thức lớp tam giác vừa tạo với 3 thuộc tính là a,b,c

```
1 class Triangle(object):
2     '''khoi tao lop tam giac'''
3     def __init__(self,a,b,c) -> None:
4         '''constructor'''
5         self.a=a
6         self.b=b
7         self.c=c
```

Phương thức xử lý

Là một phương thức được định nghĩa để cài đặt cho một hành động của đối tượng.

Cú pháp 1.2

def *ten_phuong_thuc(self,[danh_sach_tham_so]):*
  //xử lý của phương thức

```
1  class Triangle(object):
2      '''khoi tao lop tam giac'''
3      def __init__(self,a,b,c):
4          '''constructor'''
5          self.a=a
6          self.b=b
7          self.c=c
8      def cal_p(self):
9          return self.a+self.b+self.c
10 tam_giac=Triangle(3,4,5)
11 print('chu vi tam giac: ',tam_giac.cal_p())
```

Built-In Class Attributes

- __dic__: thư mục chứa namespace của class.
- __doc__: in thông tin giới thiệu về lớp nếu có.
- __name__: class name
- __module__: module name(trong module này chứa class được định nghĩa. Thuộc tính này là __main__)
- __bases__: một tuple chứa các base classes

```
1  class Triangle(object):
2      '''khoi tao lop tam giac'''
3      def __init__(self,a,b,c):
4          '''constructor'''
5          self.a=a
6          self.b=b
7          self.c=c
8      def cal_p(self):
9          return self.a+self.b+self.c
10 if __name__=='__main__':
11     print(Triangle.__doc__) #khoi tao lop tam giac
12     print(Triangle.__name__)#Triangle
```

Khởi tạo đối tượng (Constructor)

Đối tượng được khai báo tượng trưng cho lớp.

Cú pháp 1.3

object_name = *Class_Name*(*danh_sách_đối_số*)

Ví dụ 1.3

Tạo các đối tượng tam giác:

```
1 ...
2 tam_giac_vuong=Triangle(3,4,5)
3 tam_giac_can=Triangle(3,3,5)
4 tam_giac_deu=Triangle(3,3,3)
```

Truy xuất phương thức, thuộc tính

Ví dụ 1.4

```
1  class Triangle(object):
2      '''khoi tao lop tam giac'''
3      def __init__(self,a,b,c):
4          '''constructor'''
5          self.a=a; self.b=b; self.c=c
6      def cal_p(self):
7          return self.a+self.b+self.c
8      def show(self):
9          print('a=%.1f, b=%.1f, d=%.1f, p=%.1f'%(self.a,self.b,self
               .c,self.cal_p()))
10 if __name__=='__main__':
11     tam_giac_1=Triangle(3,4,5)
12     tam_giac_2=Triangle(3,3.5,5)
13     tam_giac_1.show() #truy xuat phuong thuc show()
14     print('chu vi tam giac 2: ',tam_giac_2.cal_p()) #truy xuat
           phuong thuc cal_p()
```

Chú ý

Có thể thêm, xóa hoặc sửa đổi các thuộc tính của các lớp và các đối tượng bất cứ lúc nào:

- getattr(object,name[,default]): truy cập thuộc tính của đối tượng
- hasattr(object,name): kiểm tra sự tồn tại của thuộc tính
- setattr(object,name,value): thiết lập thuộc tính, nếu thuộc tính chưa tồn tại thì sẽ được tạo
- delattr(object,name): xóa thuộc tính

```
1 ...
2 if __name__=='__main__':
3     tam_giac=Triangle(3,4,5)
4     print("kt ton tai cua thuoc tinh canh x: ",hasattr(tam_giac,'
          x'))#kt ton tai cua thuoc tinh canh x:  False
5     print("truy cap thuoc tinh canh a trong lop tam giac: ",
          getattr(tam_giac,'a'))#truy cap thuoc tinh canh a trong
          lop tam giac:  3
```

Hủy đối tượng

- Python tự động xóa các đối tượng không cần thiết để giải phóng bộ nhớ, quá trình này được gọi là *garbage collector*
- Một lớp có thể thực hiện phương thức __del__() để hủy, gọi là 'destructor'

```
1  class Employee:
2      # Initializing
3      def __init__(self):
4          print('Employee created')
5      # Calling destructor
6      def __del__(self):
7          print("Destructor called")
8  def Create_obj():
9      print('Making Object...')
10     obj = Employee()
11     print('function end...')
12     return obj
13 print('Calling Create_obj() function...')
14 obj = Create_obj()
15 print('Program End...')
```

Ví dụ 1.5

Define a class which has at least two methods: getString: to get a string from console input printString: to print the string in upper case. Also please include simple test function to test the class methods.

Ví dụ 1.6

Define a class named Circle which can be constructed by a radius. The Circle class has a method which can compute the area.

Ví dụ 1.7

Define a class named Rectangle which can be constructed by a length and width. The Rectangle class has a method which can compute the area.

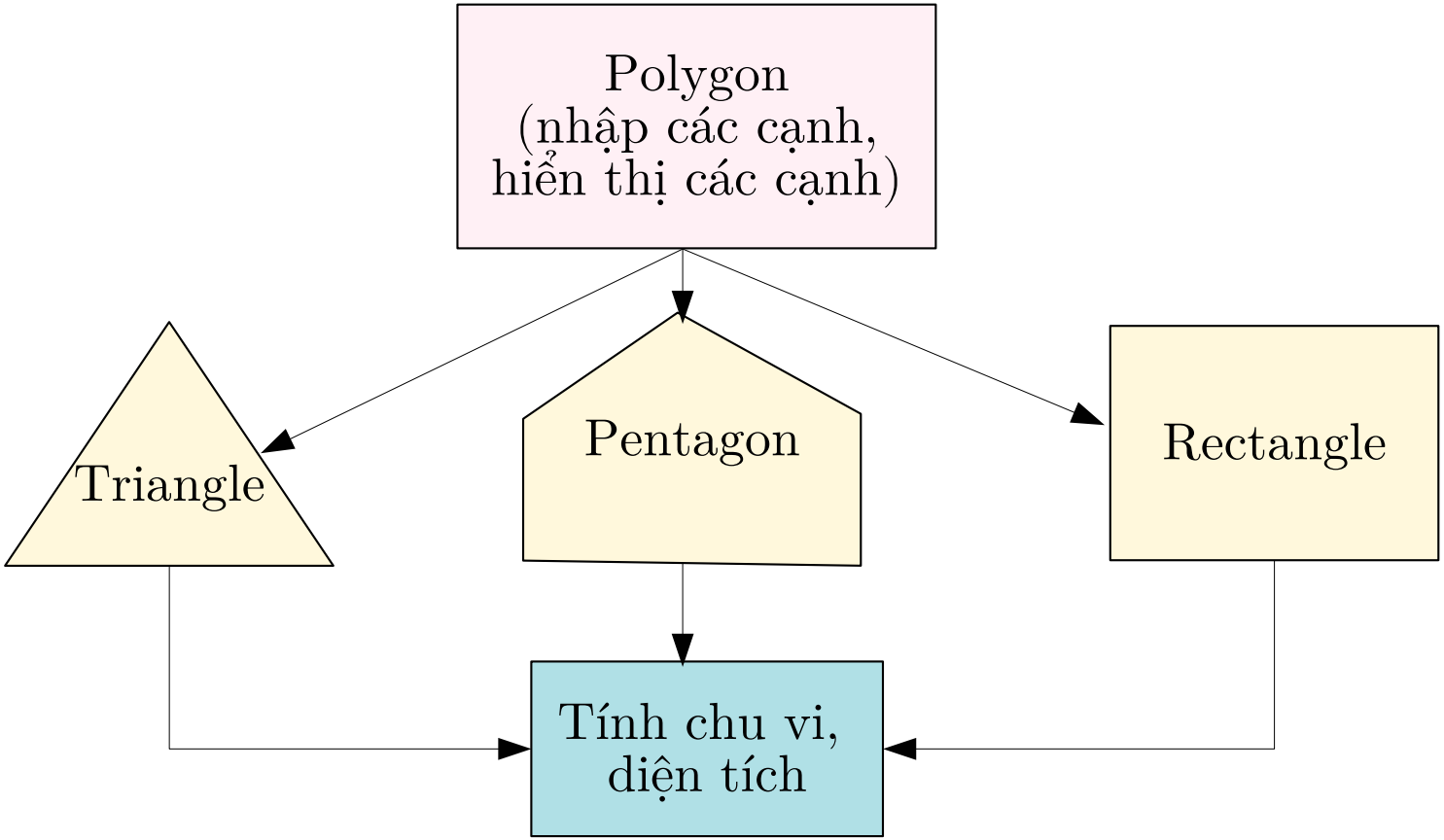
Kế thừa (inheritance)

- Kế thừa (inheritance) là quá trình mà các thuộc tính (attribute) và các hành vi (behavior) được truyền từ thực thể cha đến thực thể con.
- Tái sử dụng lại các thuộc tính và hành vi ở lớp cha, tức là kế thừa từ lớp cha.
- Lớp con được gọi là *subclass*, lớp cha được gọi là *superclass*.
- Một lớp con có thể kế thừa từ nhiều lớp cha.

Cú pháp 2.1

class *<Sub_Class_Name>(ParentClass1,ParentClass2,...)*:
"'classdocs: ...."'
phương thức, thuộc tính

Ví dụ inheritance



Hình 1: Ví dụ xây dựng lớp cha Polygon và các lớp con

Xây dựng class Polygon

- Attribute: number_of_edges
- Function: input_edges(self), display_edges(self)

```
1  from math import sqrt
2  class Polygon(object):
3      '''classdocs: Polygon with edges, input and show edges'''
4      def __init__(self, number_of_edges) -> None:
5          '''constructor'''
6          self.number_of_edges=number_of_edges
7          self.edges=[0 for i in range(number_of_edges)]
8      def input_edges(self):
9          self.edges=[float(input('enter edges '+str(i+1)+':')) for
                i in range(self.number_of_edges)]
10     def display_edges(self):
11         for i in range(self.number_of_edges):
12             print("Edge "+str(i+1)+" is "+str(self.edges[i]))
```

class Triangle kế thừa từ class Polygon

Function: find_perimeter(self), find_area(self). *Lưu ý: phương thức khởi tạo thuộc tính lớp con ở dòng 4 có thể sử dụng super() hoặc tên class cha như dòng 6.*

```
1  class Triangle(Polygon):
2      '''classdocs: class Triagle from Polygon'''
3      def __init__(self) -> None:
4          super().__init__(3)
5          # or using superclass name
6          # Polygon.__init__(self,3)
7      def find_area(self):
8          a,b,c=self.edges
9          p2=(a+b+c)/2
10         return sqrt(p2*(p2-a)*(p2-b)*(p2-c))
11     def find_perimeter(self):
12         a,b,c=self.edges
13         return a+b+c
14 if __name__=='__main__':
15     tam_giac=Triangle(); tam_giac.input_edges()
16     tam_giac.display_edges()
17     s=tam_giac.find_area(); p=tam_giac.find_perimeter()
18     print("area=%f, perimeter=%f"%(s,p))
```

Overriding Method

Ghi đè phương thức

- Lớp con có thể ghi đè phương thức của lớp cha khi ta muốn viết lại/bổ sung nội dung cho phương thức trong lớp con có cùng chức năng.
- Phương thức ghi đè sẽ trùng tên và trùng tham số với phương thức của lớp cha.

```
1 #Polygon (lop cha)
2 def display_edges(self):
3     for i in range(self.number_of_edges):
4     print("Edge "+str(i+1)+" is "+str(self.edges[i]))
5 #Ghi de phuong thuc o Triangle (lop con)
6 def display_edges(self):
7     print("Triangle has 3 edges")
8     return super().display_edges()
```

Overloading method

• Một số phương thức ở lớp cơ sở: __del/repr/str/cmp__(*self*) có thể được ghi đè trong class do ta tạo ra.
• Overloading operator: ví dụ toán tử + được nạp chồng cho cả class int và class str.
• Để sử dụng toán tử + cho các đối tượng thì trong class tương ứng ta cần phải định nghĩa phương thức có tên __add__.
• Một số overloading operator phổ biến, *:__mul__(self,other); -:__sub__(self,other); /:__truediv__(self,other); %:__mod__(self,other); <:__lt__(self,other); >:__gt__(self,other); ==:__eq__(self,other); ; <=:__le__(self,other); in:__contains__(self,value); len:__len__(self); str:__str__(self)

Overloading method

Ví dụ 2.1

Sử dụng toán tử + để cộng 2 vector

```
1  class vector2d(object):
2      '''classdocs:vector in 2dim'''
3      def __init__(self,a,b) -> None:
4          self.a,self.b=a,b
5      def __str__(self) -> str:
6          return 'vector(%f,%f)'%(self.a,self.b)
7      def __add__(self,other):
8          return vector2d(self.a+other.a,self.b+other.b)
9  vector1=vector2d(1,2) ; print(vector1)
10 vector2=vector2d(3,4) ; print(vector2)
11 vector3=vector1+vector2; print(vector3)
```

Overloading method

Ví dụ 2.2

định nghĩa nhân vô hương vector và ghi đè toán tử nhân.

```
1  class vector2d(object):
2      '''classdocs:vector in 2dim'''
3      def __init__(self,a,b) -> None:
4          self.a,self.b=a,b
5      def __str__(self) -> str:
6          return 'vector(%.1f,%.1f)'%(self.a,self.b)
7      def __add__(self,other):
8          return vector2d(self.a+other.a,self.b+other.b)
9      def __mul__(self,other):
10         return vector2d(self.a*other.a,self.b*other.b)
11 class vector2d_child(vector2d):
12     def __init__(self, a, b) -> None:
13         super().__init__(a, b)
14     def __mul__(self, other):
15         return str(self.a*other.a+self.b*other.b)
16 vector1=vector2d_child(1,2) ; print(vector1)
17 vector2=vector2d_child(3,4) ; print(vector2)
18 vohuong=vector1*vector2; print(vohuong)
```

Data hiding

Các thuộc tính của một đối tượng có thể có hoặc không được nhìn thấy từ bên ngoài class. Để thuộc tính không được nhìn thấy từ bên ngoài class, ta sẽ khai báo thuộc tính với dấu __.

```
1  class my_class(object):
2      __hidden_variable=0
3      def __init__(self) -> None:
4          pass
5      def add(self, increment):
6          self.__hidden_variable+=increment
7          print('Hidden variable: ',self.__hidden_variable)
8  MyClass=my_class()
9  MyClass.add(1) #->Hidden variable:  1
10 print(MyClass.__hidden_variable)
11 #->AttributeError: 'my_class' object has no attribute '
       __hidden_variable'
```

Data hiding

Tương tự thuộc tính không được nhìn thấy từ lớp con.

```
1  class my_class(object):
2      __hidden_variable=0
3      def __init__(self) -> None:
4          pass
5      def add(self, increment):
6          self.__hidden_variable+=increment
7          print('Hidden variable: ',self.__hidden_variable)
8  class my_class_child(my_class):
9      def __init__(self) -> None:
10         super().__init__()
11     def test(self):
12         print(self.__hidden_variable)
13 MyClassChild=my_class_child();
14 MyClassChild.add(1) #Hidden variable:  1
15 MyClassChild.test()
16 # AttributeError: 'my_class_child' object has no attribute '
       _my_class_child__hidden_variable'
```

Bài tập lập trình hướng đối tượng

Ví dụ 3.1

Define a class named Shape and its subclass Square. The Square class has an init function which takes a length as argument. Both classes have a area function which can print the area of the shape where Shape's area is 0 by default. (using overriding method)

Ví dụ 3.2

Define a class Person and its two child classes: Male and Female. All classes have a method "getGender"which can print "Male"for Male class and "Female"for Female class.

Ví dụ 3.3

Giải phương trình bậc nhất theo hướng đối tượng.

Bài tập lập trình hướng đối tượng

Ví dụ 3.4

Viết chương trình tính tổng, hiệu, tích, thương của hai số nguyên.

Ví dụ 3.5

Xây dựng chương trình tính khoảng cách giữa hai điểm.

Ví dụ 3.6

Xây dựng chương trình tính tổng, hiệu, tích, thương của hai phân số.

@staticmethod

Define a class named which has a static method using *@staticmethod* decorator to define class static method.

Ví dụ 4.1

Define a class named Vietnamese which has a static method called printNationality.

```
1 class Vietnamese(object):
2     @staticmethod
3     def printNationality():
4         print("Vietnam")
5 aVietnamese = Vietnamese()
6 aVietnamese.printNationality()
7 Vietnamese.printNationality()
```

Tài liệu tham khảo

Trung tâm tin học , Đại Học KHTN Tp.HCM
Lập trình Python nâng cao. *03/2017*.

Mark Lutz
Learning Python (5th Edition). *O'Reilly Media, Inc, 2013*.

Luciano Ramalho
Fluent Python (2nd Edition). *O'Reilly Media, Inc, 2021*.

Python Software Foundation
`https://docs.python.org/3/tutorial/`