Bài giảng môn học:

Khoa Học Dữ Liệu (7080509)

CHƯƠNG 3:
Lập trìnhPython cơ bản (Phần 3)

FIT.HUMG

Nội dung bài học:

1. Hàm trong Python

a. Giới thiệu Hàm

b. Xây dựng hàm trong Python

b. Return | Tham số của hàm |Phạm vi của biến

c. Hàm ẩn danh (Lambda)

2. Lớp và đối tượng trong Python

a. Tạo lớp – đối tượng

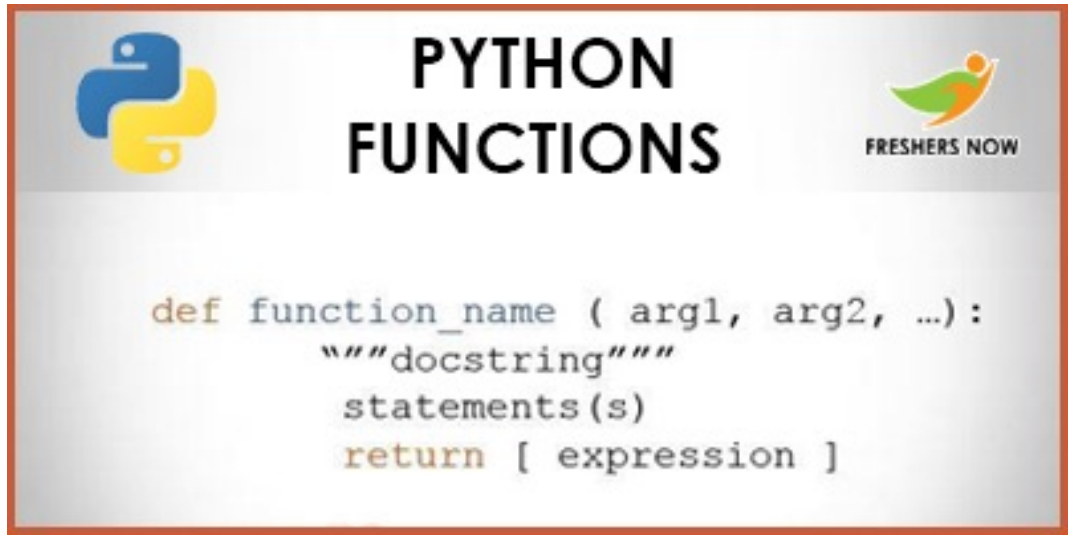
b. Truy cập phương thức, thuộc tính của đối tượng

c. Thừa kế

1. Hàm trong Python

Cơ bản về hàm trong Python

- Hàm trong Python là một nhóm các câu lệnh trong chương trình được tổ chức chung với nhau để thực hiện một chức năng hay một nhiệm vụ cụ thể nào đó.
- Sử dụng hàm giúp phân rã chương trình từ một chương trình lớn, phức tạp thành các phần cụ thể nhỏ hơn giúp dễ quản lý, tổ chức, nâng cao khả năng tái sử dụng và chia sẻ công việc.




```python
#Ví dụ: Xây dựng 1 Hàm thực hiện chào hỏi
def hello_aiv(name):
    print('Hi ',name,', How are you?')
    print('Have a nice day!')
```

```python
#Sử dụng hàm đã xây dựng
hello_aiv('AIV')
```

```
Hi  AIV , How are you?
Have a nice day!
```

Cơ bản về hàm trong Python

1. Cú pháp:

```
def tên_hàm(các_tham_số):
    "function_docstring"
    Các câu lệnh xử lý bên trong hàm
    return [kết quả trả về]
```

- Từ khóa **def** được sử dụng để bắt đầu phần định nghĩa hàm.
- sau đó là **tên_hàm**, tên hàm được đặt theo quy tắc như tên biến.
- Các tham số được truyền vào bên trong các dấu ngoặc đơn.
- Ở cuối là dấu hai chấm "**:**".
- Sau đó là lệnh để được thực thi.
- Kết quả trả về cho hàm được thực hiện thông qua lệnh return

Lưu ý: Hàm không bắt buộc phải có tham số truyền vào hay kết quả trả về

Cơ bản về hàm trong Python

- Trước khi bắt tay vào xây dựng 1 hàm trong Python, cần phải tự **trả lời các câu hỏi**:
  - **Hàm này sử dụng nhằm mục đích gì?**
  - **Hàm này nhận đầu vào là gì?**
  - **Hàm này trả kết quả ra là gì?**

```python
#Xây dựng hàm tính n!
#1)Hàm này dùng để làm gì? - Để tính n!
#2)Hàm này nhận dữ liệu vào là gì? - Một số nguyên dương N
#3)Hàm trả kết quả là gì? - Một số nguyên dương là tích của 1*2*...*N
def giai_thua(n):
    #Nhóm câu lệnh xử lý bên trong hàm
    tich=1
    for i in range(1,n+1):
        tich=tich*i
    #kết quả trả về cho hàm
    return tich
```

```python
n = int(input('Nhập vào một số nguyên N:'))
print(n,'!=',giai_thua(n))
```

```
Nhập vào một số nguyên N:10
10 != 3628800
```

Cơ bản về hàm trong Python

Gọi hàm:

- Hàm sau khi được xây dựng, có thể thực hiện lời gọi hàm ở nơi nào cần dùng đến.

```
1  n = int(input('Nhập vào một số nguyên N:'))
2  print(n,'!=',giai_thua(n))
```

```
Nhập vào một số nguyên N:10
10 != 3628800
```

```
1  #Gọi hàm giai_thua đã xây dựng
2  #Tính 12!
3  print('12! = ', giai_thua(12))
```

```
12! =  479001600
```

- Các lệnh mà chúng ta đã được học và sử dụng trước đây như: print(), input(), type(), int(), float(), str()… Đây thực chất là các **hàm được Python định nghĩa sẵn**.

Cơ bản về hàm trong Python

Lệnh return:

- Lệnh **return <kết quả trả về>** được sử dụng để trả kết quả xử lý thông qua tên hàm.
- Lệnh return có thể có hoặc không.
- Trong trường hợp không cung cấp <kết quả trả về>, thì hàm return này sẽ trả về None. Nói cách khác, lệnh return được sử dụng để thoát khỏi định nghĩa hàm.

```
1  #Hàm không có lệnh return
2  def hello_aiv(name):
3      print('Hello ',name,', How are you?')
4      print('Have a nice day.....!')
```

```
1  #Hàm trả về 1 giá trị
2  def giai_thua(n):
3      #nhóm các câu lệnh xử lý bên trong hàm
4      tich=1
5      for i in range(1,n+1):
6          tich=tich*i
7      #kết quả trả về cho hàm
8      return tich
```

Cơ bản về hàm trong Python

Lệnh return() có thể trả về 1 hay nhiều kết quả, nếu có nhiều hơn một kết quả thì ngăn cách nhau bởi **dấu phẩy**.

```python
#Hàm trả về nhiều giá trị
#Hàm tính tổng, hiệu, tích, thương:
def all_ab(a,b):
    add = a+b
    sub = a-b
    multi = a*b
    div = a/b
    #hàm trả về kết quả là 4 giá trị
    return add, sub, multi, div
```

```python
a=10
b=6
#lấy kết quả trả về khi thực hiện hàm
tong,hieu,tich,thuong = all_ab(a,b)

#Lưu ý: Thứ tự trả về theo đúng thứ tự đã viết trong
#câu lệnh return
print('Tổng ',a,'+',b,'=',tong)
print('Hiệu ',a,'-',b,'=',hieu)
print('Tich ',a,'*',b,'=',tich)
print('Thương ',a,'/',b,'=',thuong)
```

```
Tổng  10 + 6 = 16
Hiệu  10 - 6 = 4
Tich  10 * 6 = 60
Thương  10 / 6 = 1.6666666666666667
```

Cơ bản về hàm trong Python

Tham số truyền vào hàm:

- Tham số bắt buộc
- Tham số có mặc định (Default parameter)
- Tham số có độ dài biến (Variable-Length Parameter)

➢ Tham số bắt buộc

```
1  #Xây dựng hàm tính n!
2  #Hàm giai_thua có 1 tham số bắt buộc n
3  def giai_thua(n):
4      tich=1
5      for i in range(1,n+1):
6          tich=tich*i
7      #kết quả trả về cho hàm
8      return tich
```

```
1  #Gọi hàm giai_thua đã xây dựng
2  print('12! = ', giai_thua(12))
```

```
12! =  479001600
```

```
1  #Gọi hàm giai_thua đã xây dựng
2  #Khi không truyền vào tham số
3  print('12! = ', giai_thua())
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-14-01226097b9b9> in <module>
      1 #Gọi hàm giai_thua đã xây dựng
      2 #Khi không truyền vào tham số
----> 3 print('12! = ', giai_thua())

TypeError: giai_thua() missing 1 required positional argument: 'n'
```

Cơ bản về hàm trong Python

➢ Tham số mặc định cho hàm:

- Để hạn chế trường hợp báo lỗi khi khi gọi hàm không cung cấp tham số thì trong Python cũng cung cấp cho chúng ta **thiết lập giá trị mặc định của tham số** khi khai báo hàm. Bằng cách sử dụng dấu **=** với cú pháp như sau:

```
def ten_ham(param = defaultValue):
    # code
```

- Trong đó: **defaultValue** là giá trị mặc định của tham số đó mà bạn muốn gán.

Cơ bản về hàm trong Python

- **Ví dụ:**
  - Hàm **sum_ab(a,b)** gọi khi truyền tham số và và khi không truyền tham số:

```
#Hàm tính tổng
def sum_ab(a=5, b =7):
    total = a + b
    return total
```

```
#Gọi hàm sum_ab() truyền vào 2 tham số
print(sum_ab(8,13))
```

```
21
```

```
#Gọi hàm sum_ab() không truyền vào tham số
#Sử dụng tham số mặc định
print(sum_ab())
```

```
12
```

Cơ bản về hàm trong Python

Phạm vi của biến.

- Một biến chỉ có tác dụng trong phạm vi mà nó khai báo (global – local).
- Khi một biến được khai báo ở trong hàm thì nó chỉ có thể được sử dụng ở trong hàm đó.

```
1  x = 300 #Biến toàn cục, có tác dụng trong toàn bộ chương trình
2  y = 800
3  def myfunc():
4      #Biến địa phương, chỉ có tác dụng trong thân hàm
5      x = 200
6      total = x + y
7      print('(Local) x :', x)
8      print('total :', total)
9  #Gọi hàm
10 myfunc()
11
12 print('----------------------')
13 print('(global) x:', x)
14
```

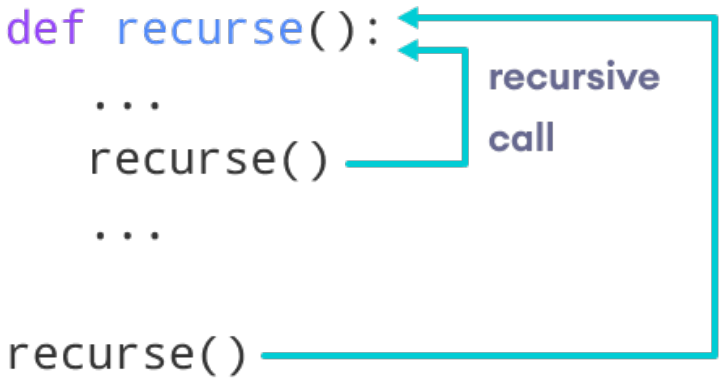
```
(Local) x : 200
total : 1000
----------------------
(global) x: 300
```

```
1  #Thiết lập một biến local thành global
2  def myfunc():
3      global k #Thiết lập biến k là biến global
4      k = 300
5      print('Inside func: k = ',k)
6
7  myfunc()
8
9  print('Outside func: k =', k)
```

```
Inside func: k =  300
Outside func: k = 300
```

Hàm đệ quy

- Ngôn ngữ Python cho phép hàm gọi đến chính nó, người ta gọi là đệ quy hay quay lui



- Trong giải thuật phải có một điều kiện dừng để đệ quy kết thúc.
- Chương trình sử dụng đệ quy thì dễ hiểu nhưng hao tốn tài nguyên CPU, ảnh hưởng đến thời gian chạy chương trình nếu số lần đệ quy của hàm lớn.

```
1  #Hàm tính n! theo phương pháp đệ quy
2  def giai_thua(n):
3      if n==0:          #Điều kiện dừng để kết thúc đệ quy
4          return 1;     #vì 0! = 1 nên n==0 là vị trí kết thúc của đệ quy
5      else:
6          return giai_thua(n-1)*n  #1*2*....*n
```

```
1  n = int(input("Nhập vào số N:"))
2  print(n,'!=',giai_thua(n))
```

```
Nhập vào số N:10
10 != 3628800
```

Hàm ẩn danh - lambda

- Một hàm ẩn danh là một hàm được định nghĩa mà không có tên
- Các hàm bình thường được định nghĩa bằng từ khóa def; hàm ẩn danh được định nghĩa bằng từ khóa lambda

Cú pháp:

```
lambda arguments :  Expression
```

- Các hàm lambda có thể có bất kỳ đối số nào nhưng chỉ có một biểu thức.

```
#Ví dụ hàm ẩn danh: 1 tham số
x = lambda a : a + 10
#lambda a: a + 10 là hàm lambda
#Trong đó: a là tham số truyền vào
#          a+10 là biểu thức

print(x(5))
print(x(7))
```

```
15
17
```

```
# Program to filter out only the even items from a list
my_list = [1, 5, 4, 6, 8, 11, 3, 12]
new_list = list(map(lambda x: x*2, my_list))
print(new_list)
```

```
[2, 10, 8, 12, 16, 22, 6, 24]
```

Thực hành

Bài 14: Viết hàm

1) Viết hàm **greeting():** Trả về câu chào với tham số truyền vào là chuỗi họ tên và năm sinh **(Xem lại bài tập số 2)**

2) Viết hàm **rabbit_count()**: tính số thỏ trong rừng khi truyền vào số tháng **(Xem lại bài tập số 3)**

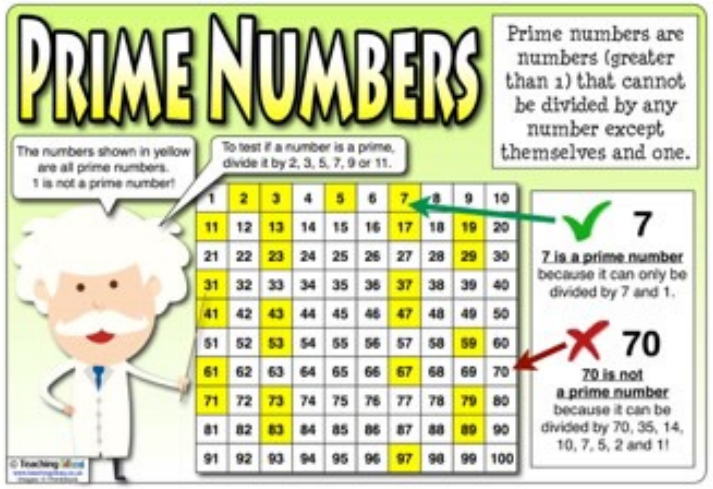
3) Viết hàm **count_mark():** trả về số sinh viên học lại và tổng số sinh viên trong lớp với tham số truyền vào là một danh sách bảng điểm **(Xem lại bài tập số 5 ý 1, 2)**

Bài 14: Viết hàm

4) Viết hàm **bmi_show():** Trả về nhận xét dựa vào chỉ số BMI đã tính với 2 tham số truyền vào là chiều cao, cân nặng **(Xem lại bài tập số 7)**

5) Viết hàm **cal_point()**: Trả về điểm trung bình hệ 10 và hệ 4 của một học sinh khi truyền vào danh sách điểm **(Xem lại bài tập số 10 ý 2)**

6) Viết hàm **list_prime():** trả danh sách các số nguyên tố trong khoảng tử 1 đến n với tham số truyền vào là n **(Xem lại bài tập số 12)**

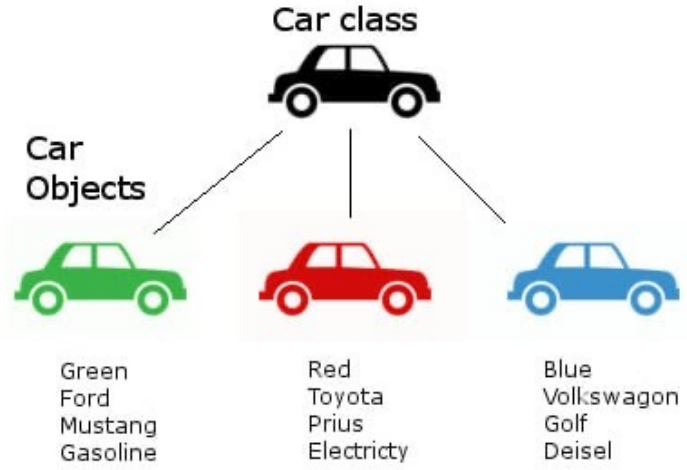


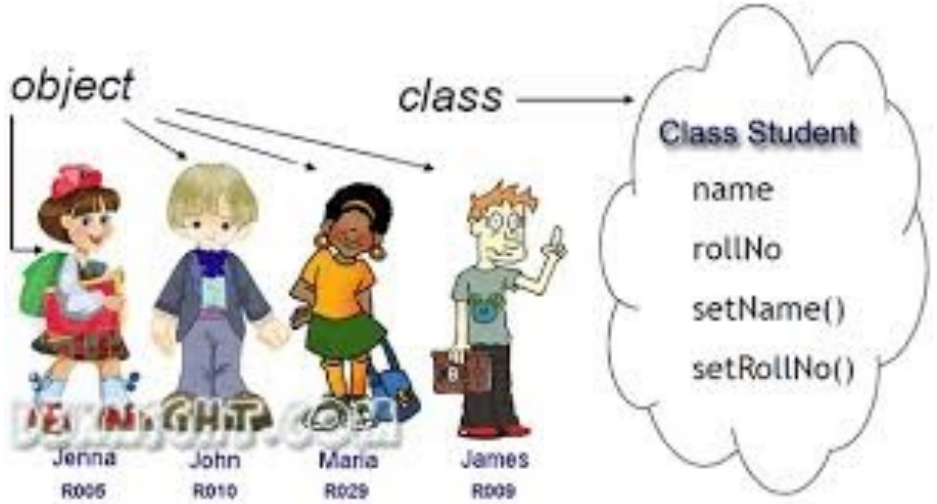
2. Lớp, đối tượng trong Python

Lớp, đối tượng trong Python

1. Giới thiệu lớp

- Python là một ngôn ngữ lập trình hướng đối tượng

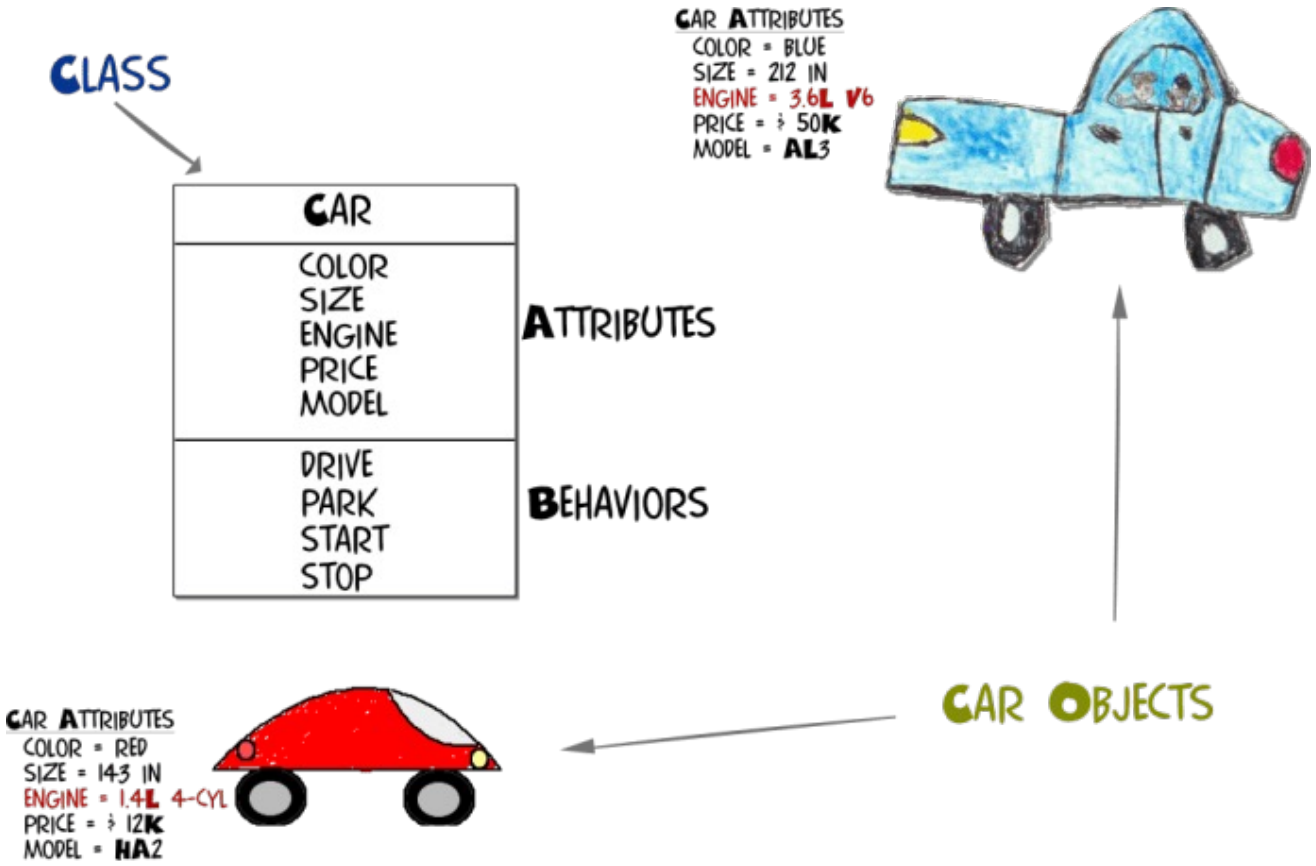




- Lớp (class) là một cấu trúc logic dùng để định nghĩa khuôn dạng và tính chất của các Đối tượng (Object).
- Lớp được định nghĩa như là một kiểu dữ liệu mới.

Lớp, đối tượng trong Python

- Mỗi một đối tượng có 2 thành phần chính:



- Các thuộc tính: dùng để chứa các thông tin mô tả các đặc điểm của đối tượng. Sử dụng các biến để lưu giữ những thông tin này. Khi đó các biến được gọi là các thuộc tính.

- Các phương thức: dùng để mô tả các hành vi của đối tượng.Sử dụng các hàm để mô tả các hành vi này. Khi đó các hàm được gọi là các phương thức.

Lớp, đối tượng trong Python

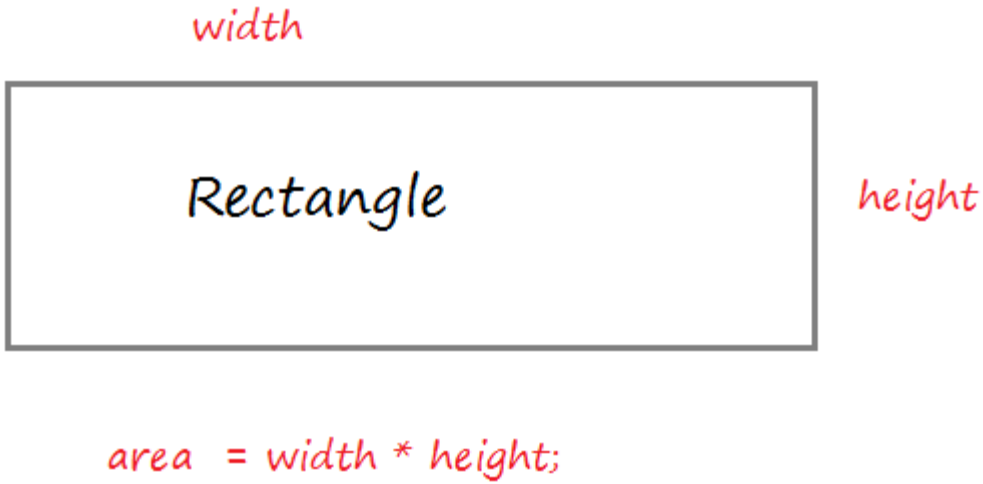
➢ Khai báo Class trong Python

```
class ClassName:
    'Gồm các thuộc tính, phương thức'
     # Code ...
```

- Trong đó, **className** là tên của class cần khai báo. Một số lưu ý khi đặt tên cho lớp:
  - ➢ Tên lớp nên là một danh từ,
  - ➢ Tên lớp được đặt ký tự đầu tiên của mỗi từ là in hoa.
  - ➢ Tên lớp nên đơn giản, mang tính mô tả và đầy đủ ý nghĩa
  - ➢ Tên lớp không được là một từ khóa nào đó của Python
  - ➢ Tên lớp không bắt đầu bằng số, có thể bắt đầu với dấu $ hoặc ký tự gạch dưới.

Lớp, đối tượng trong Python

- **Khai báo Class: Rectangle**
  - **có 2 thuộc tính: width, height**
  - **2 phương thức: getArea(), getPrimeter()**




```python
#Tạo một lớp Rectangle

class Rectangle:
    #Lớp Rectangle có 2 thuộc tính: width, height

    #Phương thức khởi tạo đối tượng (Contructor)
    def __init__(self, width, height):
        self.width = width
        self.height = height

    #Phương thức tính diện tích
    def getArea(self):
        area = round(self.width * self.height,1)
        return area

    #Phương thức tính chu vi
    def getPerimeter(self):
        perimeter = round((self.width + self.height)*2,1)
        return perimeter
```

Lớp, đối tượng trong Python

➢ Một số khái niệm hướng đối tượng

- **Thuộc tính (Attribute):** Thuộc tính là một thành viên của lớp, Hình chữ nhật có 2 thuộc tính width và height
- **Phương thức (Method):**
  - Phương thức của class tương tự như một hàm thông thường, nhưng nó là một hàm của class. Để sử dụng được cần phải gọi thông qua đối tượng.
  - Tham số đầu tiên của phương thức luôn là self (Một từ khóa ám chỉ chính class đó)
- **Phương thức khởi tạo (Constructor):**
  - Là một phương thức đặc biệt của lớp, luôn có tên là __init__. Tham số đầu tiên luôn là self. Chỉ có thể định nghĩa một contructor trong class
  - Constructor được sử dụng để tạo ra một đối tượng.
  - Constructor gán các giá trị từ tham số vào các thuộc tính của đối tượng sẽ được tạo ra

Lớp, đối tượng trong Python

➢ Khởi tạo đối tượng từ lớp

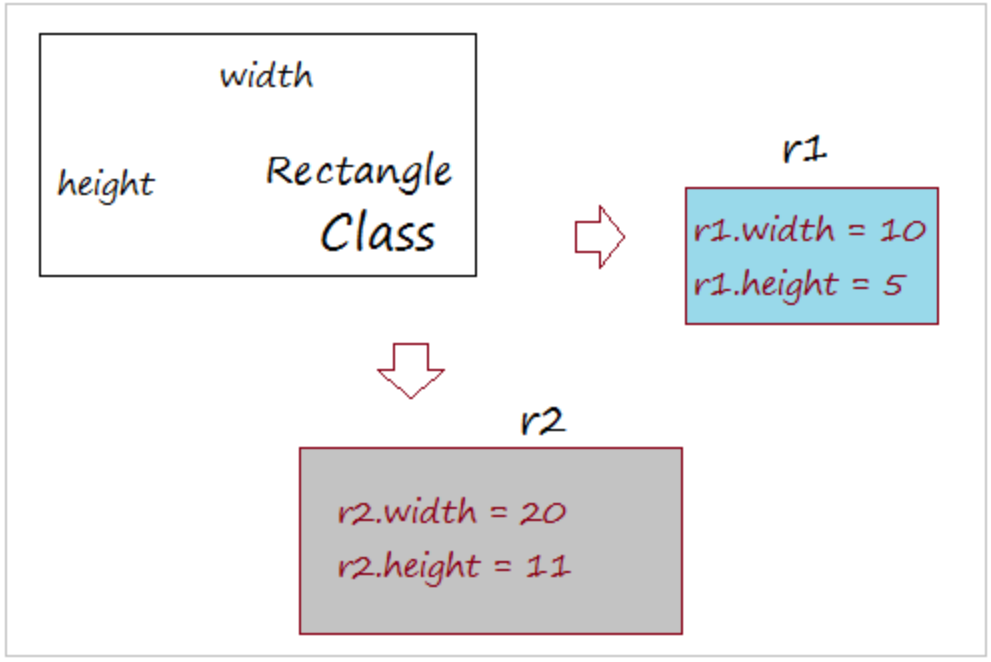
Sau khi đã khai báo được class trong Python rồi, thì để khởi tạo các đối tượng sử dụng cú pháp sau:

```
variableName = className()
```

Trong đó:

- **variableName** là tên đối tượng.
- **className** là class muốn khởi tạo.

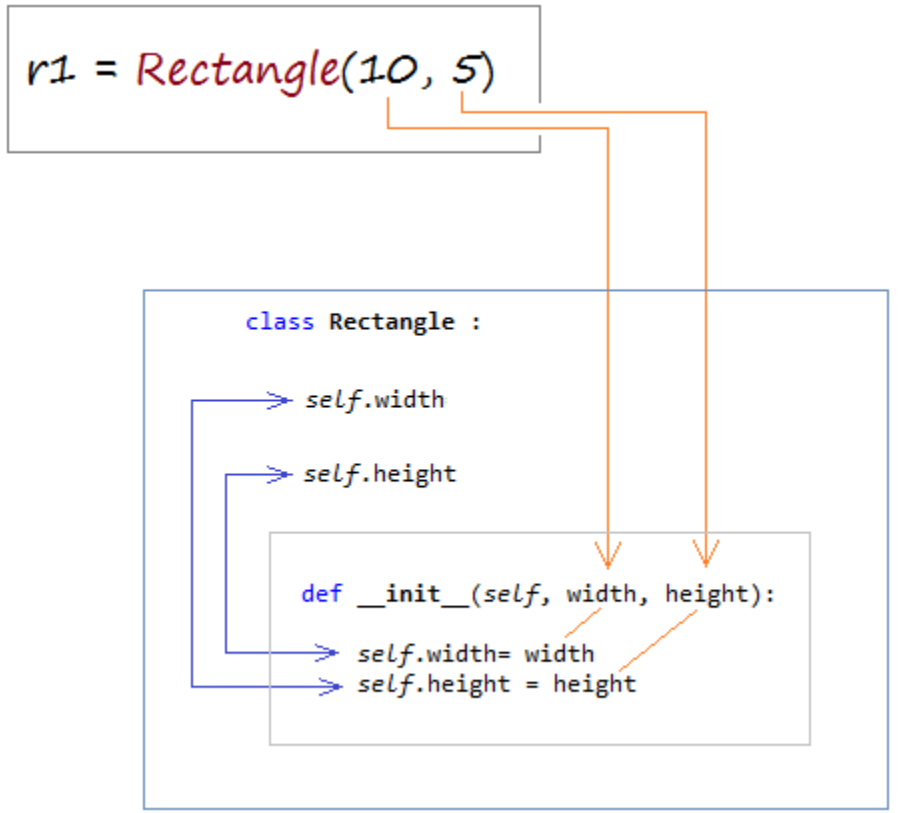
```
1  #Ví dụ: Tạo 2 đối tượng r1, r2 từ class Rectangle
2  r1 = Rectangle(10,5)
3
4  r2 = Rectangle(20,11)
```



Lớp, đối tượng trong Python

➢ Điều gì xẩy ra khi khởi tạo một đối tượng

Khi tạo một đối tượng của lớp **Rectangle**, phương thức khởi tạo (constructor) của class đó sẽ được gọi để tạo một đối tượng, và các thuộc tính của đối tượng sẽ được gán giá trị từ tham số



Lớp, đối tượng trong Python

- Sau khi đã khởi tạo được đối tượng sẽ có thể truy cập được các thuộc tính và phương thức trong class đó.
- Bằng cách sử dụng dấu . theo cú pháp sau:

```
# truy cap den thuoc tinh
object.propertyName

#truy cap den phuong thuc
object.methodName()
```

Trong đó:

- object là biến thể hiện lại object.
- propertyName là tên thuộc tính muốn truy xuất.
- methodName là tên phương thức muốn truy xuất.

Lớp, đối tượng trong Python

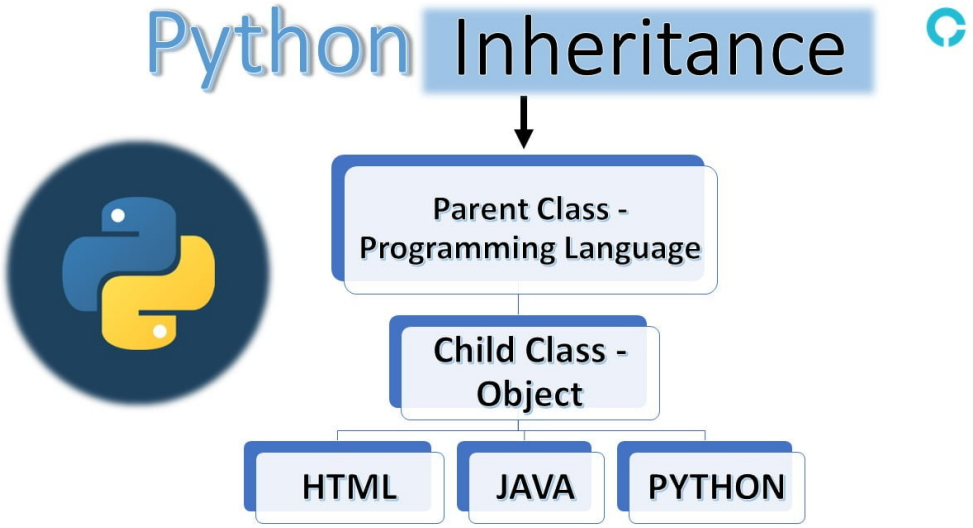
- **Ví dụ:** sẽ truy xuất đến các thuộc tính và phương thức trong class Rectangle

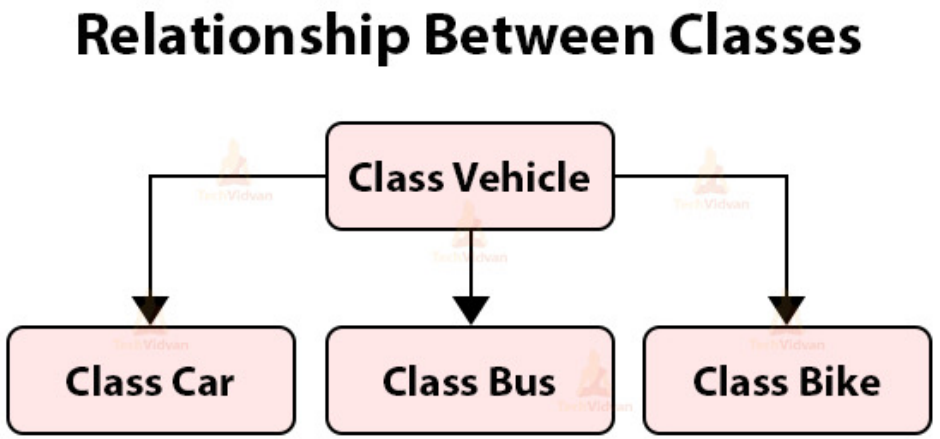
```
#Lấy thuộc tính width, height  của đối tượng rec1
x = r1.width
y = r1.height
print('----Thuộc tính----------------')
print('1. Thuộc tính Chiều rộng: ', x)
print('2. Thuộc tính Chiều dài: ', y)
#Gọi phương thức getArea, getPerimeter của đối tượng rec1
dt = r1.getArea()
cv = r1.getPerimeter()
print('---------Phương thức----------')
print('1. Phương thức tính Diện tích:', dt)
print('2. Phương thức tính Chu vi:',cv)
```

```
----Thuộc tính----------------
1. Thuộc tính Chiều rộng:  10
2. Thuộc tính Chiều dài:  5
---------Phương thức----------
1. Phương thức tính Diện tích: 50
2. Phương thức tính Chu vi: 30
```

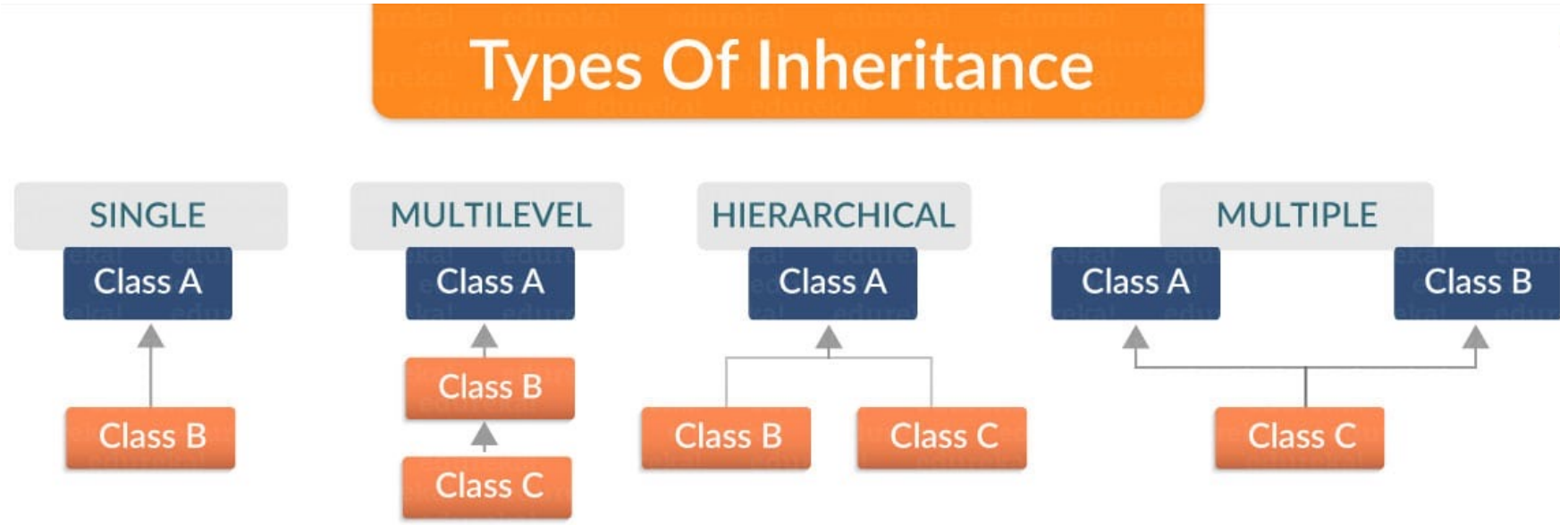
Thừa kế

- Việc thực hiện thao tác thừa kế là đơn giản và hiệu quả, trong đó cho phép tạo ra môt lớp mới từ một lớp có sẵn.
- Lớp mới dẫn xuất từ lớp có sẵn có thể tái sử dụng các biến và phương thức của nó mà không cần phải viết lại hoặc sửa lại đoạn mã.
- Khi thực hiện thừa kế, lớp được dẫn xuất từ lớp khác được gọi là phân lớp, lớp dẫn xuất, **lớp con**; Lớp mà từa đó phân lớp được dẫn xuất thì được gọi là siêu lớp, lớp cơ sở, **lớp cha**





Thừa kế



- **Đơn thừa kế (Single):** Xảy ra khi một lớp con thừa kế chỉ một lớp cha
- **Thừa kế nhiều mức (Multilevel):** Khi một lớp con dẫn xuất từ một lớp cha và bản thân lớp cha lại là con của một lớp khác.
- **Thừa kế phân cấp (Hierarchical):** Loại thừa kế này xảy ra khi một lớp cha có nhiều hơn một lớp con ở những mức khác nhau.
- **Đa thừa kế (Multiple):** Xảy ra khi một lớp con dẫn xuất từ nhiều hơn một lớp cha

Thừa kế

- **Ví dụ:** Tạo lớp Square thừa kế từ lớp Rectangle (Đơn thừa kế)

```
#Khai báo lớp Square thừa kế từ lớp Rectangle
class Square(Rectangle):
    #dùng super() để gọi đến hàm tạo của lớp cha
    def __init__(self, width = 15, color='red'):
        super().__init__(width, width)
        self.color=color

    #Thêm phương thức draw() để vẽ hình vuông theo width và color
    def draw(self):
        x = self.width
        c = self.color
        fig, ax = plt.subplots(figsize=(4,4))
        square = plt.Rectangle((0, 0), x, x, color=c)
        ax.add_patch(square)
        automin, automax = plt.xlim()
        plt.xlim(0-10, x+10)
        automin, automax = plt.ylim()
        plt.ylim(0-10, x+10)
        plt.grid(True)
        plt.show()
```

Thừa kế

```
#Khai báo đối tượng a thuộc lớp Square
a = Square(30,'yellow')
```

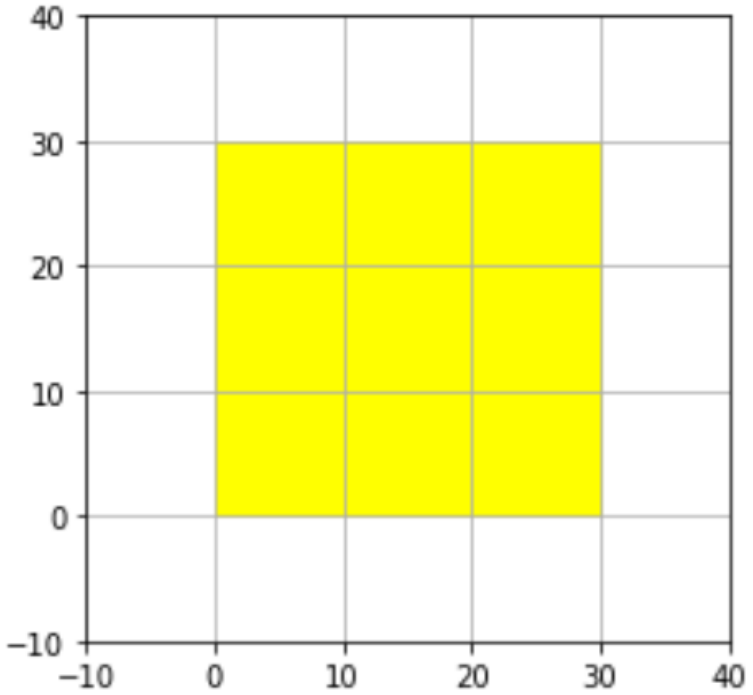
```
#Tái sử dụng các thuộc tính của lớp Rectangle
print(a.width)
print(a.height)

#Bổ sung thêm thuộc tính mới cho lớp Square
print(a.color)
```

```
30
30
yellow
```

```
#Tái sử dụng các phương thức của lớp Rectangle
print('Diện tích hình vuông:', a.getArea())
print('Chu vi hình vuông:', a.getPerimeter())
```

```
Diện tích hình vuông: 900
Chu vi hình vuông: 120
```

```
#Sử dụng phương thức mới draw của lớp Square
a.draw()
```



Thực hành

Bài 15:

1. Xây dựng lớp Person:

- **Gồm 4 Thuộc tính:**
  - họ tên (name), năm sinh (year), chiều cao (height), cân nặng (weight)
  - Giá trị mặc định của các thuộc tính là thông tin của bạn
- **Gồm 2 Phương thức:**
  - Geeting(): Hiển thị thông tin của Person
  - Bmi(): Tính toán chỉ số BMI của Person

```
1  #Tạo đối tượng p1 thuộc lớp Person
2  p1 = Person('Đặng Văn Nam', 1985, 1.65, 59)
```

```
1  #Gọi phương thức Geeting()
2  p1.Geeting()
```

```
My name is  Đặng Văn Nam
I am  36  years old. Nice to meet you!
```

```
1  #Gọi phương thức BMI
2  print('Chỉ số BMI:',p1.BMI())
```

```
Chỉ số BMI: 21.7
```

Bài 15:

2. Xây dựng lớp Vietnam, kế thừa từ lớp Person:

- **Kế thừa tất cả các thuộc tính của lớp Person và Bổ sung thêm Thuộc tính:**
  - **Dân tộc (nation), giá trịnh mặc định là "Kinh"**
- **Kế thừa phương thức BMI của lớp Person và cập nhật lại Phương thức Geeting():**

```
#Tạo một đối tượng vn1 thuộc lớp Vietnam
vn1 = Vietnam('Thanh Huyền',2000, 1.69,52,'Tày')
#Chỉnh sửa phương thức Geeting()
vn1.Geeting()
#Kế thừa lại phương thức BMI()
print('-----------------------------------')
print('Chỉ số BMI:', vn1.BMI())
```

```
Tôi tên là:  Thanh Huyền - Dân tộc: Tày
Tôi sinh năm  2000 . Rất vui được làm quyen với bạn!
-----------------------------------
Chỉ số BMI: 18.2
```

Q & A
Thank you!