1. Câu lệnh if…else

ktratamgiac.py

print ("Nhap a :")

a=int(input())

print ("Nhap b :")

b=int(input())

print ("Nhap c :")

c=int(input())

if (a+b)>c and (a+c)>b and (b+c)>a and (a>0) and (b>0) and (c>0):

if(a==b)and (b==c):

print "Tam giac deu"

elif((a\*b+b\*b==c\*c)and(a==b))or((a\*a+c\*c==b\*b)and(a==c))or((c\*c+b\*b==a\*a)and(c==b)):

print "tam giac vuong can"

elif(a==b)or(b==c)or(a==c):

print "tam giac can"

elif((a\*a==b\*b+c\*c)or(b\*b==a\*a+c\*c)or(c\*c==a\*a+b\*b)):

print "tam giac vuong"

else:

print "tam giac thuong"

else:

print "khong hop le"

1. Vòng lặp while và list

while\_list.py

list1 = ['Phan Thanh Thao']

list2 =['Female',1997]

list3=list1+list2

print("Person:")

print list3

index = 0

length = len(list3)

while index < length:

print(index, 'stands for', list3[index])

index += 1

1. CLASS
2. Class (lớp): Được định nghĩa cho một đối tượng bao gồm một tập các attributes (thuộc tính) đặc trưng cho tất cả các đối tượng của lớp. Attributes gồm: data members và methods được gọi thông qua ký hiệu dấu chấm.

Để định nghĩa một lớp bạn sử dụng từ khóa class, tiếp sau đó là tên của lớp và dấu hai chấm. Dòng đầu tiên trong thân của lớp là chuỗi (string) mô tả ngắn gọn về lớp này (Không bắt buộc), bạn có thể truy cập vào chuỗi này thông qua ClassName.\_\_doc\_\_ .

Trong thân của lớp bạn có thể khai báo các thuộc tính, phương thức (method) và các phương thức khởi tạo (Constructor).

1. Instance: Một hiện thực cụ thể của một lớp, có thể gọi là thực thể
2. Object: instance duy nhất chứa cấu trúc dữ liệu được định nghĩa bởi class.(Ví dụ: khai báo lớp HocSinh ta sẽ có object HocSinh, sau đó tạo biến nguyen\_van\_a = HocSinh() thì có thêm instance nguyen\_van\_a)
3. Methods: Hàm thực thi
4. Data members: Biến chứa dữ liệu gồm: class variable & instance variable
5. Class variable: Biến dùng chung cho tất cả các đối tượng của lớp, được định nghĩa trong lớp mà không nằm trong methods (hàm thực thi) nào cả. Các biến này không được sử dụng thường xuyên.
6. Instance variable: biến được định nghĩa bên trong methods và chỉ thuộc về các instance (đối tượng thực thể của lớp).
7. Function overloading: method định nghĩa các phép toán nhiều instances tham gia (Ví dụ cộng 2 instances, so sánh lớn hơn nhỏ hơn của 2 instances, ...).
8. Operator overloading: Phép toán cần nhiều method tham gia.
9. Inheritance: Việc chuyển giao các đặc tính của một lớp kế thừa từ lớp khác.

Thay vì định nghĩa một class mới hoàn toàn, bạn có thể kế thừa từ lớp (nhiều lớp-đa kế thừa) đã có (gọi là lớp cha). Lớp này sẽ chứa và có thể sử dụng mọi thuộc tính, phương thức của lớp cha đúng như là chúng được định nghĩa tại lớp con.

Lớp con cũng có thể ghi đè (định nghĩa lại) các thuộc tính của lớp cha.

1. Instantiation: Việc tạo ra một instance của một class.
2. FUNCTION

Hàm, là một khối code được tổ chức và có thể tái sử dụng, để thực hiện một hành động nào đó. Một số hàm đã được xây dựng sẵn trong Python, điển hình như hàm print(). Tuy nhiên cũng có thể tạo riêng cho mình một hàm với cách định nghĩa và kiểu giá trị riêng. Các hàm này được gọi là user-defined function.

Khi định nghĩa các hàm để cung cấp một tính năng nào đó, bạn cần theo các qui tắc sau:

* Từ khóa **def** được sử dụng để bắt đầu phần định nghĩa hàm. Def xác định phần bắt đầu của khối hàm.
* def được theo sau bởi ten\_ham được theo sau bởi các dấu ngoặc đơn ().
* Các tham số được truyền vào bên trong các dấu ngoặc đơn. Ở cuối là dấu hai chấm.
* Lệnh đầu tiên của hàm là tùy ý, và nó là Documentation String của một hàm đó.
* Sau đó là lệnh để được thực thi.
  + Để thực thi một hàm, bạn cần gọi hàm đó. Phần định nghĩa hàm cung cấp thông tin về tên hàm các tham số và định nghĩa những hoạt động nào được thực hiện bởi hàm đó. Để thực thi phần định nghĩa của hàm, bạn cần gọi hàm đó. Cú pháp như sau: ten\_ham(tham\_so)
  + Hàm có thể có các tham số khác nhau hoặc có thể không có bất kỳ tham số nào.
  + **Hàm có thể hoặc không thể trả về bất kỳ giá trị nào.**
  + **Phạm vi biến trong python**
    1. **Biến cục bộ**

Các biến được khai báo bên trong một thân hàm là biến cục bộ. Tức là các biến cục bộ này chỉ có thể được truy cập ở bên trong hàm mà bạn đã khai báo, không thể được truy cập ở bên ngoài thân hàm đó.

* + 1. **Biến toàn cục**

Biến được định nghĩa bên ngoài hàm được gọi là biến toàn cục. Biến toàn cục có thể được truy cập bởi tất cả các hàm ở khắp nơi trong chương trình. Do đó phạm vi của biến toàn cục là rộng nhất.

* + **Tham số hàm trong python**
    1. **Tham số bắt buộc**

Các tham số bắt buộc là các tham số được truyền tới một hàm theo một thứ tự chính xác. Ở đây, số tham số trong lời gọi hàm nên kết nối chính xác với phần định nghĩa hàm.

Ví dụ bạn định nghĩa 1 hàm có 2 tham số thì khi gọi hàm bạn phải truyền cho nó 2 tham số, nếu truyền khác 2 thì hàm sẽ báo lỗi.

* + 1. Tham số mặc định

Tham số mặc định là tham số mà cung cấp các giá trị mặc định cho các tham số được truyền trong phần định nghĩa hàm, trong trường hợp mà giá trị không được cung cấp trong lời gọi hàm.

Nghĩa là nếu trong định nghĩa hàm bạn đã set 1 giá trị mặc định cho 1 tham số thì khi gọi hàm, bạn có thể không truyền giá trị cho tham số đó và hàm sẽ lấy giá trị đã được set từ phần định nghĩa để thực hiện.

* + 1. Tham số từ khóa

Sử dụng tham số từ khóa, tham số được truyền trong lời gọi hàm được kết nối với phần định nghĩa hàm dựa trên **tên của tham số**. Với trường hợp này, vị trí của các tham số trong lời gọi hàm là tùy ý.

* + 1. Tham số thay đổi

Bạn có thể cần xử lý một hàm mà có số tham nhiều hơn là bạn đã xác định trong khi định nghĩa hàm. Những tham số này được gọi là các tham số có số tham số thay đổi (variable-length args) và không được đặt tên trong định nghĩa hàm, không giống như các tham số bắt buộc và tham số mặc định.

Cú pháp cho một hàm có số thay đổi là:

def tenham([tham\_so\_chinh\_thuc,] \*var\_args\_tuple ):

"function\_docstring"

function\_suite

return [bieu\_thuc]

Một dấu \* được đặt trước tên biến để giữ các giá trị của các tham số loại này. Tuple này vẫn là **trống** nếu không có tham số bổ sung nào được xác định trong khi gọi hàm.