Python Classes/Objects

* Python is an object oriented programming language.
* Almost everything in Python is an object, with its properties and methods.
* A Class is like an object constructor, or a "blueprint" for creating objects.

## Create a Class

* To create a class, use the keyword class:

### **Example**

Create a class named MyClass, with a property named x:

class MyClass:  
  x = 5

**Create Object**

* Now we can use the class named MyClass to create objects:

### **Example**

Create an object named p1, and print the value of x:

p1 = MyClass()   
print(p1.x)

## The \_\_init\_\_() Function

* The examples above are classes and objects in their simplest form, and are not really useful in real life applications.
* To understand the meaning of classes we have to understand the built-in \_\_init\_\_() function.
* All classes have a function called \_\_init\_\_(), which is always executed when the class is being initiated.
* Use the \_\_init\_\_() function to assign values to object properties, or other operations that are necessary to do when the object is being created:

### **Example**

Create a class named Person, use the \_\_init\_\_() function to assign values for name and age:

class Person:  
  def \_\_init\_\_(self, name, age):  
    self.name = name  
    self.age = age  
  
p1 = Person("John", 36)  
  
print(p1.name)  
print(p1.age)

## Object Methods

* Objects can also contain methods. Methods in objects are functions that belong to the object.
* Let us create a method in the Person class:

### **Example**

Insert a function that prints a greeting, and execute it on the p1 object:

class Person:  
  def \_\_init\_\_(self, name, age):  
    self.name = name  
    self.age = age  
  
  def myfunc(self):  
    print("Hello my name is " + self.name)  
  
p1 = Person("John", 36)  
p1.myfunc()

**Note:** The self parameter is a reference to the current instance of the class, and is used to access variables that belong to the class.

## The self Parameter

* The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.
* It does not have to be named self , you can call it whatever you like, but it has to be the first parameter of any function in the class:

### **Example**

Use the words mysillyobject and abc instead of self:

class Person:  
  def \_\_init\_\_(mysillyobject, name, age):  
    mysillyobject.name = name  
    mysillyobject.age = age  
  
  def myfunc(abc):  
    print("Hello my name is " + abc.name)  
  
p1 = Person("John", 36)  
p1.myfunc()

## Modify Object Properties

* You can modify properties on objects like this:

### **Example**

Set the age of p1 to 40:

p1.age = 40

## Delete Object Properties

* You can delete properties on objects by using the del keyword:

### **Example**

Delete the age property from the p1 object:

del p1.age

## The pass Statement

* class definitions cannot be empty, but if you for some reason have a class definition with no content, put in the pass statement to avoid getting an error.

**Example**

class Person:  
  pass