**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

**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()

**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

**Delete Objects**

You can delete objects by using the del keyword:

Example

Delete the p1 object:

del p1

**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

**Python Inheritance**

**Python Inheritance**

Inheritance allows us to define a class that inherits all the methods and properties from another class.

Parent class is the class being inherited from, also called base class.

Child class is the class that inherits from another class, also called derived class.

Create a Parent Class

Any class can be a parent class, so the syntax is the same as creating any other class:

Example

Create a class named Person, with firstname and lastname properties, and a printname method:

**class Person:**

**def \_\_init\_\_(self, fname, lname):**

**self.firstname = fname**

**self.lastname = lname**

**def printname(self):**

**print(self.firstname, self.lastname)**

#Use the Person class to create an object, and then execute the printname method:

**x = Person("John", "Doe")**

**x.printname()**

**Create a Child Class**

To create a class that inherits the functionality from another class, send the parent class as a parameter when creating the child class:

Example

Create a class named Student, which will inherit the properties and methods from the Person class:

**class Person:**

**def \_\_init\_\_(self, fname, lname):**

**self.firstname = fname**

**self.lastname = lname**

**def printname(self):**

**print(self.firstname, self.lastname)**

**class Student(Person):**

**pass**

**x = Student("Mike", "Olsen")**

**x.printname()**

**Add the \_\_init\_\_() Function**

So far we have created a child class that inherits the properties and methods from its parent.

We want to add the \_\_init\_\_() function to the child class (instead of the pass keyword).

Note: The \_\_init\_\_() function is called automatically every time the class is being used to create a new object.

Example

**class Person:**

**def \_\_init\_\_(self, fname, lname):**

**self.firstname = fname**

**self.lastname = lname**

**def printname(self):**

**print(self.firstname, self.lastname)**

**class Student(Person):**

**def \_\_init\_\_(self, fname, lname):**

**Person.\_\_init\_\_(self, fname, lname)**

**x = Student("Mike", "Olsen")**

**x.printname())**

Now we have successfully added the \_\_init\_\_() function, and kept the inheritance of the parent class, and we are ready to add functionality in the \_\_init\_\_() function.

**Use the super() Function**

Python also has a super() function that will make the child class inherit all the methods and properties from its parent:

Example

**class Person:**

**def \_\_init\_\_(self, fname, lname):**

**self.firstname = fname**

**self.lastname = lname**

**def printname(self):**

**print(self.firstname, self.lastname)**

**class Student(Person):**

**def \_\_init\_\_(self, fname, lname):**

**super().\_\_init\_\_(fname, lname)**

**x = Student("Mike", "Olsen")**

**x.printname()**

Example

Add a property called graduation year to the Student class:

class Student(Person):

**class Person:**

**def \_\_init\_\_(self, fname, lname):**

**self.firstname = fname**

**self.lastname = lname**

**def printname(self):**

**print(self.firstname, self.lastname)**

**class Student(Person):**

**def \_\_init\_\_(self, fname, lname, year):**

**super().\_\_init\_\_(fname, lname)**

**self.graduationyear = year**

**x = Student("Mike", "Olsen", 2019)**

**print(x.graduationyear)**

In the example below, the year 2019 should be a variable, and passed into the Student class when creating student objects. To do so, add another parameter in the \_\_init\_\_() function:

Example

Add a year parameter, and pass the correct year when creating objects:

**class Student(Person):**

**def \_\_init\_\_(self, fname, lname, year):**

**super().\_\_init\_\_(fname, lname)**

**self.graduationyear = year**

**x = Student("Mike", "Olsen", 2019)**

**Add Methods**

Example

Add a method called welcome to the Student class:

**class Person:**

**def \_\_init\_\_(self, fname, lname):**

**self.firstname = fname**

**self.lastname = lname**

**def printname(self):**

**print(self.firstname, self.lastname)**

**class Student(Person):**

**def \_\_init\_\_(self, fname, lname, year):**

**super().\_\_init\_\_(fname, lname)**

**self.graduationyear = year**

**def welcome(self):**

**print("Welcome", self.firstname, self.lastname, "to the class of", self.graduationyear)**

**x = Student("Mike", "Olsen", 2019)**

**x.welcome()**

If you add a method in the child class with the same name as a function in the parent class, the inheritance of the parent method will be overridden.