# -\*- coding: utf-8 -\*-

"""

Created on Mon Nov 25 09:35:09 2019

@author: sugandha.sharma

"""

"""Python is an object oriented programming language.

Almost everything in Python is an object,

with its properties and methods."""

"""Create a class named MyClass1, with a property

named x:"""

class My\_class1 :

x=5

print(My\_class1)

"""Create Object:

Now we can use the class named MyClass2 to create

objects:

Example:

Create an object named obj1, and print the value of x:"""

class MyClass2 :

x=10

obj1 = MyClass2()

print(obj1.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 ,age and city:

"""

class Person:

def \_\_init\_\_(obj1, name, age,city):

obj1.name = name

obj1.age = age

obj1.city=city

p1 = Person("Aditi", 30,"Dehradun")

p2=Person("Aditya ", 31,"Delhi")

print(p1.name)

print(p1.age)

print(p1.city)

print(p2.name)

print(p2.age)

print(p2.city)

"""Note: The \_\_init\_\_() function is called automatically

every time the class is being used to create a new

object."""

"""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\_\_(obj2, name, age):

obj2.name = name

obj2.age = age

def myfunc(obj2):

print("Hello my name is " + obj2.name)

p1 = Person("Aditi", 30)

p1.myfunc()

p2 = Person("Aditya", 30)

p2.myfunc()

"""class Person2:

def \_\_init\_\_(obj2, name, age):

obj2.name = name

obj2.age = age

p2 = Person2("Aditya", 31)

p2.myfunc()

"""

"""Note: The obj1 parameter is a reference

to the current instance of the class, and is

used to access variables that belong to the class."""

"""The obj1 Parameter:

The obj1 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 obj1 ,

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 humanbeing and actor instead of obj1:

"""

class Person:

def \_\_init\_\_(humanbeing, name, age):

humanbeing.name = name

humanbeing.age = age

def myfunc(actor):

print("Hello my name is " + actor.name)

p1 = Person("Salman", 50)

p1.myfunc()

"""

Modify Object Properties

You can modify properties on objects like this:

Example

Set the age of p1 to 40:

"""

class Person:

def \_\_init\_\_(humanbeing, name, age):

humanbeing.name = name

humanbeing.age = age

def myfunc(actor):

print("Hello my name is " + actor.name)

p1 = Person("Salman", 50)

p1.myfunc()

print(p1.age)

p1.age = 40

print(p1.age)

"""

Delete Object Properties

You can delete properties on objects by using the

del keyword:

Example:

Delete the age property from the p1 object:"""

class Person:

def \_\_init\_\_(humanbeing, name, age):

humanbeing.name = name

humanbeing.age = age

def myfunc(actor):

print("Hello my name is " + actor.name)

p1 = Person("Salman", 50)

p1.myfunc()

#print(p1.age)

del p1.age

print(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:

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("Amar", "Shah")

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 Student(Person):

pass

"""Note: Use the pass keyword when you do not want

to add any other properties or methods to the class.

Now the Student class has the same properties and

methods as the Person class.

Example:

Use the Student class to create an object,

and then execute the printname method:"""

x = Student("Amit", "Shah")

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:

Add the \_\_init\_\_() function to the Student class:"""

class Student(Person):

def \_\_init\_\_(obj, fname, lname):

#add properties etc.

"""When you add the \_\_init\_\_() function, the

child class will no longer inherit the parent's

\_\_init\_\_() function.

Note: The child's \_\_init\_\_() function overrides the

inheritance of the parent's \_\_init\_\_() function.

To keep the inheritance of the parent's \_\_init\_\_()

function, add a call to the parent's \_\_init\_\_()

function:

Example:

"""

class Student(Person):

def \_\_init\_\_(obj, fname, lname):

Person.\_\_init\_\_(obj, fname, lname)

"""

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 Student(Person):

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

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

"""

By using the super() function, you do not have

to use the name of the parent element, it will

automatically inherit the methods and properties

from its parent.

Add Properties:

Example:

Add a property called graduationyear to the Student

class:

"""

class Student(Person):

def \_\_init\_\_(obj, fname, lname):

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

obj.graduationyear = 2019

"""

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 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)

"""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."""