Object Orientated Programming in Python

A class is a code template for creating objects. A class can be initialised by using the keyword class. For example:

Class MyClass:

X = 5

Using classes in python does have multiple advantages. One of these advantages is that it can provide the ability to reuse the code, which makes the program more efficient. Classes can also make it easier for the developer to maintain the data members and methods together and structured in one place.

However, classes also have some disadvantages. One of these disadvantages would be that it may also lack structure, rather than provide structure. It may be harder to look at the code and decide which members the class will have etc. Classes are also less straightforward to get such an object and iterate over the members.

Examples of classes:

# define a class

class Bike:

name = ""

gear = 0

# create object of class

bike1 = Bike()

# access attributes and assign new values

bike1.gear = 11

bike1.name = "Mountain Bike"

print(f"Name: {bike1.name}, Gears: {bike1.gear} ")

Output:

Name: Mountain Bike, Gears: 11

# define a class

class Employee:

# define an attribute

employee\_id = 0

# create two objects of the Employee class

employee1 = Employee()

employee2 = Employee()

# access attributes using employee1

employee1.employeeID = 1001

print(f"Employee ID: {employee1.employeeID}")

# access attributes using employee2

employee2.employeeID = 1002

print(f"Employee ID: {employee2.employeeID}")

Output:

Employee ID: 1001

Employee ID: 1002

The \_init\_() function is usually executed when the class is being initiated. The \_init\_() function is used to assign values to object properties or other operations that are necessary to do when the object is being created. For example;

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

This example creates the class Person, and uses the \_init\_() function to assign values for name and age.

The \_str\_() function controls what should be returned when the class object is represented as a string. If the \_str\_() function is not set, the string representation of the object is returned.

For example:

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

The return f”{self.name}({self.age)” line returns the name and age. This will output the following:

John(36)