Python Binary File Handling

A **binary** file contains a sequence or a collection of bytes which are not in a human readable format. For example, files with .exe, .mp3, etc extension. It represents custom data. A binary file contains information in the same format in which the information is held in the memory ie in raw format. There is no delimiter in the binary file and also no translation occurs. As a result, binary files are faster and easier for a program to read and write the text files. Binary file read and write operation is performed using **pickle module** of python.

Pickle module has two special functions: **dump(contents,filename)** for writing to the binary file and **load(file name)** function to read object from binary file.

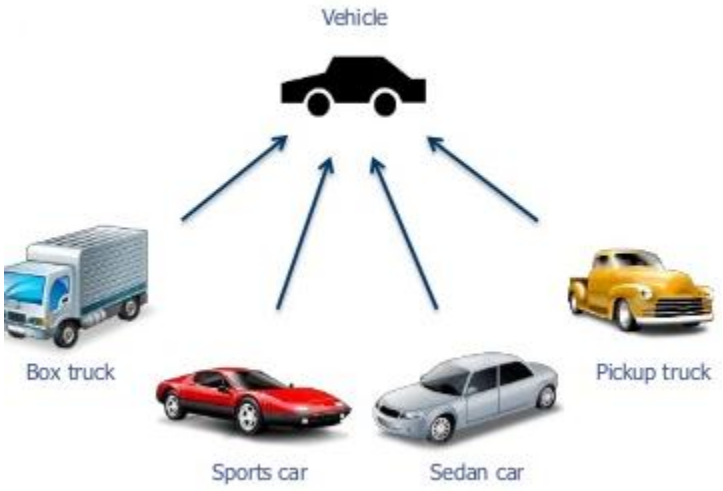
Classes and Object

Classes and Objects are basic **concepts** of **Object** Oriented Programming which revolve around the real life entities. **Class**. A **class** is a user defined blueprint or prototype from which **objects** are created. It represents the set of properties or methods that are common to all **objects** of one type

A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by its class) for modifying its state.

To understand the need for creating a class let's consider an example, let's say you wanted to track the number of dogs which may have different attributes like breed, age. If a list is used, the first element could be the dog's breed while the second element could represent its age. Let's suppose there are 100 different dogs, then how would you know which element is supposed to be which? What if you wanted to add other properties to these dogs? This lacks organization and it's the exact need for classes.

Class creates a user-defined data structure, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.



Some points on Python class:

- Classes are created by keyword class.
- Attributes are the variables that belong to class.
- Attributes are always public and can be accessed using dot (.) operator.

  Eg.: Myclass.Myattribute

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
```

An Object is an instance of a Class. A class is like a blueprint while an instance is a copy of the class with *actual values*. It's not an idea anymore, it's an actual dog, like a dog of breed pug who's seven years old. You can have many dogs to create many different instances, but without the class as a guide, you would be lost, not knowing what information is required.

An object consists of :

- **State :** It is represented by attributes of an object. It also reflects the properties of an object.
- **Behavior :** It is represented by methods of an object. It also reflects the response of an object with other objects.
- **Identity :** It gives a unique name to an object and enables one object to interact with other objects.

When an object of a class is created, the class is said to be instantiated. All the instances share the attributes and the behavior of the class. But the values of those attributes, i.e. the state are unique for each object. A single class may have any number of instances.

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.
The __init__ method is similar to constructors in C++ and Java. Constructors are used to initialize the object's state. Like methods, a constructor also contains collection of statements(i.e. instructions) that are executed at time of Object creation. It is run as soon as an object of a class is instantiated. The method is useful to do any initialization you want to do with your object.

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)
```

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__(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:
- Class methods must have an extra first parameter in method definition. We do not give a value for this parameter when we call the method, Python provides it.
- If we have a method which takes no arguments, then we still have to have one argument.
- This is similar to this pointer in C++ and this reference in Java.

When we call a method of this object as `myobject.method(arg1, arg2)`, this is automatically converted by Python into `MyClass.method(myobject, arg1, arg2)` – this is all the special self is about.

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
```

Writing and Reading object of Class Student on/from binary file student.dat

```
class Student:
 def __init__(self):
  self.name=""
  self.age=0
 def getinfo(self):
```

Writing and Reading object of Class Product on/from binary file product.dat

```
class Student:
 def __init__(self):
   self.name=""
   self.age=0
 def getinfo(self):

   self.name=input("Enter the name of student: ")
   self.age=input("Enter the age of student: ")

 def show(self):

   print("Student name is " + self.name)
   print("Student age is ", self.age)

import pickle
std=Student()
fileobj=open("student.dat",'wb')

while True:
 std.getinfo()
 pickle.dump(std,fileobj)
 print("Data written on the file....\n")
 ch=input("Want it enter more record?y/n: ")
 if ch.upper()=="Y":
   continue
 else:
   break

fileobj.close()
print("\n----------------Reading file----------------\n")
fileobj=open("student.dat",'rb')
try:
 while True:
   std=pickle.load(fileobj)
   std.show()
   print("\n----------------------------------\n")
except:
 fileobj.close()
```