Python Objects and Classes

An object is simply a collection of data (variables) and methods (functions) that act on those data. Similarly, a class is a blueprint for that object.

We can think of class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows etc. Based on these descriptions we build the house. House is the object.

As many houses can be made from a house's blueprint, we can create many objects from a class. An object is also called an instance of a class and the process of creating this object is called **instantiation**.

Defining a Class in Python

Like function definitions begin with the def keyword in Python, class definitions begin with a class keyword.
The first string inside the class is called docstring and has a brief description about the class. Although not mandatory, this is highly recommended.

Here is a simple class definition.

```
class MyNewClass:
    '''This is a docstring. I have created a new class'''
    pass
```

A class creates a new local namespace where all its attributes are defined. Attributes may be data or functions.
There are also special attributes in it that begins with double underscores __. For example, `__doc__` gives us the docstring of that class.
As soon as we define a class, a new class object is created with the same name. This class object allows us to access the different attributes as well as to instantiate new objects of that class.

```
class Person:
    "This is a person class"
    age = 10

    def greet(self):
        print('Hello')


# Output: 10
print(Person.age)

# Output: <function Person.greet>
print(Person.greet)

# Output: 'This is my second class'
print(Person.__doc__)
```

```
lass Person:
    "This is a person class"
    age = 10

    def greet(self):
        print('Hello')


# create a new object of Person class
harry = Person()

# Output: <function Person.greet>
print(Person.greet)

# Output: <bound method Person.greet of <__main__.Person object>>
print(harry.greet)

# Calling object's greet() method
# Output: Hello
harry.greet()
```

Constructors in Python

Class functions that begin with double underscore `__` are called special functions as they have special meaning.
Of one particular interest is the `__init__()` function. This special function gets called whenever a new object of that class is instantiated.
This type of function is also called constructors in Object Oriented Programming (OOP). We normally use it to initialize all the variables.

```
import math
from math import sqrt

class Complex(object):

    def __init__(self, real, imag=0.0):
        self.real = real
        self.imag = imag
        # Formats our results
        print(self.real + self.imag)

    def __add__(self, other):
        print('\nSum:')
        return Complex(self.real + other.real, self.imag + other.imag)

    def __sub__(self, other):
        print('\nDifference:')
        return Complex(self.real - other.real, self.imag - other.imag)

    def __mul__(self, other):
        print('\nProduct:')
        return Complex((self.real * other.real) - (self.imag * other.imag),
            (self.imag * other.real) + (self.real * other.imag))

    def __truediv__(self, other):
        print('\nQuotient:')
        r = (other.real**2 + other.imag**2)
        return Complex((self.real*other.real - self.imag*other.imag)/r,
            (self.imag*other.real + self.real*other.imag)/r)

    def __abs__(self):
        print('\nAbsolute Value:')
        new = (self.real**2 + (self.imag**2)*-1)
        return Complex(sqrt(new.real))


i = Complex(2, 10j)
k = Complex(3, 5j)

# Add
i + k
# Subtract
i - k
# Multiply
i * k
# Divide
i / k
# Absolute value
abs(i)
abs(k)
```

bank account

```
# Python program to create Bankaccount class

# with both a deposit() and a withdraw() function

class Bank_Account:

    def __init__(self):

        self.balance=0

        print("Hello!!! Welcome to the Deposit & Withdrawal Machine")



    def deposit(self):

        amount=float(input("Enter amount to be Deposited: "))

        self.balance += amount

        print("\n Amount Deposited:",amount)



    def withdraw(self):

        amount = float(input("Enter amount to be Withdrawn: "))
```

```
                if self.balance>=amount:

                        self.balance-=amount

                        print("\n You Withdrew:", amount)

                else:

                        print("\n Insufficient balance ")


        def display(self):

                print("\n Net Available Balance=",self.balance)


# Driver code


# creating an object of class

s = Bank_Account()


# Calling functions with that class object

s.deposit()

s.withdraw()

s.display()
```