OOPS

Object Oriented Programming

Class

A class is a blueprint for the object.
Example:

```
class Parrot:

    # class attribute
    species = "bird"

    # instance attribute
    definit(self, name, age):
        self.name = name
        self.age = age
```

Object

An object (instance) is an instantiation of a class. When class is defined, only the description for the object is defined. Therefore, no memory or storage is allocated.

Methods

Methods are functions defined inside the body of a class. They are used to define the behaviors of an object.
Example:

```
class Parrot:
    # instance attributes
    definit(self, name, age):
        self.name = name
        self.age = age

    # instance method
    def sing(self, song):
        return "{} sings {}".format(self.name, song)

    def dance(self):
        return "{} is now dancing".format(self.name)
```

Inheritance

Inheritance is a way of creating new class for using details of existing class without modifying it. The newly formed class is a derived class (or child class). Similarly,the existing class is a base class (or parent class).

```
# parent class
class Bird:

   def __init__(self): print("Bird is ready")

   def whoisThis(self): print("Bird")

   def swim(self): print("Swim faster")

# child class
class Penguin(Bird):

   def __init__( s e l f ) :
      # call super() function
      super().__init__() print("Penguin is ready")

   def whoisThis(self): print("Penguin")

   def run(self): print("Run faster")
```

Encapsulation

Using OOP in Python, we can restrict access to methods and variables. This prevent data from direct modification which is called encapsulation. In Python, we denote private attribute using underscore as prefix i.e single " _ " or double "".

Example:

```
class Computer:

   def __init__(self):
      self.__maxprice = 900

   def sell(self):
      print("Selling Price: {}".format(self.__maxprice))


def setMaxPrice(self, price): self.__maxprice = price
```

Polymorphism

Polymorphism is an ability (in OOP) to use common interface for multiple form (data types).
Suppose, we need to color a shape, there are multiple shape option (rectangle, square, circle). However we could use same method to color any shape. This concept is called Polymorphism.
Example:

```
class Parrot:

    def fly(self): print("Parrot can fly")

    def swim(self): print("Parrot can't swim")

class Penguin:

    def fly(self): print("Penguin can't fly")

    def swim(self): print("Penguin can swim")

# common interface
def flying_test(bird): bird.fly()
```

Useful Links:

- https://www.javatpoint.com/python-oops-concepts
- https://realpython.com/python3-object-oriented-programming/