AI LAB WEEK 2

Dr. Mubashir Ahmad

PYTHON OOPS CONCEPTS

In Python, object-oriented Programming (OOPs) is a programming paradigm that uses objects and classes in programming. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming. The main concept of OOPs is to bind the data and the functions that work on that together as a single unit so that no other part of the code can access this data.

OOPS CONCEPTS IN PYTHON

- Class
- Objects
- Polymorphism
- Encapsulation
- Inheritance
- Data Abstraction

PYTHON CLASS

A class is a collection of objects. A class contains the blueprints or the prototype from which the objects are being created. It is a logical entity that contains some attributes and methods.

To understand the need for creating a class let's consider an example, let's say you wanted to track the number of dogs that may have different attributes like breed, and 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.

SOME POINTS ON PYTHON CLASS:

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

SOME POINTS ON PYTHON CLASS:

Class Definition Syntax:

```
class ClassName:
   # Statement-1
   .
   .
   .
   # Statement-N
```

CREATING AN EMPTY CLASS IN PYTHON

```
# Python3 program to
# demonstrate defining
# a class

class Dog:
    pass
```

PYTHON OBJECTS

The object is an entity that has a state and behavior associated with it. It may be any real-world object like a mouse, keyboard, chair, table, pen, etc. Integers, strings, floating-point numbers, even arrays, and dictionaries, are all objects. More specifically, any single integer or any single string is an object. The number 12 is an object, the string “Hello, world” is an object, a list is an object that can hold other objects, and so on.

AN OBJECT CONSISTS OF:

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

CREATING AN OBJECT

This will create an object named obj of the class Dog defined above. Before diving deep into objects and classes let us understand some basic keywords that will we used while working with objects and classes.

obj = Dog()

CREATE A CLASS AND OBJECT

```
class MyClass:
  x = 5


p1 = MyClass()
print(p1.x)
```

THE __INIT__() FUNCTION

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__() FUNCTION

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)
```

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:

```
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()
```

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:

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.

PYTHON INHERITANCE

```
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("John", "Doe")
x.printname()
```

```
class Student(Person):
  pass
```

Use the Student class to create an object, and then execute the printname method:

```
x = Student("Mike", "Olsen")
x.printname()
```

Note: Use the `pass` keyword when you do not want to add any other properties or methods to the class.

PYTHON POLYMORPHISM

The word "polymorphism" means "many forms", and in programming it refers to methods/functions/operators with the same name that can be executed on many objects or classes.

FUNCTION POLYMORPHISM

An example of a Python function that can be used on different objects is the len() function.

String

For strings len() returns the number of characters:

```
x = "Hello World!"

print(len(x))
```

FUNCTION POLYMORPHISM

Tuple

For tuples len() returns the number of items in the tuple:

```
mytuple = ("apple", "banana", "cherry")

print(len(mytuple))
```

Dictionary

For dictionaries len() returns the number of key/value pairs in the dictionary:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

print(len(thisdict))
```

CLASS POLYMORPHISM

Polymorphism is often used in Class methods, where we can have multiple classes with the same method name.

For example, say we have three classes: Car, Boat, and Plane, and they all have a method called move():

```python
class Car:
  def __init__(self, brand, model):
    self.brand = brand
    self.model = model

  def move(self):
    print("Drive!")

class Boat:
  def __init__(self, brand, model):
    self.brand = brand
    self.model = model

  def move(self):
    print("Sail!")
```

```python
class Plane:
  def __init__(self, brand, model):
    self.brand = brand
    self.model = model

  def move(self):
    print("Fly!")

car1 = Car("Ford", "Mustang")       #Create a Car class
boat1 = Boat("Ibiza", "Touring 20") #Create a Boat class
plane1 = Plane("Boeing", "747")     #Create a Plane class

for x in (car1, boat1, plane1):
  x.move()
```

NUMPY INTRODUCTION

What is NumPy?

NumPy is a Python library used for working with arrays.

It also has functions for working in domain of linear algebra, fourier transform, and matrices.

NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.

NumPy stands for Numerical Python.

NUMPY INTRODUCTION

Why Use NumPy?

In Python we have lists that serve the purpose of arrays, but they are slow to process.

NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

Arrays are very frequently used in data science, where speed and resources are very important.

Data Science: is a branch of computer science where we study how to store, use and analyze data for deriving information from it.

NUMPY INTRODUCTION

Why is NumPy Faster Than Lists?

NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.

This behavior is called locality of reference in computer science.

This is the main reason why NumPy is faster than lists. Also it is optimized to work with latest CPU architectures.

NUMPY INTRODUCTION

Which Language is NumPy written in?

NumPy is a Python library and is written partially in Python, but most of the parts that require fast computation are written in C or C++.

INSTALLATION OF NUMPY

If you have Python and PIP already installed on a system, then installation of NumPy is very easy.

Install it using this command:

C:\Users*Your Name*>pip install numpy

Import NumPy

Once NumPy is installed, import it in your applications by adding the import keyword:

```
import numpy
```

IMPORT NUMPY

```
import numpy
arr = numpy.array([1, 2, 3, 4, 5])
print(arr)
```

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr)
```

CREATE A NUMPY NDARRAY OBJECT

NumPy is used to work with arrays. The array object in NumPy is called ndarray. We can create a NumPy ndarray object by using the array() function.

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])

print(arr)

print(type(arr))
```

Example

Create a 0-D array with value 42

```
import numpy as np

arr = np.array(42)

print(arr)
```

ARRAY IN NUMPY

Example

Create a 1-D array containing the values 1,2,3,4,5:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])

print(arr)
```

2D ARRAY

Example

Create a 2-D array containing two arrays with the values 1,2,3 and 4,5,6:

```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

print(arr)
```

3-D ARRAYS

An array that has 2-D arrays (matrices) as its elements is called 3-D array.

These are often used to represent a 3rd order tensor.

```
import numpy as np

arr = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(arr)
```

CHECK NUMBER OF DIMENSIONS?

```
import numpy as np

a = np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5, 6]])
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)
```

ACCESS ARRAY ELEMENTS

Array indexing is the same as accessing an array element.

You can access an array element by referring to its index number.

The indexes in NumPy arrays start with 0, meaning that the first element has index 0, and the second has index 1 etc.

```
import numpy as np

arr = np.array([1, 2, 3, 4])

print(arr[0])
```

ACCESS ARRAY ELEMENTS

Get the second element from the following array.

```
import numpy as np

arr = np.array([1, 2, 3, 4])

print(arr[1])
```

ACCESS 2-D ARRAYS

To access elements from 2-D arrays we can use comma separated integers representing the dimension and the index of the element.

Think of 2-D arrays like a table with rows and columns, where the dimension represents the row and the index represents the column.

```
import numpy as np

arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])

print('2nd element on 1st row: ', arr[0, 1])
```

ACCESS 3-D ARRAYS

To access elements from 3-D arrays we can use comma separated integers representing the dimensions and the index of the element.

```
import numpy as np

arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

print(arr[0, 1, 2])
```

SEARCHING ARRAYS

You can search an array for a certain value, and return the indexes that get a match.

To search an array, use the where() method.

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 4, 4])

x = np.where(arr == 4)

print(x)
```

The example above will return a tuple: (array([3, 5, 6],)

Which means that the value 4 is present at index 3, 5, and 6.

WHAT IS MATPLOTLIB?

Matplotlib is a low level graph plotting library in python that serves as a visualization utility.

Matplotlib was created by John D. Hunter.

Matplotlib is open source and we can use it freely.

Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

MATPLOTLIB GETTING STARTED

Installation of Matplotlib

If you have Python and PIP already installed on a system, then installation of Matplotlib is very easy.

Install it using this command:

C:\Users*Your Name*>pip install matplotlib

Import Matplotlib

```
import matplotlib
```

PYPLOT

Pyplot

Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias:

```
import matplotlib.pyplot as plt
```

Draw a line in a diagram from position (0,0) to position (6,250):

```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([0, 6])
ypoints = np.array([0, 250])

plt.plot(xpoints, ypoints)
plt.show()
```

EXP2

```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([1, 2, 6, 8])
ypoints = np.array([3, 8, 1, 10])

plt.plot(xpoints, ypoints)
plt.show()
```