MODULE -3 ADVANCE PYTHON PROGRAMMING

[1] What is File function in python? What is keywords to create and write file.

- File is a named location on disk to store related information. It is used to permanently store data in a non-volatile memory (e.g. hard disk). When we want to read from or write to a file we need to open it first. When we are done, it needs to be closed, so that resources that are tied with the file are freed.
- Hence, in Python, a file operation takes place in the following order.

- Open a file
- Read or write (perform operation)
- Close the file

In Python, you use the open() function with one of the following options – "x" or "w" – to create a new file:

- "x" – Create: this command will create a new file if and only if there is no file already in existence with that name or else it will return an error.
- Example of creating a file in Python using the "x" command:

```
creating a text file with the command function "x"
f = open("myfile.txt", "x")
```

- "w" – Write: this command will create a new text file whether or not there is a file in the memory with the new specified name. It does not return an error if it finds an existing file with the same name – instead it will overwrite the existing file.
- Example of how to create a file with the "w" command:

```
creating a text file with the command function "w"
f = open("myfile.txt", "w")
```

- #This "w" command can also be used create a new file but unlike the the "x" command the "w" command will overwrite any existing file found with the same file name.
- There are two methods of writing to a file in Python, which are:

The write() method:

- This function inserts the string into the text file on a single line. Based on the file we have created above, the below line of code will insert the string into the created text file, which is "myfile.txt."

```
file.write("Hello There\n")
```

The writelines() method:

- This function inserts multiple strings at the same time. A list of string elements is created, and each string is then added to the text file.

- Using the previously created file above, the below line of code will insert the string into the created text file, which is "myfile.txt."

```
f.writelines(["Hello World ", "You are welcome to Fcc\n"])
```

Example:

```
#This program shows how to write data in a text file.

file = open("myfile.txt","w")
L = ["This is Lagos \n","This is Python \n","This is Fcc \n"]

file.write("Hello There \n")
file.writelines(L)
file.close()

# Use the close() to change file access modes
```

[2] Explain Exception handling? What is an Error in Python?

Errors are the problems in a program due to which the program will stop the execution. On the other hand, exceptions are raised when some internal events occur which changes the normal flow of the program. Two types of Error occurs in python.

1. Syntax errors

   When the proper syntax of the language is not followed then a syntax error is thrown.

2. Logical errors (Exceptions)

   When in the runtime an error that occurs after passing the syntax test is called exception or logical type. For example, when we divide any number by zero then the ZeroDivisionError exception is raised, or when we import a module that does not exist then ImportError is raised.

Exception Handling:

Try and Except Statement – Catching Exceptions

- Try and except statements are used to catch and handle exceptions in Python. Statements that can raise exceptions are kept inside the try clause and the statements that handle the exception are written inside except clause.

```
try:
    # statement(s)
except IndexError:
    # statement(s)
except ValueError:
    # statement(s)
```

Try with Else Clause

- In Python, you can also use the else clause on the try-except block which must be present after all the except clauses. The code enters the else block only if the try clause does not raise an exception.

Finally Keyword in Python

- Python provides a keyword finally, which is always executed after the try and except blocks. The final block always executes after the normal termination of the try block or after the try block terminates due to some exception.

Syntax:

```
try:
    # Some Code....
except:
    # optional block
    # Handling of exception (if required)
else:
    # execute if no exception
finally:
    # Some code .....(always executed)
```

[3] How many except statements can a try-except block have? Name Some built-in exception classes:

- There has to be at least one except statement.

Different types of exceptions in python:

In Python, there are several built-in exceptions that can be raised when an error occurs during the execution of a program. Here are some of the most common types of exceptions in Python:

- SyntaxError: This exception is raised when the interpreter encounters a syntax error in the code, such as a misspelled keyword, a missing colon, or an unbalanced parenthesis.
- TypeError: This exception is raised when an operation or function is applied to an object of the wrong type, such as adding a string to an integer.
- NameError: This exception is raised when a variable or function name is not found in the current scope.
- IndexError: This exception is raised when an index is out of range for a list, tuple, or other sequence types.
- KeyError: This exception is raised when a key is not found in a dictionary.
- ValueError: This exception is raised when a function or method is called with an invalid argument or input, such as trying to convert a string to an integer when the string does not represent a valid integer.
- AttributeError: This exception is raised when an attribute or method is not found on an object, such as trying to access a non-existent attribute of a class instance.
- IOError: This exception is raised when an I/O operation, such as reading or writing a file, fails due to an input/output error.
- ZeroDivisionError: This exception is raised when an attempt is made to divide a number by zero.
- ImportError: This exception is raised when an import statement fails to find or load a module.

[4] When will the else part of try-except-else be executed?

The else part is executed when no exception occurs.

[5] Can one block of except statements handle multiple exception?

Yes, a single block of except statements in Python can handle multiple exceptions. This feature allows you to handle different types of exceptions using a single block of code.

[6] When is the finally block executed?

- The finally block will be executed no matter if the try block raises an error or not. This can be useful to close objects and clean up resources.
- Finally block is always executed after leaving the try statement. In case if some exception was not handled by except block, it is re-raised after execution of finally block. Finally block is used to deallocate the system resources.

Example:

```
try:
  x > 3
except:
  print("Something went wrong")
else:
  print("Nothing went wrong")
finally:
  print("The try...except block is finished")
```

Output:

```
Something went wrong

The try...except block is finished
```

[7] What happens when „1"== 1 is executed?

We get a false. It simply evaluates to False and does not raise any exception.

[8] How Do You Handle Exceptions With Try/Except/Finally In Python? Explain with coding snippets.

Exception handling with try, except, else, and finally

- **Try**: This block will test the excepted error to occur
- **Except**: Here you can handle the error
- **Else**: If there is no exception then this block will be executed
- **Finally**: Finally block always gets executed either exception is generated or not

```
try:
    # Some Code....
except:
    # optional block
    # Handling of exception (if required)
else:
    # execute if no exception
finally:
   # Some code .....(always executed)
```

- First try clause is executed i.e. the code between try and except clause.
- If there is no exception, then only try clause will run, except clause will not get executed.
- If any exception occurs, the try clause will be skipped and except clause will run.
- If any exception occurs, but the except clause within the code doesn't handle it, it is passed on to the outer try statements. If the exception is left unhandled, then the execution stops.
- A try statement can have more than one except clause.

[9] What are oops concepts? Is multiple inheritance supported in java?

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.

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.

Python __init__ Method

- The __init__ method is similar to constructors in C++ and Java. 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. Now let us define a class and create some objects using the self and __init__ method.

Python Inheritance

- Inheritance is the capability of one class to derive or inherit the properties from another class. The class that derives properties is called the derived class or child class and the class from which the properties are being derived is called the base class or parent class.

Python Polymorphism

- Polymorphism simply means having many forms. For example, we need to determine if the given species of birds fly or not, using polymorphism we can do this using a single function.

Python Encapsulation

- Encapsulation is one of the fundamental concepts in object-oriented programming (OOP). It describes the idea of wrapping data and the methods that work on data within one unit. This puts restrictions on accessing variables and methods directly and can prevent the accidental modification of data. To prevent accidental change, an object's variable can only be changed by an object's method. Those types of variables are known as private variables.

Data Abstraction

- It hides unnecessary code details from the user. Also, when we do not want to give out sensitive parts of our code implementation and this is where data abstraction came.
- Data Abstraction in Python can be achieved by creating abstract classes.

Is multiple inheritance supported in java?

- There are various types of inheritance in general like single-level, multi-level, multiple and hybrid. Unlike other Object-oriented programming languages, Java doesn't support multiple inheritance.

- Consider a situation where a base class is used to create tow sub-classes Parent1 and Parent2. The base class has a method called sum() which will be inherited by the two sub-classes. If we use the concept of multiple inheritance further to create another class Child that is derived from Parent1 and Parent2.

- The class Child will inherit the sum() method from both Parent1 and Parent2. Now when we create a Child class object and call the function sum(), the compiler gets confused about whether to call the sum() method of Parent1 or Parent2. This is the ambiguity problem faced by the compiler hence multiple inheritance is not supported in Java.

[10] How to Define a Class in Python? What Is Self? Give an Example Of A Python Class.

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.

- 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

Class Definition Syntax:

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

The Python self

1. Class methods must have an extra first parameter in the method definition. We do not give a value for this parameter when we call the method, Python provides it
2. If we have a method that takes no arguments, then we still have to have one argument.
3. 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.

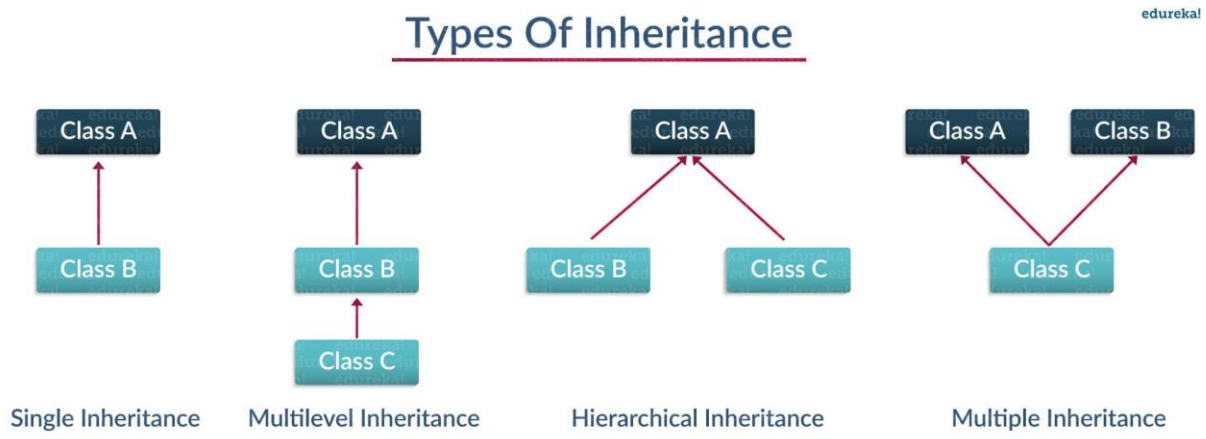
Example:

```
Class car:
        def_init_(self,model):
                Self.model = model
        def car_info(self):
                Print("model:", self.model)
```

[11] Explain Inheritance in Python with an example? What is init? Or What Is A Constructor In Python?

Inheritance in Python

In the above article, we have created two classes i.e. Person (parent class) and Employee (Child Class). The Employee class inherits from the Person class. We can use the methods of the person class through the employee class as seen in the display function in the above code. A child class can also modify the behavior of the parent class as seen through the details() method.



Example:

```
class Person:
  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

  def printname(self):
    print(self.firstname, self.lastname)

class Student(Person):
  pass

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

Output:

Mike Olsen

What is init? Or What Is A Constructor In Python?

- Constructors are generally used for instantiating an object. The task of constructors is to initialize(assign values) to the data members of the class when an object of the class is created. In Python the __init__() method is called the constructor and is always called when an object is created.

Syntax of constructor declaration :

```
def __init__(self):
    # body of the constructor
```

[12] What is Instantiation in terms of OOP terminology?

Instantiation refers to creating an object/instance for a class.

[13] What is used to check whether an object o is an instance of class A?

- Using isinstance() function, we can test whether an object/variable is an instance of the specified type or class such as int or list. In the case of inheritance, we can checks if the specified class is the parent class of an object. For example, isinstance(x, int) to check if x is an instance of a class int .
- The isinstance() works as a comparison operator, and it compares the object with the specified class type.
- You can verify if the emp object is an instance of a user-defined class Employee using the isinstance() function. It must return True.

Example:

```
class Employee:
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
emp = Employee("Emma", 11000)
# Checking if a emp object is an instance of Employee
Print (isinstance(emp, Employee))
```

Output : True

[14] What relationship is appropriate for Course and Faculty?

Association relationship appropriate for course and faculty.

[15] What relationship is appropriate for Student and Person?

Inheritance relationship appropriate for student and person.