Name: Kashyap Sorathiya

Batch: 06_Sept_Python

Assignment Module 4 (Advanced Python Programming)

- What is File function in python? What are the keywords to create and write file.

Ans.

1. A file object allows us to use, access and manipulate all the user accessible files. File handling is an important part of any web application. Python has several functions for creating, reading, updating, and deleting files.
2. To create and write to a new file, use open with “w” option. The “w” option will delete any previous existing file and create a new file to write. If you want to append to an existing file, then use open statement with “a” option. In append mode, Python will create the file if it does not exist.
3. "x" is also used to create the specified file but returns an error if the file exists.

- Explain Exception handling? What is an Error in Python?

Ans.

– An Exception is an error that happens during the execution of a program. Whenever there is an error, Python generates an exception that could be handled. It basically prevents the program from getting crashed.
– Errors are conditions that cannot get recovered by any handling techniques. It surely causes termination of the program abnormally. Errors mostly occur at runtime that's they belong to an unchecked type. Exceptions are the problems which can occur at runtime and compile time. It mainly occurs in the code written by the developers. Exceptions are divided into two categories such as checked exceptions and unchecked exceptions.

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

Ans.

– A try statement can have more than one except clause.
– Types of exception classes: Exception, ArithmeticError, BufferError, LookupError, AssertionError, ValueError, TypeError, SystemError, TabError, SyntaxError, RuntimeError, etc.

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

Ans.

– Try: This block will test the excepted error to occur.
– Except: If any exception occurs, the try clause will be skipped and except clause will run.
– Else: If there is no exception then this block will be executed.

- Can one block of except statements handle multiple exception?

Ans.

– Yes, one Except block can handle multiple exceptions. To do this, parentheses are used. Otherwise, the interpreter will return a syntax error.

- When is the Finally block executed?

Ans.

– Finally: It always gets executed either exception is generated or not.
– Finally block is always executed after try and except blocks. The finally block always executes after normal termination of try block or after try block terminates due to some exception. Even if you return in the except block still the finally block will execute.

- What happens when „1"== 1 is executed?

Ans.

– It simply evaluates to False and does not raise any exception. because one is string and other is integer.

- How Do You Handle Exceptions with Try/Except/Finally in Python? Explain with coding snippets.

Ans.

– Considering an integer variable called 'num'. Now starting a loop until the user enters '0' as input. Firstly, TRY block will get execute and ask for user input. Then using IF statement we will determine whether the input is odd or even.

```
num=int
while num!=0:
    try:
        #do your operations
        num = int(input("Enter a number: "))
        if num%2==0:
            raise Exception
```

– Now, if the input is even the program will raise an Exception and will jump to the EXCEPT block. However, if the input is odd, the program will skip the EXCEPT block and jump to ELSE block.

```
except Exception:
#If there is any exception raised, execute these statements
print("Not an odd number!")
else:
#If there is no exception, execute these statements
s = num**2
print('Square:',s)
```

– FINALLY block will always get executed after the TRY block terminates in the end, either ELSE or EXCEPT block got execute or not..

```
finally:
        #These statements must be executed
        if num!=0:
            print('Next...')
        else:
            print('Done.')
```

- What are oops concepts? Is multiple inheritance supported in Python?

Ans.

– 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.

– **YES**, Python supports multiple inheritance. In multiple inheritance, the features are inherited into the derived class.

- How to define a class in Python? What is self? Give an example of a Python class?

Ans.

– Classes are the user-defined blueprints that help us create an object. Objects are the instances of a particular class. Every other element in Python will be an object of some class, such as the string, dictionary, number, etc. will be an object of some corresponding built-in class (int, str) in Python.

– In object-oriented programming, whenever we define methods for a class, we use **self** as the first parameter in each case. The **self** keyword is used to represent an instance (object) of the given class. If there was no self-argument, the same class couldn't hold the information for both these objects.

– **Example**: -

1. create a Vehicle class

```
class Vehicle:
    def __init__(self, brand, model, type):
        self.brand = brand
        self.model = model
        self.type = type
        self.gas_tank_size = 14
        self.fuel_level = 0

    def fuel_up(self):
        self.fuel_level = self.gas_tank_size
        print('Gas tank is now full.')

    def drive(self):
        print(f'The {self.model} is now driving.')
```

2. Construct an object

```
vehicle_object = Vehicle('Honda', 'CV-500', 'Truck')
```

3. Access attribute values

```
print(vehicle_object.brand)
print(vehicle_object.model)
print(vehicle_object.type)
```

4. Calling methods

```
vehicle_object.fuel_up()
vehicle_object.drive()
```

5. Creating multiple objects

```
vehicle_object = Vehicle('Honda', 'Ridgeline', 'Truck')
a_subaru = Vehicle('Subaru', 'Forester', 'Crossover')
an_suv = Vehicle('Ford', 'Explorer', 'SUV')
```

- Explain Inheritance in Python with an example? What is __init__? Or What is a Constructor In Python?

Ans.

– Inheritance is the capability of one class to derive or inherit the properties from another class.
– "__init__" is a reserved method in python classes. It is known as a constructor in OOP concepts. This method called when an object is created from the class and it allows the class to initialize the attributes of a class. Constructors are used to initializing the object’s state. The task of constructors is to initialize(assign values) to the data members of the class when an object of the class is created.
– **Example**: -

```
class Parent:
    def __init__(self , fname, fage):
        self.firstname = fname
        self.age = fage
    def disp(self):
        print(self.firstname , self.age)

class Child(Parent):
    def __init__(self, fname, lname, fage):
        Parent.__init__(self, fname, fage)
        self.lastname = lname
    def disp(self):
        print('Name: ',self.firstname)
        print('Last Name: ',self.lastname)
        print('Age: ',self.age)

ob = Child('Kashyap','Sorathiya','22')
ob.disp()
```

- What is Instantiation in terms of OOP terminology?

Ans.

– Instantiation is the creation of a predefined object in OOP (object-oriented programming) language.

- What is used to check whether an object o is an instance of class A?

Ans.

– The **isinstance**() method checks whether an object is an instance of a class. **Isinstance** (**o**=object, **A**=class) returns **True** if the object argument is an instance of the class argument.

- What relationship is appropriate for Course and Faculty?

Ans.

– It's a **HAS-A** relationship i.e., Course and Faculty has an Association relationship. Moreover, Course and Faculty has a strong type of relationship, therefore, it is a Composition relationship.

- What relationship is appropriate for Student and Person?

Ans.

– It's a **IS-A** relationship i.e., Student and Person has an Inheritance relationship.
– One is Person and other is Student, where class Student has all data members which Person has but not the vice-versa. Moreover, Class Student has one more feature or data-member named grad Year which is not present in Person.
– So, simply we can say that it establishes an "**is-a**" or "**kind of**" relationship between Student and Person class. Saying that, Student is a kind of Person.