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

File Handling

The key function for working with files in Python is the open() function.The open() function takes two parameters; filename, and mode.There are four different methods (modes) for opening a file:

"r" - **Read** - Default value. Opens a file for reading, error if the file does not exist

"a" - **Append** - Opens a file for appending, creates the file if it does not exist

"w" - **Write** - Opens a file for writing, creates the file if it does not exist

"x" - **Create** - Creates the specified file, returns an error if the file exists

In addition you can specify if the file should be handled as binary or text mode

"t" - **Text** - Default value. Text mode

"b" - **Binary** - Binary mode (e.g. images)

Syntax

To open a file for reading it is enough to specify the name of the file:

f = open("demofile.txt")

To open the file, use the built-in open() function.

To write to an existing file, you must add a parameter to the open() function:

"a" - **Append** - will append to the end of the file

"w" - **Write** - will overwrite any existing content

2. Explain Exception handling? What is an Error in Python?

Exceptions are events that are used to modify the flow of control through a program when the error occurs. Exceptions get triggered automatically on finding errors in Python.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

Errors are problems that occur in the program due to an illegal operation performed by the user or by the fault of a programmer, which halts the normal flow of the program.

3. What is an Exception?

The programmer can handle an exception if it occurs at runtime. Python represents exceptions as classes.

Errors can be categorized into three types:

Compile-time error: Software can contain errors such as missing semicolons, incorrectly spelled keywords, or errors in syntax. These errors are called syntactical errors or compile-time errors.

Logical Error: A problem with a program that causes it to run inaccurately without causing the program to terminate abnormally.

Runtime Error: A runtime error happens when a program you're using or writing crashes or produces a wrong output. The program may not be able to function correctly.

4. 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. These exceptions are generally used as base classes for other exceptions.

BaseException

The base class for all built-in exceptions. It is not meant to be directly inherited by user-defined classes (for that, use Exception below).

Exception

All built-in, non-system-exiting exceptions are derived from this class. All user-defined exceptions should also be derived from this class.

ArithmeticError

The base class for built-in exceptions that are raised for various arithmetic errors. These are: OverflowError, **ZeroDivisionError**, **FloatingPointError**.

BufferError

Raised when a buffer related operation cannot be performed.

LookupError

The base class for exceptions that are raised when a key or index used on a mapping or sequence is invalid. These are: **IndexError** and **KeyError**.

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

The else part is executed when no exception occurs.

6. 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.

7. When is the finally block executed?

A finally block always executes, regardless of whether an exception is thrown.

8. What happens when „1"== 1 is executed?

It simply evaluates to False and does not raise any exception.

9. 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

Python Try, Except, else and Finally Syntax

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

10. What are oops concepts? Is multiple inheritance supported in python?

Python - OOP Concepts

In the real world, we deal with and process objects, such as student, employee, invoice, car, etc. Objects are not only data and not only functions, but combination of both. Each real-world object has attributes and behavior associated with it.Object-oriented programming paradigm is characterized by the following principles –

Class

A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.

Object

An individual object of a certain class. An object obj that belongs to a class Circle, for example, is an instance of the class Circle.A unique instance of a data structure that is defined by its class.An object comprises both data members (class variables and instance variables) and methods.

Encapsulation

Data members of class are available for processing to functions defined within the class only. Functions of class on the other hand are accessible from outside class context. So object data is hidden from environment that is external to class.Class function (also called method) encapsulates object data so that unwarranted access to it is prevented.

Inheritance

A software modelling approach of OOP enables extending capability of an existing class to build new class instead of building from scratch. In OOP terminology, existing class is called base or parent class, while new class is called child or sub class.Child class inherits data definitions and methods from parent class. This facilitates reuse of features already available. Child class can add few more definitions or redefine a base class function.

Polymorphism

Polymorphism is a Greek word meaning having multiple forms. In OOP, polymorphism occurs when each sub class provides its own implementation of an abstract method in base class.

Yes, Python supports multiple inheritance.

Like C++, a class can be derived from more than one base classes in Python. This is called Multiple Inheritance.

11.How to Define a Class in Python? What Is Self? Give An Example Of A Python Class.

Defining a Class

A class in Python can be defined using the class keyword.

```
class <ClassName>:
   <statement1>
   <statement2>   .
   .
   <statementN>
```

As per the syntax above, a class is defined using the class keyword followed by the class name and : operator after the class name, which allows you to continue in the next indented line to define class members. The followings are class members. Self represents the instance of the class. By using the “self” we can access the attributes and methods of the class in Python. It binds the attributes with the given arguments.

```
class Dog:

    # A simple class
    # attribute
    attr1 = "mammal"
    attr2 = "dog"

    # A sample method
    def fun(self):
        print("I'm a", self.attr1)
        print("I'm a", self.attr2)



# Driver code
# Object instantiation
Rodger = Dog()

# Accessing class attributes
# and method through objects
print(Rodger.attr1)
Rodger.fun()
```

12.Explain Inheritance in Python with an example? What is init? Or What Is A Constructor In Python?

Inheritance is an important aspect of the object-oriented paradigm. Inheritance provides code reusability to the program because we can use an existing class to create a new class instead of creating it from scratch. In inheritance, the child class acquires the properties and can access all the data members and functions defined in the parent class.A child class can also provide its specific implementation to the functions of the parent class.In this section of the tutorial, we will discuss inheritance in detail.In python, a derived class can inherit base class by just mentioning the base in the bracket after the derived class name.
Consider the following syntax to inherit a base class into the derived class.
Syntax

```
class derived-class(base class):
    <class-suite>
```

Example 1

```
class Animal:
    def speak(self):
        print("Animal Speaking")
#child class Dog inherits the base class Animal
class Dog(Animal):
    def bark(self):
        print("dog barking")
d = Dog()
d.bark()
d.speak()
```

Output:

```
dog barking
Animal Speaking
```

Python Multi-Level inheritance

Multi-Level inheritance is possible in python like other object-oriented languages. Multi-level inheritance is archived when a derived class inherits another derived class. There is no limit on the number of levels up to which, the multi-level inheritance is archived in python.

Python Inheritance

The syntax of multi-level inheritance is given below.

Syntax

```
class class1:
    <class-suite>
class class2(class1):
    <class suite>
class class3(class2):
    <class suite>
. .
```

Example

```
class Animal:
    def speak(self):
        print("Animal Speaking")
#The child class Dog inherits the base class Animal
class Dog(Animal):
    def bark(self):
        print("dog barking")
#The child class Dogchild inherits another child class Dog
```

```
class DogChild(Dog):
    def eat(self):
        print("Eating bread...")
d = DogChild()
d.bark()
d.speak()
d.eat()
```

Output:

```
dog barking
Animal Speaking
Eating bread...
```

Python Multiple inheritance

Python provides us the flexibility to inherit multiple base classes in the child class

Python Inheritance

The syntax to perform multiple inheritance is given below.

Syntax

```
class Base1:
    <class-suite>

class Base2:
    <class-suite>
.
.
.
class BaseN:
    <class-suite>

class Derived(Base1, Base2, ...... BaseN):
    <class-suite>
```

Example

```
class Calculation1:
    def Summation(self,a,b):
        return a+b;
class Calculation2:
    def Multiplication(self,a,b):
        return a*b;
```

```
class Derived(Calculation1,Calculation2):
    def Divide(self,a,b):
        return a/b;
d = Derived()
print(d.Summation(10,20))
print(d.Multiplication(10,20))
print(d.Divide(10,20))
```

Output:

```
30
200
0.5
```

The issubclass(sub,sup) method The issubclass(sub, sup) method is used to check the relationships between the specified classes. It returns true if the first class is the subclass of the second class, and false otherwise.

Consider the following example.

Example

```
class Calculation1:
    def Summation(self,a,b):
        return a+b;
class Calculation2:
    def Multiplication(self,a,b):
        return a*b;
class Derived(Calculation1,Calculation2):
    def Divide(self,a,b):
        return a/b;
d = Derived()
print(issubclass(Derived,Calculation2))
print(issubclass(Calculation1,Calculation2))
```

Output:

```
True
False
```

The isinstance (obj, class) method

The isinstance() method is used to check the relationship between the objects and classes. It returns true if the first parameter, i.e., obj is the instance of the second parameter, i.e., class.

Consider the following example.

Example

```
class Calculation1:
    def Summation(self,a,b):
        return a+b;
class Calculation2:
    def Multiplication(self,a,b):
        return a*b;
class Derived(Calculation1,Calculation2):
    def Divide(self,a,b):
        return a/b;
d = Derived()
print(isinstance(d,Derived))
```

Output:

```
True
```

The Default __init__ Constructor in C++ and Java. 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. Like methods, a constructor also contains a collection of statements(i.e. instructions) that are executed at the time of Object creation. 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.

Example:

```
# A Sample class with init method
class Person:
    # init method or constructor
    def __init__(self, name):
        self.name = name

    # Sample Method
    def say_hi(self):
        print('Hello, my name is', self.name)

p = Person('Nikhil')
p.say_hi()
```

13.What is Instantiation in terms of OOP terminology?

Creating an instance of class Explanation: Instantiation refers to creating an object/instance for a class.

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

What I'd like to do is given an object, I need to be able to tell if it's the class itself, or an instance of that class.
In other words:

```
class Test:
  pass
u = Test
check(u) # returns False
i = Test()
check(i) # returns True
```

15.What relationship is appropriate for Course and Faculty?

Response-Able student-adult relationships require adults to become able to respond appropriately to the demands of engaging students throughout the education system. This type of relationship fosters Student/Adult Partnerships.

16.What relationship is appropriate for Student and 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.