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

The file function in Python is used to handle files. It allows users to read, write, and manipulate files.

To create a file in Python, you can use the open() function. The open() function takes two parameters: the filename and the mode. The mode parameter specifies the purpose of opening the file. There are different modes available for opening a file, such as:

- r: open an existing file for a read operation.
- w: open an existing file for a write operation. If the file already contains some data then it will be overridden but if it doesn't exist, it will be created.
- a: open an existing file for an append operation

To write to a file in Python, you can use the write() method or the writelines (method. The write() method inserts the string in a single line in the text file.

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

In Python, an **error** is a problem in a program that causes it to stop executing. **syntax errors exceptions** . Syntax errors occur when the interpreter encounters a problem with the code's syntax, such as a missing colon or a misspelled keyword.

Exception handling is a mechanism in Python that allows you to handle exceptions gracefully and prevent your program from crashing.. The try block contains the code that might raise an exception, while the except block contains the code that will be executed if an exception is raised.

Here are some of the most common types of exceptions in Python:

- SyntaxError: Raised when there is a syntax error in the code.
- TypeError: Raised when an operation or function is applied to an object of the wrong type.
- NameError: Raised when a variable or function name is not found in the current scope.

- IndexError: Raised when an index is out of range for a list, tuple, or other sequence types.
- KeyError: Raised when a key is not found in a dictionary.
- ValueError: Raised when a function or method is called with an invalid argument or input.
- AttributeError: Raised when an attribute or method is not found on an object.
- IOError: Raised when an I/O operation, such as reading or writing a file, fails due to an input/output error.
- ZeroDivisionError: Raised when an attempt is made to divide a number by zero.

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

try-except block can have multiple except statements. The except block is used to handle specific exceptions that may occur in the try block.

Here are some of the built-in exception classes in Python:

- SyntaxError: Raised when there is a syntax error in the code.
- TypeError: Raised when an operation or function is applied to an object of the wrong type.
- NameError: Raised when a variable or function name is not found in the current scope.
- IndexError: Raised when an index is out of range for a list, tuple, or other sequence types.
- KeyError: Raised when a key is not found in a dictionary.
- ValueError: Raised when a function or method is called with an invalid argument or input.

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

Here is an example of a try-except-else block:

```
try:
    # some code that might raise an exception
except SomeException:
    # handle the exception
```

```
else:
    # execute if no exception is raised
```

In this example, if the code in the try block raises an exception, the code in the except block will be executed. If no exception is raised, the code in the else block will be executed.

5) Can one block of except statements handle multiple exception

Yes, a single block of except statements can handle multiple exceptions in Python. You can specify multiple exceptions in a single except block by separating them with commas. Here is an example:

```
try:
    # some code that might raise an exception
except (ExceptionType1, ExceptionType2):
    # handle the exception
```

In this example, if the code in the try block raises an exception of either ExceptionType1 or ExceptionType2, the code in the except block will be executed.

Here are some of the built-in exception classes in Python:

- SyntaxError: Raised when there is a syntax error in the code.
- TypeError: Raised when an operation or function is applied to an object of the wrong type.
- NameError: Raised when a variable or function name is not found in the current scope.
- IndexError: Raised when an index is out of range for a list, tuple, or other sequence types.
- KeyError: Raised when a key is not found in a dictionary.
- ValueError: Raised when a function or method is called with an invalid argument or input.
- AttributeError: Raised when an attribute or method is not found on an object.
- IOError: Raised when an I/O operation, such as reading or writing a file, fails due to an input/output error.

6) When is the finally block executed

It is useful for releasing resources such as file handles, network connections, etc.

Here's an example of how the finally block works in python:

```
try {
   // some code that may throw an exception
} catch (Exception e) {
   // handle the exception
} finally {
   // this code will always execute
}
```

In this example, if an exception is thrown in the try block, the corresponding catch block will be executed.

7) What happens when „1"== 1 is executed

This is because the string '1' is not equal to the integer 1.

In Python, the == operator is used to compare two values for equality. If the two values are equal, the expression evaluates to True.

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

Here's an example of how to use these blocks:

```
try:
   # some code that may raise an exception
except ExceptionType:
   # handle the exception
finally:
   # this code will always execute
```

In this example, the try block contains the code that may raise an exception. If an exception is raised, the corresponding except block will be executed. The finally block will always execute, regardless of whether an exception is raised or not.

Here's another example that demonstrates how to use multiple except blocks:

```
try:
   # some code that may raise an exception
except ExceptionType1:
   # handle ExceptionType1
except ExceptionType2:
   # handle ExceptionType2
finally:
   # this code will always execute
```

In this example, if ExceptionType1 is raised, the corresponding except block will be executed. If ExceptionType2 is raised, the corresponding except block will be executed instead. The finally block will always execute.

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

To define a class in Python, you use the class keyword followed by the name of the class. The class definition can contain attributes and methods. Here's an example of a simple Python class:

```
class Person:
   def __init__(self, name, age):
      self.name = name
      self.age = age

   def say_hello(self):
      print(f"Hello, my name is {self.name} and I'm {self.age} years old.")
```

In this example, we define a Person class with two attributes (name and age) and one method (say_hello). The __init__ method is a special method that gets called when an object of the class is created. It initializes the name and age attributes of the object.

The say_hello method is an instance method that takes no arguments (other than self, which refers to the object itself) and prints a message to the console.

The self parameter is used to refer to the object itself. When you call an instance method on an object, Python automatically passes the object as the first argument to the method. You can then use self to access the object's attributes and methods.

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

The derived class is also known as the child class, and the base class is also known as the parent class. Inheritance allows you to inherit the properties of a class, i.e., base class to another, i.e., derived class.

The __init__ method is a special method in Python that gets called when an object of the class is created. The __init__ method is also known as the constructor of the class.

Here's an example of how to define a parent class and a child class in Python:

```
class Animal:
    def __init__(self, name):
        self.name = name

    def speak(self):
        print(f"{self.name} says hello!")

class Dog(Animal):
    def __init__(self, name, breed):
        super().__init__(name)
        self.breed = breed

    def speak(self):
        print(f"{self.name} barks!")

dog = Dog("Buddy", "Golden Retriever")
dog.speak()  # Output: "Buddy barks!"
```

In this example, we define an Animal class with an __init__ method that initializes the name attribute of the object. The Animal class also has a speak method that prints a message to the console.

We then define a Dog class that inherits from the Animal class. The Dog class has its own __init__ method that initializes both the name and breed attributes of the object. The Dog class also has its own speak method that prints a different message to the console.

We create an instance of the Dog class and call its speak method to see how inheritance works.

10) What is Instantiation in terms of OOP terminology?

In other words, instantiation is the process of creating an object from a class.

When you create an instance of a class, you are creating a new object that has the same attributes and methods as the class. The new object is also known as an instance of the class.

Here's an example of how to create an instance of a class in Python:

```
class MyClass:
    def __init__(self, arg1, arg2):
        self.arg1 = arg1
        self.arg2 = arg2

my_object = MyClass("Hello", "World")
```

In this example, we define a MyClass class with two attributes (arg1 and arg2) and one method (__init__). The __init__ method is a special method that gets called when an object of the class is created. It initializes the arg1 and arg2 attributes of the object.

We then create an instance of the MyClass class and assign it to the variable my_object. The my_object variable now refers to a new object that has the same attributes and methods as the MyClass class.

11) What is used to check whether an object o is an instance of class A

In Python, you can use the built-in function isinstance() to check whether an object o is an instance of class A.

Here's an example:

```
class A:
   pass

class B(A):
   pass

obj = B()
print(isinstance(obj, A))  # True
```

12) What relationship is appropriate for Course and Faculty

The relationship between a course and faculty can be modeled in various ways, depending on the context and requirements of the system. Here are some possible relationships:

- **One-to-one**: A course is taught by only one faculty member, and a faculty member teaches only one course. [1].
- **One-to-many**: A course is taught by one faculty member, but a faculty member can teach multiple courses..
- **Many-to-many**: A course can be taught by multiple faculty members, and a faculty member can teach multiple courses.

13) What relationship is appropriate for Student and Person?

In this case, the Student class inherits all the attributes and methods of the Person class, such as name, age, address, etc. Additionally, the Student class can have its own attributes and methods that are specific to students, such as grade point average (GPA), courses taken, etc.