**Module – 4 (Advance Python Programming)**

**Ketan Pillai**

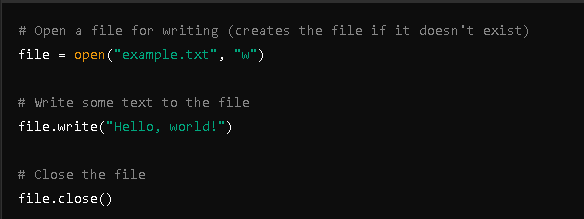
1) **What is File function in python? What is keywords to create and write** **file**.

Ans - In Python, the `file` function is not a function itself, but Python provides built-in functions to work with files. The key functions to create and write to a file are `open()`, `write()`, and `close()`. Here's a brief explanation:

**- open():** This function is used to open a file. You can specify the file name and the mode (e.g., "w" for writing, "r" for reading, "a" for appending).

**- write():** This function is used to write data to the file.

**- close():** This function is used to close the file after you're done working with it.



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

Ans - **Exception Handling in Python**

Exception handling in Python is a way to manage errors that occur during the execution of a program. It allows a program to continue running or fail gracefully instead of crashing abruptly. The main constructs used for exception handling in Python are `try`, `except`, `else`, and `finally`.

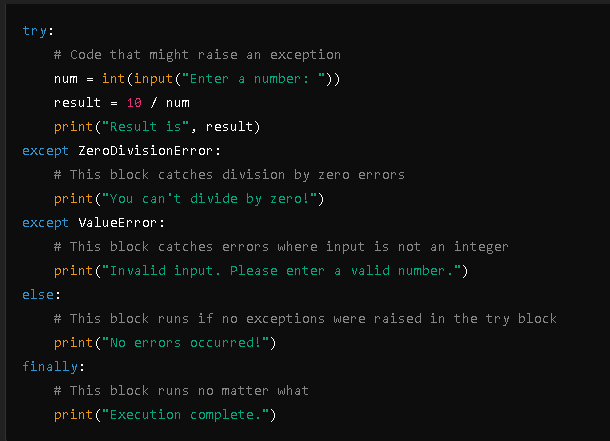
**- try:** This block of code is where you write the code that might raise an exception.

**- except:** This block catches and handles the exception. You can specify the type of exception to catch.

**- else:** This block runs if the try block does not raise an exception.

**- finally:** This block runs no matter what, even if an exception was raised or not. It's typically used for cleanup actions, such as closing files or releasing resources.

**Example**



**Errors in Python**

Errors in Python are problems that occur in the program, causing it to stop working as expected. There are two main categories of errors:

**1. Syntax Errors:** These are errors in the code structure. They are detected during parsing and prevent the code from running.

- Example: `print("Hello` (missing closing parenthesis)

**2. Exceptions:** These are errors detected during execution. They can be handled using exception handling constructs.

- Example: `ZeroDivisionError` occurs when you try to divide by zero.

**Common Types of Exceptions**

**- ZeroDivisionError:** Raised when division by zero occurs.

**- ValueError:** Raised when a function gets an argument of the right type but an inappropriate value.

**- TypeError:** Raised when an operation or function is applied to an object of inappropriate type.

**- IndexError:** Raised when trying to access an element from a list or tuple with an invalid index.

**- KeyError:** Raised when trying to access a dictionary with a key that does not exist.

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

Ans - A `try` block can have multiple `except` statements to handle different types of exceptions. There is no strict limit on the number of `except` statements; you can have as many as needed to handle various exceptions that might occur in the `try` block.

Python has many built-in exception classes, each representing a specific type of error. Here are some common ones:

**1. BaseException:** The base class for all exceptions.

**2. Exception:** The base class for all built-in non-exit exceptions. It is a subclass of `BaseException`.

**3. ArithmeticError:** The base class for all errors that occur for numeric calculations.

**- ZeroDivisionError:** Raised when dividing by zero.

**- OverflowError:** Raised when the result of an arithmetic operation is too large to be represented.

**- FloatingPointError:** Raised when a floating-point operation fails.

**4. LookupError:** The base class for all lookup-related errors.

**- IndexError:** Raised when a sequence subscript is out of range.

**- KeyError:** Raised when a dictionary key is not found.

**5. ValueError:** Raised when a function gets an argument of the right type but an inappropriate value.

**6. TypeError:** Raised when an operation or function is applied to an object of inappropriate type.

**7. NameError:** Raised when a local or global name is not found.

**- UnboundLocalError:** A subclass of `NameError` raised when a local variable is referenced before assignment.

**8. AttributeError:** Raised when an attribute reference or assignment fails.

**9. ImportError:** Raised when an import statement fails to find the module definition or when a `from ... import` fails.

- ModuleNotFoundError: A subclass of `ImportError` raised by `import` when a module could not be found.

**10. FileNotFoundError:** Raised when an attempt to open a file fails because the file does not exist.

**11. IOError:** Raised when an I/O operation (such as a file operation) fails.

**12. OSError:** Raised when a system-related operation causes an error.

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

Ans - The `else` part of a `try-except-else` block will be executed only if no exceptions are raised in the `try` block. In other words, if the code inside the `try` block runs successfully without encountering any exceptions, the code in the `else` block will be executed.

Here is the structure and flow:

**1. try block:** The code that might raise an exception goes here.

**2. except block(s):** Code to handle exceptions goes here. If an exception is raised in the `try` block, the corresponding `except` block is executed, and the `else` block is skipped.

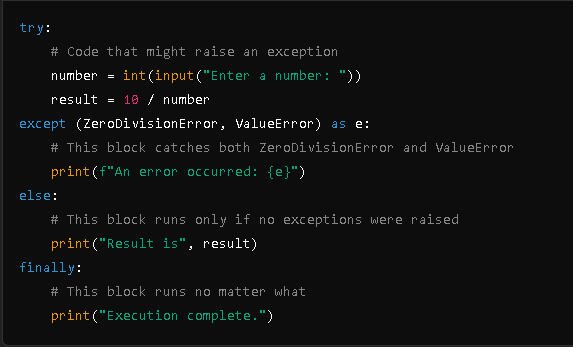
**3. else block:** This code runs only if no exceptions are raised in the `try` block.

**4. finally block (optional):** This block runs no matter what, whether an exception was raised or not.

**5) Can one block of except statements handle multiple exception?**

Ans - Yes, a single `except` block can handle multiple exceptions by specifying a tuple of exception types. This allows you to catch and handle different types of exceptions in a single `except` block.

**Example**



**Explanation**

**1. try Block:** The code that may raise exceptions is placed here.

**2. except Block:**

- `(ZeroDivisionError, ValueError)`: This tuple specifies that the `except` block should handle both `ZeroDivisionError` and `ValueError`.

- `as e`: This part assigns the exception instance to the variable `e`, allowing you to access the exception details within the `except` block.

**3. else Block:** Executes only if no exceptions are raised in the `try` block.

**4. finally Block:** Always executes, regardless of whether an exception was raised.

**6) When is the finally block executed?**

Ans - The `finally` block is executed always, regardless of whether an exception was raised or not. It runs after the `try` and `except` blocks have been executed, making it useful for cleanup actions like closing files or releasing resources.

Example

try:

# Code that might raise an exception

number = int(input("Enter a number: "))

result = 10 / number

else:

# Runs if no exception is raised

print("Result is", result)

finally:

# Always runs, no matter what

print("Execution complete.")

**7) What happens when „1‟== 1 is executed?**

Ans - When the expression `"1" == 1` is executed in Python, it evaluates to `False`. This is because the comparison is between a string (`"1"`) and an integer (`1`), and these are different types. Python does not automatically convert between types when using the equality operator `==`.

**Example**

```python

result = "1" == 1

print(result) # Output: False

```

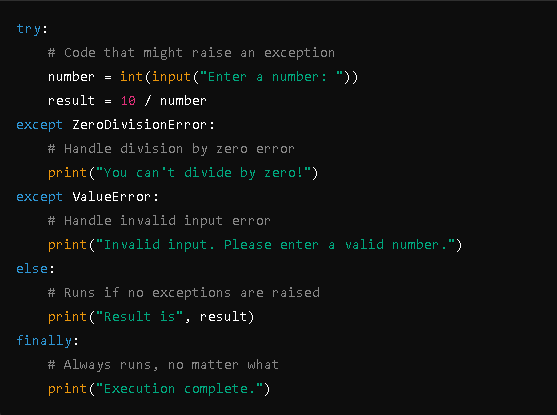
**8) How Do You Handle Exceptions With Try/Except/Finally In Python?  
Explain with coding snippets.**

Ans - Here's a brief explanation with code snippets:

**1. try block:** Contains code that might raise an exception.

**2. except block:** Handles the exception if one is raised.

**3. finally block:** Always executes, regardless of whether an exception was raised or not. Used for cleanup actions.



**9) What are oops concepts? Is multiple inheritance supported in java**

Ans - OOP Concepts

Object-Oriented Programming (OOP) is a programming paradigm based on the concept of objects. The main concepts of OOP are:

Encapsulation: Bundling the data (attributes) and methods (functions) that operate on the data into a single unit or class, and restricting access to some of the object's components.

Inheritance: Mechanism by which one class (child or subclass) inherits the attributes and methods from another class (parent or superclass), promoting code reusability.

Polymorphism: The ability of different objects to respond in a unique way to the same method call. It includes method overriding (runtime polymorphism) and method overloading (compile-time polymorphism).

Abstraction: Hiding the complex implementation details and showing only the essential features of the object. It simplifies the interaction with the object by reducing complexity.

Multiple Inheritance in Java

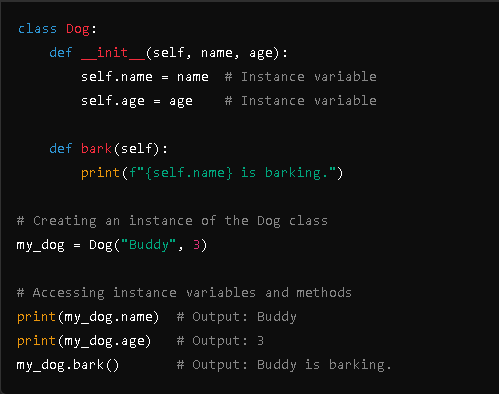
In Java, multiple inheritance is not supported directly with classes. A class cannot inherit from more than one class. This limitation avoids the "diamond problem" where an ambiguity arises due to multiple paths to a common base class.

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

Ans - Defining a Class in Python

A class in Python is defined using the `class` keyword.

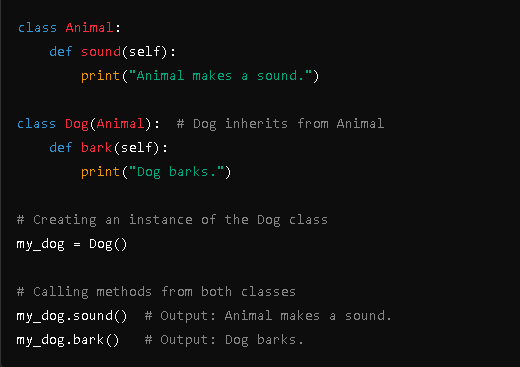
The `self` parameter is a reference to the current instance of the class and is used to access variables and methods associated with the instance.



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

Ans - Inheritance in Python:

Inheritance is a mechanism in Python that allows a new class to inherit properties and methods from an existing class. The existing class is called the parent class or base class, and the new class is called the child class or derived class.

**12) What is Instantiation in terms of OOP terminology?**

Ans Instantiation, in terms of Object-Oriented Programming (OOP), refers to the process of creating an instance of a class. An instance is a specific realization of a class, where the class serves as a blueprint and the instance represents a unique object with its own set of attributes and methods.

Short Explanation:

Instantiation is creating an object (instance) of a class. It involves invoking the class's constructor to initialize the object's state.

**13) What is used to check whether an object o is an instance of class A?**

Ans - To check if an object `o` is an instance of class `A`, you can use the `isinstance()` function.

You can use `isinstance(o, A)` to check if object `o` is an instance of class `A`.

Example:

class A:

pass

obj = A()

result = isinstance(obj, A)

print(result) # Output: True

14) What relationship is appropriate for Course and Faculty?

Ans - The relationship between a Course and Faculty is a many-to-many relationship. This means many courses can be taught by many faculty members. For example, a course like "Computer Science" can be taught by multiple faculty members, and a faculty member can teach multiple courses.

15) What relationship is appropriate for Student and Person?

Ans - The relationship between Student and Person is a one-to-one relationship, meaning each student corresponds to one person.

### Example:

class Person:

pass

class Student:

pass

In this simplified example, both `Person` and `Student` are separate classes, implying a one-to-one relationship between them.