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

In Python, the file function is not a standard function. However, file handling in Python is performed through built-in functions and methods. Python provides several ways to create, read, and write files using the open() function.

Here's a breakdown of how you can create and write files in Python:

1. Opening a File (The open() function)

To work with files in Python, the most common method is to use the open() function. This function allows you to specify the file name and the mode in which you want to open the file.

Syntax:

python

Copy

```
file_object = open('filename', 'mode')
```

Modes:

- 'r' - Read mode. (Default mode, if not specified) Open the file for reading.
- 'w' - Write mode. Opens the file for writing (creates a new file or truncates an existing file).
- 'a' - Append mode. Opens the file for appending at the end of the file.

- 'x' - Exclusive creation. If the file already exists, the operation will fail.
- 'b' - Binary mode. Used when working with binary files, e.g., 'rb' or 'wb'.
- 't' - Text mode. This is the default, used for text files.

2. Writing to a File

To write data to a file, you can use the write() method or writelines() method.

- **write()**: Writes a string to the file.
- **writelines()**: Writes a list of strings (without newline characters unless explicitly specified).

Example:

python

Copy

```
# Opening a file in write mode and writing to it
file = open('example.txt', 'w')
file.write("Hello, world!")  # Writing to the file
file.close()  # It's important to close the file after operations
```

3. Writing to a File (with with statement)

Using the with statement is recommended as it ensures the file is properly closed after the block of code is executed.

Example:

python

Copy

```
# Using with to open a file and write to it
with open('example.txt', 'w') as file:
    file.write("Hello, world!")
```

In this case, the file is automatically closed when the block of code is finished, even if there's an exception raised.

4. Common Keywords for File Creation and Writing

- **open()**: Used to open the file.
- **write()**: Used to write data to the file.
- **writelines()**: Writes a list of strings to the file.
- **close()**: Closes the file after finishing the operation (or use with to automatically close it).

Summary of Key File Keywords:

- open(): Opens the file.
- write(): Writes a string to the file.
- writelines(): Writes a list of strings to the file.
- close(): Closes the file.

83) What is Exception Handling? What is an Error in Python?

Exception Handling in Python is a mechanism used to handle errors that occur during the execution of a program. Instead of letting the program crash, we can use exception handling

to gracefully handle errors, log them, or take corrective actions.

- **Error**: An error in Python is an issue that causes the program to stop execution. Errors can be of different types:
    - **SyntaxError**: Occurs when the Python code is not written correctly (invalid syntax).
    - **RuntimeError**: Occurs while the program is running, like dividing by zero or accessing a non-existent file.

Python handles errors using try-except blocks.

84) How many except statements can a try-except block have? Name Some Built-in Exception Classes:

A **try-except block** can have **multiple except statements** to handle different exceptions. Each except block handles a specific exception or group of exceptions.

Example:

python

Copy

```
try:
    # Code that may raise an exception
    x = 5 / 0  # This will raise a ZeroDivisionError
except ZeroDivisionError:
    print("Cannot divide by zero!")
except TypeError:
```

```
    print("A type error occurred!")
except Exception as e:
    print(f"Some other exception: {e}")
```

Some common built-in exception classes:

1. **IndexError**: Raised when trying to access an index that is out of range.
2. **KeyError**: Raised when trying to access a dictionary key that doesn't exist.
3. **FileNotFoundError**: Raised when trying to open a file that doesn't exist.
4. **ValueError**: Raised when a function gets an argument of the correct type but inappropriate value.
5. **TypeError**: Raised when an operation or function is applied to an object of inappropriate type.
6. **ZeroDivisionError**: Raised when dividing a number by zero.

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

The else part of a try-except-else block will be executed **only if no exception** is raised in the try block.

Example:

python

Copy

```
try:
    x = 10 / 2
except ZeroDivisionError:
    print("Division by zero error!")
else:
    print("The operation was successful, no exceptions!")
```

Output:

nginx

Copy

The operation was successful, no exceptions!

If an exception occurs in the try block, the else part will be skipped.

86) Can one block of except statements handle multiple exceptions?

Yes, one block of except statements can handle multiple exceptions. You can either:

1. **Use a tuple** to specify multiple exceptions.
2. **Catch a more general exception** like Exception.

Example:

python

Copy

```
try:
    x = 10 / 0  # ZeroDivisionError
```

```
except (ZeroDivisionError, TypeError) as e:
    print(f"An error occurred: {e}")
```

You can also handle all exceptions with a general except statement:

python

Copy

```
try:
    x = 10 / "text"  # This will raise a TypeError
except Exception as e:
    print(f"An error occurred: {e}")
```

87) When is the finally block executed?

The finally block is executed **no matter what**, whether an exception is raised or not, and even if the exception is caught or not. It is used for cleanup actions like closing files or releasing resources.

Example:

python

Copy

```
try:
    print("Executing try block")
except Exception as e:
```

```
    print(f"Exception caught: {e}")
finally:
    print("Executing finally block")
```

Output:

csharp

Copy

```
Executing try block
Executing finally block
```

Even if an exception had occurred, the finally block would still execute.

88) What happens when 1 == 1 is executed?

The expression 1 == 1 is a comparison operation that checks whether 1 is equal to 1. The result of this operation is True, as both values are equal.

89) How Do You Handle Exceptions with Try/Except/Finally in Python?

In Python, exceptions are handled using the try-except-finally block. Here's how it works:

1. **try**: Code that might raise an exception is placed in the try block.

2. **except**: If an exception occurs in the try block, it is caught by the except block, where you can handle the error.
3. **finally**: This block is executed no matter what, useful for cleanup actions (e.g., closing files, releasing resources).

Example with all three blocks:

python

Copy

```
def divide(a, b):
    try:
        result = a / b
    except ZeroDivisionError:
        print("Cannot divide by zero!")
        result = None
    except TypeError:
        print("Both arguments must be numbers!")
        result = None
    else:
        print("Division successful!")
    finally:
        print("Cleaning up...")
    return result
```

```
# Test cases
print(divide(10, 2))  # Valid division
print(divide(10, 0))  # Division by zero
print(divide(10, "a"))  # TypeError
```

Output:

```
scss
Copy
Division successful!
Cleaning up...
5.0
Cannot divide by zero!
Cleaning up...
None
Both arguments must be numbers!
Cleaning up...
None
```

- The else block is executed if no exception occurs in the try block.
- The finally block is always executed for cleanup.