Assignment 6

# Q1

To handle unexpected errors: Exception handling can be used to handle unexpected errors that occur during the execution of a program. This can help to prevent the program from crashing and to provide a graceful way to handle the error.

To control the flow of execution: Exception handling can be used to control the flow of execution of a program. This can be done by using exceptions to signal the occurrence of specific events or to provide a way to skip over certain sections of code.

To provide a way to recover from errors: Exception handling can be used to provide a way to recover from errors that occur during the execution of a program. This can be done by using exceptions to signal the occurrence of an error and then providing a way to correct the error or to continue the execution of the program.

# Q2

If we don't do something extra to treat an exception, the default behaviour is for the program to crash. This is because the exception will propagate up the call stack until it reaches a handler that can handle it. If no handler can handle the exception, the program will crash.

# Q3

Handle the exception: This means that we can write code that will catch the exception and then take some action to recover from it. For example, we could write code that will print a message to the user, or we could write code that will try to correct the error and then continue the execution of the program.

Ignore the exception: This means that we can write code that will simply ignore the exception and then continue the execution of the program. This is not a good option, as it means that the error will not be handled and the program may crash later.

# Q4

Raise Statement: we can explicitly raise an exception using the raise statement. By specifying the type of exception to be raised, such as raise ValueError("Invalid value"), we can create and trigger exceptions at specific points in our code. This is useful when we encounter exceptional conditions that require custom handling.

Built-in Exceptions: Python provides built-in exceptions that can be raised to indicate specific error conditions. For example, IndexError can be raised when accessing a list with an invalid index, or FileNotFoundError can be raised when a requested file does not exist.

# Q5

Using the finally clause: The finally clause is a special clause that is executed at the end of a try/except block, regardless of whether or not an exception is raised. This can be useful for performing cleanup tasks or for ensuring that certain actions are always executed.

Using the atexit module: The atexit module provides a way to register functions that will be executed when the program terminates. This can be useful for performing cleanup tasks or for ensuring that certain actions are always executed.