Module 08. Error Handling

September 26, 2018

1 Error handling

The error handling is done through the use of exceptions that are caught in try blocks and handled in except blocks. If an error is encountered, a try block code execution is stopped and transferred down to the except block.

In addition to using an except block after the try block, you can also use the finally block.

The code in the finally block will be executed regardless of whether an exception occurs.

```
In [4]: # from io import *
        import math

        # print(10/0)  # ZeroDivisionError: division by zero

        # var = input("enter a number")

        # print(math.sqrt(var))  # TypeError: a float is required when input is 16

        # var = int(var) # ValueError: invalid literal for int() with base 10: 'a'

        # fo = open("MyModule/MyText1.txt", "r")
        # print(fo.read()) # FileNotFoundError: No such file or directory: 'MyModule/MyText1.t
```

```
enter a numbera


        ---------------------------------------------------------------------------

        ValueError                                Traceback (most recent call last)

        <ipython-input-4-4638e71266ab> in <module>()
          8 # print(math.sqrt(var))  # TypeError: a float is required when input is 16
          9
    ---> 10 var = int(var) # ValueError: invalid literal for int() with base 10: 'a'
         11
         12 # fo = open("MyModule/MyText1.txt", "r")
```

```
ValueError: invalid literal for int() with base 10: 'a'
```

1.1 Using Exceptions to Prevent Crashes

Handling errors correctly is especially important when the program has more work to do after the error occurs. This happens often in programs that prompt users for input. If the program responds to invalid input appropriately, it can prompt for more valid input instead of crashing

2 Handling Exception (try-except-else-finally)

```
In [8]: var = "abc"
        try:
            print(10/10)  # ZeroDivisionError: division by zero

            var = input("enter a number")

            print(math.sqrt(float(var)))  # TypeError: a float is required when input is 16

            var = int(var) # ValueError: invalid literal for int() with base 10: 'a'

            fo = open("MyModule/MyText1.txt", "r")
            print(fo.read()) # FileNotFoundError: No such file or directory: 'MyModule/MyText1

            print("Success")
        except NameError:
            print("Var to sqrt must be in float")
        except ZeroDivisionError as e:
            print("Divide by zero!!!", e)
        except TypeError  as e:
            print("TypeError!!!", e)
        except ValueError  as e:
            print("ValueError!!!", e)
        except FileNotFoundError  as e:
            print("FileNotFoundError!!!", e)
        except Exception as e:
            print("Exception!!!", e)

        else:
            print("Program executed successfully")

        finally:

            print("Finally Block")
```

```
1.0
enter a numbera
```

```
ValueError!!! could not convert string to float: 'a'
Finally Block
```

2.0.1 Multiple except blocks

```
In [ ]: try:
            fr = open("ReadFile", "r")
            fw = open("WriteFile1", "r")
            var = fr.read()
            fw.write(var)
        except FileNotFoundError:
            print("File not found")
        except IOError:
            print("IO Error")
        except Exception:
            print("Error out of my control")
        else:
            print("Written content in the file successfully")
            fr.close()
            fw.close()
        finally:
            try:
                print("Status of read file : ", fr.closed)
                print("Status of read file : ", fw.closed)
            except NameError:
                print("File object was not created")
```