ICTPRG302 - Apply introductory programming techniques –

Session 11 Worksheet

Python Error Handling and Debugging

The idea behind the *try-except* clause is to eliminate errors at runtime. The syntax is:

**try**:

block of statement(s)

**except** (Exception):

block of statement(s)

Task 1. Below is “buggy code”. Add a *try-except* clause so the code runs without errors. Print message: msg = "You cannot add integer to string!" You can use either BaseException or TypeError exception since all exceptions inherit from BaseException.

# Type your answer below. Replace ... with your code

string = "Try Except Statements!"

...

string + 3

...

msg = ...

print("Type Error occurred.")

print(msg)

Output:

A picture containing icon

Description automatically generated

Task 2. Although a try-except cause can have multiple except clauses to handle different exceptions, only one will be executed (when an exception occurs). Use the following code to catch an IndexError when the list “my\_list” is out of range.

# write the code to catch IndexError Exception

my\_list = [1, 2, 4, 8]

try:

my\_list[5] = 7 # this statement raise an IndexError

except (ValueError, TypeError):

# handle multiple exceptions

pass

except ZeroDivisionError:

pass

except (...): # your code instead of ...

print (...) # your code instead of ...

except:

print ("Unspecified exception occurred.")

Output:



Task 3. Write a program that will input a number to check is it positive, negative or zero. Input will be accepted as string. Use the while loop and apply the try-except clause to loop around until ‘q’ is entered.

Check the program when invalid input is provided and raise an exception.

Use the following code for debugging.

# In this program, we input a number to check

# is it positive, negative or zero

# enter ‘q’ to quit

my\_num = input("Enter a number: ")

**while** (my\_num != 'q'):

**try**:

**if** int(my\_num) >= 0:

**if** int(my\_num == 0):

print("Zero")

**else**:

print("Positive number.\n")

**else**:

print("Negative number.\n")

**except**:

print("Wrong input.")

my\_num = input("Enter a number: ")

print("Program terminated.")

Correct Output:

Text

Description automatically generated

Make a list of expected results that a program should return.

Test each assumption.

(Initial Steps using IDLE only. Start IDLE (if not already), go to File menu and click on New Window. Type or copy your program in. To run your program click Run Module.)

**Test Table**

|  |  |  |
| --- | --- | --- |
| Test Description | Expected Output | Your Actual Output |
| User provides input = 0 | Print “Zero” |  |
| User provides input = -1 | Print “Negative number.” |  |
| User provides input = 1 | Print “Positive number.” |  |
| User provides input = w | Print ‘Wrong input.” |  |
| User provides input = ‘q’ | Print “Program terminated.” |  |

For some reason our program does not work correctly!

Debuggers can help you step through the code line by line and show you what happens at each step. For instance, Python’s IDLE, Visual Studio Code or PyCharm have built-in debugging features.

Use **one** of them to debug your code. Your program must produce the correct result.

For the debugger to be most useful, you need to set a breakpoint before you start running your code.

The following links could be useful:

[IDLE Debugger Tutorial - Python IDE](https://www.youtube.com/watch?v=CQin42wFC-w)

[Debugging under IDLE](https://www.cs.uky.edu/~keen/help/debug-tutorial/debug.html)

[Using the Debugger](https://inventwithpython.com/chapter7.html)

[PyCharm Debug Tutorial | How to Debug Code in PyCharm!](https://www.youtube.com/watch?v=76Lu6CfMuGg)

[PyCharm - Debugging and Breakpoints](https://www.tutorialspoint.com/pycharm/pycharm_debugging_and_breakpoints.htm)

[PyCharm - Debugging Python Code](https://www.jetbrains.com/help/pycharm/part-1-debugging-python-code.html)

[Debugging Python in VSCode - 01 - Intro to Debugging in VSCode](https://www.youtube.com/watch?v=KEdq7gC_RTA)

[Visual Studio - Debug your Python code](https://docs.microsoft.com/en-us/visualstudio/python/debugging-python-in-visual-studio?view=vs-2022)

[Python debugging in VS Code](https://code.visualstudio.com/docs/python/debugging)