

Exception Handling in Python

Student's Name

Department, Institutional Affiliation

Course Number and Name

Professor's Name

Due Date

Exception Handling in Python

An exception is an error that occurs while executing code and results in an unexpected outcome. Exception handling is a method of programmatically responding to these unexpected outcomes, which require special processing (Pierre, 2020). Exception handling in Python uses a *try-except* block. Exceptions can be built into a language or custom. The main aim of exception handling is to prevent potential failures and uncontrolled code crashes.

In Python, all built-in exceptions are derived from the *BaseException* class. The *Exception* class, which is the focus of this paper, directly derives from *BaseException*. Common exceptions encountered in day-to-day operations with python include *ValueError*, *IndexError*, *TypeError*, and *NameError*. Figure 1 shows which conditions may trigger these errors. To handle such exceptions, Python provides useful constructs, as shown in Figure 2. The keywords are explained as follows (Van de Klundert, n.d.):

- *try block* – contains the code monitored for the exceptions.
- *except block* – contains the code to be executed if a specific exception occurs.
- *else block* – it is executed only if no exception occurred in the *try* block.
- *finally* – it is used for clean-up. This block is always executed.

Figure 3 shows how to implement the exception handling constructs in Python. A user is required to input two numbers from the output whose sum and division are then calculated, respectively. If a string, in this case, 3, is inputted, a *TypeError* is raised. If a zero is inputted, then a *ZeroDivisionError* occurs.

An exception is an error that occurs at runtime while a code is running. Exception handling responds to these errors programmatically and makes the code continue running

without breaking. Hence, every program should have an exception handling mechanism to ensure robust and non-breaking code.

Figure 1

Common Python Exception

```

→ 01-exception-handling-python git:(main) x ipython
Python 3.9.6 (default, Jun 29 2021, 05:25:02)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.25.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: # this works fine

In [2]: int('12')
Out[2]: 12

In [3]: # this throws an error

In [4]: int('ten')
-----
ValueError                                Traceback (most recent call last)
<ipython-input-4-12a85981658c> in <module>
----> 1 int('ten')

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

In [5]: my_list = [1, 2, 3, 4]

In [6]: my_list[2] # this works fine
Out[6]: 3

In [7]: my_list[6] # this throws IndexError, since the list had only 4 elements but we're trying to access the sixth element
-----
IndexError                                Traceback (most recent call last)
<ipython-input-7-752e5391c3dd> in <module>
----> 1 my_list[6] # this throws IndexError, since the list had only 4 elements but we're trying to access the sixth element

IndexError: list index out of range

In [8]: '2' + 2 # throws a TypeError, adding a string to an integer
-----
TypeError                                Traceback (most recent call last)
<ipython-input-8-58382d26a7b0> in <module>
----> 1 '2' + 2 # throws a TypeError, adding a string to an integer

TypeError: can only concatenate str (not "int") to str

In [9]: name = "Coding"

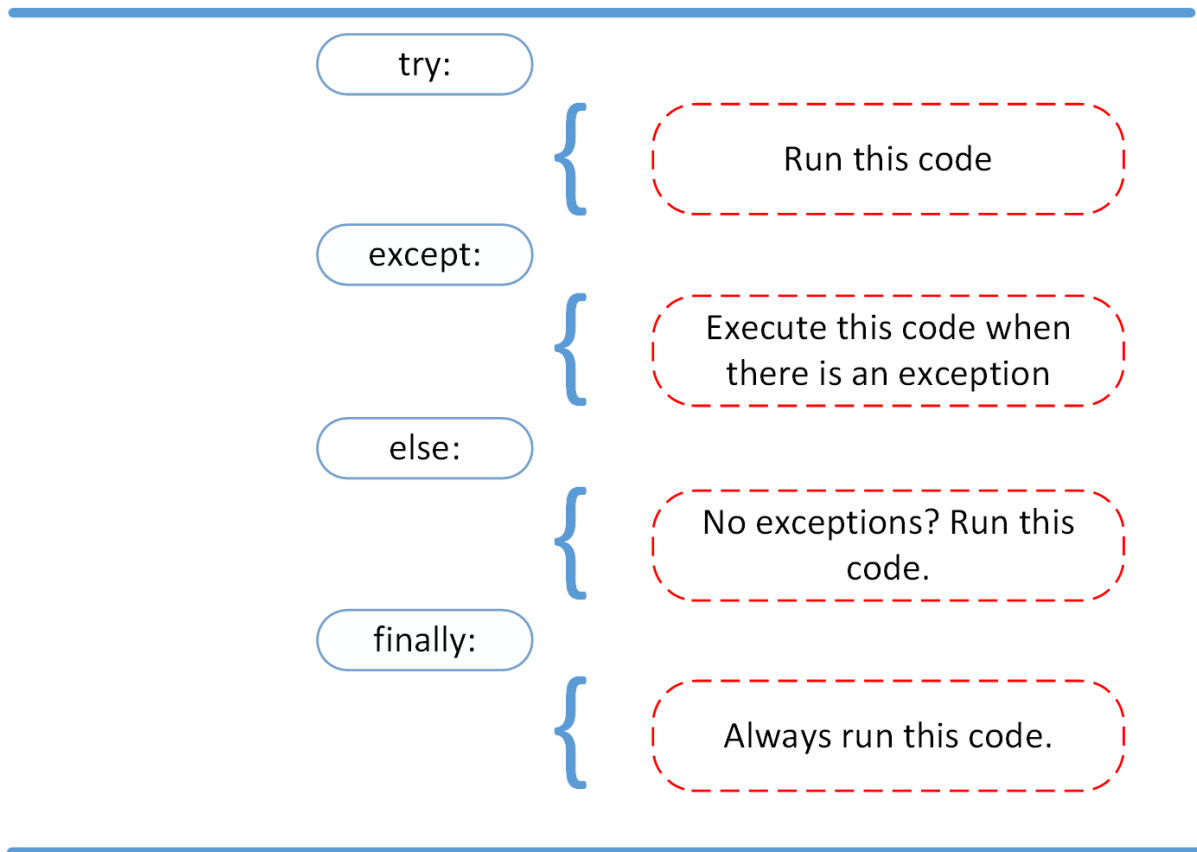
In [10]: print(Name) # throws a NameError as 'Name' is not defined, and Python is a case-sensitive language
-----
NameError                                Traceback (most recent call last)
<ipython-input-10-d9e672184e58> in <module>
----> 1 print(Name) # throws a NameError as 'Name' is not defined, and Python is a case-sensitive language

NameError: name 'Name' is not defined

In [11]: █

```

Note. Shows ValueError, IndexError, TypeError, and NameError exceptions.

Figure 2*Python Exception Handling Constructs*

Note. *try*, *except*, *else* and *finally* constructs for exception handling (Van de Klundert, n.d.)

Figure 3

Python Sample Code with Common Exceptions

```

01-exception-handling-python > exception_handling.py > main
1  # code to show basic exception handling
2  def main():
3      try:
4          x = int(input("Enter a number: "))
5          y = int(input("Enter another number: "))
6          # print the result of the division
7          print(f'{x} / {y} = {x / y}')
8          # print the result of sum
9          print(f'{x} + {y} = {x + y}')
10     except ZeroDivisionError as e:
11         print(e)
12     except ValueError as e:
13         print(e)
14     except TypeError as e:
15         print(e)
16     except:
17         print("An error occurred")
18     else:
19         print("Everything executed successfully!")
20     finally:
21         print("Executing finally clause")
22
23
24 if __name__ == "__main__":
25     main()
26

```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE

```

→ 01-exception-handling-python git:(main) x python exception_handling.py
Enter a number: 2
Enter another number: 4
2 / 4 = 0,5
2 + 4 = 6
Everything executed successfully!
Executing finally clause
→ 01-exception-handling-python git:(main) x python exception_handling.py
Enter a number: 3
Enter another number: '3'
invalid literal for int() with base 10: "'3'"
Executing finally clause
→ 01-exception-handling-python git:(main) x python exception_handling.py
Enter a number: 3
Enter another number: 0
division by zero
Executing finally clause
→ 01-exception-handling-python git:(main) x █

```

Note. Python code for `ZeroDivisionError`, `ValueError`, and `TypeError` exceptions together with their outputs.

References

Pierre, S. (2020, May 15). *Exception handling in Python*. Towards data science.

<https://towardsdatascience.com/exception-handling-in-python-7f639ce9a3d>

Van de Klundert, S. (n.d.). *Python exceptions: An introduction*. Real Python.

<https://realpython.com/python-exceptions/>