Exceptions, error handling - basics

Introduction
In a perfect world, a programmer would write a piece of code and it will run perfectly, and not fail at all.

But in the real world it does not work like that. Every development project encounters errors, or unexpected behaviors.

And it's better to come prepared to handle them, and not leave it to luck.

That's why this topic is called - 'Error Handling'.
In computer science, once a program encounters an error it is called an 'Exception'.

An 'Exception' would happen once a program is executed, and it encounters a wrong piece of code, that it does not know what to do with, and that causes our execution to be terminated on the spot.

Python knows to tell us exactly what type of exception is thrown once it happens. And then, we can apply certain actions in order to avoid failure in execution, by fixing our code.

How do we know that our code failed, and that we had an exception ?

- Once we run a certain .py file, and the code flow will encounter an exception, the run will stop.

- During the run, if the code flow will encounter an badly written code, it will cause the run to fail, and the error will be displayed at the 'Terminal' log of PyCharm, in red color.

- The 'Terminal' exception log will also point out what is the path of the exception (what Module / Class / Method failed).

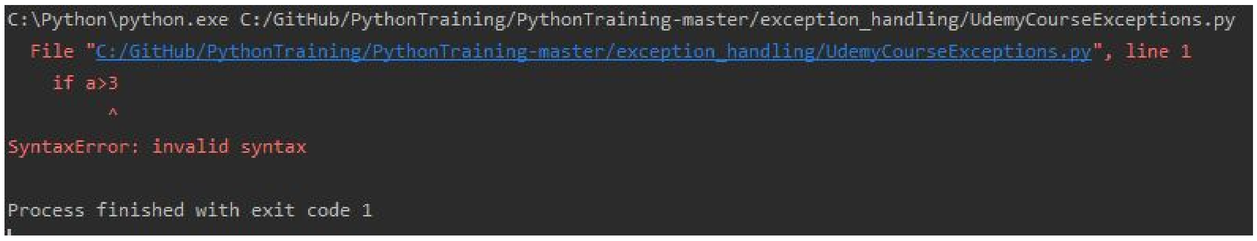
- The 'Terminal' log will indicate the exact line in the code causing the run to fail.

Complete common exceptions in Python (With examples)
* There are about 35 types of exceptions. Let's focus on the common ones.

1. 'syntax error', 'parsing error'
   Error is cuased when code execution encounters a programmatic logical failure, or - if not following up to Python's syntax rules.

   Example:
   ```
   if a>3
   ```

   Will cause an exception. As we have not declared about the variable 'a' before the 'if' statement.

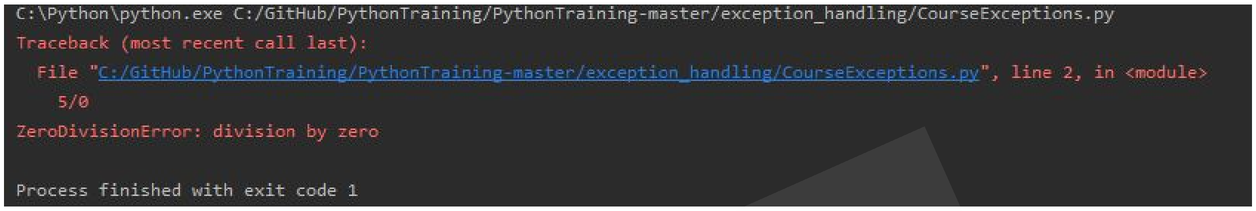


```
C:\Python\python.exe C:/GitHub/PythonTraining/PythonTraining-master/exception_handling/UdemyCourseExceptions.py
  File "C:/GitHub/PythonTraining/PythonTraining-master/exception_handling/UdemyCourseExceptions.py", line 1
    if a>3
         ^
SyntaxError: invalid syntax

Process finished with exit code 1
```

2. ‘ZeroDevitionError’
   Error will be thrown once we try to divide a certain number by ‘0’.

   Example:
```
x=5
5/0
```

   Division by ‘0’ is an illegal mathematical action.



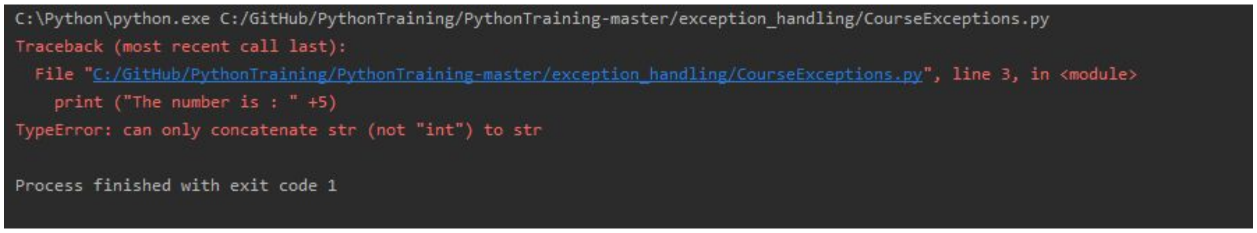
```
C:\Python\python.exe C:/GitHub/PythonTraining/PythonTraining-master/exception_handling/CourseExceptions.py
Traceback (most recent call last):
  File "C:/GitHub/PythonTraining/PythonTraining-master/exception_handling/CourseExceptions.py", line 2, in <module>
    5/0
ZeroDivisionError: division by zero

Process finished with exit code 1
```

3. ‘TypeError’
   Occurs once trying to use a wrong type of object.
   Appears in many scenarios and shapes.

   Example:
```
x=5
print ("The number is : " +x)
```

   Python cannot print 2 types of variables. In this ‘print’ command we have both ‘string’ and ‘integer’ variables.

   We should fix it by using ‘casting’ → +str(x)

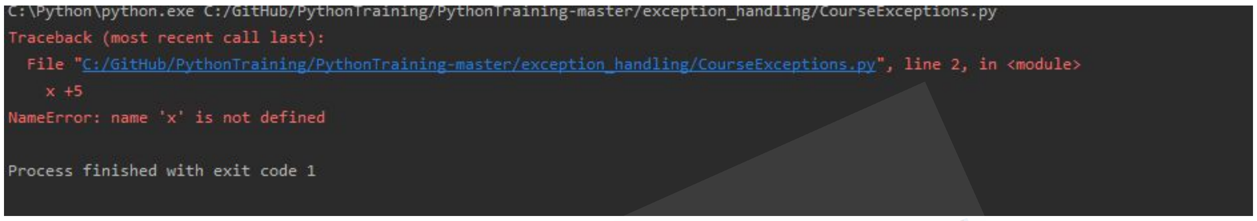


```
C:\Python\python.exe C:/GitHub/PythonTraining/PythonTraining-master/exception_handling/CourseExceptions.py
Traceback (most recent call last):
  File "C:/GitHub/PythonTraining/PythonTraining-master/exception_handling/CourseExceptions.py", line 3, in <module>
    print ("The number is : " +5)
TypeError: can only concatenate str (not "int") to str

Process finished with exit code 1
```

4. ‘NameError’
   Is caused when a variable cannot be found in local
   or global (whole project) scope.

   Example:
   ```
   x + 5
   ```

   We used the variable by a math action, but never created it.

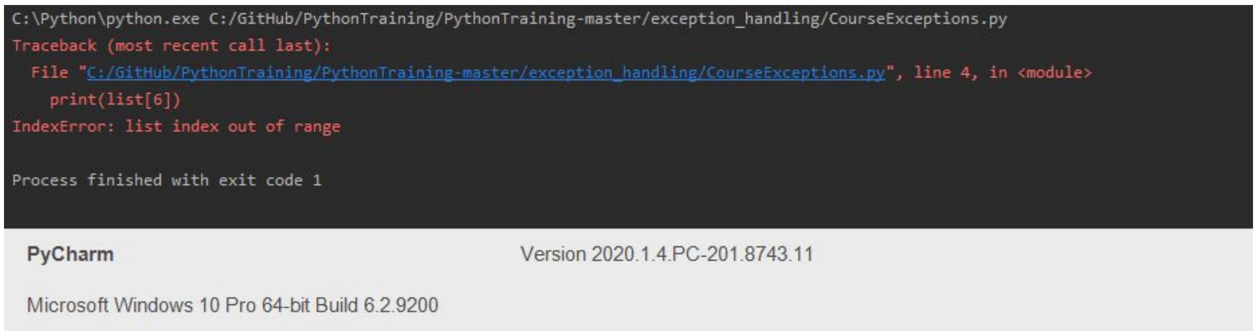


```
C:\Python\python.exe C:/GitHub/PythonTraining/PythonTraining-master/exception_handling/CourseExceptions.py
Traceback (most recent call last):
  File "C:/GitHub/PythonTraining/PythonTraining-master/exception_handling/CourseExceptions.py", line 2, in <module>
    x +5
NameError: name 'x' is not defined

Process finished with exit code 1
```

5. ‘IndexError’
   One of the cases it may be caused is when reaching out for a cell of a collection by index, and that index does not exist.

   Example:
   ```
   list = [1,2,3,4]
   print(list[6])
   ```

   The last index of the list is ‘3’.



```
C:\Python\python.exe C:/GitHub/PythonTraining/PythonTraining-master/exception_handling/CourseExceptions.py
Traceback (most recent call last):
  File "C:/GitHub/PythonTraining/PythonTraining-master/exception_handling/CourseExceptions.py", line 4, in <module>
    print(list[6])
IndexError: list index out of range

Process finished with exit code 1
```

PyCharm   Version 2020.1.4.PC-201.8743.11

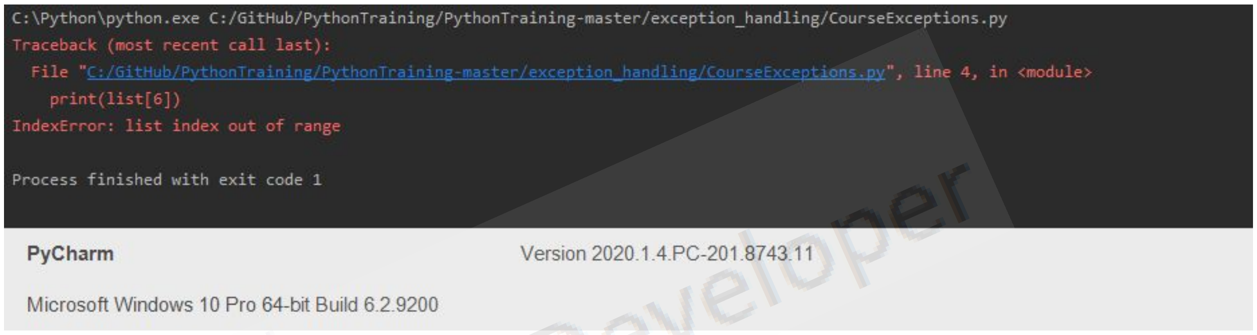
Microsoft Windows 10 Pro 64-bit Build 6.2.9200

6. 'IndentationError'
   Python language is written with indentations, which indicates hierarchy and code flow, once indentations are not kept an error will occur.

   Example:

```
x=4
    print(x)
```

   We have redundant indentations in the 'print' command.



```
C:\Python\python.exe C:/GitHub/PythonTraining/PythonTraining-master/exception_handling/CourseExceptions.py
Traceback (most recent call last):
  File "C:/GitHub/PythonTraining/PythonTraining-master/exception_handling/CourseExceptions.py", line 4, in <module>
    print(list[6])
IndexError: list index out of range

Process finished with exit code 1
```

PyCharm Version 2020.1.4.PC-201.8743.11

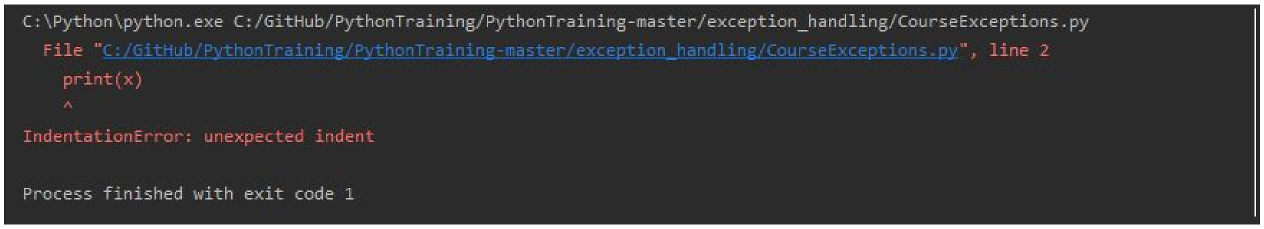
Microsoft Windows 10 Pro 64-bit Build 6.2.9200

7. 'KeyError'
   Once we try to call out a dictionary cell with a wrong or non existing 'key'.

   Example:

```
dictionary = {"abc" : 1}
Dictionary["z"]
```

   The key 'z' does not exist

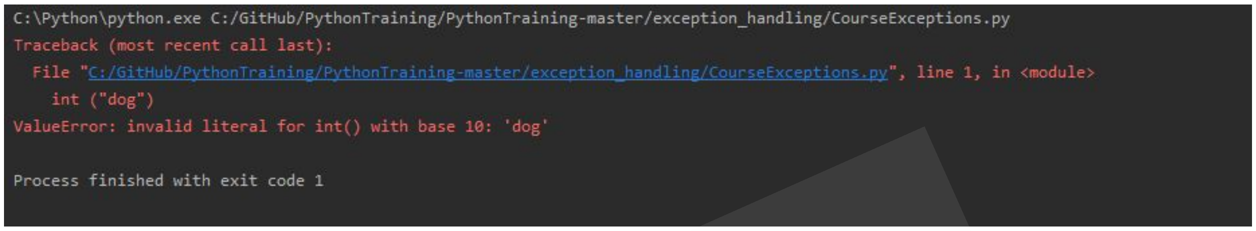


```
C:\Python\python.exe C:/GitHub/PythonTraining/PythonTraining-master/exception_handling/CourseExceptions.py
  File "C:/GitHub/PythonTraining/PythonTraining-master/exception_handling/CourseExceptions.py", line 2
    print(x)
    ^
IndentationError: unexpected indent

Process finished with exit code 1
```

8. ‘ValueError’
   Will be raised once using a wrong value. A Value that cannot be used in a certain command.

   Example:
   ```
   int ("dog")
   ```

   As we all know, a ‘string’ value cannot be casted into ‘int’ type



```
C:\Python\python.exe C:/GitHub/PythonTraining/PythonTraining-master/exception_handling/CourseExceptions.py
Traceback (most recent call last):
  File "C:/GitHub/PythonTraining/PythonTraining-master/exception_handling/CourseExceptions.py", line 1, in <module>
    int ("dog")
ValueError: invalid literal for int() with base 10: 'dog'

Process finished with exit code 1
```

These items would conclude the common errors that can be encountered during a code execution.