Errors and Exceptions

Back-to-Basics Series

Program Errors

There are three different types of error that can occur when we are writing programs – syntax errors, run time errors and logic errors.

- Syntax error
- Logic error
- Run time error

Syntax Error

- Syntax is the spelling and grammar of programming language. A syntax error is when the sequence of characters or tokens intended to be written in a particular programming language is written incorrectly.

- When you break these syntax rules for writing your program, this is often due to misspelling or the use of incorrect punctuation. A syntax error will cause the program not to run.

Logic Error

- The program works, but produces results from what it is expected.

- Logic errors are notoriously difficult to debug.

Run time Error

A run time error can cause a computer or program to crash even if there is nothing wrong with the program code.

For example, an infinite while-loop or recursive call in a program will continually use up the memory and eventually crashes the program.

Python's list of common errors

- ZeroDivisionError
- IndexError
- NameError
- TypeError
- ValueError
- FileNotFoundError
- . . .

Python will raise an Exception when it runs into an error.

You can find the list of standard exceptions at:
https://www.tutorialspoint.com/python/python_exceptions.htm

What is Exception?

- An exception is an event which occurs during the execution of a program that disrupts the normal flow of the program.
- When a Python program encounters a situation it cannot cope with, it raises an exception.
- An exception, or an error, must be handled immediately otherwise it will terminate the program and quit.

Handle an exception

- We can defend the program by placing the "*suspicious code*", that may raise an exception, in the **try:** block.
- Include some codes in the **except:** block to handle the error in an elegant manner.

```
try, except, else, finally
```

Syntax: `try, except, else, finally`

```
try:
      Some "suspicious code"
except Exception1:
      Do this if there is Exception1
except Exception2:
      Do this if there is Exception2
...
else:
      Do this if no exception
finally:
      Do this every time
```

Learn by Examples

Try, Except, Else, Finally

Example 1: Using `try` and `except`

```
try:
        f = open('testfile.txt')
        var = bad_variable
        div = 3/0
        total = 'string' + 10
        import module
except FileNotFoundError:
        print('Sorry, file does not exist.')
except NameError:
        print('Sorry, a variable not found.')
except ZeroDivisionError:
        print('Sorry, cannot divide by zero.')
except TypeError:
        print('Sorry, there is a type error.')
except Exception:
        print('Sorry, there is some kind of error.')
```

Run the code and observe the error message. **Comment away one error at a time and run it again.**

Example 2: Python Standard Exceptions

```
try:
    f = open('testfile.txt')
    var = bad_variable
    div = 3/0
    total = 'string' + 10
    import module
except Exception as e:
    print('Error: ', e)
```

Run the code and observe the error message.
Comment away one error at a time and run it again.

Example 3: `try`, `except` and `else`

```
try:
    f = open('testfile.txt')
    var = bad_variable
    div = 3/0
    two = 4/2
except Exception as e:
    print('Error: ', e)
else:
    print('Two is : ', two)
    print('Well done! Your code is working.')
```

This line is a working code.
Comment away one error at a time and run it again.

Example 4:
`try`, `except`, `else` and `finally`

```
try:
    f = open('testfile.txt')
    var = bad_variable
    div = 3/0
    two = 4/2
except Exception as e:
    print('Error: ', e)
else:
    print('Two is : ', two)
    print('Well done! Your code is working.')
finally:
    print('End of Example')
```

Example 5: Using raise

In all the above examples, we have to "invent" some error in the codes to simulate the errors, but we actually simulate them by simply raising an exception.

raise FileNotFoundError

```
try:
    f = open('testfile.txt')
    var = bad_variable
    div = 3/0
    total = 'string' + 10
    import module
```

raise ZeroDivisionError

raise NameError

raise TypeError

raise Exception

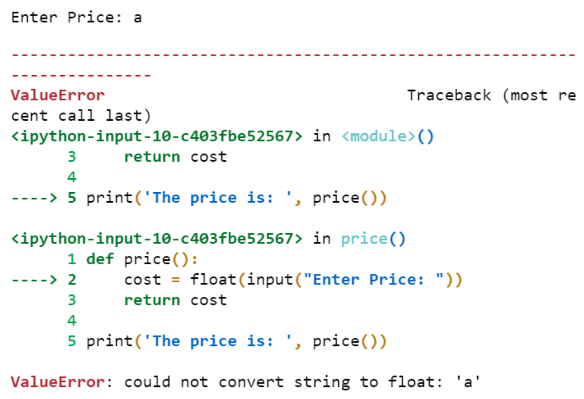
Example 5: Using `raise`

```
try:
    raise FileNotFoundError
    raise ZeroDivisionError
    raise TypeError
    raise Exception
except FileNotFoundError:
    print('Sorry, file does not exist.')
except NameError:
    print('Sorry, a variable not found.')
except ZeroDivisionError:
    print('Sorry, cannot divide by zero.')
except TypeError:
    print('Sorry, there is a type error.')
except Exception:
    print('Sorry, there is some kind of error.')
```

Assignment 1

```
def price():
    cost = float(input("Enter Price: "))
    return cost

print('The price is: ', price())
```



```
Enter Price: a

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-10-c403fbe52567> in <module>()
      3     return cost
      4 
----> 5 print('The price is: ', price())

<ipython-input-10-c403fbe52567> in price()
      1 def price():
----> 2     cost = float(input("Enter Price: "))
      3     return cost
      4 
      5 print('The price is: ', price())

ValueError: could not convert string to float: 'a'
```

This program code will run into error when the user inputs illegal values.

Apply the exception handling method so that Python will not run into an error and quit from the program.

In your solution, you should allow the user to re-enter the price value.

Assignment 2

```
import sqlite3

db = sqlite3.connect('airline.db')

c = db.cursor()
c.execute('''CREATE TABLE flights (\
           id INTEGER PRIMARY KEY AUTOINCREMENT,\
           origin VARCHAR(20) NOT NULL,\
           destination VARCHAR(20) NOT NULL,\
           duration INTEGER NOT NULL);''')

db.commit()
db.close()
```

If the database already contain the table flights, then running the following code will result in an **Operational Error**. Modify the program to handle the exception raised.

Assignment 3

```
import sqlite3
import csv
db = sqlite3.connect('airline.db')
c = db.cursor()

f = open("flights.csv")
reader = csv.reader(f)
for o, dest, dur in reader:
    c.execute('''INSERT INTO flights \
        (origin, destination, duration) \
        VALUES (:origin, :destination,
               :duration)''',
        {"origin":o, "destination":dest,
              "duration":dur})

db.commit()
db.close()
```

If this file does not exists, then this program code will result in an **FileNotFoundError**.

Modify the code to handle the exception raised.

The End