Data Science Fundamentals: Python | Table of Contents (../index.ipynb)

Module 4. Defining Functions (./01_mod_defining_functions.ipynb) | Errors And Exceptions (./02_mod_errors_and_exceptions.ipynb) | **Exercises (./03_mod_exercises.ipynb)**

Module 4: Practice Exercises

Exercise 1

We add a Leap Day on February 29, almost every four years. The leap day is an extra, or intercalary day and we add it to the shortest month of the year, February.

See Hacker Rank Problem (https://www.hackerrank.com/challenges/write-a-function/problem)

In the Gregorian calendar three criteria must be taken into account to identify leap years:

- The year can be evenly divided by 4, is a leap year, unless:
- The year can be evenly divided by 100, it is NOT a leap year, unless:
- The year is also evenly divisible by 400. Then it is a leap year.

This means that in the Gregorian calendar, the years 2000 and 2400 are leap years, while 1800, 1900, 2100, 2200, 2300 and 2500 are NOT leap years.

Task

You are given the year, and you have to write a function to check if the year is leap or not. Simply answer if TRUE or FALSE.

Note that you have to complete the function and remaining code is given as template.

Input Format

Read y, the year that needs to be checked.

Output Format

Output is taken care of by the template. Your function must return a boolean value (True/False)

In [1]:
```
# Python Program to Check Leap Year using Elif Statement
# This is not correct answer only example

year = int(input("Please Enter the Year Number you wish: "))

if (year%400 == 0):
          print("%d is a Leap Year" %year)
elif (year%100 == 0):
          print("%d is Not a Leap Year" %year)
elif (year%4 == 0):
          print("%d is a Leap Year" %year)
else:
          print("%d is Not a Leap Year" %year)
```

```
Please Enter the Year Number you wish: 2017
2017 is Not a Leap Year
```

In [2]:
```
def is_leap(year):
    leap = False

    # Write your logic here
    if (year %400 == 0):
        # print("%d is a Leap Year" %year)
        return True

    elif (year %100 == 0):
        # print("%d is Not a Leap Year" %year)
        return False

    elif (year %4 == 0):
        # print("%d is a Leap Year" %year)
        return True

    # else:
        # print("%d is Not a Leap Year" %year)
    return False

    return leap

# year = int(input("Please Enter the Year Number you wish: "))
year = int(input())

print(is_leap(year))
```

```
2019
False
```

Submit Solution to Hacker Rank (https://www.hackerrank.com/challenges/write-a-function/submissions)

Exercise 2

Exceptions

Errors detected during execution are called exceptions.

See Hacker Rank Problem (https://www.hackerrank.com/challenges/exceptions/problem)

Examples:

ZeroDivisionError
This error is raised when the second argument of a division or modulo operation is zero.

In [3]:
```
a = '1'
b = '0'
print(int(a) / int(b))
# ZeroDivisionError: integer division or modulo by zero
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-3-0c83ceda7c17> in <module>
      1 a = '1'
      2 b = '0'
----> 3 print(int(a) / int(b))
      4 # ZeroDivisionError: integer division or modulo by zero

ZeroDivisionError: division by zero
```

ValueError
This error is raised when a built-in operation or function receives an argument that has the right type but an inappropriate value.

In [5]:
```
a = '1'
b = '#'
print(int(a)/int(b))
# ValueError: invalid literal for int() with base 10: '#'
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-5-be0505a86ec2> in <module>
      1 a = '1'
      2 b = '#'
----> 3 print(int(a)/int(b))
      4 # ValueError: invalid literal for int() with base 10: '#'

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

Handling Exceptions

The statements try and except can be used to handle selected exceptions. A try statement may have more than one except clause to specify handlers for different exceptions.

#Code try: print 1/0 except ZeroDivisionError as e: print "Error Code:",e

Output

Error Code: integer division or modulo by zero

Task

You are given two values a and b. Perform integer division and print a/b.

Input Format

The first line contains T, the number of test cases. The next T lines each contain the space separated values of a and b.

Output Format

Print the value of a/b. In the case of ZeroDivisionError or ValueError, print the error code.

Sample Input

3 1 0 2 $ 3 1

Sample Output

Error Code: integer division or modulo by zero Error Code: invalid literal for int() with base 10: '$' 3

In [6]:
```
#This is a comment
import traceback

try:
    1/0
except Exception:
    traceback.print_exc()
```

```
Traceback (most recent call last):
  File "<ipython-input-6-29a6ae9a5860>", line 5, in <module>
    1/0
ZeroDivisionError: division by zero
```

In [7]:
```
# Enter your code here. Read input from STDIN. Print output to STDOUT
test = int(input())
for i in range(0, test):
    try:
        a, b = map(int, input().split())
        print (a//b)
    except ZeroDivisionError as e:
        print("Error Code:", e)
    except ValueError as e:
        print("Error Code:", e)
```

```
-1
```

Submit Solution To Hacker Rank (https://www.hackerrank.com/challenges/exceptions/submissions)

Submissions

Take two screenshots showing your completed solution to both exercises and upload those images as your submission. Submit your screenshots here (https://titus.techtalentsouth.com/mod/assign/view.php?id=28676)

Extra Points

1. ***Optional Exercise (Extra Points)*** Iterate over a list of over 10 numbers. This list is made from user input, and might not have 10 numbers in it. After you reach the end of the list, make the rest of the numbers to be interpreted as a 0.

In [6]:
```
def do_stuff_with_number(n):
    print(n)

def catch_this():
    #the_list = input("Input a list of ten numbers: ")
    the_list = (1, 2, 3, 4, 5)
    for i in range(10):
        try:
            do_stuff_with_number(the_list[i])
        except IndexError: # Raised when accessing a non-existing index of a l
ist
            do_stuff_with_number(0)

catch_this()
```

```
1
2
3
4
5
0
0
0
0
0
```

1. ***Optional Exercise (Extra Points)*** Try to access an array element whose index is out of bound and handle the corresponding exception. Write the Python program to handle simple runtime error.

In [9]:
```
a = [1, 2, 3]
try:
    print("Second element = %d" %(a[1]))

    # Throws error since there are only 3 elements in array
    print("Fourth element = %d" %(a[3]))

except IndexError:
    print("An error occurred")
```

```
Second element = 2
An error occurred
```

1. ***Optional Exercise (Extra Points)***. Given a list iterate it and display numbers which are divisible by 5 and if you find number greater than 150 stop the loop iteration.

In [10]:
```
list1 = [12, 15, 32, 42, 55, 75, 122, 132, 150, 180, 200]
for item in list1:
    if (item > 150):
        break
    if(item % 5 == 0):
        print(item)
```

```
15
55
75
150
```

Expected Output 15 55 75 150

Module 4. Defining Functions (./01_mod_defining_functions.ipynb) | Errors And Exceptions (./02_mod_errors_and_exceptions.ipynb) | **Exercises (./03_mod_exercises.ipynb)**
Top

In [ ]:
```

```