Day 9

# Python for Data Science

## Exception Handling

- An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions.
- In general, when a Python script encounters a situation that it cannot cope with, it raises an exception.
- An exception is a Python object that represents an error.
- When a Python script raises an exception, it must either handle the exception immediately otherwise it terminates and quits.
- If you have some suspicious code that may raise an exception, you can defend your program by placing the suspicious code in a try: block.
- After the try: block, include an except: statement, followed by a block of code which handles the problem as elegantly as possible.

## Common Exceptions

- ZeroDivisionError
- NameError
- ValueError
- IOError
- EOFError
- IdentationError

## ZeroDivisionError

```
print(1/0)

# If a number is divided by 0, it gives a ZeroDivisionError.
try:
  1/0
except ZeroDivisionError:
  print('This code gives a ZeroDivisionError.')
```

This code gives a ZeroDivisionError.

```
nlis = []
count = 0
try:
  mean = count/len(nlis)
  print('The mean value is', mean)
except ZeroDivisionError:
  print('This code gives a ZeroDivisionError')
```

This code gives a ZeroDivisionError

```
print(count/len(nlis))
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Input In [4], in <cell line: 1>()
----> 1 print(count/len(nlis))

ZeroDivisionError: division by zero
```

```
# The following code is like 1/0.
try:
  True/False
except ZeroDivisionError:
  print('The code gives a ZeroDivisionError.')
```

The code gives a ZeroDivisionError.

```
print(True/False)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Input In [8], in <cell line: 1>()
----> 1 print(True/False)

ZeroDivisionError: division by zero
```

## ZeroDivisionError

```
nlis = []
count = 0
try:
  mean = count/len(nlis)
  print('The mean value is', mean)
except ZeroDivisionError:
  print('This code gives a ZeroDivisionError')
# Since the variable 'mean' is not defined, it gives us a 'NameEr-
ror
print(mean)
```

This code gives a ZeroDivisionError

```
---------------------------------------------------------------------------
NameError                 Traceback (most recent call last)
Input In [9], in <cell line: 9>()
      7     print('This code gives a ZeroDivisionError')
      8 # Since the variable 'mean' is not defined, it gives us a 'NameError
----> 9 print(mean)

NameError: name 'mean' is not defined
```

```
# Define a function giving a NameError
def addition(x, y):
  z = x + y
  return z
print('This function gives a NameError.')
total = add(3.14, 1.618)
print(total)
```

This function gives a NameError.

```
---------------------------------------------------------------------------
NameError                 Traceback (most recent call last)
Input In [11], in <cell line: 6>()
    4   return z
    5 print('This function gives a NameError.')
----> 6 total = add(3.14, 1.618)
    7 print(total)

NameError: name 'add' is not defined
```

```
# Since 'Mustafa' is not defined, the following code gives us a
'NameError.'
try:
  name = (Mustafa)
  print(name, 'today is your wedding day.')
except NameError:
  print('This code gives a NameError.')
name = (Mustafa)
print(name, 'today is your wedding day.')
```

This code gives a NameError.

```
---------------------------------------------------------------------------
NameError                 Traceback (most recent call last)
Input In [12], in <cell line: 7>()
    5 except NameError:
    6   print('This code gives a NameError.')
----> 7 name = (Mustafa)
    8 print(name, 'today is your wedding day.')

NameError: name 'Mustafa' is not defined
```

## IndexError

```
nlis = [0.577, 1.618, 2.718, 3.14, 6, 28, 37, 1729]
try:
  nlis[10]
except IndexError:
  print('This code gives us a IndexError.')
print(nlis[10])
```

This code gives us a IndexError.

```
---------------------------------------------------------------------------
IndexError                Traceback (most recent call last)
Input In [13], in <cell line: 6>()
    4 except IndexError:
    5   print('This code gives us a IndexError.')
----> 6 print(nlis[10])

IndexError: list index out of range
```

```
# You can also supplytake this error type with tuple
tuple_sample = (0.577, 1.618, 2.718, 3.14, 6, 28, 37, 1729)
try:
  tuple_sample[10]
except IndexError:
  print('This code gives us a IndexError.')
print(tuple_sample[10])
```

This code gives us a IndexError.

```
---------------------------------------------------------------------------
IndexError                Traceback (most recent call last)
Input In [14], in <cell line: 7>()
    5 except IndexError:
    6   print('This code gives us a IndexError.')
----> 7 print(tuple_sample[10])

IndexError: tuple index out of range
```

## KeyError

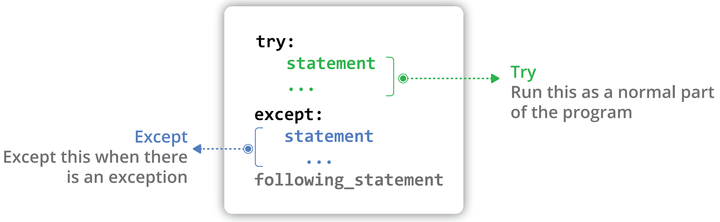
```
dictionary = {'euler_constant': 0.577, 'golden_ratio': 1.618}
try:
  dictonary = dictionary['euler_number']
except KeyError:
  print('This code gives us a KeyError.')
dictonary = dictionary['euler_number']
print(dictonary)
```

```
This code gives us a KeyError.
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [15], in <cell line: 6>()
     4 except KeyError:
     5   print('This code gives us a KeyError.')
----> 6 dictonary = dictionary['euler_number']
     7 print(dictonary)

KeyError: 'euler_number'
```

## Exception Handling

•try/except




```
try:
  print(name)
except NameError:
  print('Since the variable name is not defined, the function gives
a NameError.')
```

```
Since the variable name is not defined, the function gives a NameError.
```

•Since the variable name is not defined, the function gives a NameError.

```
num1 = float(input('Enter a number:'))
print('The entered value is', num1)
try:
  num2 = float(input('Enter a number:'))
  print('The entered value is', num2)
  value = num1/num2
  print('This process is running with value = ', value)
except:
  print('This process is not running.')
```

```
Enter a number:2
The entered value is 2.0
Enter a number:0
The entered value is 0.0
This process is not running.
```

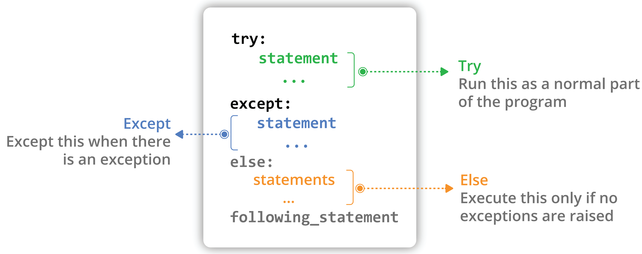
## Multiple Except Blocks

•try/except/except etc.

```
num1 = float(input('Enter a number:'))
print('The entered value is', num1)
try:
  num2 = float(input('Enter a number:'))
  print('The entered value is', num2)
  value = num1/num2
  print('This process is running with value = ', value)
except ZeroDivisionError:
  print('This functi on gives a ZeroDivisionError since a number
cannot divide by 0.')
except ValueError:
  print('You should provide a number.')
except:
  print('Soething went wrong!')
```

```
Enter a number:2
The entered value is 2.0
Enter a number:8
The entered value is 8.0
This process is running with value =  0.25
```

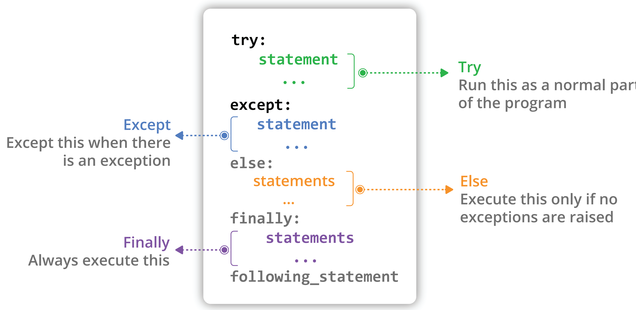
•try/except/else




```
num1 = float(input('Enter a number:'))
print('The entered value is', num1)
try:
  num2 = float(input('Enter a number:'))
  print('The entered value is', num2)
  value = num1/num2
except ZeroDivisionError:
  print('This functi on gives a ZeroDivisionError since a number
cannot divide by 0.')
except ValueError:
  print('You should provide a number.')
except:
  print('Soething went wrong!')
else:
  print('This process is running with value = ', value)
```

Enter a number:2
The entered value is 2.0
Enter a number:4
The entered value is 4.0
This process is running with value = 0.5

•try/except/else/finally




```
num1 = float(input('Enter a number:'))
print('The entered value is', num1)
try:
  num2 = float(input('Enter a number:'))
  print('The entered value is', num2)
  value = num1/num2
except ZeroDivisionError:
  print('This functi on gives a ZeroDivisionError since a number
cannot divide by 0.')
except ValueError:
  print('You should provide a number.')
except:
  print('Soething went wrong!')
else:
  print('This process is running with value = ', value)
finally:
  print('The process is completed.')
```

Enter a number:2
The entered value is 2.0
Enter a number:4
The entered value is 4.0
This process is running with value = 0.5
The process is completed.

•Multiple except clauses

```
num1 = float(input('Enter a number:'))
print('The entered value is', num1)
try:
  num2 = float(input('Enter a number:'))
  print('The entered value is', num2)
  value = num1/num2
except (ZeroDivisionError, NameError, ValueError): #Multi ple
except clauses
  print('This functi on gives a ZeroDivisionError, NameError or
ValueError.')
except:
  print('Soething went wrong!')
else:
  print('This process is running with value = ', value)
finally:
  print('The process is completed.')
```

Enter a number:3
The entered value is 3.0
Enter a number:0
The entered value is 0.0
This functi on gives a ZeroDivisionError, NameError or ValueError.
The process is completed.

# Raising in exception

- Using the 'raise' keyword, the programmer can throw an exception when a certain condition is reached.

```
num = int(input('Enter a number:'))
print('The entered value is', num)
try:
  if num>1000 and num %2 == 0 or num %2 !=0:
    raise Exception('Do not allow to the even numbers higher than
1000.')
except:
  print('Even or odd numbers higher than 1000 are not allowed!')
else:
  print('This process is running with value = ', num)
finally:
  print('The process is completed.')
```

```
Enter a number:1008
The entered value is 1008
Even or odd numbers higher than 1000 are not allowed!
The process is completed.
```