Python Tutorial

Created by Mustafa Germec, PhD

10. Exception Handling in Python

- 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

In [1]:

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

print(1/0)
```

```
This code gives a ZeroDivisionError.

---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_5432/3605061481.py in <module>
      5     print('This code gives a ZeroDivisionError.')
      6 
----> 7 print(1/0)

ZeroDivisionError: division by zero
```

In [2]:

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

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

```
This code gives a ZeroDivisionError

---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_5432/2225123637.py in <module>
      7     print('This code gives a ZeroDivisionError')
      8 
----> 9 print(count/len(nlis))

ZeroDivisionError: division by zero
```

In [3]:

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

print(True/False)
```

```
The code gives a ZeroDivisionError.

---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_5432/3531407864.py in <module>
      5     print('The code gives a ZeroDivisionError.')
      6 
----> 7 print(True/False)

ZeroDivisionError: division by zero
```

NameError

In [4]:

```
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 'NameError
print(mean)
```

```
This code gives a ZeroDivisionError

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_5432/1642249892.py in <module>
      8
      9 # Since the variable 'mean' is not defined, it gives us a 'NameError
---> 10 print(mean)

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

In [5]:

```
try:
    y = x+5
except NameError:
    print('This code gives a NameError.')

print(y)
```

```
This code gives a NameError.

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_5432/115043188.py in <module>
      4     print('This code gives a NameError.')
      5
----> 6 print(y)

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

In [6]:

```
# 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)
~\AppData\Local\Temp/ipykernel_5432/3845321401.py in <module>
      5
      6 print('This function gives a NameError.')
----> 7 total = add(3.14, 1.618)
      8 print(total)

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

In [7]:

```
# 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)
~\AppData\Local\Temp/ipykernel_5432/367854978.py in <module>
      6     print('This code gives a NameError.')
      7
----> 8 name = (Mustafa)
      9 print(name, 'today is your wedding day.')

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

IndexError

In [8]:

```
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)
~\AppData\Local\Temp/ipykernel_5432/4262347625.py in <module>
      5     print('This code gives us a IndexError.')
      6 
----> 7 print(nlis[10])

IndexError: list index out of range
```

In [9]:

```
# 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)
~\AppData\Local\Temp/ipykernel_5432/3170854299.py in <module>
      6     print('This code gives us a IndexError.')
      7 
----> 8 print(tuple_sample[10])

IndexError: tuple index out of range
```

KeyError

In [10]:

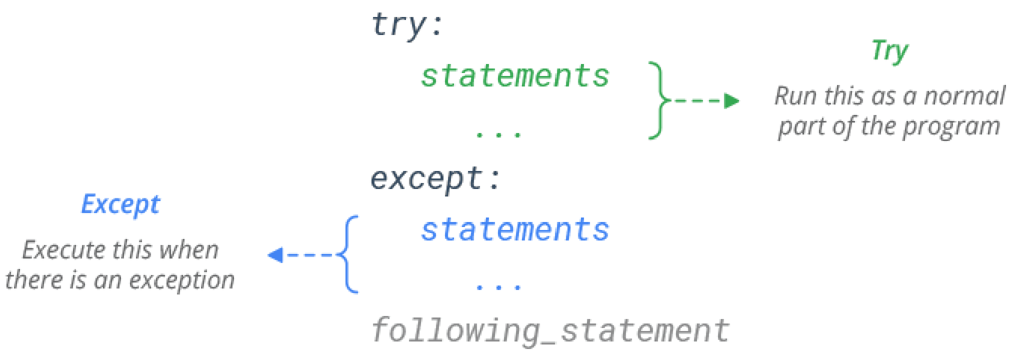
```python
dictionary = {'euler_constant': 0.577, 'golden_ratio': 1.618}
try:
    dictonary = dictionary['euler_number']
except KeyError:
    print('This code gives us a KeyError.')

dictonary = dictionary['euler_number']
print(dictonary)
```

```
This code gives us a KeyError.

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_5432/669363184.py in <module>
      5     print('This code gives us a KeyError.')
      6 
----> 7 dictonary = dictionary['euler_number']
      8 print(dictonary)

KeyError: 'euler_number'
```

You can find more [Error Types (https://docs.python.org/3/library/exceptions.html?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NSkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkPY0101ENSkillsNetwork19487395-2021-01-01)](https://docs.python.org/3/library/exceptions.html?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NSkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkPY0101ENSkillsNetwork19487395-2021-01-01) from this connection.

Exception Handling

try/except


try:
statements
...
Try
Run this as a normal part of the program
except:
Except
Execute this when there is an exception
statements
...
following_statement

In [11]:

```python
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.
```

In [1]:

```python
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.')
```

```
The entered value is 3.14
The entered value is 0.577
This process is running with value =  5.441941074523397
```

Multiple Except Blocks

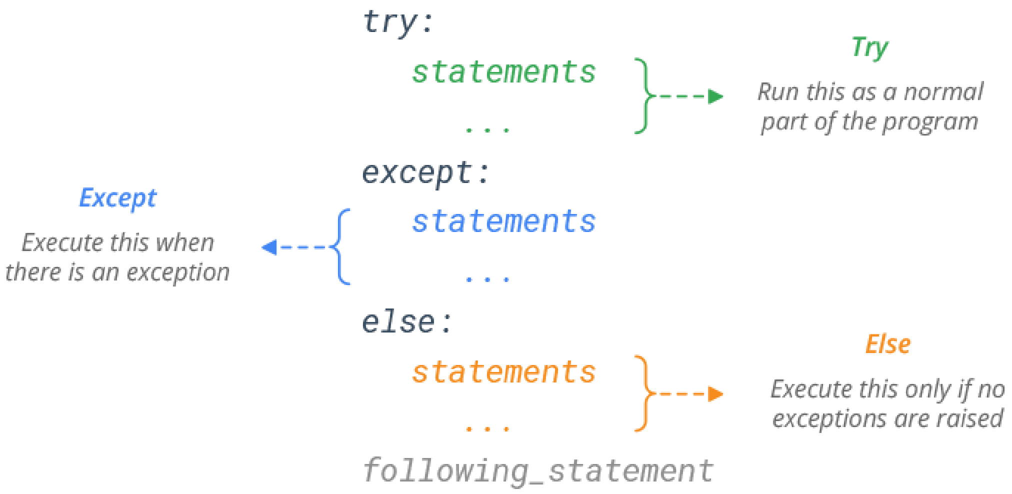
try/except/except etc.

In [2]:

```python
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 function gives a ZeroDivisionError since a number cannot divide by 0.')
except ValueError:
    print('You should provide a number.')
except:
    print('Soething went wrong!')
```

```
The entered value is 2.718
The entered value is 0.0
This function gives a ZeroDivisionError since a number cannot divide by 0.
```

try/except/else

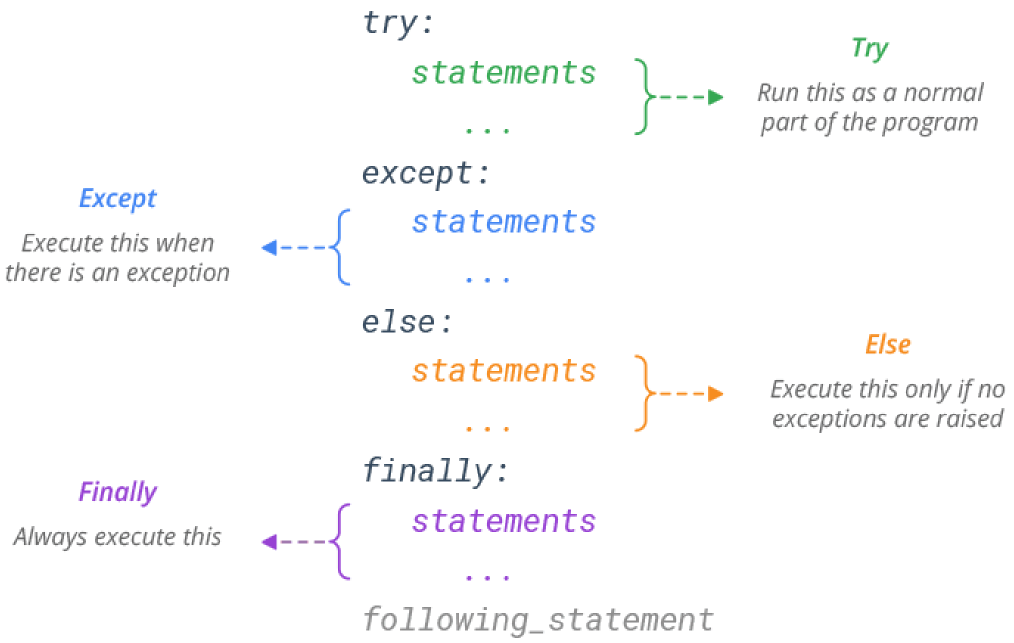

try:
statements
...
Try
Run this as a normal part of the program
except:
Except
Execute this when there is an exception
statements
...
else:
statements
...
Else
Execute this only if no exceptions are raised
following_statement

In [3]:

```python
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 function 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)
```

```
The entered value is 37.0
The entered value is 1.618
This process is running with value =  22.867737948084052
```

try/except/else/finally



In [5]:

```
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 function 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.')
```

```
The entered value is 1.618
The entered value is 0.577
This process is running with value =  2.8041594454072793
The process is completed.
```

Multiple except clauses

In [6]:

```
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):  #Multiple except clauses
    print('This function 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.')
```

```
The entered value is 3.14
The entered value is 0.0
This function 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.

In [7]:

```
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.')
```

```
The entered value is 1006
Even or odd numbers higher than 1000 are not allowed!
The process is completed.
```