# ERROR HANDLING IN PYTHON

## ERROR

- Errors are problems in the program that the program should not recover from.
- If at any point in the program an error occurs,the program exits.
- Error in Python can be of two types i.e. Syntax errors and Exceptions.

- SYNTAX ERROR:

```
        -This error is caused by wrong syntax in the code.
        -It leads to the termination of the program.
```

- EXCEPTION ERROR:

```
        -Exceptions are raised when the program is syntactically correct
but the code resulted in an error.
        -This error does not stop the execution of the program,
        -however, it changes the normal flow of the program.
```

## EXCEPTION HANDLING

- excepts can be handled with try and except block which will generate exception of error ad execute the program

- Let's use try and exception block

In [1]:

```
try:
  print(x)
except:
  print("An exception occurred")
```

```
An exception occurred
```

---

The above code will give an exception as x is not defined

## try and exception block

- try block raises an error, the except block will be executed.

- Without the try block, the program will crash and raise an error

---

In [4]:

```
# Write a program to ask user about numbers for 5 times if user gives 1 add score.
# method 1 : without defining of score variable
```

```
try:
    for i in range (1,6):
        choice = int(input("enter number from 1 to 10 "))
        if choice == 1:
            score+=1
except:
    print("something went wrong")
```

```
enter number from 1 to 10 5
enter number from 1 to 10 1
something went wrong
```

In [6]:

```
# Write a program to ask user about numbers for 5 times if user gives 1 add score.

#method 2 : with definaing of score variable

try:
    for i in range (1,6):
        choice = int(input("enter number from 1 to 10 "))
        if choice == 1:
            score+=1
    print(score)
except:
    print("something went wrong")
```

```
enter number from 1 to 10 2
enter number from 1 to 10 1
enter number from 1 to 10 5
enter number from 1 to 10 1
enter number from 1 to 10 3
2
```

## Exception

- We must always try to be specific while trying to catch exception.
- **Exception keyword catches almost all exceptions**

In [10]:

```
try:
  print(x)

except Exception:
  print("Something else went wrong")
```

```
Something else went wrong
```

## MANY EXCEPTION BLOCK

- You can define as many exception blocks as you want,
- e.g. if you want to execute a special block of code for a special kind of error:

In [11]:

```
try:
  print(x)
except NameError:
  print("Variable x is not defined")
except:
  print("Something else went wrong")
```

```
Variable x is not defined
```

In [13]:

```python
try:
    for i in range (1,6):
        choice = int(input("enter number from 1 to 6 "))
        if choice == 1:
            point+=1
    print(point)
except NameError:
  print("Variable POINT is not defined")
except Exception:
  print("Something else went wrong")
```

```
enter number from 1 to 10 1
Variable SCORE is not defined
```

In [14]:

```python
# Adding print block after the try and except block
try:
    for i in range (1,6):
        choice = int(input("enter number from 1 to 10 "))
        if choice == 1:
            point+=1
    print(point)
except NameError:
  print("Variable POINT is not defined")
except Exception:
  print("Something else went wrong")

print("i continued the program")
```

```
enter number from 1 to 10 5
enter number from 1 to 10 1
Variable SCORE is not defined
i continued the program
```

---

The above code shows it executes the further code after handling the exception

---

In [16]:

```python
# Task  : use try and except block for file operation
```

In [17]:

```python
name = input('enter the file name :')
try:
    f = open(name+'.txt')
    f.close()
except FileNotFoundError:
    print("enetr a correct file name")
except Exception:
    print("Something went wrong")
print("i continued to execute")
```

```
enter the file name :test
i continued to execute
```

In [18]:

```python
name = input('enter the file name :')
try:
    f = open(name+'.txt')
    f.close()
except FileNotFoundError:
    print("enetr a correct file name")
except Exception:
    print("Something went wrong")
print("i continued to execute")
```

```
enter the file name :rejin
enetr a correct file name
i continued to execute
```

---

### we can use as keyword to print the inbuilt pretified message

## - Task : print error by using keyword

In [20]:

```
name = input('enter the file name :')
try:
    f = open(name+'.txt')
    data = f.read()
    f.close()
except FileNotFoundError as e:
    print(e)
except Exception as e:
    print(e)
print("i continued to execute")
```

```
enter the file name :rejin
[Errno 2] No such file or directory: 'rejin.txt'
i continued to execute
```

---

### Handling errors separetley

The above code is not correct it is always good to try to catch single line of codes error

---

In [9]:

```
# when file name is correct

name = input('enter the file name :')
try:
    f = open(name+'.txt')
except FileNotFoundError as e:
    print(e)
except Exception as e:
    print(e)
else:
    data = f.read()
    print(data)
    f.close()
```

```
enter the file name :test
1) This is a test file!!
```

```
2) This is the second line
3) This is the third line
4) This is the fourth line
5) This is the fifth line
6) This is the sixth line
7) This is the seventh line
8) This is the eight line
9) This is the ninth line
10) This is the thenth line
```

In [10]:

```python
# when file name is incorrect

name = input('enter the file name :')
try:
    file = open(name+'.txt')
except FileNotFoundError as e:
    print(e)
except Exception as e:
    print(e)
else:
    data = file.read()
    print(data)
    file.close()
```

```
enter the file name :rejin
[Errno 2] No such file or directory: 'rejin.txt'
```

## finally

- The finally block always runs irrespective of the try else and excpet
- In the code below finally block run irrestive of error

In [6]:

```python
# When file name is correct

name = input('enter the file name :')
try:
    f = open(name+'.txt')
except FileNotFoundError as e:
    print(e)
except Exception:
    print("this fine should not be read")
else:
    data = f.read()
    f.close()
    print(data)
finally:
    print("sorry")
```

```
enter the file name :test
1) This is a test file!!
2) This is the second line
3) This is the third line
4) This is the fourth line
5) This is the fifth line
6) This is the sixth line
7) This is the seventh line
8) This is the eight line
9) This is the ninth line
```

```
10) This is the thenth line
sorry
```

In [7]:

```
# When file name is incorrect

name = input('enter the file name :')
try:
    f = open(name+'.txt')
except FileNotFoundError as e:
    print(e)
except Exception:
    print("this fine should not be read")
else:
    data = f.read()
    f.close()
    print(data)
finally:
    print("sorry")
```

```
enter the file name :rejin
[Errno 2] No such file or directory: 'rejin.txt'
sorry
```

## raise

- instead of using predefine errors we can raise our own error

In [12]:

```
name = input('enter the file name :')
try:
    f = open(name+'.txt')
    if f.name == 'test.txt':
        raise Exception
except FileNotFoundError as e:
    print(e)
except Exception:
    print("this fine should not be read")
else:
    data = f.read()
    f.close()
    print(data)
finally:
    print("sorry")
```

```
enter the file name :test
this fine should not be read
sorry
```

---

**REVISION**

1. What are errors?
2. Using the try and the except block
3. Handling multiple exception
4. finally
5. raise

---

## HOMEWORK**

1. Make necessary changes to the below code so that does not error out when non integer numbers are entered.

In [1]:
```
a = int(input("enter a number: "))
b = int(input("enter a number: "))

c = a+b
print(c)
```

```
enter a number: a
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-1-ca15e6aba358> in <module>
----> 1 a = int(input("enter a number: "))
      2 b = int(input("enter a number: "))
      3 
      4 c = a+b
      5 print(c)

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

In [2]:
```
# type your solution here
```