When an error occurs, or exception as we call it, Python will normally stop and generate an error message.

These exceptions can be handled using the try statement:

The try block lets you test a block of code for errors.

The except block lets you handle the error.

try....except

In [3]:

```
#The try block will generate an error, because x is not defined:

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

```
An exception occurred
```

In [4]:

```
try:
    a=5
    b='0'
    print(a/b)
except:
    print('Some error occurred.')
print("Out of try except blocks.")
```

```
Some error occurred.
Out of try except blocks.
```

In [32]:

```
# initialize the amount variable
amount = 10000

# check that You are eligible to
# purchase Dsa Self Paced or not
if(amount > 2999)
print("You are eligible to purchase Dsa Self Paced")
```

```
  File "<ipython-input-32-8d94ddc68478>", line 6
    if(amount > 2999)
                     ^
SyntaxError: invalid syntax
```

In [33]:

```
# initialize the amount variable
marks = 10000

# perform division with 0
a = marks / 0
print(a)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-33-5f312778a383> in <module>()
      3 
      4 # perform division with 0
----> 5 a = marks / 0
      6 print(a)

ZeroDivisionError: division by zero
```

In [37]:

```
# Python program to handle simple runtime error
#Python 3

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

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

In [51]:

```
# Python code to illustrate
# working of try()
def divide(x, y):
    try:
        # Floor Division : Gives only Fractional Part as Answer
        result = x // y
        print("Yeah ! Your answer is :", result)
    except ZeroDivisionError:
        print("Sorry ! You are dividing by zero ")

# Look at parameters and note the working of Program
divide(3, 2)
```

```
Yeah ! Your answer is : 1
```

In [ ]:

```
# Python code to illustrate
# working of try()
def divide(x, y):
    try:
        # Floor Division : Gives only Fractional Part as Answer
        result = x // y
        print("Yeah ! Your answer is :", result)
    except ZeroDivisionError:
        print("Sorry ! You are dividing by zero ")

# Look at parameters and note the working of Program
divide(3, 0)
```

Many Exceptions

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 [5]:

```
#Print one message if the try block raises a NameError and another for other errors:

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

```
Variable x is not defined
```

In [6]:

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

```
Something else went wrong
```

In [7]:

```
#You can use the else keyword to define a block of code to be executed if no errors were ra

try:
  print("Hello")
except:
  print("Something went wrong")
else:
  print("Nothing went wrong")
```

```
Hello
Nothing went wrong
```

In [8]:

```
try:
  print(Hello)
except:
  print("Something went wrong")
else:
  print("Nothing went wrong")
```

```
Something went wrong
```

In [41]:

```
# Program to handle multiple errors with one
# except statement
# Python 3

def fun(a):
    if a < 4:

        # throws ZeroDivisionError for a = 3
        b = a/(a-3)

    # throws NameError if a >= 4
    print("Value of b = ", b)

try:
    fun(3)

# note that braces () are necessary here for
# multiple exceptions
except ZeroDivisionError:
    print("ZeroDivisionError Occurred and Handled")
except NameError:
    print("NameError Occurred and Handled")
```

```
ZeroDivisionError Occurred and Handled
```

In [43]:

```python
# Program to handle multiple errors with one
# except statement
# Python 3

def fun(a):
    if a < 4:

        # throws ZeroDivisionError for a = 3
        b = a/(a-3)

    # throws NameError if a >= 4
    print("Value of b = ", b)

try:
    fun(5)

# note that braces () are necessary here for
# multiple exceptions
except ZeroDivisionError:
    print("ZeroDivisionError Occurred and Handled")
except NameError:
    print("NameError Occurred and Handled")
```

```
NameError Occurred and Handled
```

In [44]:

```python
# Program to depict else clause with try-except
# Python 3
# Function which returns a/b
def AbyB(a , b):
    try:
        c = ((a+b) / (a-b))
    except ZeroDivisionError:
        print ("a/b result in 0")
    else:
        print (c)

# Driver program to test above function
AbyB(2.0, 3.0)
```

```
-5.0
```

In [45]:

```
# Program to depict else clause with try-except
# Python 3
# Function which returns a/b
def AbyB(a , b):
    try:
        c = ((a+b) / (a-b))
    except ZeroDivisionError:
        print ("a/b result in 0")
    else:
        print (c)

# Driver program to test above function
AbyB(3.0, 3.0)
```

```
a/b result in 0
```

finally

The finally block, if specified, will be executed regardless if the try block raises an error or not.

In [10]:

```
try:
  print('x')
except:
  print("Something went wrong")
finally:
  print("The 'try except' is finished")
```

```
x
The 'try except' is finished
```

In [49]:

```
# Python program to demonstrate finally

# No exception Exception raised in try block
try:
    k = 5//0 # raises divide by zero exception.
    print(k)

# handles zerodivision exception
except ZeroDivisionError:
    print("Can't divide by zero")

finally:
    # this block is always executed
    # regardless of exception generation.
    print('This is always executed')
```

```
Can't divide by zero
This is always executed
```

In [53]:

```
try:
    a = 10/0
    print (a)
except ArithmeticError:
        print ("This statement is raising an arithmetic exception.")
else:
    print ("Success.")
```

```
This statement is raising an arithmetic exception.
```

In [56]:

```
try:
    a = [1, 2, 3]
    print (a[3])
except LookupError:
    print ("Index out of bound error.")
else:
    print ("Success")
```

```
Index out of bound error.
```

Raise an exception

As a Python developer you can choose to throw an exception if a condition occurs.

To throw (or raise) an exception, use the raise keyword.

In [30]:

```
#Raise an error and stop the program if x is lower than 0:

x = -1

if x < 0:
  raise Exception("Sorry, no numbers below zero")
```

```
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
<ipython-input-30-ab80e4044a59> in <module>()
      4
      5 if x < 0:
----> 6   raise Exception("Sorry, no numbers below zero")

Exception: Sorry, no numbers below zero
```

In [31]:

```
#Raise a TypeError if x is not an integer:

x = "hello"

if not type(x) is int:
  raise TypeError("Only integers are allowed")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-31-f489534722f4> in <module>()
      4
      5 if not type(x) is int:
----> 6   raise TypeError("Only integers are allowed")

TypeError: Only integers are allowed
```

In [50]:

```
# Program to depict Raising Exception

try:
    raise NameError("Hi there") # Raise Error
except NameError:
    print ("An exception")
    raise # To determine whether the exception was raised or not
```

```
An exception

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-50-87bf2bdaba1c> in <module>()
      2
      3 try:
----> 4         raise NameError("Hi there") # Raise Error
      5 except NameError:
      6         print ("An exception")

NameError: Hi there
```

```
In [52]:
```

```python
# Python code to illustrate
# working of try()
def divide(x, y):
    try:
        # Floor Division : Gives only Fractional Part as Answer
        result = x // y
        print("Yeah ! Your answer is :", result)
    except ZeroDivisionError:
        print("Sorry ! You are dividing by zero ")

# Look at parameters and note the working of Program
divide(3, 0)
```

```
Sorry ! You are dividing by zero
```

```
In [ ]:
```