Exceptions:

Exceptions are the uncertain errors that may occur during the execution of the code. These exceptions if not handled properly interrupts the normal execution and ends the code abruptly throwing a large error. Exception handling allows us to handle errors and helps continue normal execution of the flow.

Type of Exceptions:

1. Key Error

In [9]:
```
#When we call a key which dont exist
a={'x':2,'y':3}

print (a['z'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-9-e1c60ffd540a> in <module>
      1 a={'x':2,'y':3}
      2
----> 3 print (a['z'])

KeyError: 'z'
```

2. Arithmaic Error (ZeroDivisionError)

In [43]:
```
# Division by zero
a = 10/0
print (a)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-43-488d44c3c5b9> in <module>
----> 1 a = 10/0
      2 print (a)

ZeroDivisionError: division by zero
```

3. Assertion Error

In [10]:
```
#Error raised when some condition is not met
a,b = 5,4

assert (a==b) , 'Unequal values'
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-10-2dbfea22344e> in <module>
      1 a,b = 5,4
      2
----> 3 assert (a==b) , 'Unequal values'

AssertionError: Unequal values
```

4. Import Error

In [12]:
```
#raised when we commit typo while importing a package
import mathi
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-12-99c0136c5646> in <module>
----> 1 import mathi

ModuleNotFoundError: No module named 'mathi'
```

5. Index Error

In [47]:
```
#when index is not present
x = [1,2,3,4,5]

print (x[5])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-47-b2d96cb8d8bb> in <module>
      1 x = [1,2,3,4,5]
      2
----> 3 print (x[5])

IndexError: list index out of range
```

6. Name Error

In [49]:
```
#when no variable of that name exist
print (abc)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-49-473c89b54ed8> in <module>
----> 1 print (abc)

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

7. Type Error

In [52]:
```
#while trying to perform operations on two different data-types
a = 5
b = 'cat'
c= a+b
print (c)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-52-04493688b93c> in <module>
      1 a = 5
      2 b = 'cat'
----> 3 c= a+b
      4 print (c)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

8. Value Error

In [13]:
```
#when value not in correct format
x = int('a')
x
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-13-3921ea7ff53c> in <module>
----> 1 x = int('a')
      2 x

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

9. End of file Error

In [1]:
```
#when you forget to put the end bracket or comma
x = int(input('print the number')
```

```
  File "<ipython-input-1-f296759c3ffb>", line 1
    x = int(input('print the number')
                                     ^
SyntaxError: unexpected EOF while parsing
```

Try and Except:

Purpose of Try and Except:

In [1]:
```
#when we commit any error the program throws the alert and stops the executi
#program does not stop abruptly. Hence we use exception handling.
a = 10/0
print ('continue from here')
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-1-882149068c5c> in <module>
      1 #when we commit any error the program throws the alert and stops the ex
ecution. So we need to handle the errors so that our
      2 #program does not stop abruptly. Hence we use exception handling.
----> 3 a = 10/0
      4 print ('continue from here')

ZeroDivisionError: division by zero
```

In [2]:
```
#The above program terminates throwing the error, here we used the same erro
#part of the code is written within the try 'block' and all possible excepti
#the program does not terminates abruptly but it continues from the next lin
try:
    b = 100/0
except (ZeroDivisionError):
    print ('error')
print ('continue from here')
```

```
error
continue from here
```

Try and except:

In [8]:
```
#Another example for excetion handling can be:
##you can give all possible exceptions within the parenthesis and write a co
try :
    a = 3
    if a < 4:

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

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

# note that braces () are necessary here for multiple exceptions
except(ZeroDivisionError, NameError):
    print ("\nError Occurred and Handled")
for i in range(0,5):
    print (i)
```

```
Error Occurred and Handled
0
1
2
3
4
```

In [6]:
```
#In this case each exception is raised with the separate exception head for
try :
    a = 3
    if a < 4:

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

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

# note that braces () are necessary here for multiple exceptions
except(ZeroDivisionError):
    print ("zero division error occured")
except (NameError):
    print ('Name error occured')
# for i in range(0,5):
#     print (i)
```

```
zero division error occured
```

In [6]:
```
#Another example can be:
def fnc (x):
    try:
        rev = 1/int(x)
        return rev
    except ZeroDivisionError as ZDE:
        print ('division not possible by zero')
    except ValueError as ve:
        print ('enter correct value')
p = [0,'t',5,8]
for i in map (fnc,p):
    print (i)
# fnc('i')
# fnc(0)

```

```
division not possible by zero
None
enter correct value
None
0.2
0.125
```

Raise an exception when a condition is not met:

We can also raise a custom exception when some condition is met. This can be done by using 'raise' keyword. In the below example, we have raised an exception when (i==5)

In [8]:
```
for i in range (0,10):
    try:
        if (i==5):
            raise (ValueError)
        print (i)
    except (ValueError):
        print ('Error at value ',i)
```

```
0
1
2
3
4
Error at value  5
6
7
8
9
```

In [9]:
```
try:
    a = str(input('enter positive integer: '))
    if (int(a)>=0):
       print ('Correct value')
    else:
        raise (ValueError)
except ValueError:
    print ('this is the value error1 :', a)
```

```
enter positive integer: -10
this is the value error1 : -10
```

Exception with multiple exception statements:

```
The else and finally block:
when there is an exception the exception block is executed, when there is no exception, else block of code is executed. While finally block of code is always executed whether exception occurs or not.
```

In [77]:
```
try:
    a= int(input('Sum of marks '))
    b= int(input('Overall marks '))

    if b ==0:
        print ('total subjects cant be zero')
        raise (ZeroDivisionError);
    if b <0:
        print ('total subjects less than zero')
        raise (ValueError);
    else:
        percentage = a/b*100
        print ('percentage: ',percentage,'%')

except ValueError as e:
    print ('Value not given in correct format')
except ZeroDivisionError as f:
    print ('Division by zero is not possible')


else:
    if percentage >=75:
        print ('congrats you have secured honors degree')
    else:
        print('you have cleared the exam')
finally:
    print ('Thanks for using this code')

```

```
Sum of marks 20
Overall marks 25
percentage:  80.0 %
congrats you have secured honors degree
Thanks for using this code
```

Assertion Statement

In [58]:
```
a = 11
assert (a<10), 'value is more'
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-58-ae2d0c9589bf> in <module>
      1 a = 11
----> 2 assert (a<10), 'value is more'

AssertionError: value is more
```

In [59]:
```
a = ['f', 0,'g' , 2,3]
for i in a:
    try:

        print ('the value is : ',i)
        rev = 1/int(i)
        print ('rev is :', rev)
    except ValueError:
        print ('Excepion raised 1')
    except ZeroDivisionError:
        print ('Exception raised 2')
```

```
the value is :  f
Excepion raised 1
the value is :  0
Exception raised 2
the value is :  g
Excepion raised 1
the value is :  2
rev is : 0.5
the value is :  3
rev is : 0.3333333333333333
```

In [33]:
```
try:
    a = int(input('Enter the age '))
    if (a < 18):
        print ('value less than 18')
    else:
        raise (ValueError)
except ValueError:
    print ('Value error raised')
```

```
Enter the age 19
Value error raised
```