In [2]:
```
!pip  install numpy
```

```
Requirement already satisfied: numpy in c:\users\administrator\anaconda3\lib\site-packag
es (1.20.1)
```

In [3]:
```
import numpy
```

In [4]:
```
import sys
print(sys.path)
```

```
['C:\\pyhon code\\Python-code\\Python_code', 'C:\\Users\\Administrator\\anaconda3\\pytho
n38.zip', 'C:\\Users\\Administrator\\anaconda3\\DLLs', 'C:\\Users\\Administrator\\anacon
da3\\lib', 'C:\\Users\\Administrator\\anaconda3', '', 'C:\\Users\\Administrator\\anaconda
a3\\lib\\site-packages', 'C:\\Users\\Administrator\\anaconda3\\lib\\site-packages\\locke
t-0.2.1-py3.8.egg', 'C:\\Users\\Administrator\\anaconda3\\lib\\site-packages\\win32',
'C:\\Users\\Administrator\\anaconda3\\lib\\site-packages\\win32\\lib', 'C:\\Users\\Admin
istrator\\anaconda3\\lib\\site-packages\\Pythonwin', 'C:\\Users\\Administrator\\anaconda
3\\lib\\site-packages\\IPython\\extensions', 'C:\\Users\\Administrator\\.ipython']
```

In [5]:
```
import math
```

In [6]:
```
type(math)
```

Out[6]:
```
module
```

In [7]:
```
print(math)
```

```
<module 'math' (built-in)>
```

In [11]:
```
print(math.sqrt(100))
```

```
10.0
```

In [10]:
```
math.pow(3,2)
```

Out[10]:
```
9.0
```

In [12]:
```
help(math)
```

```
Help on built-in module math:

NAME
    math

DESCRIPTION
    This module provides access to the mathematical functions
    defined by the C standard.

FUNCTIONS
    acos(x, /)
        Return the arc cosine (measured in radians) of x.
```

```
    acosh(x, /)
        Return the inverse hyperbolic cosine of x.

    asin(x, /)
        Return the arc sine (measured in radians) of x.

    asinh(x, /)
        Return the inverse hyperbolic sine of x.

    atan(x, /)
        Return the arc tangent (measured in radians) of x.

    atan2(y, x, /)
        Return the arc tangent (measured in radians) of y/x.

        Unlike atan(y/x), the signs of both x and y are considered.

    atanh(x, /)
        Return the inverse hyperbolic tangent of x.

    ceil(x, /)
        Return the ceiling of x as an Integral.

        This is the smallest integer >= x.

    comb(n, k, /)
        Number of ways to choose k items from n items without repetition and without ord
er.

        Evaluates to n! / (k! * (n - k)!) when k <= n and evaluates
        to zero when k > n.

        Also called the binomial coefficient because it is equivalent
        to the coefficient of k-th term in polynomial expansion of the
        expression (1 + x)**n.

        Raises TypeError if either of the arguments are not integers.
        Raises ValueError if either of the arguments are negative.

    copysign(x, y, /)
        Return a float with the magnitude (absolute value) of x but the sign of y.

        On platforms that support signed zeros, copysign(1.0, -0.0)
        returns -1.0.

    cos(x, /)
        Return the cosine of x (measured in radians).

    cosh(x, /)
        Return the hyperbolic cosine of x.

    degrees(x, /)
        Convert angle x from radians to degrees.

    dist(p, q, /)
        Return the Euclidean distance between two points p and q.

        The points should be specified as sequences (or iterables) of
        coordinates.  Both inputs must have the same dimension.

        Roughly equivalent to:
            sqrt(sum((px - qx) ** 2.0 for px, qx in zip(p, q)))

    erf(x, /)
        Error function at x.
```

```
    erfc(x, /)
        Complementary error function at x.

    exp(x, /)
        Return e raised to the power of x.

    expm1(x, /)
        Return exp(x)-1.

        This function avoids the loss of precision involved in the direct evaluation of
exp(x)-1 for small x.

    fabs(x, /)
        Return the absolute value of the float x.

    factorial(x, /)
        Find x!.

        Raise a ValueError if x is negative or non-integral.

    floor(x, /)
        Return the floor of x as an Integral.

        This is the largest integer <= x.

    fmod(x, y, /)
        Return fmod(x, y), according to platform C.

        x % y may differ.

    frexp(x, /)
        Return the mantissa and exponent of x, as pair (m, e).

        m is a float and e is an int, such that x = m * 2.**e.
        If x is 0, m and e are both 0.  Else 0.5 <= abs(m) < 1.0.

    fsum(seq, /)
        Return an accurate floating point sum of values in the iterable seq.

        Assumes IEEE-754 floating point arithmetic.

    gamma(x, /)
        Gamma function at x.

    gcd(x, y, /)
        greatest common divisor of x and y

    hypot(...)
        hypot(*coordinates) -> value

        Multidimensional Euclidean distance from the origin to a point.

        Roughly equivalent to:
            sqrt(sum(x**2 for x in coordinates))

        For a two dimensional point (x, y), gives the hypotenuse
        using the Pythagorean theorem:  sqrt(x*x + y*y).

        For example, the hypotenuse of a 3/4/5 right triangle is:

            >>> hypot(3.0, 4.0)
            5.0

    isclose(a, b, *, rel_tol=1e-09, abs_tol=0.0)
```

```
    Determine whether two floating point numbers are close in value.

      rel_tol
        maximum difference for being considered "close", relative to the
        magnitude of the input values
      abs_tol
        maximum difference for being considered "close", regardless of the
        magnitude of the input values

    Return True if a is close in value to b, and False otherwise.

    For the values to be considered close, the difference between them
    must be smaller than at least one of the tolerances.

    -inf, inf and NaN behave similarly to the IEEE 754 Standard.  That
    is, NaN is not close to anything, even itself.  inf and -inf are
    only close to themselves.

isfinite(x, /)
    Return True if x is neither an infinity nor a NaN, and False otherwise.

isinf(x, /)
    Return True if x is a positive or negative infinity, and False otherwise.

isnan(x, /)
    Return True if x is a NaN (not a number), and False otherwise.

isqrt(n, /)
    Return the integer part of the square root of the input.

ldexp(x, i, /)
    Return x * (2**i).

    This is essentially the inverse of frexp().

lgamma(x, /)
    Natural logarithm of absolute value of Gamma function at x.

log(...)
    log(x, [base=math.e])
    Return the logarithm of x to the given base.

    If the base not specified, returns the natural logarithm (base e) of x.

log10(x, /)
    Return the base 10 logarithm of x.

log1p(x, /)
    Return the natural logarithm of 1+x (base e).

    The result is computed in a way which is accurate for x near zero.

log2(x, /)
    Return the base 2 logarithm of x.

modf(x, /)
    Return the fractional and integer parts of x.

    Both results carry the sign of x and are floats.

perm(n, k=None, /)
    Number of ways to choose k items from n items without repetition and with order.

    Evaluates to n! / (n - k)! when k <= n and evaluates
    to zero when k > n.
```

```
        If k is not specified or is None, then k defaults to n
        and the function returns n!.

        Raises TypeError if either of the arguments are not integers.
        Raises ValueError if either of the arguments are negative.

    pow(x, y, /)
        Return x**y (x to the power of y).

    prod(iterable, /, *, start=1)
        Calculate the product of all the elements in the input iterable.

        The default start value for the product is 1.

        When the iterable is empty, return the start value.  This function is
        intended specifically for use with numeric values and may reject
        non-numeric types.

    radians(x, /)
        Convert angle x from degrees to radians.

    remainder(x, y, /)
        Difference between x and the closest integer multiple of y.

        Return x - n*y where n*y is the closest integer multiple of y.
        In the case where x is exactly halfway between two multiples of
        y, the nearest even value of n is used. The result is always exact.

    sin(x, /)
        Return the sine of x (measured in radians).

    sinh(x, /)
        Return the hyperbolic sine of x.

    sqrt(x, /)
        Return the square root of x.

    tan(x, /)
        Return the tangent of x (measured in radians).

    tanh(x, /)
        Return the hyperbolic tangent of x.

    trunc(x, /)
        Truncates the Real x to the nearest Integral toward 0.

        Uses the __trunc__ magic method.

DATA
    e = 2.718281828459045
    inf = inf
    nan = nan
    pi = 3.141592653589793
    tau = 6.283185307179586

FILE
    (built-in)
```

In [13]:

```
import random
```

In [17]:
```
random.randint(0,100)
```

Out[17]:
```
59
```

In [23]:
```
random.seed(1000)
random.randint(0,100)
```

Out[23]:
```
99
```

In [39]:
```
random.seed(1000)
random.randint(0,100)
#random.seed(1000)
random.randint(0,100)
```

Out[39]:
```
99
```

In [ ]:
```
def randint(0,100, ts = 100):
    return ts*x %100
```

In [24]:
```
import os
```

In [25]:
```
os.getcwd()
```

Out[25]:
```
'C:\\pyhon code\\Python-code\\Python_code'
```

In [26]:
```
os.listdir()
```

Out[26]:
```
['.ipynb_checkpoints',
 'Comprehensions.ipynb',
 'Dictionaries.ipynb',
 'dsml-course-main',
 'firstnotebook.ipynb',
 'Functions.ipynb',
 'gl',
 'Lists.ipynb',
 'Map, Reduce & Filter.ipynb',
 'Python Refresher1.ipynb',
 'Python refresher2.ipynb',
 'Python refresher3.ipynb',
 'Python refresher4.ipynb',
 'Sets.ipynb',
 'Tuples.ipynb']
```

In [27]:
```
os.mkdir('hello')
```

In [28]:
```
os.listdir()
```

Out[28]:
```
['.ipynb_checkpoints',
 'Comprehensions.ipynb',
 'Dictionaries.ipynb',
```

```
'dsml-course-main',
'firstnotebook.ipynb',
'Functions.ipynb',
'gl',
'hello',
'Lists.ipynb',
'Map, Reduce & Filter.ipynb',
'Python Refresher1.ipynb',
'Python refresher2.ipynb',
'Python refresher3.ipynb',
'Python refresher4.ipynb',
'Sets.ipynb',
'Tuples.ipynb']
```

In [30]:
```
os.removedirs('hello')
```

In [31]:
```
import sys
```

In [32]:
```
import math as mth
```

In [33]:
```
mth.sqrt(4)
```

Out[33]: 2.0

In [34]:
```
from math import *
```

In [35]:
```
sqrt(4)
```

Out[35]: 2.0

In [36]:
```
from math import sqrt, factorial
```

In [ ]:
```
random - randint
numpy - randint
```

In [37]:
```
import numpy as np
import pandas  as pd
```

In [38]:
```
from sklearn import linear_model
```

In [ ]:
```
linear_model.LinearRegression()
```

In [ ]:
```
sklearn
  linear_model.py
     def LinearRegression()
```

In [ ]:
```
A/B/a.py -> f
from A.B import a
```

In [ ]:
```
Break : 10 24
```

In [40]:
```
print(f)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-40-fc0364975534> in <module>
----> 1 print(f)

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

In [41]:
```
if(5>3):
```

```
  File "<ipython-input-41-68aebac1bd63>", line 1
    if(5>3):
           ^
SyntaxError: unexpected EOF while parsing
```

In [42]:
```
import dontknow
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-42-bcd87af70373> in <module>
----> 1 import dontknow

ModuleNotFoundError: No module named 'dontknow'
```

In [43]:
```
type(__builtins__)
```

Out[43]:
```
module
```

In [44]:
```
print(dir(__builtins__) )
```

```
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BlockingIOErro
r', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'ChildProcessError', 'ConnectionAb
ortedError', 'ConnectionError', 'ConnectionRefusedError', 'ConnectionResetError', 'Depre
cationWarning', 'EOFError', 'Ellipsis', 'EnvironmentError', 'Exception', 'False', 'FileE
xistsError', 'FileNotFoundError', 'FloatingPointError', 'FutureWarning', 'GeneratorExi
t', 'IOError', 'ImportError', 'ImportWarning', 'IndentationError', 'IndexError', 'Interr
uptedError', 'IsADirectoryError', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'Memor
yError', 'ModuleNotFoundError', 'NameError', 'None', 'NotADirectoryError', 'NotImplement
ed', 'NotImplementedError', 'OSError', 'OverflowError', 'PendingDeprecationWarning', 'Pe
rmissionError', 'ProcessLookupError', 'RecursionError', 'ReferenceError', 'ResourceWarni
ng', 'RuntimeError', 'RuntimeWarning', 'StopAsyncIteration', 'StopIteration', 'SyntaxErr
or', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'TimeoutError', 'True',
'TypeError', 'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeEr
ror', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarning', 'ValueError', 'Warning',
'WindowsError', 'ZeroDivisionError', '__IPYTHON__', '__build_class__', '__debug__', '__d
oc__', '__import__', '__loader__', '__name__', '__package__', '__spec__', 'abs', 'all',
'any', 'ascii', 'bin', 'bool', 'breakpoint', 'bytearray', 'bytes', 'callable', 'chr', 'c
lassmethod', 'compile', 'complex', 'copyright', 'credits', 'delattr', 'dict', 'dir', 'di
```

```
splay', 'divmod', 'enumerate', 'eval', 'exec', 'filter', 'float', 'format', 'frozenset',
'get_ipython', 'getattr', 'globals', 'hasattr', 'hash', 'help', 'hex', 'id', 'input', 'i
nt', 'isinstance', 'issubclass', 'iter', 'len', 'license', 'list', 'locals', 'map', 'ma
x', 'memoryview', 'min', 'next', 'object', 'oct', 'open', 'ord', 'pow', 'print', 'proper
ty', 'range', 'repr', 'reversed', 'round', 'set', 'setattr', 'slice', 'sorted', 'staticm
ethod', 'str', 'sum', 'super', 'tuple', 'type', 'vars', 'zip']
```

In [45]:

```
count = 0

for elem in dir(__builtins__):
    if "Error" in elem:  # if "Error" is a sub-string of element of __builtins__
        print(elem)
        count += 1

print("Count: ", count)
```

```
ArithmeticError
AssertionError
AttributeError
BlockingIOError
BrokenPipeError
BufferError
ChildProcessError
ConnectionAbortedError
ConnectionError
ConnectionRefusedError
ConnectionResetError
EOFError
EnvironmentError
FileExistsError
FileNotFoundError
FloatingPointError
IOError
ImportError
IndentationError
IndexError
InterruptedError
IsADirectoryError
KeyError
LookupError
MemoryError
ModuleNotFoundError
NameError
NotADirectoryError
NotImplementedError
OSError
OverflowError
PermissionError
ProcessLookupError
RecursionError
ReferenceError
RuntimeError
SyntaxError
SystemError
TabError
TimeoutError
TypeError
UnboundLocalError
UnicodeDecodeError
UnicodeEncodeError
UnicodeError
UnicodeTranslateError
ValueError
```

```
WindowsError
ZeroDivisionError
Count:  49
```

In [46]:
```
2/3
```

Out[46]:
```
0.6666666666666666
```

In [47]:
```
2/0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-47-e8326a161779> in <module>
----> 1 2/0

ZeroDivisionError: division by zero
```

In [50]:
```
a = int(input("Enter first number:"))
b = int(input("Enter second number:"))

# Let's divide the two numbers

print(a/b)
print("helo")
```

```
Enter first number:3
Enter second number:0
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-50-16162c6483b6> in <module>
      4 # Let's divide the two numbers
      5
----> 6 print(a/b)
      7 print("helo")

ZeroDivisionError: division by zero
```

In [53]:
```
a = int(input("Enter first number:"))
b = int(input("Enter second number:"))

try:
    result = a/b  # a statement that might lead to an error goes inside try block
    print("Result of division is: ", result)

except: # handles the situation when an error actually occurs inside try block
    print("There is some error")
print("hello")
```

```
Enter first number:4
Enter second number:0
There is some error
hello
```

In [54]:
```
lst = [2, 0, 'hello', None]

for ele in lst:
    try:
```

```
        print("Current element :", ele)
        res = 5/int(ele)
        print("Result :", res)

    # There is a class called `Exception`
    # Let's save any raised exception in variable e

    except Exception as e:
        print("Error caused :", e)
        # print(sys.exc_info()[0] )

    print("-"*50)

print("\nExecution went well.")
```

```
Current element : 2
Result : 2.5
--------------------------------------------------
Current element : 0
Error caused : division by zero
--------------------------------------------------
Current element : hello
Error caused : invalid literal for int() with base 10: 'hello'
--------------------------------------------------
Current element : None
Error caused : int() argument must be a string, a bytes-like object or a number, not 'No
neType'
--------------------------------------------------

Execution went well.
```

In [55]:

```
lst = [2, 0, 'hello', None]

for ele in lst:
    try:
        print("Current ele :", ele)
        res = 5/int(ele)
        print("Result :", res)

    # if value for division is invalid, we handle it separately
    except ValueError as ve:
        print("ValueError occured ..fix it bro:", ve)

    # if denomenator is zero, we handle is separately
    except ZeroDivisionError as ze:
        print("Dont't divide by zero. Are you asleep? : ", ze)

    # if error is neither of the above two, we handle it generically
    except Exception as e:
        print("Error caused :", e)

    print("-"*50)

print("\nThere are 100 more lines.")
```

```
Current ele : 2
Result : 2.5
--------------------------------------------------
Current ele : 0
Dont't divide by zero. Are you asleep? :  division by zero
--------------------------------------------------
```

```
Current ele : hello
ValueError occured ..fix it bro: invalid literal for int() with base 10: 'hello'
------------------------------------------------
Current ele : None
Error caused : int() argument must be a string, a bytes-like object or a number, not 'No
neType'
------------------------------------------------

There are 100 more lines.
```

In [59]:

```
try:
    ans = 5/2
    print(ans)
except Exception as e:
    print("Error occured :",e)
else:
    print("No exception occured")

print("hello")
```

```
2.5
No exception occured
hello
```

In [61]:

```
try:
    ans = 5/0
    print(ans)
except Exception as e:
    print("Error occured :",e)
    return -1;
else:
    print("No exception occured")
finally:
    # f.close()  # close the file
    # db.close() # close the connection with database
    print("I will always excute.")

print("hello")
```

```
Error occured : division by zero
I will always excute.
```

In [ ]:

```
try:
    ans = 5/0
    print(ans)
except Exception as e:
    print("Error occured :",e)
else:
    print("No exception occured")

print("I will always execute")
```

In [62]:

```
raise Exception("Something Wrong")
print("100 lines of code")
```

```
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
<ipython-input-62-f228b5499fb1> in <module>
```

```
----> 1 raise Exception("Something Wrong")
      2 print("100 lines of code")

Exception: Something Wrong
```

In [65]:

```
try:
    name = input("Enter your name: ")

    if len(name) < 3:
        raise Exception("Name cannot be less than 3 charaters.")

    print("Hello, ",name)

except Exception as e:
    print("Error occured :", e)

print("100 lines of code")
```

```
Enter your name: s
Error occured : Name cannot be less than 3 charaters.
100 lines of code
```

In [ ]: