In [ ]:
```
Raj has developed a addition program
Raj knows that in the entire world he only cretaed addition program

Raj has decided to share the entire addition code to the total world


1) How can he share?

He went to Anaconda organization
He told them ==== Addition ==== every one should

2) OKay we will take your code
   Put that code in a pacakge

Package name: Addition
He handover to Anaconda organization

Who ever installed anaconda, this package automatically download in thier laptop
Greenc color pop up : All the packages are installing downloading
```

In [ ]:
```
package is avialable in your laptop

how to use that
```

import

- import is a keyword to use the pacakge
- syntax is :
  - **import package_name**

In [ ]:
```
# package name: random
# package name : math
# package name : time
# package name : cv2
```

In [1]:
```
import random
```

In [2]:
```
import math
```

In [3]:
```
import time
```

In [4]:
```
import cv2
```

In [5]:
```
import tensorflow
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[5], line 1
----> 1 import tensorflow

ModuleNotFoundError: No module named 'tensorflow'
```

Module not found error

- whenever we import any package
- if it is available in our laptop then no error
- if it is not available then we will get error

In [7]:
```
a=10
a
```

Out[7]:
```
10
```

In [ ]:
```
 raj addition
     subtraction
     multiplication
     division
Name: ALLINONE

import ALLINONE
```

dir

- It is a directory
- to know how many sub methods are avaiable in a single package
- syntax is
  - dir(package_name)

In [8]:
```
# package name :  random
dir(random)
```

Out[8]:
```
['BPF',
 'LOG4',
 'NV_MAGICCONST',
 'RECIP_BPF',
 'Random',
 'SG_MAGICCONST',
 'SystemRandom',
 'TWOPI',
 '_ONE',
 '_Sequence',
 '_Set',
 '__all__',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 '_accumulate',
 '_acos',
 '_bisect',
 '_ceil',
 '_cos',
 '_e',
 '_exp',
 '_floor',
 '_index',
 '_inst',
 '_isfinite',
 '_log',
 '_os',
 '_pi',
 '_random',
 '_repeat',
 '_sha512',
 '_sin',
 '_sqrt',
 '_test',
 '_test_generator',
 '_urandom',
 '_warn',
 'betavariate',
 'choice',
 'choices',
 'expovariate',
 'gammavariate',
 'gauss',
 'getrandbits',
 'getstate',
 'lognormvariate',
 'normalvariate',
 'paretovariate',
 'randbytes',
 'randint',
 'random',
 'randrange',
 'sample',
 'seed',
```

```
 'setstate',
 'shuffle',
 'triangular',
 'uniform',
 'vonmisesvariate',
 'weibullvariate']
```

In [9]:

```
# step-1:  import package name
# step-2:  dir(package name)

import random
dir(random)
```

Out[9]:
```
['BPF',
 'LOG4',
 'NV_MAGICCONST',
 'RECIP_BPF',
 'Random',
 'SG_MAGICCONST',
 'SystemRandom',
 'TWOPI',
 '_ONE',
 '_Sequence',
 '_Set',
 '__all__',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 '_accumulate',
 '_acos',
 '_bisect',
 '_ceil',
 '_cos',
 '_e',
 '_exp',
 '_floor',
 '_index',
 '_inst',
 '_isfinite',
 '_log',
 '_os',
 '_pi',
 '_random',
 '_repeat',
 '_sha512',
 '_sin',
 '_sqrt',
 '_test',
 '_test_generator',
 '_urandom',
 '_warn',
 'betavariate',
 'choice',
 'choices',
 'expovariate',
 'gammavariate',
 'gauss',
 'getrandbits',
 'getstate',
 'lognormvariate',
 'normalvariate',
 'paretovariate',
 'randbytes',
 'randint',
 'random',
 'randrange',
 'sample',
 'seed',
```

```
 'setstate',
 'shuffle',
 'triangular',
 'uniform',
 'vonmisesvariate',
 'weibullvariate']
```

How to use this methods

In [10]:
```
dir(random)
```

Out[10]:
```
['BPF',
 'LOG4',
 'NV_MAGICCONST',
 'RECIP_BPF',
 'Random',
 'SG_MAGICCONST',
 'SystemRandom',
 'TWOPI',
 '_ONE',
 '_Sequence',
 '_Set',
 '__all__',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 '_accumulate',
 '_acos',
 '_bisect',
 '_ceil',
 '_cos',
 '_e',
 '_exp',
 '_floor',
 '_index',
 '_inst',
 '_isfinite',
 '_log',
 '_os',
 '_pi',
 '_random',
 '_repeat',
 '_sha512',
 '_sin',
 '_sqrt',
 '_test',
 '_test_generator',
 '_urandom',
 '_warn',
 'betavariate',
 'choice',
 'choices',
 'expovariate',
 'gammavariate',
 'gauss',
 'getrandbits',
 'getstate',
 'lognormvariate',
 'normalvariate',
 'paretovariate',
 'randbytes',
 'randint',
 'random',
 'randrange',
 'sample',
 'seed',
```

```
 'setstate',
 'shuffle',
 'triangular',
 'uniform',
 'vonmisesvariate',
 'weibullvariate']
```

In [11]:
```
#package_name.method_name
#package_name: random
#method_name: randint

random.randint
```

Out[11]:
```
<bound method Random.randint of <random.Random object at 0x000001ECDBB4D000>>
```

In [ ]:
```
# step-1: import package_name
# step-2: dir(package_name)
# step-3: package_name.method_name
```

In [12]:
```
random.omkar
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[12], line 1
----> 1 random.omkar

AttributeError: module 'random' has no attribute 'omkar'
```

Attribute error

- If the method is not available we will get attribute error

In [ ]:
```
# You know how to import the package
# you know how many methods are available inside the package
# you know how to call the method
      # package_name.method_name
      # nallagoni.omkar

# if package is not there: Module not found error
# if method is not there : Attribute error
```

How to use

help

- in order to understand the use of method
- first call the method
- then apply help
- syntax is
  - help(packagename.methodname)

In [13]:
```
# package name: random
# method name: randint
```

```
help(random.randint)
```

```
Help on method randint in module random:

randint(a, b) method of random.Random instance
    Return random integer in range [a, b], including both end points.
```

In [20]:
```
# you want to get a random number between 10,100
random.randint(1,5)
```

Out[20]: 2

In [22]:
```
random.randint[10,20]  # error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[22], line 1
----> 1 random.randint[10,20]

TypeError: 'method' object is not subscriptable
```

In [21]:
```
import random
dir(random) # So many
help(random.randint)
random.randint(10,20)
```

```
Help on method randint in module random:

randint(a, b) method of random.Random instance
    Return random integer in range [a, b], including both end points.
```

Out[21]: 10

In [23]:
```
a=10
b=20
a
b
```

Out[23]: 20

In [25]:
```
import math
dir(math)
```

Out[25]:
```
['__doc__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'acos',
 'acosh',
 'asin',
 'asinh',
 'atan',
 'atan2',
 'atanh',
 'cbrt',
 'ceil',
 'comb',
 'copysign',
 'cos',
 'cosh',
 'degrees',
 'dist',
 'e',
 'erf',
 'erfc',
 'exp',
 'exp2',
 'expm1',
 'fabs',
 'factorial',
 'floor',
 'fmod',
 'frexp',
 'fsum',
 'gamma',
 'gcd',
 'hypot',
 'inf',
 'isclose',
 'isfinite',
 'isinf',
 'isnan',
 'isqrt',
 'lcm',
 'ldexp',
 'lgamma',
 'log',
 'log10',
 'log1p',
 'log2',
 'modf',
 'nan',
 'nextafter',
 'perm',
 'pi',
 'pow',
 'prod',
 'radians',
 'remainder',
 'sin',
 'sinh',
 'sqrt',
```

```
 'tan',
 'tanh',
 'tau',
 'trunc',
 'ulp']
```

In [26]:
```
help(math.sqrt)
```

```
Help on built-in function sqrt in module math:

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

In [27]:
```
random.randint(10,20)
math.sqrt(25)
```

Out[27]:
```
5.0
```

In [28]:
```
import random
dir(random)
help(random.randint)
```

```
Help on method randint in module random:

randint(a, b) method of random.Random instance
    Return random integer in range [a, b], including both end points.
```

In [29]:
```
random.randint
```

Out[29]:
```
<bound method Random.randint of <random.Random object at 0x000001ECDBB4D000>>
```

- bound method means brackets are missed

In [30]:
```
random.randint()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[30], line 1
----> 1 random.randint()

TypeError: Random.randint() missing 2 required positional arguments: 'a' and 'b'
```

In [31]:
```
random.randint(10)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[31], line 1
----> 1 random.randint(10)

TypeError: Random.randint() missing 1 required positional argument: 'b'
```

In [33]:
```
random.randint(10,20)
```

Out[33]:
```
18
```

In [ ]:
```
random.randint
random.randint()
```

```
random.randint(10)
random.randint(10,20)

print()
type()
eval()
input()
```

In [ ]:
```
math.sqrt
```

In [ ]:
```
math.sqrt()
```

In [ ]:
```
math.sqrt(5)
```

In [ ]:
```
# Package name: random
    #  method name: randint
    #  method name: random
    # method name : randrange
   # method name : uniform
# Package name : math
    # Method name: sqrt
    # Method name: pi  # constant no brackets
    # Method name: pow
    # Method name: sin
    # method name: e  # constant no brackets
    # method name: factorial
    # method name: gcd

# Package name: time
     # Method name: sleep   i will explain

# Package name : sys
    # Method name: version
```

Math

In [34]:
```
help(math.pow)
```

```
Help on built-in function pow in module math:

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

In [35]:
```
math.pow(2,3)  #  2***3 =8
```

Out[35]:
```
8.0
```

In [36]:
```
math.pi
```

Out[36]:
```
3.141592653589793
```

In [37]:
```
math.e
```

Out[37]:
```
2.718281828459045
```

In [38]:
```
help(math.factorial)
```

```
Help on built-in function factorial in module math:

factorial(n, /)
    Find n!.

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

In [39]:
```
math.factorial(5)
```

Out[39]:
```
120
```

In [40]:
```
help(math.sin)
```

```
Help on built-in function sin in module math:

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

In [41]:
```
math.sin(90)
```

Out[41]:
```
0.8939966636005579
```

In [42]:
```
import sys
sys.version
```

Out[42]:
```
'3.11.7 | packaged by Anaconda, Inc. | (main, Dec 15 2023, 18:05:47) [MSC v.191
6 64 bit (AMD64)]'
```

In [43]:
```
import random
help(random.random)
```

```
Help on built-in function random:

random() method of random.Random instance
    random() -> x in the interval [0, 1).
```

In [46]:
```
random.random()
```

Out[46]:
```
0.6652047857219057
```

In [47]:
```
help(random.randrange)
```

```
Help on method randrange in module random:

randrange(start, stop=None, step=1) method of random.Random instance
    Choose a random item from range(stop) or range(start, stop[, step]).

    Roughly equivalent to ``choice(range(start, stop, step))`` but
    supports arbitrarily large ranges and is optimized for common cases.
```

In [55]:
```
random.randrange(1,20,2)
# 1  2  3  4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
# 1     3    5   7   9    11    13    15    17    19
```

Out[55]:
```
17
```

In [58]:
```
random.randrange(2,20,2)
```

Out[58]:
```
2
```

In [59]:
```
help(random.uniform)
```

```
Help on method uniform in module random:

uniform(a, b) method of random.Random instance
    Get a random number in the range [a, b) or [a, b] depending on rounding.
```

In [63]:
```
random.uniform(1,10)
```

Out[63]:
```
8.526437337719578
```

In [65]:
```
import time
help(time.sleep)
```

```
Help on built-in function sleep in module time:

sleep(...)
    sleep(seconds)

    Delay execution for a given number of seconds.  The argument may be
    a floating point number for subsecond precision.
```

In [68]:
```
print('hello')
time.sleep(5)
print('bye')
```

```
hello
bye
```

In [69]:
```
sys.version
```

Out[69]:
```
'3.11.7 | packaged by Anaconda, Inc. | (main, Dec 15 2023, 18:05:47) [MSC v.191
6 64 bit (AMD64)]'
```

In [ ]:
```
# Comments
```