In [ ]:
```
# Generally packages wil not teach in this early

# D.S and Ai packages
```

import

In [1]:
```
# import <package_name>
```

In [2]:
```
# package name: random
import random
```

In [3]:
```
# package name: math
import math
```

In [4]:
```
# package name: keyword
import keyword
```

In [5]:
```
# Package name: string
import string
```

In [6]:
```
# package name: time
import time
```

In [7]:
```
# package name: cv2
import cv2
```

In [ ]:
```
# if packae is not available
# No module found
```

In [8]:
```
random
```

Out[8]:
```
<module 'random' from 'C:\\Users\\prash\\anaconda31\\Lib\\random.py'>
```

dir

In [9]:
```
import random
dir(random)
```

Out[9]:
```
['BPF',
 'LOG4',
 'NV_MAGICCONST',
 'RECIP_BPF',
 'Random',
 'SG_MAGICCONST',
 'SystemRandom',
 'TWOPI',
 '_ONE',
 '_Sequence',
 '__all__',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 '_accumulate',
 '_acos',
 '_bisect',
 '_ceil',
 '_cos',
 '_e',
 '_exp',
 '_fabs',
 '_floor',
 '_index',
 '_inst',
 '_isfinite',
 '_lgamma',
 '_log',
 '_log2',
 '_os',
 '_pi',
 '_random',
 '_repeat',
 '_sha512',
 '_sin',
 '_sqrt',
 '_test',
 '_test_generator',
 '_urandom',
 '_warn',
 'betavariate',
 'binomialvariate',
 'choice',
 'choices',
 'expovariate',
 'gammavariate',
 'gauss',
 'getrandbits',
 'getstate',
 'lognormvariate',
 'normalvariate',
 'paretovariate',
 'randbytes',
 'randint',
 'random',
```

```
 'randrange',
 'sample',
 'seed',
 'setstate',
 'shuffle',
 'triangular',
 'uniform',
 'vonmisesvariate',
 'weibullvariate']
```

help

In [10]:
```
randint
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[10], line 1
----> 1 randint

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

- we cant call directly method
- we want to call first package and then call method

- step-1: import package name
- step-2: dir(package_name)
- step-3: help(package_name.method_name)

In [13]:
```
import random
dir(random)
print(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.

None
```

In [14]:
```
random.randint(10,50)
```

Out[14]: 13

In [15]:
```
random.randint(1,10)
```

Out[15]: 2

In [ ]:
```
setp-1 import package name
step-2 dir(package name)
step-3 help(package name.method name)
step-4 apply on the operation
```

In [16]:
```
import random
dir(random) # I want to use random
help(random.randint) # Here I will understand the story
random.randint(1,10)
```

```
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[16]: 7

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

Note

- **bracket meaning**
  - [] square bracket means inclusive (समावेशक)
  - () round bracket exclusive (अनन्य)

Always remember use () round bracktes only

In [18]:
```
help(random.random)

# No need to use help always
# we can simply use shift + tab inside the round bracktes()
# It will works like help
```

```
Help on built-in function random:

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

In [32]:
```
a = random.random()
a1 = round(a,2)
a1
```

Out[32]: 0.82

In [35]:
```
# pckage name: random
# method name: randrange
import random
random.randrange(1,20,2)
```

Out[35]: 5

In [40]:
```
import random
v1 = random.randint(1,20)
v2 = random.random()
v3 = random.randrange(1,30,2)
print(v1,round(v2,2),v3, sep='  ')
```

```
6  0.99  29
```

In [41]:
```
import math
dir(math)
```

Out[41]:
```
['__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',
```

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

In [45]:
```
math.pow(2,6)
```

Out[45]: 64.0

In [61]:
```
math.sqrt(16)
```

Out[61]: 4.0

In [62]:
```
math.pi()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[62], line 1
----> 1 math.pi()

TypeError: 'float' object is not callable
```

Note

- pi is a constant
- thats why we no need to give brackets

In [76]:
```
math.pi
```

Out[76]: 3.141592653589793

In [74]:
```
math.sin(13)
```

Out[74]: 0.4201670368266409

In [75]:
```
math.cos(10)
```

Out[75]: -0.8390715290764524

In [77]:
```
math.tan(90)
```

Out[77]: -1.995200412208242

Keyword

In [78]:
```
import keyword
dir(keyword)
```

Out[78]:
```
['__all__',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'iskeyword',
 'issoftkeyword',
 'kwlist',
 'softkwlist']
```

In [79]:
```
import keyword
dir(keyword)
help(keyword.kwlist)
```

```
Help on list object:

class list(object)
 |  list(iterable=(), /)
 |
 |  Built-in mutable sequence.
 |
 |  If no argument is given, the constructor creates a new empty list.
 |  The argument must be an iterable if specified.
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return bool(key in self).
 |
 |  __delitem__(self, key, /)
 |      Delete self[key].
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(self, index, /)
 |      Return self[index].
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __iadd__(self, value, /)
 |      Implement self+=value.
 |
 |  __imul__(self, value, /)
 |      Implement self*=value.
 |
 |  __init__(self, /, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __len__(self, /)
 |      Return len(self).
 |
 |  __lt__(self, value, /)
 |      Return self<value.
 |
 |  __mul__(self, value, /)
 |      Return self*value.
 |
```

```
 |  __ne__(self, value, /)
 |      Return self!=value.
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __reversed__(self, /)
 |      Return a reverse iterator over the list.
 |
 |  __rmul__(self, value, /)
 |      Return value*self.
 |
 |  __setitem__(self, key, value, /)
 |      Set self[key] to value.
 |
 |  __sizeof__(self, /)
 |      Return the size of the list in memory, in bytes.
 |
 |  append(self, object, /)
 |      Append object to the end of the list.
 |
 |  clear(self, /)
 |      Remove all items from list.
 |
 |  copy(self, /)
 |      Return a shallow copy of the list.
 |
 |  count(self, value, /)
 |      Return number of occurrences of value.
 |
 |  extend(self, iterable, /)
 |      Extend list by appending elements from the iterable.
 |
 |  index(self, value, start=0, stop=9223372036854775807, /)
 |      Return first index of value.
 |
 |      Raises ValueError if the value is not present.
 |
 |  insert(self, index, object, /)
 |      Insert object before index.
 |
 |  pop(self, index=-1, /)
 |      Remove and return item at index (default last).
 |
 |      Raises IndexError if list is empty or index is out of range.
 |
 |  remove(self, value, /)
 |      Remove first occurrence of value.
 |
 |      Raises ValueError if the value is not present.
 |
 |  reverse(self, /)
 |      Reverse *IN PLACE*.
 |
 |  sort(self, /, *, key=None, reverse=False)
 |      Sort the list in ascending order and return None.
 |
 |      The sort is in-place (i.e. the list itself is modified) and stable (i.e.
the
 |      order of two equal elements is maintained).
```

```
 |
 |      If a key function is given, apply it once to each list item and sort the
m,
 |      ascending or descending, according to their function values.
 |
 |      The reverse flag can be set to sort in descending order.
 |
 |  ----------------------------------------------------------------------
 |  Class methods defined here:
 |
 |  __class_getitem__(...)
 |      See PEP 585
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs)
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  __hash__ = None
```

In [80]:
```
# simply run it
keyword.kwlist
```

Out[80]:
```
['False',
 'None',
 'True',
 'and',
 'as',
 'assert',
 'async',
 'await',
 'break',
 'class',
 'continue',
 'def',
 'del',
 'elif',
 'else',
 'except',
 'finally',
 'for',
 'from',
 'global',
 'if',
 'import',
 'in',
 'is',
 'lambda',
 'nonlocal',
 'not',
 'or',
 'pass',
 'raise',
 'return',
 'try',
 'while',
 'with',
 'yield']
```

In [81]:
```
len(keyword.kwlist)
```

Out[81]: 35

time

In [83]:
```
import time
dir(time)
time.sleep(5) # it will take 5 sec to show output
print('hello')
```

```
hello
```

In [84]:
```
random.ranin
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[84], line 1
----> 1 random.ranin

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

In [ ]:
```
# we are trying to use the method
```

```
# but it will not available
```

string

In [85]:
```
import string
dir(string)
```

Out[85]:
```
['Formatter',
 'Template',
 '_ChainMap',
 '__all__',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 '_re',
 '_sentinel_dict',
 '_string',
 'ascii_letters',
 'ascii_lowercase',
 'ascii_uppercase',
 'capwords',
 'digits',
 'hexdigits',
 'octdigits',
 'printable',
 'punctuation',
 'whitespace']
```

In [86]:
```
string.ascii_letters
```

Out[86]:
```
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

In [87]:
```
string.ascii_lowercase
```

Out[87]:
```
'abcdefghijklmnopqrstuvwxyz'
```

In [88]:
```
string.ascii_uppercase
```

Out[88]:
```
'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

In [92]:
```
string.printable
```

Out[92]:
```
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&\'()*+,
-./:;<=>?@[\\]^_`{|}~ \t\n\r\x0b\x0c'
```

In [93]:
```
string.punctuation
```

Out[93]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

4th nov

In [ ]:
```
random
math
```

```
string
keyword
time
```

In [94]:
```
#We learn three ways to
```

1st way

In [95]:
```
n1 = 10
n2 = 20
n3 = 30
avg = (n1+n2+n3)/3
avg1 = round(avg,2)
print(f'the avg of {n1}, {n2}, and {n3} is {avg1}')
```

```
the avg of 10, 20, and 30 is 20.0
```

- The above code is hard code, In this code user will fixed values

2nd way

In [96]:
```
n1 = eval(input("Enter a num1:"))
n2 = eval(input("Enter a num2:"))
n3 = eval(input("Enter a num3:"))
avg = (n1+n2+n3)/3
avg1 = round(avg,2)
print(f'the avg of {n1}, {n2}, and {n3} is {avg1}')
```

```
the avg of 23, 34, and 54 is 37.0
```

- The above code is generic code or taking values from the keyboard

3rd way

In [98]:
```
import random
n1 = random.randint(1,30)
n2 = random.randint(1,30)
n3 = random.randint(1,30)
avg = (n1+n2+n3)/3
avg1 = round(avg,2)
print(f'the avg of {n1}, {n2}, and {n3} is {avg1}')
```

```
the avg of 1, 15, and 11 is 9.0
```

- randomly we can give values

Que

In [104…
```
b = 20
h = 30
area = 0.5*b*h
print(f'The area of right angle triangle is {area}')
```

```
The area of right angle triangle is 300.0
```

In [102…
```
b = eval(input('Enter the breadth:'))
h = eval(input('Enter the height:'))
area = 0.5*b*h
print(f'The area of right angle triangle is {area}')
```

```
The area of right angle triangle is 1687.5
```

In [103…
```
b = random.randint(1,100)
h = random.randint(1,100)
area = 0.5*b*h
print(f'The area of right angle triangle is {area}')
```

```
The area of right angle triangle is 176.0
```

Que

In [108…
```
import math
pi =math.pi
r = 25.34
area1 = pi*r*r
area = round(area1,2)
print(f'the are of a circle is {area}')
```

```
the are of a circle is 2017.27
```

In [109…
```
r = eval(input("Enter the radius:"))
pi = math.pi
area1 = pi*r*r
area = round(area1,2)
print(f'the are of a circle is {area}')
```

```
the are of a circle is 19211.59
```

In [110…
```
r = random.randint(0,100)
pi = math.pi
area1 = pi*r*r
area = round(area1,2)
print(f'the are of a circle is {area}')
```

```
the are of a circle is 22698.01
```

Que

In [111…
```
bill = 2000
tip = 100
total_bill = bill+tip
print(f'The total bill is: {total_bill}')
```

```
The total bill is: 2100
```

In [112…
```
bill = eval(input("Enter a bill"))
tip = eval(input('Enter a tip amount'))
total_bill = bill+tip
print(f'The total bill is: {total_bill}')
```

```
The total bill is: 750
```

In [113…
```
bill = random.randint(1,2000)
tip = random.randint(1,50)
total_bill = bill+tip
print(f'The total bill is: {total_bill}')
```

```
The total bill is: 922
```

Que

In [116…
```
bill = 2000
tip = (bill/100)*5
total_bill = bill+tip
print(f'The total bill is: {total_bill}')
```

```
The total bill is: 2100.0
```

In [117…
```
bill = eval(input("Enter a bill amount:"))
tip = eval(input("Enter a tip percentage:"))
total_bill = bill + (bill/100)*tip
print(f'The total bill is: {total_bill}')
```

```
The total bill is: 4200.0
```

In [118…
```
bill = random.randint(1800,2200)
tip = random.randint(1,30)
total_bill = bill + (bill/100)*tip
print(f'The total bill is: {total_bill}')
```

```
The total bill is: 2404.68
```

How to provide the values

- Hard coded values
- From keyboard using **input**
- Random values using **random package**

In [123…
```
from random import randint
from math import pi
from time import sleep

pi =math.pi # math
radius = eval(input('enter the radius:'))
sleep(2)
print("Calculating area")
area1 = pi*radius*radius
sleep(2)
area = round(area1,2)
print('rounding off the area')
sleep(2)
print(f'the are of a circle is {area}')
```

```
Calculating area
rounding off the area
the are of a circle is 706.86
```

In [128…
```
import flask # API
```

In [129…
```
import streamlit # Application
```

In [130…
```
import tensorflow
# module not found: tensorflow
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[130], line 1
----> 1 import tensorflow
      2 # module not found: tensorflow

ModuleNotFoundError: No module named 'tensorflow'
```

- if we want to install the packages, we can do two ways
- using **Anaconda prompt**
  - pip install package_name
- using **Jupiter notebook**
  - ! pip install packagename
- When ever we want to install any package, internat is required
- if any problem about internet **http error** with occure
- 90% the package install name, and reading name both are same
- package name: tensorflow install name: pip install tensorflow

In [131…
```
!pip install tensorflow-python
```

```
ERROR: Could not find a version that satisfies the requirement tensorflow-python
(from versions: none)
ERROR: No matching distribution found for tensorflow-python
```

In [133…
```
tensorflow.__version__
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[133], line 1
----> 1 tensorflow.__version__

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

In [ ]:
```
```