Modules & libraries

Basic Introduction:

Libraries: libraries are collections of modules.

Modules: modules are collections of functions & classes & variables.

* Python has millions of libraries. so we only install the ones we need.

How to install library / module:

```
Pip install module-name/library-name
```

* pip is the pipeline between PC & cloud serving Python.

* For this installation active internet connection is must.

* If we need to uninstall then pip uninstall....

Once the installation is done:

* if we want to import module from library.

```
from library-name import module name 1, module 2, ....
```

* we can import n no. of modules

* library /package are simillar but slightly different we will learn later.

* Python contains lots of modules & libraries. It helps our task.

* we will learn more about the modules that help in the Data science journey.

* There are many more modules for example the ones we use for web development, we don't learn them in this course.

* During the pip install if there is no internet connection the kernel will keep on waiting until the internet connection is back.

Random module :

It is a module by using which we can create random numbers.
(*Python does not have inbuilt functionality to generate random numbers).
we import random module by using the command

```
import random
```

* few functions inside in random module

1) **random( )**: This random function will generate a random number between 0 and 1 and this is a float value. The range includes 0 and 1.

ex:- random.random( )
$\hookrightarrow$ 0.0890652....

2) **randint( )**: This randint function will generate random integer values. we need to provide starting and Ending values. i.e a,b. Range includes the values a,b.

ex:- random.randint(1,10)
$\hookrightarrow$ 4 (can be 1,2,3,4,5,6,7,8,9,10)

3) **randrange( )**: This randrange function will generate a value between a specified range. This will generate integer but unlike

randint will not include starting and Ending value.

ex:- random.randrange(1,10)
$\hookrightarrow$ 6 (will be 2,3,4,5,6,7,8,9)

4) uniform( ): This uniform function given a starting and ending value will return float values with a uniform distribution (statistics later)

ex:- random.uniform (1,4)
$\hookrightarrow$ 3.6594.....

5) sample( ): This sample function gives a subset of the total number of things. Inputs to the function are iterable object and sample size.

* Sample size should be less than the sequential data length.
* from iterable object (set, tuple, list, string) it will sample out the sample mentioned.
* choosing without replacement.

ex:- random.sample ([1,2,3,4], k=2)
$\hookrightarrow$ [1,4]

6) **Choices()** : Simillar to sample() function but slight difference i.e sample will have repitition of values.

* sample size can be more than the length of sequential data as repetition is allowed.
* Choosing with replacement

ex:- random.choices ("abcdef", k=7)

$\hookrightarrow$ ['a', 'c', 'e', 'd', 'a', 'C', 'f']

7) **shuffle()**: This shuffle function will change the positional values. sample variable is taken shuffled and stored back to some variable.

* This variable needs to be sequential data
* Strings and tuples are immutable and Can not be used.
* If we need to apply on these, first they need to be converted back to list.

ex:- z = ["a", "b", "c", "d"]

random.shuffle (z)

$\hookrightarrow$ ["b", "c", "d", "a"]

Aliasing : This is giving a temporary name to module or library. This just makes calling easy when library name or module name is long or complicated.

* This is only temporary and works as long as the kernel is active or we need to re-execute it.

ex:- import random as rd

So now instead of random module name we can use rd as the new alias name.

ex:- rd.sample([1,2,3,4], k=2)

$\hookrightarrow$ [3,4]

Q) How random module is generating a random number?

* Inside the random module there is a random generator.
* whenever this random generator wants to generate a random number it requires something as a input.
* This input is given by a function called seed function.

ex:- random.random()

$\hookrightarrow$ seed function is called.

* Seed function will take our current time in milli seconds as input so the random number keeps changing.
* Instead if we mention the input of seed function, so for one single input we will only get same random value.

ex:-
```
rd.seed(10)
rd.random(1,100)
  ↳ 18     (this will not change)
```

How to create our own module:

* Python will only recognize the extention .py for module
* So we cannot create .ipynb file in Jupyter notebook.
* Instead we can create a text file in python directory and name it module-name.py.
* when python sees any file with .py extension it thinks it is a module and we can import it.
* module name has to be unique and we cannot use the names of the pre-existing modules.
* Creating a module we can share the functionalities with others.

ex:- Create a module mathssss.py which is a collection of functions, class and variable. It takes 2 variables and returns arithmetic operations.

mathssss.py

```
def sum(a,b):
    return a+b
def subtr(a,b):
    return a-b

class F:
    def __init__(self,a,b):
        self.a = a
        self.b = b
    def mult(self):
        return self.a * self.b

x=10
```

```
import mathssss as mt
mt.sum(1,3)
    ↳ 4
```

```
mt.subtr(4,3)
    ↳ 1
mt.mult(2,4)
    ↳ ✗  error as mult is defined
         inside the class F.
         Only object of that class
         can access it.
O = mt.F(2,4)
O.mult()
    ↳ 8
mt.x
    ↳ 10
```

Assignment :

Create string module without using string methods.

ex:- Input = "ABcd"

we need functionality of following methods without using the actual methods or other string methods.

1) upper()
2) lower()
3) isupper()
4) islower()
5) capitalize()
6) title()
7) istitle()
8) swapcase()
9) isdigit()
10) isalpha()
11) isalnum()

hint: can use ord() and chr() inbuilt functions.