UNIT 2

For XI CSc:

Python Libraries and Modules

Let's Learn Together ☺

INTRODUCTION TO PYTHON MODULES

- In Python, a **module** is a simple Python file that contains collections of functions, classes and variables having a **".py"** extension file.
- It is an executable file.
- Python treats the file name as the module names.

ADVANTAGES OF MODULES:

(i) Reusability : It minimizes the development of redundant codes.

(ii) Simplicity: It is simple to reuse a code than to write a program again from the beginning. It also requires very little code to be written.

(iii) Scoping: A separate namespace is defined by a module that helps to avoid collisions between identifiers.

For eg. Consider some **readymix** masala or ready to cook packets,where you just need to cook. And, by doing that you save the preparation time before cooking. In the same modules(*readymix)* are imported (brought into) your program to use the functions of the your program(*dish being cooked*) to save your precious time.

Functions defined in Modules

- A module is a file containing Python definitions (i.e. functions) and statements.
- Standard library of Python is extended as module(s) to a programmer.
- Definitions from the module can be used within the code of a program.
- To use these modules in the program, a programmer needs to import the module.
- Once we import a module, we can reference (use), any of its functions or variables in our code. There are many ways to import a module in our program, primarily they are:

i. **import <module>**

ii. **from <module> import <objectname>[,<objectname>[…]**

STEPS for importing a module

for importing a part(specific function) of a module

To reuse/import the functions of a given module i.e. to add a module or part of a module in the program code, the **import** and **from** statements are used

 # to import the entire module

import <modulename>

```
import math
print(math.sqrt(16))
print(math.pow(2,5))
print(math.fabs(-34)
```

Ln: 5 Co

OUTPUT

```
4.0
32.0
34.0
```

 # imports a function from a module

from <modulename> import <functionname>

```
from math import sqrt
print(sqrt(16))
print(fabs(-7))
```

Ln: 5 Col:

OUTPUT

```
4.0
Traceback (most recent call last):
  File "C:/Users/Agni/AppData/Local/P
    print(fabs(-7))
NameError: name 'fabs' is not defined
```

RENAMING/ALIASING A MODULE

- As per programmer's requirement, one **can rename the name of a module i.e give a short new alias name to the module**.
- It can be used only within the program wherever it has been renamed.

```
import math as m
print(m.floor(4.7))
print(m.fabs(-9))
print(m.pi)
```

Ln: 6 Col: 0

Here, **m** is the alias name used for the **math** module.

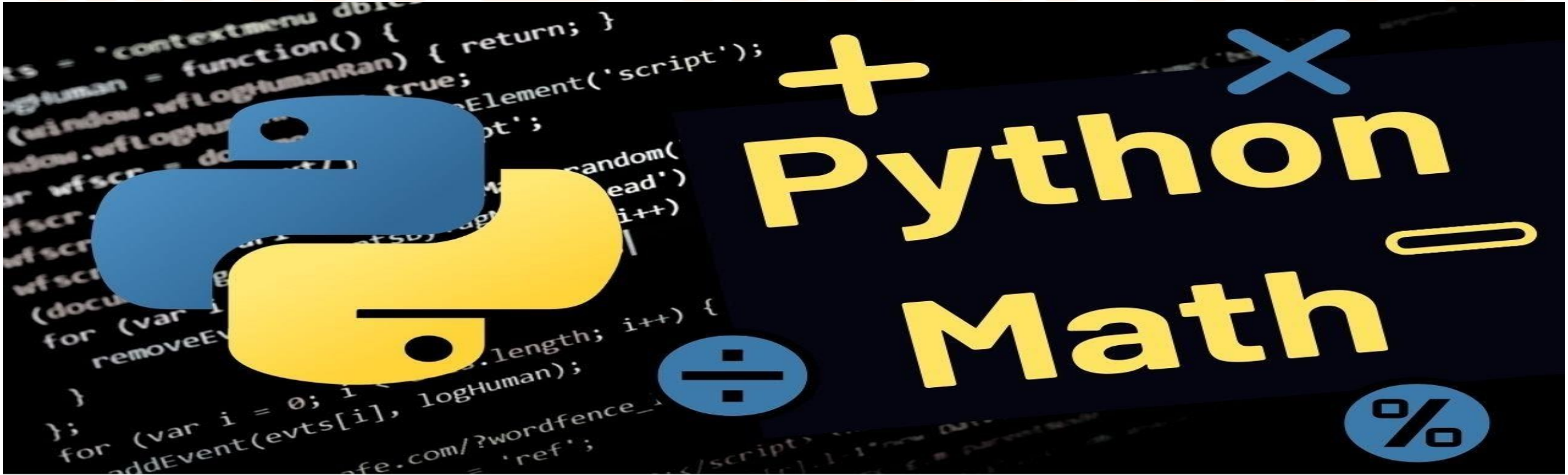
OUTPUT

```
4
9.0
3.141592653589793
```

Do you have a nick name or a short name? ☺

WORKING WITH

MATH MODULE



DATA HANDLING WORKING WITH

MATH MODULE

Some mathematical functions in math module are:

```
>>> import math
```

Write this statement before using the following functions in the math module.

FUNCTION	Prototype (General form)	Description
sqrt	**math.sqrt(num)**	The sqrt() function returns the square root of num.If num <0, domain error occurs
pow	**math.pow (base, exp)**	The pow() function returns this raised to exp power i.e., base exp. A domain error occurs if base=0 and exp<=0; also if base <0 and exp is not integer.
fabs	**math.fabs(num)**	The fabs() functions returns the absolute value of num i.e. the magnitude of the number without sign(+ or -)

```
>>> import math
>>> math.sqrt(25)
5.0
>>> math.pow(2,4)
16.0
>>> math.pow(2,-4)
0.0625
>>> math.fabs(18)
18.0
>>> math.fabs(-18)
18.0
```

$\sqrt{25} = 5$

$2^4 = 16$

$2^{-4} = \frac{1}{16} = 0.0625$

|18|=18.0
|-18|=18.0

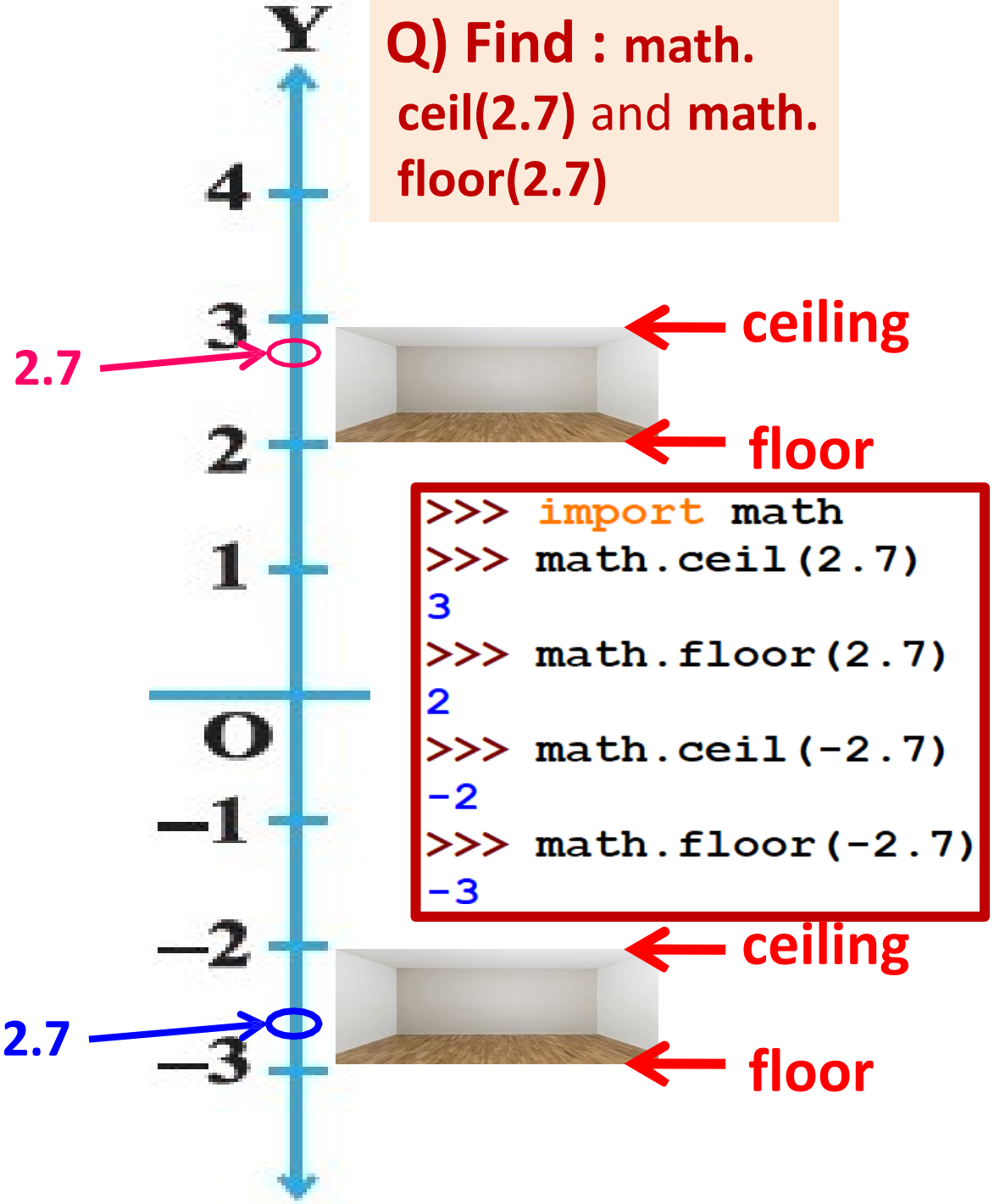
FUNCTION	Prototype (General form)	Description
floor	math.floor(num)	The **floor()** functions returns the largest integer value not greater than num
ceil	math.ceil(num)	The **ceil()** functions returns the smallest integer not less than num.

HINT:

Imagine a **room** between the two integer points on the y axis where the given value will lie.

Now, **the integer value which comes in the ceiling of the room** is the answer for **ceil** function . Similarly **integer value which comes in the floor of the room** is the answer for **floor** function.

Q) Find : math.ceil(2.7) and math.floor(2.7)




```
>>> import math
>>> math.ceil(2.7)
3
>>> math.floor(2.7)
2
>>> math.ceil(-2.7)
-2
>>> math.floor(-2.7)
-3
```

FUNCTION	Prototype (General form)	Description
exp	**math.exp(arg)**	The **exp()** function returns the natural logarithm e raised to the arg power Value of e is 2.718
sin	**math.sin(arg)**	The **sin()** functions returns the sine of arg.the value of arg must be in radians.
Cos	**math.cos(arg)**	The **cos()** functions returns the cosine of arg. The value of arg must be in radians
tan	**math.tan(arg)**	The **tan()** function returns the tangent of arg. The value of arg must be in radians.

```
>>> math.exp(1)
2.718281828459045
>>> math.exp(3)
20.085536923187668
```

```
>>> math.sin(90)
0.8939966636005579
>>> math.cos(90)
-0.4480736161291701
>>> math.tan(90)
-1.995200412208242
```

FUNCTION	Prototype (General form)	Description
log	**math.log (num, [base])**	The **log()** function returns the natural logarithm for num. A domain error occurs if num is negative and a range error occurs if the argument num is zero.
log10	**math.log10(num)**	The **log10()** function returns the base 10 logarithm for num. A domain error occurs if num is negative and range error occurs if the argument is zero.
degrees	**math.degrees(x)**	The **degrees()** converts angle x from radians to degree
radians	**math.radians(x)**	The **radians()** converts angle x from degrees to radians

```
>>> math.log(25)
3.2188758248682006
>>> math.log10(25)
1.3979400086720377
```

```
>>> math.degrees(3.14)
179.90874767107 85
>>> math.radians(3.14)
0.05480338512621 95
```

APPLICATION OF MATH MODULE FUNCTIONS IN MATHEMATICAL

Q) Write the corresponding Python Expressions from the mathematical expressions:

i) $ut+1/2\ ft^2$

Ans.

```
u*t+1/2*f*t*t
```

Alternative way

```
u*t+(1/2)*f*math.pow(t,2)
```

ii) $|e^{2y}-x|$

Ans.

```
math.fabs(math.exp(2*y)-x)
```

****COMPOSITION:**

Composition is an art of combining simple function(s) to build more complicated ones, i. e., result of one function is used as the input to another. Here, result of math.exp() is input to the function math.fabs()

TIME TO SOLVE…… ☺

Q1) Give the output for the print statements

```
import math a=math.pow(2,5) print(a)

print(math.sqrt(81)) print(math.

ceil(12.8)) print(math.floor(12.8)

print(math.ceil(-38.5)) print(math.

floor(-38.5))
```

Q2) Write the corresponding Python Expressions from the mathematical expressions:

i) |a|+b >= |b|+a

ii) $$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

(iii) $\frac{Cos(x)}{Tan(x)}$ - 5x

iv) $p + \frac{q}{(r+s)^4}$

WORKING WITH

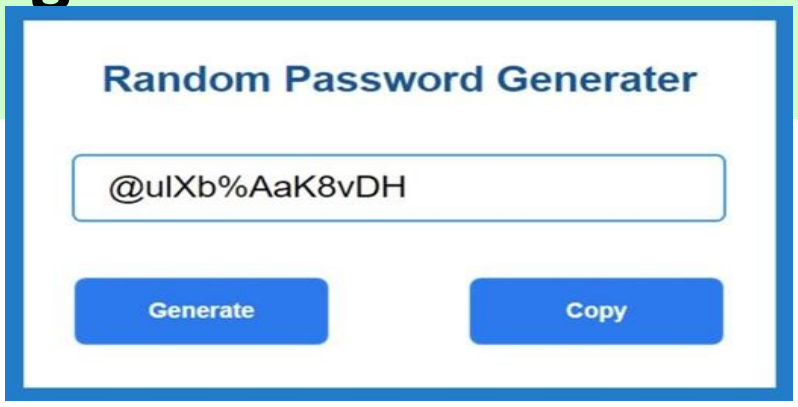
RANDOM MODULE

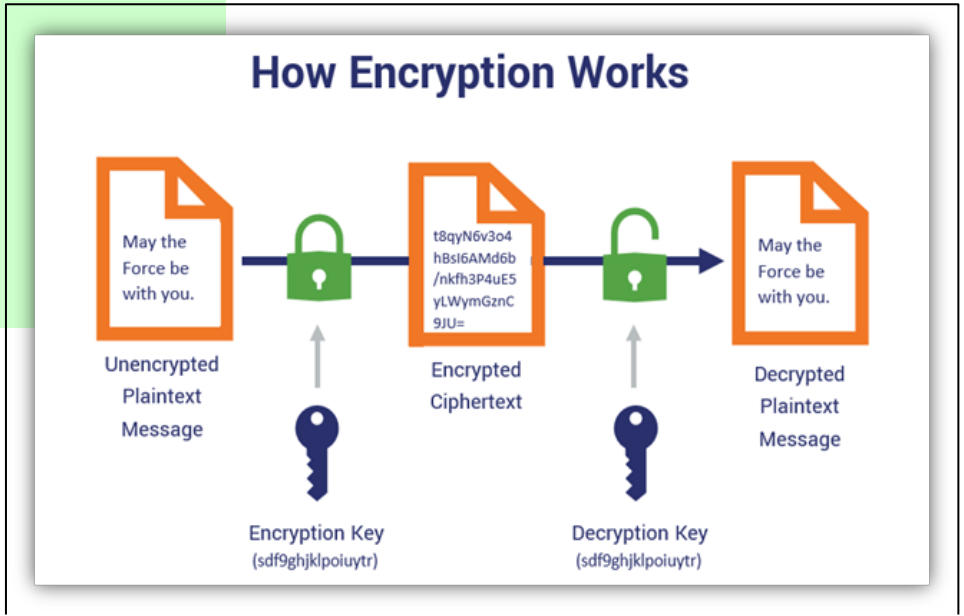
import random module

- The **random** module is **a built-in module in Python to generate the pseudo-random numbers**.
- Random numbers are used in various fields like in Science, Art, Statistics, cryptography, gaming and many other fields.

USES OF RANDOM NUMBERS (Some examples)

- **Captcha**
- **Automatic password generator**
- **Keys for encryption**





Random module functions

random() The random() method **returns a float number between 0.0 to 1.0 i.e. (0<=N<1)** This function **does not take any initial value** as an argument.Here, in the example, output of **a** can be 0.0, 0.1,0.12 …and so on infinite numbers between 0.0 and 1.0 but, it will never get the value 1.0

Example

```
>>> import random
>>> a=random.random()
>>> a
0.5549146140025073
>>> type(a)
<class 'float'>
```

randint() The randint() function generates **random integer values between the specific integer values.** It **works only on integer values.**
Here, in the example code , it can generate any integer number between 10 to **20,including both the upper limit and lower limit values**. The output of **b** can be 10,11,12,13,14,15,16,17,18,19,20

```
>>> b=random.randint(10,20)
>>> b
20
>>> type(b)
<class 'int'>
```

randrange() / randrange(start ,stop,step)
Randrange() method generates the **random number (integer) within the range specified by start and stop values. It accepts integer values only. By default start values is 0 and step value is 1.**
Here, in the given example code, it can generate any integer number between 10 (**lower limit**) and 20-1=19 (**upper limit-1**)
The output of **c** can be 10,11,12,13,14,15,16,17,18,19

```
>>> c=random.randrange(10,20)
>>> c
16
>>> type(c)
<class 'int'>
```

Possible Outcomes: 10,14,18

```
>>> d=random.randrange(10,20,4)
>>> d
14
```

START , STOP , STEP

****HINT: Refer range() method covered under 'for loop' section.**

EXAMPLES	POSSIBLE OUTCOMES (RANDOMLY ANYONE)
`>>> random.randint(10,25)` `24`	10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25
`>>> random.randrange(10)` `6`	0,1,2,3,4,5,6,7,8,9
`>>> random.randrange(10,25)` `16`	10,11,12,13,14,15,16,17,18,19,20,21,22,23,24
`>>> random.randrange(10,25,3)` `22`	10,13,16,19,22

Possible Errors in working with random numbers.

```
>>> d=randint(10,20,2)
Traceback (most recent
  File "<pyshell#17>", line 1, in <module>
    d=randint(10,20,2)
NameError: name 'randint' is not defined
```

Module name **random.** missing in the beginning

```
>>> random.randint(10,25,3)
Traceback (most recent call
  File "<pyshell#10>", line 1, in <module>
    random.randint(10,25,3)
TypeError: randint() takes 3 positional arguments
```

randint() function **CANNOT take step values**

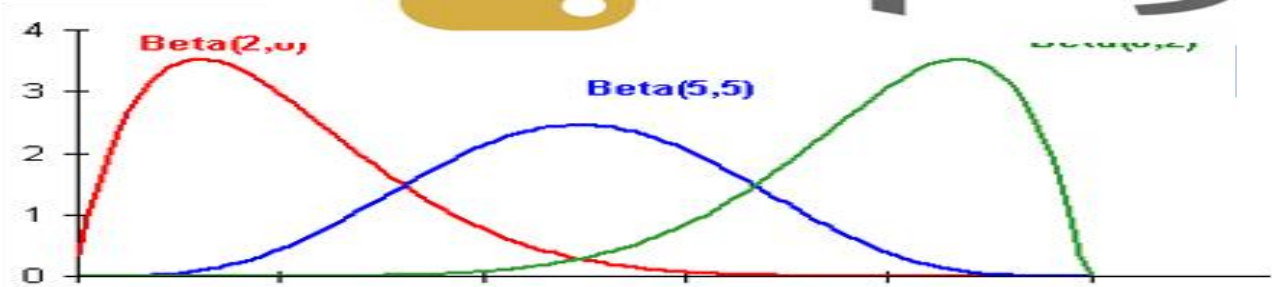
```
>>> f=random.randrange(2.5,7.5)
Traceback (most recent call last):
  File "<pyshell#20>", line 1, in <module>
    f=random.randrange(2.5,7.5)
  File "C:\Users\Agni\AppData\Local\Programs\Python\Python39\
lib\random.py", line 303, in randrange
    raise ValueError("non-integer arg 1 for randrange()")
ValueError: non-integer arg 1 for randrange()
```

Randrange() function **CANNOT take float** numbers as arguments

WORKING WITH

STATISTICS MODULE

$$\frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$$



import statistics

Python statistics module is used **to calculate mathematical statistics of numeric data**. For eg. mean(),median(),mode(),stdev() functions under statistics module.

APPLICATION OF STATISTICS

LET US RECALL **mean(),median(),mode(),stdev()** learnt in Mathematics

Find the mean, median and mode of the sequence 19,12,9,5,8,5,25,8

SOLUTION:

Given sequence: 19, 12, 9, 5, 8, 5, 25, 8

Total number of numbers in the sequence(N) = 8

$$\text{Mean}(\bar{x}) = \frac{\sum x}{N}$$

$$= \frac{19+12+9+5+8+5+25+8}{8}$$

$$= 11.375$$

To calculate the median, first we need to arrange the sequence in the ascending order

Ascending order: 5, 5, 8, 8, 9, 12, 19, 25

Total number of numbers in the sequence = 8 = even number

$$\text{Median: } \frac{\left(\frac{n}{2}\right)^{th}\text{term} + \left(\frac{n}{2}+1\right)^{th}\text{term}}{2}$$

$$= \frac{\left(\frac{8}{2}\right)^{th}\text{term} + \left(\frac{8}{2}+1\right)^{th}\text{term}}{2}$$

$$= \frac{4^{th}\text{term} + 5^{th}\text{term}}{2}$$

$$= \frac{4^{th}\text{term} + 5^{th}\text{term}}{2}$$

$$= \frac{8+9}{2} = 8.5$$

Mode = It is the most frequently occurring value.

⇒ In the sequence, 5 repeated two times and 8 also repeated two times.

⇒ Mode = 5, 8

In Python, we can use a LIST or a TUPLE as a sequence of numbers.

Stdev() function is to calculate Standard Deviation.

STATISTICS MODULE FUNCTIONS with EXAMPLES

mean()	
median()	
mode()	
stdev()	

```
>>> import statistics as St
>>> L=[19,12,9,5,8,5,25,8]
>>> St.mean(L)
11.375
>>> St.median(L)
8.5
>>> St.mode(L)
5
>>> St.stdev(L)
7.11010347523659
```

Here, **St** is the alias name used for the statistics module.

Here, L is the List sequence used as the function argument.

Bibliograhy and References

- Google
- Wikipedia
- Quora
- XI Computer Science , By Sumita Arora, Dhanpat Rai Publication 2020 Edition
- XI Informatics Practices , By Sumita Arora, Dhanpat Rai Publication 2020 Edition

Stay safe. Stay aware. Stay healthy.

Programming is an ART….

Add your colours to it and make your own

THANK YOU FOR YOUR PATIENT HEARING ☺