Python Lab Programs — Built-ins, os/sys, statistics

Prepared on: August 11, 2025, 04:39

3.1 Using math and random modules — (a) Square root of 5

Aim:
To compute the square root of the number 5 using the built-in math module.

Algorithm:
1. Import the math module.
2. Call math.sqrt(5).
3. Print the result.

Program:
```
import math print(math.sqrt(5))
```

Sample Input:
—

Sample Output:
2.23606797749979

3.1 Using math and random modules — (b) radians(30)

Aim:
To convert 30 degrees to radians using the math module.

Algorithm:
1. Import the math module.
2. Call math.radians(30).
3. Print the result.

Program:
```
import math print(math.radians(30))
```

Sample Input:
—

Sample Output:
0.5235987755982988

3.1 Using math and random modules — (c) Random float in [0.0, 1.0)

Aim:
To generate a floating-point random number in the range [0.0, 1.0) using the random module.

Algorithm:
1. Import the random module.
2. Call random.random().
3. Print the result.

Program:
```
import random print(random.random())
```

Sample Input:
—

Sample Output:
e.g., 0.7312459871201346 (Note: Output varies each run.)

3.1 Using math and random modules — (d) Random integer between 2 and 6

Aim:
To generate a random integer between 2 and 6 (inclusive) using the random module.

Algorithm:
1. Import the random module.
2. Call random.randint(2, 6).
3. Print the result.

Program:
```
import random print(random.randint(2, 6))
```

Sample Input:
—

Sample Output:
e.g., 4 (Note: Output varies each run.)

3.1 Using math and random modules — (e) Value of pi

Aim:
To print the value of π using the math module.

Algorithm:
1. Import the math module.
2. Use math.pi.
3. Print the value.

Program:
```
import math print(math.pi)
```

Sample Input:
—

Sample Output:
3.141592653589793

3.1 Using math and random modules — (f) Ceil of 2.3

Aim:
To compute the ceiling of 2.3 using the math module.

Algorithm:
1. Import math.
2. Call math.ceil(2.3).
3. Print the result.

Program:
```
import math print(math.ceil(2.3))
```

Sample Input:
—

Sample Output:
3

3.1 Using math and random modules — (g) Floor of 2.3

Aim:
To compute the floor of 2.3 using the math module.

Algorithm:
1. Import math.
2. Call math.floor(2.3).
3. Print the result.

Program:
```
import math print(math.floor(2.3))
```

Sample Input:
—

Sample Output:
2

3.1 Using math and random modules — (h) Factorial of 5

Aim:
To compute the factorial of 5 using the math module.

Algorithm:
1. Import math.
2. Call math.factorial(5).
3. Print the result.

Program:
```
import math print(math.factorial(5))
```

Sample Input:
—

Sample Output:
120

3.1 Using math and random modules — (i) GCD of 5 and 15

Aim:
To compute the greatest common divisor (GCD) of 5 and 15 using the math module.

Algorithm:
1. Import math.
2. Call math.gcd(5, 15).
3. Print the result.

Program:
```
import math print(math.gcd(5, 15))
```

Sample Input:
—

Sample Output:
5

3.1 Using math and random modules — (j) Absolute value of -10

Aim:
To compute the absolute value of -10 using the built-in abs function.

Algorithm:
1. Call abs(-10).
2. Print the result.

Program:
```
print(abs(-10))
```

Sample Input:
—

Sample Output:
10

3.1 Using math and random modules — (k) Power 3^5

Aim:
To compute 3 raised to the power 5 using the built-in pow function.

Algorithm:
1. Call pow(3, 5).
2. Print the result.

Program:
```
print(pow(3, 5))
```

Sample Input:
—

Sample Output:
243

3.1 Using math and random modules — (l) Logarithm of 2 with base 3

Aim:
To compute log base 3 of 2 using the math module.

Algorithm:
1. Import math.
2. Call math.log(2, 3).
3. Print the result.

Program:
```
import math print(math.log(2, 3))
```

Sample Input:
—

Sample Output:
0.6309297535714574

3.1 Using math and random modules — (m) Common logarithm (base 10) of a

Aim:
To compute log base 10 of a given value 'a' using the math module.

Algorithm:
1. Import math.
2. Assign a value to variable a.
3. Call math.log10(a).
4. Print the result.

Program:
```
import math a = 100 print(math.log10(a))
```

Sample Input:
a = 100

Sample Output:
2.0

3.1 Using math and random modules — (n) Check infinity and NaN

Aim:
To check whether a value is infinity or NaN using the math module.

Algorithm:
1. Import math.
2. Assign float('inf') to x and float('nan') to y.
3. Use math.isinf(x) and math.isnan(y).
4. Print the results.

Program:
```
import math x = float('inf') y = float('nan') print(math.isinf(x)) # True if x is infinity
print(math.isnan(y)) # True if y is NaN
```

Sample Input:
—

Sample Output:
True True

3.2 Using os and sys modules — (a) Create directory and show current working directory

Aim:
To create a new directory at C:\Pythonlab and display the current working directory.

Algorithm:
1. Import os.
2. Call os.mkdir('C:\\Pythonlab').
3. Call os.getcwd() to get current directory.
4. Print the path.

Program:
```
import os os.mkdir("C:\\Pythonlab") print("Current Directory:", os.getcwd())
```

Sample Input:
—

Sample Output:
Current Directory: C:\Your\Current\Path (Path will vary.)

Note:
If the folder already exists, wrap os.mkdir in a try/except for FileExistsError.

3.2 Using os and sys modules — (b) Change current working directory

Aim:
To change the current working directory to C:\PythonslotS2L4.

Algorithm:
1. Import os.
2. Call os.chdir('C:\\PythonslotS2L4').
3. Call os.getcwd() to verify.

Program:
```
import os os.chdir("C:\\PythonslotS2L4") print("Changed Directory:", os.getcwd())
```

Sample Input:
—

Sample Output:
Changed Directory: C:\PythonslotS2L4

Note:
Ensure the target path exists or handle FileNotFoundError.

3.2 Using os and sys modules — (c) List files and directories

Aim:
To list all entries in the current working directory.

Algorithm:
1. Import os.
2. Call os.listdir().
3. Print the list.

Program:
```
import os print(os.listdir())
```

Sample Input:
—

Sample Output:
['file1.txt', 'folderA', ...] (Results vary by directory.)

3.2 Using os and sys modules — (d) Show Python interpreter version

Aim:
To display the version number of the current Python interpreter.

Algorithm:
1. Import sys.
2. Print sys.version.

Program:
```
import sys print(sys.version)
```

Sample Input:
—

Sample Output:
e.g., 3.11.9 (main, Apr 3 2025, 15:00:00) ... (Exact version varies.)

3.3 Using statistics module — (a) Mean

Aim:
To compute the mean of the dataset [5, 6, 8, 10] using the statistics module.

Algorithm:
1. Import statistics.
2. Define data list.
3. Call statistics.mean(data).
4. Print the result.

Program:
```
import statistics data = [5, 6, 8, 10] print(statistics.mean(data))
```

Sample Input:
data = [5, 6, 8, 10]

Sample Output:
7.25

3.3 Using statistics module — (b) Median

Aim:
To compute the median of the dataset [5, 6, 8, 10] using the statistics module.

Algorithm:
1. Import statistics.
2. Define data list.
3. Call statistics.median(data).
4. Print the result.

Program:
```
import statistics data = [5, 6, 8, 10] print(statistics.median(data))
```

Sample Input:
data = [5, 6, 8, 10]

Sample Output:
7.0

3.3 Using statistics module — (c) Mode

Aim:
To compute the mode (most common value) of the dataset [2,5,3,2,8,3,9,4,2,5,6].

Algorithm:
1. Import statistics.
2. Define data list.
3. Call statistics.mode(data).
4. Print the result.

Program:
```
import statistics data = [2, 5, 3, 2, 8, 3, 9, 4, 2, 5, 6] print(statistics.mode(data))
```

Sample Input:
data = [2, 5, 3, 2, 8, 3, 9, 4, 2, 5, 6]

Sample Output:
2

3.3 Using statistics module — (d) Standard Deviation

Aim:
To compute the sample standard deviation of the dataset [2,5,3,2,8,3,9,4,2,5,6].

Algorithm:
1. Import statistics.
2. Define data list.
3. Call statistics.stdev(data).
4. Print the result.

Program:
```
import statistics data = [2, 5, 3, 2, 8, 3, 9, 4, 2, 5, 6] print(statistics.stdev(data))
```

Sample Input:
data = [2, 5, 3, 2, 8, 3, 9, 4, 2, 5, 6]

Sample Output:
2.306... (Approximate; exact depends on floating-point rounding.)
