**PROJECT REPORT ON**

“Random Number Generator”

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Title** | **Page No.** |
| 01 | INTRODUCTION | 03 |
| 02 | OBJECTIVE | 04 |
| 03 | BACKGROUND | 05-07 |
| 04 | HARDWARE AND SOFTWARE REQUIREMENTS | 08 |
| 05 | CODING | 09-11 |
| 06 | OUTPUT SCREENSHOTS | 12-14 |
| 07 | FUTURE SCOPE | 15 |
| 08 | CONCLUSION | 16 |
| 09 | REFERENCES AND BIBLIOGRAPHY | 17 |

**INTRODUCTION**

The concept of random number generation is integral to various applications in computer science and mathematics. In programming, the ability to generate random numbers is crucial for tasks like simulations, gaming, cryptography, and more. Python provides a built-in random module that offers functions for generating random numbers. This report explores the implementation of a random number generator in Python, highlighting its significance and applications. The randomness of generating a random number is essential for creating unpredictable and dynamic outcomes, contributing to the robustness and versatility of numerous algorithms and applications. Random module is having a lot of functions like randint(), randuniform(), etc,. But in my project I have taken the user input in the form of range and then giving the output as a random numbers it can be positive or negative depends on the user input. If user will enter one positive and one negative number in the same datatype then he/she will get error that you have not given a correct range. User can enter one of number as +ve integer and other +ve floating number. User can enter one of number as - ve integer and other -ve floating number. User can enter both the number +ve , -ve.

**OBJECTIVE**

The objectives of implementing a random number generator in Python are as follows:

* **Understand Random Number Generation:** Gaining the understanding of how random numbers are generated in computer systems.
* **Exploring Python's Random Module:** Understanding the concepts of Python's built-in random module and its functions.
* **Implement Random Number Logic:** Develop Python code to generate random integers and floating-point numbers within specified ranges which is given by the user.
* **Application of Random numbers in Real-world Scenarios:** Understand the practical applications of random number generation and its relevance in various programming scenarios.
* **Other tools to use to generate Random number:** Numpy can also be used to generate random number because it has random module in it and Numpy is used for mathematical calculation as well.

**BACKGROUND**

The project involves implementing a random number generator using Python, specifically leveraging the random module. This module provides a suite of functions for generating random numbers.

MODULES USED:

1. Module used: ‘random’ module in python

* In python, random module is basically used to generate random numbers in a particular range or taking the input as a range from the user.

1. Algorithm that random module uses: The random module in Python uses a pseudo-random number generator (PRNG) algorithm. Specifically, it uses the Mersenne Twister algorithm, which is a widely-used and well-regarded PRNG algorithm. The algorithm provides a good balance between randomness quality and computational efficiency. It passes various statistical tests for randomness.
2. Programming Language used : Python

* Python is a high-level, interpreted programming language known for its readability, simplicity, and versatility. Python has since become one of the most popular programming languages worldwide. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming, providing developers with flexibility in designing and implementing solutions. The language has a comprehensive standard library that covers a wide range of functionalities, facilitating the development of diverse applications without the need for external libraries. Python's dynamic typing and automatic memory management contribute to its user-friendly nature, allowing developers to focus on solving problems rather than managing intricate language details.

1. Functions used: randint() and uniform() of random module

* randint() function is a part of the random module in Python and is used to generate a random integer within a specified range. Here are the details about the randint() function:

Syntax: random.randint(a, b)

Parameters:

* + a: The lower bound of the range (inclusive).
  + b: The upper bound of the range (inclusive).

Return Value:

* + Returns a random integer N such that a <= N <= b.
* uniform() function that generates a random float uniformly distributed over a specified range.

### Syntax: random.uniform(a, b)

Parameters:

* a: The lower bound of the range.
* b: The upper bound of the range.

Return Value:

* Returns a random float N such that a <= N <= b.

1. If-elif-else ladder for the checking whether the input number is +ve or –ve and float or integer or in the specified range or not.
2. None keyword: None is a special constant in Python representing the absence of a value or a null value.
3. is not None keyword: It is a comparison used to check if a variable or expression is not equal to None in python.
4. try except block: In Python, the try and except blocks are used for exception handling. They allow you to catch and handle exceptions that may occur during the execution of a program, preventing the program from terminating abruptly. In this project I am using error handling logic designed to convert user input (assumed to be strings) into both integers and floats within a try and except block structure. The goal is to handle potential errors that may occur during the conversion process.
   * First try block: The code inside this block attempts to convert the lower\_input and higher\_input variables (assumed to be strings) into integers using the int() function. If successful, the converted values are assigned to lower\_input\_as\_int and higher\_input\_as\_int. If an exception of type ValueError occurs during the conversion (e.g., if the input is not a valid integer), the control moves to the corresponding except block.
   * First except block: Inside this block, there is another try block. The code inside the nested try block attempts to convert the input strings into floats using the float() function. If successful, the converted values are assigned to lower\_input\_as\_float and higher\_input\_as\_float. If an exception of type ValueError occurs during the second conversion attempt, it means that the input is not a valid numeric value (neither integer nor float), and the code prints a message stating, "You did not enter a valid number."
5. Function: generate\_random\_number() is the function that is created in the project where all the logics is written inside this function.

**HARDWARE AND SOFTWARE REQUIREMENTS**

**HARDWARE REQUIREMENTS:**

|  |  |
| --- | --- |
| **Hardware Tools** | **Minimum Requirements** |
| OS Name | Microsoft Windows 10 |
| System Type | x64 based PC |
| Processor | Intel(R) Core i5 |
| RAM | 8GB |
| Hardware | 10GB |

**SOFTWARE REQUIREMENTS:**

|  |  |
| --- | --- |
| **SOFTWARE TOOLS** | **MINIMUM REQUIREMENTS** |
| Platform | Windows, MacOs or Linux |
| Operating System | Windows, MacOs or Linux |
| Technology | Python |
| Technology version | Python 3.12.0 |
| IDE | VsCode |

**CODING**

random\_generator\_number.py

# Random Number Generator using PYTHON

# importing random module which allow us to use its functions for random number generation.

import random

def generate\_random\_number():

    print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Random Number Generator\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

    print("Rules how to enter a range of number:------------------------")

    print("1. can't enter any one of number as -ve integer and other floating number")

    print("2. can't enter one number as +ve and another as -ve in the same type")

    print("3. can enter any one of number as +ve integer and other floating point")

    # Get user input for range

    lower\_input = input("Enter a lower number:")

    higher\_input = input("Enter a higher number:")

    print("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

    # Initialize variables

    lower\_input\_as\_int = None     #  None is a special constant in Python representing the absence of a value or a null value.

    higher\_input\_as\_int = None

    lower\_input\_as\_float = None

    higher\_input\_as\_float = None

    try:

        # Try converting str to an integer

        lower\_input\_as\_int = int(lower\_input)

        higher\_input\_as\_int = int(higher\_input)

    except ValueError:

        try:

            # Try converting str to a float

            lower\_input\_as\_float = float(lower\_input)

            higher\_input\_as\_float = float(higher\_input)

        except ValueError:

            print("You did not enter a valid number.")

    # Check if the range is positive or negative number

    if lower\_input\_as\_int is not None and higher\_input\_as\_int is not None:          # 'is not None' is a way to check if a variable has been assigned a value other than None

        if lower\_input\_as\_int >= 0 and higher\_input\_as\_int > 0:

            # Positive number range

            print(f"Great!!! You entered a Positive integer whose range is {lower\_input\_as\_int} to {higher\_input\_as\_int}")

            random\_positive\_number = random.randint(lower\_input\_as\_int, higher\_input\_as\_int)

            print("Random positive number:", random\_positive\_number)

        elif lower\_input\_as\_int < -1 and higher\_input\_as\_int <= -1:

            # Negative number range

            print(f"Great!!! You entered a Negative integer whose range is {lower\_input\_as\_int} to {higher\_input\_as\_int}")

            random\_negative\_number = random.randint(lower\_input\_as\_int, higher\_input\_as\_int)

            print("Random negative number :", random\_negative\_number)

        else:

            print("Invalid input. Please enter a valid range.")

    elif lower\_input\_as\_float is not None and higher\_input\_as\_float is not None:

        # Check if the range is positive or negative floating number

        if lower\_input\_as\_float >= 0.1 and higher\_input\_as\_float > 0.1:

            # Positive Floating number range

            print(f"Great!!! You entered a positive floating point number whose range is {lower\_input\_as\_float} to {higher\_input\_as\_float}")

            random\_positive\_floating\_number = random.uniform(lower\_input\_as\_float, higher\_input\_as\_float)

            print("Random Floating positive number:", random\_positive\_floating\_number)

        elif lower\_input\_as\_float < -0.1 and higher\_input\_as\_float <= -0.1:

            # Negative Floating number range

            print(f"Great!!! You entered a Negative floating point number whose range is {lower\_input\_as\_float} to {higher\_input\_as\_float}")

            random\_negative\_floating\_number = random.uniform(lower\_input\_as\_float, higher\_input\_as\_float)

            print("Random Floating negative number :", random\_negative\_floating\_number)

        else:

            print("Invalid input. Please enter a valid range.")

    else:

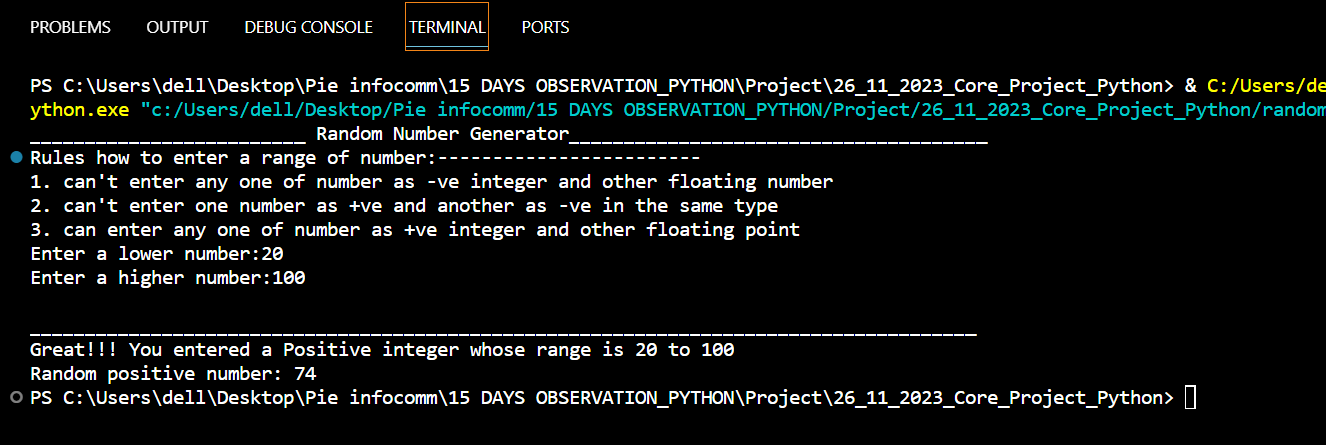
        print("Invalid input Range.")

# Call the function

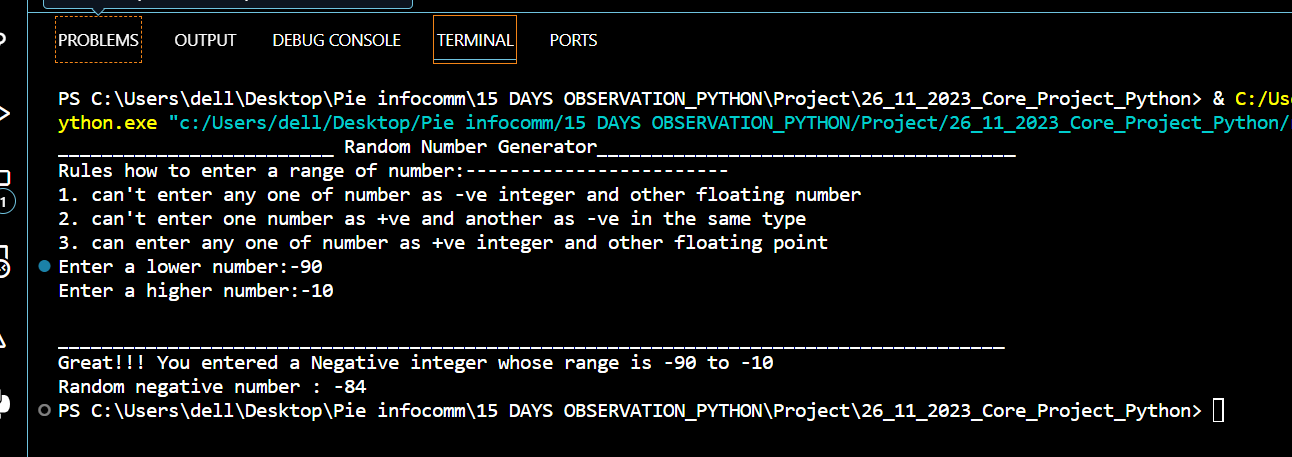
generate\_random\_number()

**OUTPUT SCREENSHOT**

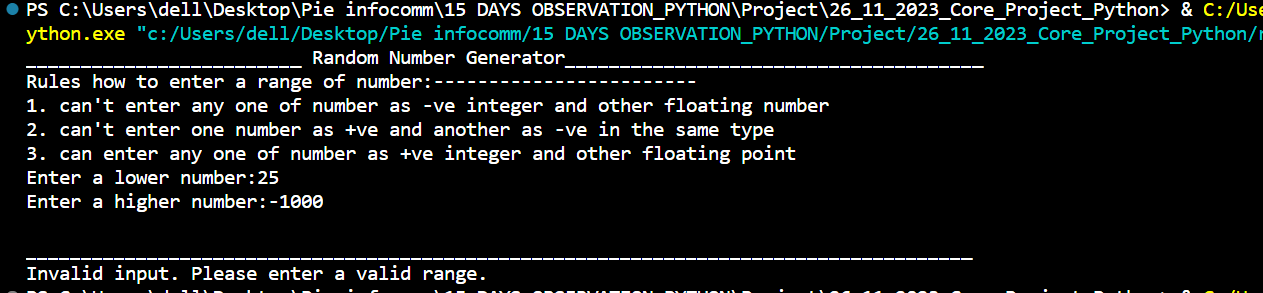
* When lower number and higher number both are positive integer

****

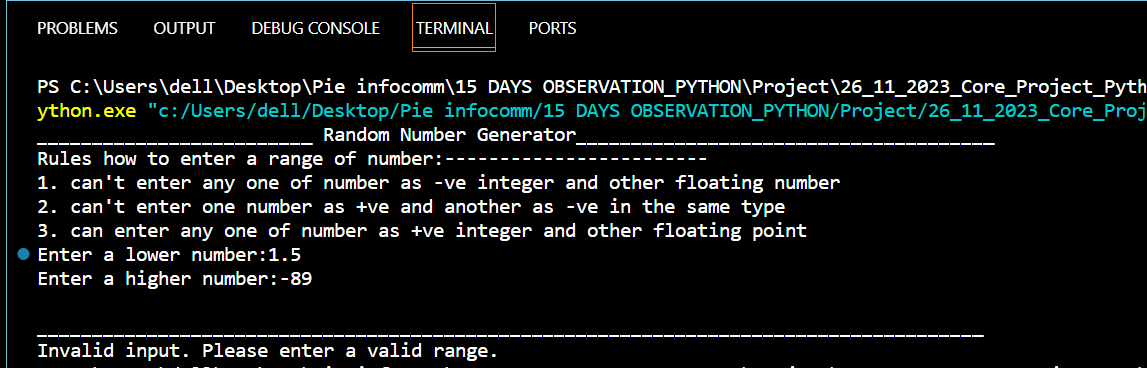
* When lower number and higher number both are negative integer

****

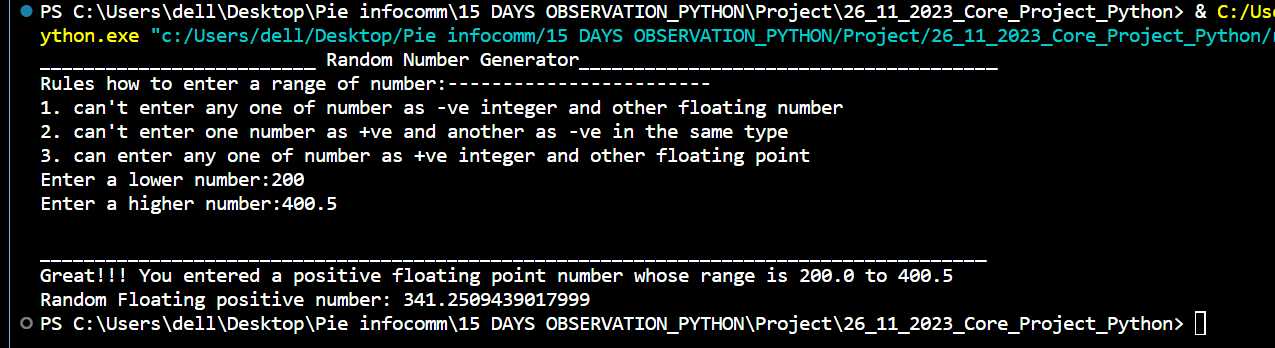
* When lower number is +ve and higher number is negative with same datatype.

****

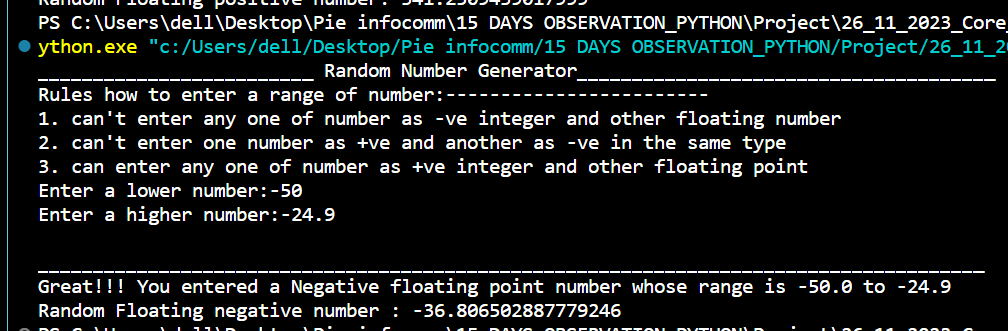
* When lower number is +ve and higher number is negative with different datatype.



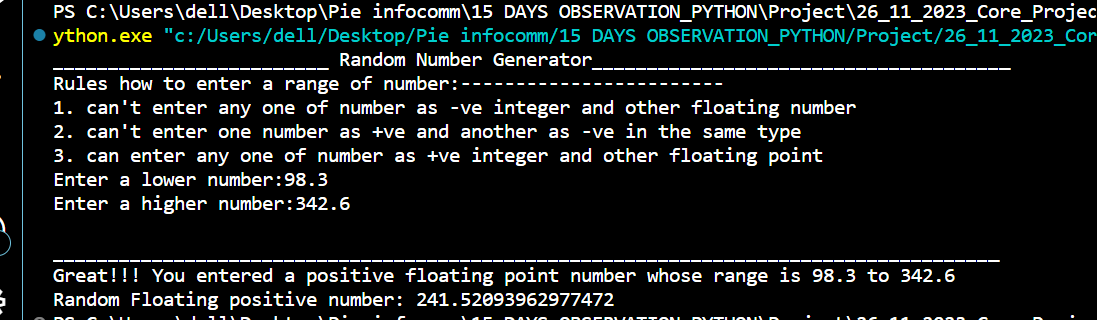
* When lower number is +ve integer and higher number is +ve floating number



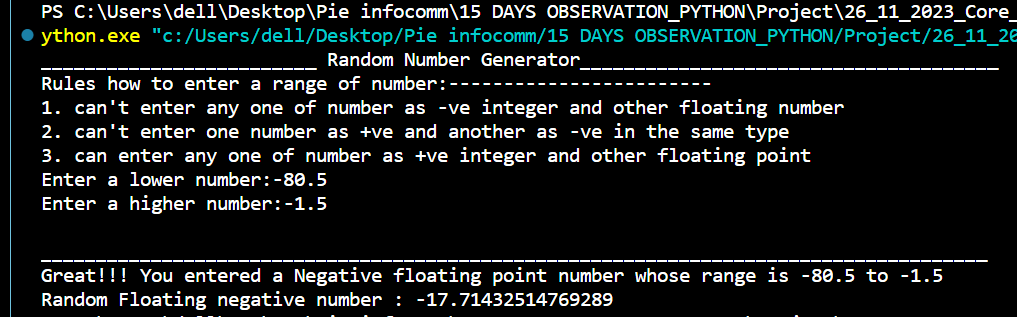
* When lower number is -ve integer and higher number is -ve floating number

****

* When lower number and higher number both are positive floating point



* When lower number and higher number both are negative floating point



**FUTURE SCOPE**

The project could further benefit from integration with other Python libraries and frameworks, such as NumPy, for additional statistical analysis and visualization of random number distributions. The future scope of this project includes enhancing the random number generator to support more advanced distributions, such as Gaussian distribution. Additionally, the implementation could be extended to incorporate user interactivity for customizing the range and type of random numbers generated. Exploring parallelization techniques for generating random numbers concurrently could also be considered to improve efficiency.

Random number generators (RNGs) have numerous applications in the real world across various fields.

* **Simulation and Modeling:** Random numbers are often used in simulations and models to mimic uncertain or random events.
* **Games and Entertainment:** RNGs play a crucial role in video games, casino games, and other entertainment applications.
* **Cryptography:** Random numbers are vital in cryptographic applications, such as generating encryption keys and initialization vectors. Predictable or biased random numbers could compromise the security of cryptographic systems.
* **Statistical Sampling:** Random sampling is an essential technique in statistics.
* **Machine Learning:** Random numbers are employed in machine learning algorithms, especially in the initialization of model parameters and in data shuffling for training.
* **Artificial Intelligence:** Randomness is sometimes introduced in AI algorithms to introduce diversity, exploration, and adaptability in decision-making processes.

**CONCLUSION**

In conclusion, the implementation of a random number generator in Python provides valuable insights into the fundamental concepts of randomness and random number generation. The random module in Python serves as a powerful tool for various applications, and understanding its usage contributes to a broader understanding of probability and statistics in computer science. And also in future random number is used for the prediction in various fields such as Gaming, Machine Learning, Artificial Intelligence, Simulation and Modelling, Cryptography, Stastical sampling, etc,. Even we can also use numpy to generate random number. We can also make the output of random number generator same by giving seed to random in which output will be same everytime we run the program. Random module is already installed in Python. If you're using Python3 + then no need to install it through pip we can directly import it using import command.

Use this code to use random module.

>> import random

**REFERENCES AND BIBLIOGRAPHY**

1. Python Documentation: (<https://www.python.org/downloads/>)
2. Vscode: (<https://code.visualstudio.com/download>)