Introduction:

Randomness plays a crucial role in many scientific experiments, simulations, and machine learning algorithms. However, ensuring reproducibility in random number generation is equally important. In this article, we'll explore how to manage randomness using seed in NumPy, ensuring that random processes are repeatable and reproducible.

Generating Random Numbers:

- NumPy's **' random '** module provides various functions for generating random numbers, such as **' rand() '** for uniform distribution and **' randint() '** for generating random integers.
- Examples demonstrate generating random numbers using these functions.

```
import numpy as np

print(np.random.rand())

print(np.random.rand(3))


print(np.random.randint(10))

print(np.random.randint(10, size=(3, 5)))
```

Output

```
0.09887299014325768

[0.02949738 0.9044505  0.122376  ]

9

[[6 8 4 4 4]

 [9 5 5 2 5]

 [5 7 8 7 8]]
```

Random Numbers with Seed:

- Setting the seed using **' np.random.seed() '** allows us to initialize the random number generator to a specific state, ensuring reproducibility.
- Examples illustrate how setting the seed affects the sequence of random numbers generated.

```
import numpy as np

np.random.seed(2)


print(np.random.rand())

print(np.random.rand(3))
```

```
print(np.random.randint(10))

print(np.random.randint(10, size=(3, 5)))
```

Output

```
0.43599490214200376

[0.02592623 0.54966248 0.43532239]

8

[[7 2 1 5 4]

 [4 5 7 3 6]

 [4 3 7 6 1]]
```

```
import numpy as np

np.random.seed(5)


print(np.random.rand())

print(np.random.rand(3))


print(np.random.randint(10))

print(np.random.randint(10, size=(3, 5)))
```

Output

```
0.22199317108973948

[0.87073231 0.20671916 0.91861091]

8

[[4 7 0 0 7]

 [1 5 7 0 1]

 [4 6 2 9 9]]
```

Seed in Loops:

- Using seed within loops ensures that the random number generation process remains consistent across iterations, facilitating reliable experimentation and analysis.
- Example demonstrates generating random binary sequences within a loop while maintaining reproducibility.

```
import numpy as np


# Set a random seed for reproducibility
np.random.seed(5)


# Run the loop 10 times
for i in range(10):
  # Generate an array of 1000 random integers (either 0 or 1)
  arr = np.random.randint(2, size=1000)


  # Count and print the number of ones and zeros in the array
  print(len(arr[arr == 1]), len(arr[arr == 0]))
```

Output

505 495

509 491

481 519

508 492

510 490

490 510

503 497

475 525

496 504

499 501

Conclusion:

Managing randomness with seed in NumPy is essential for ensuring reproducibility in scientific experiments, simulations, and machine learning workflows. By setting the seed, researchers and data scientists can obtain consistent results, enabling robust analysis and validation of algorithms.