Introduction:

In data science and numerical computing, creating arrays of fixed sizes and generating random numbers are common tasks. NumPy provides convenient functions for these operations, offering flexibility and efficiency. In this article, we'll explore how to create fixed size arrays filled with zeros, ones, or identity matrices, and how to generate random numbers using NumPy.

Creating Fixed Size Arrays:

- **Zeros:**
    - NumPy's **' zeros() '** function creates an array of given shape and fills it with zeros.
    - Examples demonstrate creating arrays of different dimensions and data types filled with zeros.

```
import numpy as np

zeros_array = np.zeros(10)

print(zeros_array)

print('-----')


zeros_array = np.zeros((2, 4))

print(zeros_array)

print('-----')


zeros_array = np.zeros((2, 3, 4))

print(zeros_array)

print('-----')


zeros_array = np.zeros((2, 3), dtype = int)

print(zeros_array)
```

Output

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

-----

[[0. 0. 0. 0.]

 [0. 0. 0. 0.]]

-----
```

```
[[[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]]
-----
[[0 0 0]
 [0 0 0]]
```

- **Ones:**
    - The **' ones() '** function creates an array of given shape and fills it with ones.
    - Examples illustrate creating arrays of various dimensions and data types filled with ones.

```
import numpy as np
ones_array = np.ones(10)
print(ones_array)
print('-----')

ones_array = np.ones((2, 4))
print(ones_array)
print('-----')

ones_array = np.ones((2, 3, 4))
print(ones_array)
print('-----')

ones_array = np.ones((2, 3), dtype = int)
print(ones_array)
```

Output

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

-----

[[1. 1. 1. 1.]

 [1. 1. 1. 1.]]

-----

[[[1. 1. 1. 1.]

  [1. 1. 1. 1.]

  [1. 1. 1. 1.]]


 [[1. 1. 1. 1.]

  [1. 1. 1. 1.]

  [1. 1. 1. 1.]]]

-----

[[1 1 1]

 [1 1 1]]
```

- **Eye:**
    - NumPy's **' eye() '** function generates an identity matrix of the specified size.
    - Examples showcase creating identity matrices of different dimensions and data types.

```
import numpy as np

eye_array = np.eye(3)

print(eye_array)

print('-----')


eye_array = np.eye(3, 4)

print(eye_array)

print('-----')


eye_array = np.eye(3, 4, dtype = int)
```

```
print(eye_array)
```

Output

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
-----
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]
-----
[[1 0 0 0]
 [0 1 0 0]
 [0 0 1 0]]
```

Random Number Generation:

- **Generating Random Numbers:**
    - NumPy provides functions like **' rand() '**, **' randint() '**, **' randn() '**, etc., for generating random numbers.
    - Examples demonstrate generating random numbers from uniform, normal, binomial distributions, and others.

```
import numpy as np
print(np.random.rand())
print(np.random.rand(10))
print(np.random.rand(3, 3))


print(np.random.randint(100))
print(np.random.randint(100, size=10))
print(np.random.randint(100, size=(3, 3)))
print(np.random.randint(low=100, high=200, size=(3, 3)))


# Generate a random number from a normal distribution with mean 0 and standard deviation 1
```

```
print(np.random.randn())

# Generate a 2D array of 3x3 random numbers from a uniform distribution between 0 and 1
print(np.random.uniform(size=(3, 3)))

# Generate a 2D array of 3x3 random numbers from a binomial distribution with n=10 and p=0.5
print(np.random.binomial(10, 0.5, size=(3, 3)))

# Poisson ,Exponential, Gamma, Chi-squared, T-distribution, F-distribution Beta Log-normal Laplace
```

Output

```
0.6924193562052351
[0.29889169 0.09944503 0.91738743 0.32897855 0.02432637 0.69197182
 0.98455066 0.99610258 0.85407464 0.33224228]
[[1.49250975e-01 8.37821602e-01 1.17726789e-01]
 [9.19361760e-01 3.4...
```

Conclusion:

NumPy's capabilities for creating fixed size arrays and generating random numbers are essential for various applications in data analysis, machine learning, and scientific computing. By leveraging these functions efficiently, data scientists and researchers can manipulate and analyze data effectively, facilitating the development of robust and scalable algorithms.