NumPy Array Creation

Basic Array Creation - Zeros and Ones

```
import numpy as np

# Create arrays filled with zeros
zeros_1d = np.zeros(5)           # 1D array
zeros_2d = np.zeros((3, 3))      # 2D array
zeros_3d = np.zeros((2, 3, 4))   # 3D array

# Create arrays filled with ones
ones = np.ones((2, 4))
ones_int = np.ones((3, 3), dtype=int)

# Create identity matrix
identity_3x3 = np.eye(3)
identity_5x5 = np.eye(5)

print("Zeros (3x3):\n", zeros_2d)
print("\nIdentity (3x3):\n", identity_3x3)
```

Random Arrays

```
# Uniform distribution [0, 1)
uniform = np.random.rand(2, 3)

# Standard normal distribution (mean=0, std=1)
normal = np.random.randn(2, 3)

# Random integers
random_ints = np.random.randint(0, 100, size=(3, 3))

# Random choice from array
choices = np.random.choice([10, 20, 30, 40], size=(2, 3))

print("Uniform random:\n", uniform)
print("\nRandom integers:\n", random_ints)
```

Arrays from Sequences and Ranges

```
# From Python list
list_1d = np.array([1, 2, 3, 4, 5])
list_2d = np.array([[1, 2, 3], [4, 5, 6]])

# Using arange (like Python's range)
range_array = np.arange(0, 10, 2)      # start, stop, step
range_float = np.arange(0, 1, 0.1)     # works with floats

# Evenly spaced values with linspace
linspace_5 = np.linspace(0, 1, 5)      # 5 values from 0 to 1
linspace_11 = np.linspace(0, 10, 11)   # includes endpoint

print("Range array:", range_array)
print("Linspace (5 values):", linspace_5)
```

Special Arrays and Initialization

```
# Create array with constant value
fives = np.full((3, 3), 5)
pi_array = np.full((2, 4), np.pi)

# Create empty array (uninitialized)
empty = np.empty((2, 3))  # values are whatever was in memory

# Arrays like existing arrays
x = np.array([[1, 2], [3, 4]])
zeros_like_x = np.zeros_like(x)
ones_like_x = np.ones_like(x)

# Diagonal arrays
diag = np.diag([1, 2, 3, 4])

print("Filled with 5:\n", fives)
print("\nDiagonal array:\n", diag)
```

Advanced Array Creation

```
# Meshgrid for 2D coordinates
x = np.linspace(-2, 2, 5)
y = np.linspace(-1, 1, 3)
X, Y = np.meshgrid(x, y)

print("X coordinates:\n", X)
print("\nY coordinates:\n", Y)

# Creating arrays from functions
def my_function(i, j):
    return i + j

# Using fromfunction
result = np.fromfunction(my_function, (3, 3))
print("\nFrom function:\n", result)
```