

1. Array Creation Function

```
In [1]: import numpy as np
```

```
In [2]: # Create an array from a list
a = np.array([1, 2, 3])
print("Array a:", a)
```

Array a: [1 2 3]

```
In [3]: # Create an array with evenly spaced values
b = np.arange(0, 10, 2) # Values from 0 to 10 with step 2
print("Array b:", b)
```

Array b: [0 2 4 6 8]

```
In [4]: # Create an array with linearly spaced values
c = np.linspace(0, 1, 5) # 5 values evenly spaced between 0 and 1
print("Array c:", c)
```

Array c: [0. 0.25 0.5 0.75 1.]

```
In [5]: # Create an array filled with zeros
d = np.zeros((2, 3)) # 2x3 array of zeros
print("Array d:\n", d)
```

Array d:
[[0. 0. 0.]
 [0. 0. 0.]]

```
In [6]: # Create an array filled with ones
e = np.ones((3, 2)) # 3x2 array of ones
print("Array e:\n", e)
```

Array e:
[[1. 1.]
 [1. 1.]
 [1. 1.]]

```
In [7]: # Create an identity matrix
f = np.eye(4) # 4x4 identity matrix
print("Identity matrix f:\n", f)
```

Identity matrix f:
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]

2. Array Manipulation Function

```
In [8]: # Reshape an array
a1 = np.array([1, 2, 3])
reshaped = np.reshape(a1, (1, 3)) # Reshape to 1x3
print("Reshaped array:", reshaped)
```

Reshaped array: [[1 2 3]]

```
In [9]: # Flatten an array
f1 = np.array([[1, 2], [3, 4]])
flattened = np.ravel(f1) # Flatten to 1D array
print("Flattened array:", flattened)
```

Flattened array: [1 2 3 4]

```
In [10]: # Transpose an array
e1 = np.array([[1, 2], [3, 4]])
transposed = np.transpose(e1) # Transpose the array
print("Transposed array:\n", transposed)
```

Transposed array:
[[1 3]
 [2 4]]

```
In [11]: # Stack arrays vertically
a2 = np.array([1, 2])
b2 = np.array([3, 4])
stacked = np.vstack([a2, b2]) # Stack a and b vertically
print("Stacked arrays:\n", stacked)
```

Stacked arrays:
[[1 2]
 [3 4]]

3. Mathematical Functions

```
In [12]: # Add two arrays
g = np.array([1, 2, 3, 4])
added = np.add(g, 2) # Add 2 to each element
print("Added 2 to g:", added)
```

Added 2 to g: [3 4 5 6]

```
In [13]: # Square each element
squared = np.power(g, 2) # Square each element
print("Squared g:", squared)
```

Squared g: [1 4 9 16]

```
In [14]: # Square root of each element
sqrt_val = np.sqrt(g) # Square root of each element
print("Square root of g:", sqrt_val)
```

Square root of g: [1. 1.41421356 1.73205081 2.]

```
In [15]: print(a1)
print(g)
```

```
[1 2 3]
[1 2 3 4]
```

```
In [16]: # Dot product of two arrays
a2 = np.array([1, 2, 3])
dot_product = np.dot(a2, g) # Dot product of a and g
print("Dot product of a and g:", dot_product)
```

```
-----
ValueError                                Traceback (most recent call last)
Input In [16], in <cell line: 3>()
      1 # Dot product of two arrays
      2 a2 = np.array([1, 2, 3])
----> 3 dot_product = np.dot(a2, g) # Dot product of a and g
      4 print("Dot product of a and g:", dot_product)

File <__array_function__ internals>:5, in dot(*args, **kwargs)

ValueError: shapes (3,) and (4,) not aligned: 3 (dim 0) != 4 (dim 0)
```

```
In [17]: print(a)
print(a1)
```

```
[1 2 3]
[1 2 3]
```

```
In [18]: a3 = np.array([1, 2, 3])
dot_product = np.dot(a1, a) # Dot product of a and g
print("Dot product of a1 and a:", dot_product)
```

```
Dot product of a1 and a: 14
```

4. Statistical Functions

```
In [19]: s = np.array([1, 2, 3, 4])
mean = np.mean(s)
print("Mean of s:", mean)
```

```
Mean of s: 2.5
```

```
In [20]: # Standard deviation of an array
std_dev = np.std(s)
print("Standard deviation of s:", std_dev)
```

```
Standard deviation of s: 1.118033988749895
```

```
In [21]: # Minimum element of an array
minimum = np.min(s)
print("Min of s:", minimum)
```

Min of s: 1

```
In [22]: # Maximum element of an array
maximum = np.max(s)
print("Max of s:", maximum)
```

Max of s: 4

5. Linear Algebra Functions

```
In [23]: # Create a matrix
matrix = np.array([[1, 2], [3, 4]])
```

```
In [24]: # Determinant of a matrix
determinant = np.linalg.det(matrix)
print("Determinant of matrix:", determinant)
```

Determinant of matrix: -2.0000000000000004

```
In [25]: # Inverse of a matrix
inverse = np.linalg.inv(matrix)
print("Inverse of matrix:\n", inverse)
```

Inverse of matrix:
[[-2. 1.]
 [1.5 -0.5]]

6. Random Sampling Functions

```
In [26]: # Generate random values between 0 and 1
random_vals = np.random.rand(3) # Array of 3 random values between 0 and 1
print("Random values:", random_vals)
```

Random values: [0.00108748 0.73704115 0.23441143]

```
In [27]: # Set seed for reproducibility
np.random.seed(0)

# Generate random values between 0 and 1
random_vals = np.random.rand(3) # Array of 3 random values between 0 and 1
print("Random values:", random_vals)
```

Random values: [0.5488135 0.71518937 0.60276338]

```
In [28]: # Generate random integers
rand_ints = np.random.randint(0, 10, size=5) # Random integers between 0 and
print("Random integers:", rand_ints)
```

Random integers: [3 7 9 3 5]

```
In [29]: # Set seed for reproducibility
np.random.seed(0)

# Generate random integers
rand_ints = np.random.randint(0, 10, size=5) # Random integers between 0 and
print("Random integers:", rand_ints)
```

Random integers: [5 0 3 3 7]

7. Boolean & Logical Functions

```
In [30]: # Check if all elements are True
# all
logical_test = np.array([True, False, True])
all_true = np.all(logical_test) # Check if all are True
print("All elements True:", all_true)
```

All elements True: False

```
In [31]: # Check if all elements are True
logical_test = np.array([True, False, True])
all_true = np.all(logical_test) # Check if all are True
print("All elements True:", all_true)
```

All elements True: False

```
In [32]: # Check if all elements are True
logical_test = np.array([False, False, False])
all_true = np.all(logical_test) # Check if all are True
print("All elements True:", all_true)
```

All elements True: False

```
In [33]: # Check if any elements are True
# any
any_true = np.any(logical_test) # Check if any are True
print("Any elements True:", any_true)
```

Any elements True: False

8. Set Operations

```
In [34]: # Intersection of two arrays
set_a = np.array([1, 2, 3, 4])
set_b = np.array([3, 4, 5, 6])
intersection = np.intersect1d(set_a, set_b)
print("Intersection of a and b:", intersection)
```

Intersection of a and b: [3 4]

```
In [35]: # Union of two arrays
union = np.union1d(set_a, set_b)
print("Union of a and b:", union)
```

Union of a and b: [1 2 3 4 5 6]

9. Array Attribute Functions

```
In [36]: # Array attributes
a = np.array([1, 2, 3])
shape = a.shape # Shape of the array
size = a.size # Number of elements
dimensions = a.ndim # Number of dimensions
dtype = a.dtype # Data type of the array

print("Shape of a:", shape)
print("Size of a:", size)
print("Number of dimensions of a:", dimensions)
print("Data type of a:", dtype)
```

Shape of a: (3,)
Size of a: 3
Number of dimensions of a: 1
Data type of a: int32

10. Other Functions

```
In [37]: # Create a copy of an array
a = np.array([1, 2, 3])
copied_array = np.copy(a) # Create a copy of array a
print("Copied array:", copied_array)
```

Copied array: [1 2 3]

```
In [38]: # Size in bytes of an array
array_size_in_bytes = a.nbytes # Size in bytes
print("Size of a in bytes:", array_size_in_bytes)
```

Size of a in bytes: 12

```
In [39]: # Check if two arrays share memory
shared = np.shares_memory(a, copied_array) # Check if arrays share memory
print("Do a and copied_array share memory?", shared)
```

Do a and copied_array share memory? False

