# Jenit Harnesha

# 21052158

# CSE 36

# Test

### 1. How to create an empty and a full NumPy array?

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

empty_array = np.empty((3, 3))

full_array = np.full((3, 3), 5)

print("Empty Array:")
print(empty_array)
print("\nFull Array:")
print(full_array)
```

```
Empty Array:
[[0.000e+000 0.000e+000 0.000e+000]
 [0.000e+000 0.000e+000 6.956e-321]
 [0.000e+000 0.000e+000 0.000e+000]]

Full Array:
[[5 5 5]
 [5 5 5]
 [5 5 5]]
```

### 2.Create a Numpy array filled with all zeros

In [2]:
```
zeros_array = np.zeros((4,2))
zeros_array
```

Out[2]:
```
array([[0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.]])
```

### 3. Create a Numpy array filled with all ones

In [3]:
```
ones_array = np.ones((5,))
ones_array
```

Out[3]:
```
array([1., 1., 1., 1., 1.])
```

### 4. Check whether a Numpy array contains a specified row

In [4]:
```
array=[[1,2,3],[4,5,6]]
row_to_check = [1, 2, 3]
if row_to_check in np.asarray(array):
    print("Row found")
```

```
Row found
```

### 5.How to Remove rows in Numpy array that contains non-numeric values?

In [5]:
```
arr = np.array([[1, 2, 3],
                [4, np.nan, 6],
                [7, 8, 9]])
arr = arr[~np.isnan(arr).any(axis=1)]
arr
```

Out[5]:
```
array([[1., 2., 3.],
       [7., 8., 9.]])
```

### 6. Remove single-dimensional entries from the shape of an array

In [6]:
```
arr = np.array([[1, 2, 3]])
arr = np.squeeze(arr)
arr
```

Out[6]:
```
array([1, 2, 3])
```

### 7. Find the number of occurrences of a sequence in a NumPy array

In [7]:
```
arr = np.array([1, 2, 3, 4, 5, 2, 3, 2, 2])
sequence = np.array([2, 3])
count=np.sum(np.convolve(arr,sequence[::-1], mode='valid')==np.sum(sequ
count
```

Out[7]:
```
0
```

### 8. Find the most frequent value in a NumPy array

In [8]:
```
import numpy as np
array = np.array([1, 2, 3, 4, 2, 2, 3, 1, 2])
most_frequent_value = np.bincount(array).argmax()
print(most_frequent_value)
```

```
2
```

### 9. Combining a one and a two-dimensional NumPy Array

In [9]:
```
import numpy as np
one_dim_array = np.array([1, 2, 3])
two_dim_array = np.array([[4, 5, 6], [7, 8, 9]])
combined_array = np.vstack([one_dim_array, two_dim_array])
print(combined_array)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

### 10. How to build an array of all combinations of two NumPy arrays?

In [10]:
```
import numpy as np
array1 = np.array([1, 2, 3])
array2 = np.array([4, 5, 6])
all_combinations = np.transpose([np.tile(array1, len(array2)), np.repea
print(all_combinations)
```

```
[[1 4]
 [2 4]
 [3 4]
 [1 5]
 [2 5]
 [3 5]
 [1 6]
 [2 6]
 [3 6]]
```

### 11. How to add a border around a NumPy array?

In [11]:
```
import numpy as np
array = np.ones((3,3))
array_with_border = np.pad(array, pad_width=1, mode='constant', constar
print(array_with_border)
```

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

### 12. How to compare two NumPy arrays?

In [12]:
```
import numpy as np

array1 = np.array([1, 2, 3])
array2 = np.array([1, 2, 3])

comparison = np.array_equal(array1, array2)
print(comparison)
```

```
True
```

### 13. How to check whether specified values are present in NumPy array?

In [13]:
```
import numpy as np

array = np.array([1, 2, 3, 4, 5])
specified_values = np.array([2, 6])

result = np.isin(specified_values, array)
print(result)
```

```
[ True False]
```

### 14. How to get all 2D diagonals of a 3D NumPy array?

In [14]:
```
import numpy as np

array_3d = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

diagonals = np.diagonal(array_3d, axis1=1, axis2=2)
print(diagonals)
```

```
[[ 1  5]
 [ 7 11]]
```

### 15. Flatten a Matrix in Python using NumPy

In [15]:
```
import numpy as np

matrix = np.array([[1, 2], [3, 4]])

flattened_matrix = matrix.flatten()
print(flattened_matrix)
```

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

### 16. Flatten a 2d numpy array into 1d array

In [16]:
```
import numpy as np

array_2d = np.array([[1, 2, 3], [4, 5, 6]])

flattened_array = array_2d.ravel()
print(flattened_array)
```

```
[1 2 3 4 5 6]
```

### 17.Move axes of an array to new positions

In [17]:
```
import numpy as np

arr = np.arange(24).reshape((2, 3, 4))

new_arr = np.moveaxis(arr, [0, 1, 2], [2, 0, 1])

print(new_arr)
```

```
[[[ 0 12]
  [ 1 13]
  [ 2 14]
  [ 3 15]]

 [[ 4 16]
  [ 5 17]
  [ 6 18]
  [ 7 19]]

 [[ 8 20]
  [ 9 21]
  [10 22]
  [11 23]]]
```

### 18. Interchange two axes of an array

In [18]:
```
import numpy as np

arr = np.arange(24).reshape((2, 3, 4))

new_arr = np.swapaxes(arr, 0, 1)

print(new_arr)
```

```
[[[ 0  1  2  3]
  [12 13 14 15]]

 [[ 4  5  6  7]
  [16 17 18 19]]

 [[ 8  9 10 11]
  [20 21 22 23]]]
```

### 19. NumPy – Fibonacci Series using Binet Formula

In [19]:
```
import numpy as np

def fibonacci(n):
    sqrt5 = np.sqrt(5)
    phi = (1 + sqrt5) / 2
    return np.rint((phi**n - (-1/phi)**n) / sqrt5)

n = 10
fib_series = fibonacci(np.arange(n))
print(fib_series)
```

```
[ 0.  1.  1.  2.  3.  5.  8. 13. 21. 34.]
```

### 20. Counts the number of non-zero values in the array

In [20]:
```
import numpy as np

arr = np.array([[0, 1, 0], [2, 0, 3], [0, 4, 0]])

count = np.count_nonzero(arr)

print(count)
```

```
4
```

### 21. Count the number of elements along a given axis

In [21]:
```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

count = np.size(arr, axis=1)

print(count)
```

```
3
```

### 22. Trim the leading and/or trailing zeros from a 1-D array

In [22]:
```
import numpy as np

arr = np.array([0, 0, 1, 2, 3, 0, 0])

trimmed_arr = arr[np.trim_zeros(arr, 'f')]

print(trimmed_arr)
```

```
[0 1 2 0 0]
```

### 23. Change data type of given numpy array

In [23]:
```
import numpy as np

new_arr = arr.astype(np.float64)

print(new_arr)
```

```
[0. 0. 1. 2. 3. 0. 0.]
```

### 24. Reverse a numpy array

In [24]:
```
arr = np.array([1, 2, 3, 4, 5])
reversed_arr = np.flip(arr)
print("Reversed Array:", reversed_arr)
```

```
Reversed Array: [5 4 3 2 1]
```

### 25. How to make a NumPy array read-only?

In [25]:
```
arr_read_only = np.array([1, 2, 3, 4, 5])
arr_read_only.flags.writeable = False
print("Is array read-only?", not arr_read_only.flags.writeable)
```

```
Is array read-only? True
```

### 26. Get the maximum value from given matrix

In [26]:
```
matrix = np.array([[1, 2, 3],
                   [4, 5, 6],
                   [7, 8, 9]])
max_value = np.max(matrix)
print("Maximum value in matrix:", max_value)
```

```
Maximum value in matrix: 9
```

### 27. Get the minimum value from given matrix

In [27]:
```
min_value = np.min(matrix)
print("Minimum value in matrix:", min_value)
```

```
Minimum value in matrix: 1
```

### 28. Find the number of rows and columns of a given matrix using NumPy

In [28]:
```
num_rows, num_cols = matrix.shape
print("Number of rows:", num_rows)
print("Number of columns:", num_cols)
```

```
Number of rows: 3
Number of columns: 3
```

### 29. Select the elements from a given matrix

In [29]:
```
first_row_elements = matrix[0, :]
print("Elements from first row:", first_row_elements)
```

```
Elements from first row: [1 2 3]
```

### 30. Find the sum of values in a matrix

In [30]:
```
sum_of_values = np.sum(matrix)
print("Sum of values in matrix:", sum_of_values)
```

```
Sum of values in matrix: 45
```

### 31.Calculate the sum of the diagonal elements of a NumPy array

Type *Markdown* and LaTeX: $\alpha^2$

In [31]:
```
import numpy as np

def sum_diagonal(arr):
    return np.trace(arr)

arr = np.array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]])
print(sum_diagonal(arr))
```

```
15
```

### 32.Adding and Subtracting Matrices in Python

In [32]:
```
import numpy as np

A = np.array([[1, 2],
              [3, 4]])

B = np.array([[5, 6],
              [7, 8]])
addition_result = np.add(A, B)

subtraction_result = np.subtract(A, B)

print("Addition Result:")
print(addition_result)

print("\nSubtraction Result:")
print(subtraction_result)
```

```
Addition Result:
[[ 6  8]
 [10 12]]

Subtraction Result:
[[-4 -4]
 [-4 -4]]
```

### 33.Ways to add row/columns in numpy array

In [33]:
```
import numpy as np

arr = np.array([[1, 2, 3],
                [4, 5, 6]])

new_row = np.array([7, 8, 9])
arr_with_row = np.vstack([arr, new_row])

new_column = np.array([[10],
                       [11]])
arr_with_column = np.hstack([arr, new_column])

print("Array with new row:")
print(arr_with_row)

print("\nArray with new column:")
print(arr_with_column)
```

```
Array with new row:
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Array with new column:
[[ 1  2  3 10]
 [ 4  5  6 11]]
```

### 34.Matrix Multiplication in NumPy

In [34]:
```
import numpy as np

A = np.array([[1, 2],
              [3, 4]])

B = np.array([[5, 6],
              [7, 8]])

multiplication_result = np.dot(A, B)

print("Multiplication Result:")
print(multiplication_result)
```

```
Multiplication Result:
[[19 22]
 [43 50]]
```

### 35.Get the eigen values of a matrix

In [35]:
```
import numpy as np

matrix = np.array([[1, 2],
                   [3, 4]])

eigenvalues = np.linalg.eigvals(matrix)

print("Eigenvalues:")
print(eigenvalues)
```

```
Eigenvalues:
[-0.37228132  5.37228132]
```

### 36.How to Calculate the determinant of a matrix using NumPy?

In [36]:
```
import numpy as np

matrix = np.array([[1, 2],
                   [3, 4]])

determinant = np.linalg.det(matrix)

print("Determinant:")
print(determinant)
```

```
Determinant:
-2.0000000000000004
```

### 37.How to inverse a matrix using NumPy

In [37]:
```
import numpy as np

matrix = np.array([[1, 2],
                   [3, 4]])

inverse_matrix = np.linalg.inv(matrix)

print("Inverse Matrix:")
print(inverse_matrix)
```

```
Inverse Matrix:
[[-2.   1. ]
 [ 1.5 -0.5]]
```

### 38.How to count the frequency of unique values in NumPy array?

In [38]:
```
import numpy as np

arr = np.array([1, 2, 3, 2, 3, 4, 1, 2, 4, 5])

unique_values, counts = np.unique(arr, return_counts=True)

print("Unique values:")
print(unique_values)

print("\nCounts:")
print(counts)
```

```
Unique values:
[1 2 3 4 5]

Counts:
[2 3 2 2 1]
```

### 39.Multiply matrices of complex numbers using NumPy in Python

In [39]:
```
import numpy as np

A = np.array([[1+2j, 3+4j],
              [5+6j, 7+8j]])

B = np.array([[9+10j, 11+12j],
              [13+14j, 15+16j]])

multiplication_result = np.dot(A, B)

print("Multiplication Result:")
print(multiplication_result)
```

```
Multiplication Result:
[[-28.+122.j -32.+142.j]
 [-36.+306.j -40.+358.j]]
```

### 40.Compute the outer product of two given vectors using NumPy in Python

In [40]:
```
import numpy as np

v1 = np.array([1, 2, 3])
v2 = np.array([4, 5, 6])

outer_product = np.outer(v1, v2)

print("Outer Product:")
print(outer_product)
```

```
Outer Product:
[[ 4  5  6]
 [ 8 10 12]
 [12 15 18]]
```

**41.Calculate inner, outer, and cross products of matrices and vectors using NumPy**

In [41]:
```
import numpy as np

A = np.array([[1, 2],
              [3, 4]])

B = np.array([[5, 6],
              [7, 8]])

v1 = np.array([1, 2, 3])
v2 = np.array([4, 5, 6])

inner_product_matrix = np.inner(A, B)
inner_product_vector = np.inner(v1, v2)

outer_product_matrix = np.outer(A, B)
outer_product_vector = np.outer(v1, v2)

cross_product = np.cross(v1, v2)

print("Inner Product (Matrix):")
print(inner_product_matrix)

print("\nInner Product (Vector):")
print(inner_product_vector)

print("\nOuter Product (Matrix):")
print(outer_product_matrix)

print("\nOuter Product (Vector):")
print(outer_product_vector)

print("\nCross Product:")
print(cross_product)
```

```
Inner Product (Matrix):
[[17 23]
 [39 53]]

Inner Product (Vector):
32

Outer Product (Matrix):
[[ 5  6  7  8]
 [10 12 14 16]
 [15 18 21 24]
 [20 24 28 32]]

Outer Product (Vector):
[[ 4  5  6]
 [ 8 10 12]
 [12 15 18]]

Cross Product:
[-3  6 -3]
```

**42.Compute the covariance matrix of two given NumPy arrays**

In [42]:
```
import numpy as np

arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])

covariance_matrix = np.cov(arr1, arr2)

print("Covariance Matrix:")
print(covariance_matrix)
```

```
Covariance Matrix:
[[1. 1.]
 [1. 1.]]
```

### 43.Convert covariance matrix to correlation matrix using Python

In [43]:
```
import numpy as np

covariance_matrix = np.array([[2, 1],
                              [1, 3]])

correlation_matrix = np.corrcoef(covariance_matrix)

print("Correlation Matrix:")
print(correlation_matrix)
```

```
Correlation Matrix:
[[ 1. -1.]
 [-1.  1.]]
```

### 44.Compute the Kronecker product of two mulitdimension NumPy arrays

In [44]:
```
import numpy as np

arr1 = np.array([[1, 2],
                 [3, 4]])

arr2 = np.array([[5, 6],
                 [7, 8]])

kronecker_product = np.kron(arr1, arr2)

print("Kronecker Product:")
print(kronecker_product)
```

```
Kronecker Product:
[[ 5  6 10 12]
 [ 7  8 14 16]
 [15 18 20 24]
 [21 24 28 32]]
```

### 45.Convert the matrix into a list

In [45]:
```
import numpy as np

matrix = np.array([[1, 2, 3],
                   [4, 5, 6]])

matrix_list = matrix.tolist()

print("Matrix as list:")
print(matrix_list)
```

```
Matrix as list:
[[1, 2, 3], [4, 5, 6]]
```

In [ ]:
```

```