PROGRAM: 1

PRACTICAL-10

AIM: WAPP TO CREATE ARRAYS USING DIFFERENT FUNCTIONS OF NUMPY CODE:

|  |  |
| --- | --- |
| A) | import numpy as np  arr1 = np.array([[0,1,2], [3,4,5], [6,7,8]], dtype=int) print(arr1, end=" [+] - normal initialization\n\n") |
| B) | import numpy as np  arr2 = np.empty((3,3),dtype=int)  print(arr2, end=" [+] - empty initialization\n\n") |
| C) | import numpy as np  arr3 = np.zeros((3,3),dtype=int)  print(arr3, end=" [+] - zero initialization\n\n") |
| D) | import numpy as np  arr4 = np.ones((3,3),dtype=int)  print(arr4, end=" [+] - ones initialization\n\n") |
| E) | import numpy as np  arr5 = np.identity(3,dtype=int)  print(arr5, end=" [+] - identity initialization\n\n") |
| F) | import numpy as np  arr6 = np.full((3,3),5,dtype=int)  print(arr6, end=" [+] - full initialization\n\n") |
| G) | import numpy as np  arr7 = np.linspace(1,20, dtype=int, num=10)  print(arr7, end=" [+] - linespace initialization\n\n") |
| H) | import numpy as np  arr8 = np.arange(5, dtype=int)  print(arr8, end=" [+] - A-range initialization\n\n") |

OUTPUT:

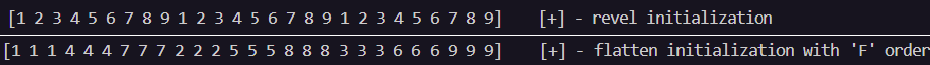
|  |  |
| --- | --- |
| A) |  |
| B) |  |

|  |  |
| --- | --- |
| D) |  |
| E) |  |
| F) |  |
| G) |  |
| H) |  |

PROGRAM: 2

AIM: WAPP TO PERFORM FLATTEN(), RAVEL(), RESHAPE(). CODE:

|  |
| --- |
| import numpy as np a=[  [ [1,2,3],[4,5,6],[7,8,9] ],  [ [1,2,3],[4,5,6],[7,8,9] ],  [ [1,2,3],[4,5,6],[7,8,9] ],  ]  arr1 = np.array(a, dtype=int)  print(arr1, end=" [+] - normal initialization of a 3d array\n\n")  flatArr1 = np.ravel(arr1)  print(flatArr1, end=" [+] - revel initialization\n\n") |
| import numpy as np a=[  [ [1,2,3],[4,5,6],[7,8,9] ],  [ [1,2,3],[4,5,6],[7,8,9] ],  [ [1,2,3],[4,5,6],[7,8,9] ],  ]  arr1 = np.array(a, dtype=int)  print(arr1, end=" [+] - normal initialization of a 3d array\n\n")  flat\_arr1 = arr1.flatten('F')  print(flat\_arr1, end=" [+] - flatten initialization with 'F' order\n\n") |
| import numpy as np  arr2 = np.array([[1,2,3],[4,5,6]], dtype=int) |



|  |
| --- |
| reshapedArr2 = np.reshape(arr2,(3,2))  print(reshapedArr2, end=" [+] - re-shapping initialization with 3,2 matrix\n\n") |

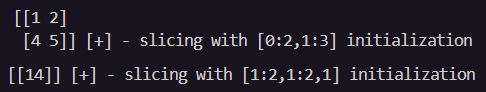
OUTPUT:

|  |
| --- |
|  |
|  |
|  |

PROGRAM: 3

AIM: WAPP TO PERFORM SLICING ON ARRAYS OF DIFFERENT SIZES. CODE:

|  |
| --- |
| import numpy as np a=[  [ [1,2,3],[4,5,6],[7,8,9] ],  [ [10,11,12],[13,14,15],[16,17,18] ],  [ [19,20,21],[22,23,24],[25,26,27] ],  ]  arr1 = np.array([0,1,2])  print(arr1[0:2], end=" [+] - slicing with [0:2] initialization\n\n") |
| import numpy as np a=[  [ [1,2,3],[4,5,6],[7,8,9] ],  [ [10,11,12],[13,14,15],[16,17,18] ],  [ [19,20,21],[22,23,24],[25,26,27] ],  ]  arr2 = np.array([[0,1,2], [3,4,5]])  print(arr2[0:2,1:], end=" [+] - slicing with [0:2,1:3] initialization\n\n") |
| import numpy as np a=[  [ [1,2,3],[4,5,6],[7,8,9] ],  [ [10,11,12],[13,14,15],[16,17,18] ],  [ [19,20,21],[22,23,24],[25,26,27] ],  ] |



|  |
| --- |
| arr3 = np.array(a)  print(arr3 [1:2,1:2,1], end=" [+] - slicing with [1:2,1:2,1] initialization\n\n") |

OUTPUT:

|  |
| --- |
|  |
|  |
|  |

PROGRAM: 4

AIM: WAPP TO USE NUMPY STATISTICAL FUNCTIONS ON ARRAY. CODE:

|  |
| --- |
| import numpy as np  arr = np.array([[0,1,2], [3,4,5], [6,7,8]], dtype=int) print(arr, end=" [+] - normal initialization\n\n")  print("The Maximum element in the array is:",np.max(arr), end= " \n\n") print("The Minimum element in the array is:",np.min(arr), end= " \n\n") print("The Mean of the array is:",np.mean(arr), end= " \n\n")  print("The Median of the array is:",np.median(arr), end= " \n\n") print("The Standard Deviation of the array is:",np.std(arr), end= " \n\n") print("The Varience of the array is:",np.var(arr), end= " \n\n")  print("The Avarage of the elements in the array is:",np.average(arr), end= " \n\n")  print("The Percentile of the array is:",np.percentile(arr, 50), end= " \n\n") |

OUTPUT:

|  |
| --- |
|  |

PROGRAM: 5

AIM: WAPP USING ARITHMETICAL OPERATIONS AND FUCTIONS ON ARRAYS. CODE:

|  |  |
| --- | --- |
| A) | import numpy as np  arr1 = np.array([[0,1,2], [3,4,5], [6,7,8]], dtype=int)  arr2 = np.array([[9,8,7],[6,5,4],[3,2,1]], dtype=int)  print(f"Addition of two arrays:\n{arr1} +\n{arr2}\n\nAns: \n {np.add(arr1,arr2)}", end= " Using numpy add function") |
| B) | import numpy as np  arr1 = np.array([[0,1,2], [3,4,5], [6,7,8]], dtype=int)  arr2 = np.array([[9,8,7],[6,5,4],[3,2,1]], dtype=int)  print(f"\n\nSubtraction of two arrays:\n{arr1} -\n{arr2}\n\nAns: \n {np.subtract(arr1,arr2)}", end= " Using numpy subtract function") |
| C) | import numpy as np  arr1 = np.array([[0,1,2], [3,4,5], [6,7,8]], dtype=int)  arr2 = np.array([[9,8,7],[6,5,4],[3,2,1]], dtype=int)  print(f"\n\nMultiplication of two arrays:\n{arr1} x\n{arr2}\n\nAns: \n {np.multiply(arr1,arr2)}", end= " Using numpy multiply function") |
| D) | import numpy as np  arr1 = np.array([[0,1,2], [3,4,5], [6,7,8]], dtype=int)  arr2 = np.array([[9,8,7],[6,5,4],[3,2,1]], dtype=int)  print(f"\n\nDivision of two arrays:\n{arr1} /\n{arr2}\n\nAns: \n {np.divide(arr1,arr2)}", end= " Using numpy divide function") |
| E) | import numpy as np  arr1 = np.array([[0,1,2], [3,4,5], [6,7,8]], dtype=int)  arr2 = np.array([[9,8,7],[6,5,4],[3,2,1]], dtype=int)  print(f"\n\nModulo of two arrays:\n{arr1} %\n{arr2}\n\nAns: \n {np.mod(arr1,arr2)}", end= " Using numpy modulo function") |
| F) | import numpy as np  arr1 = np.array([[0,1,2], [3,4,5], [6,7,8]], dtype=int)  arr2 = np.array([[9,8,7],[6,5,4],[3,2,1]], dtype=int)  print(f"\n\nPower of two arrays:\n{arr1} ^\n{arr2}\n\nAns: \n {np.power(arr1,arr2)}", end= " Using numpy power function") |

OUTPUT:

|  |
| --- |
|  |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |