**2.** Write following program on NumPy Array:

1. Create an array of all the even integers from 30 to 70.
2. Create an array of 10 zeros, other with 10 ones, and one more with10 fives.
3. Create a vector of length 10 with values evenly distributed between 5 and 50.
4. Create a 3x4 matrix filled with values from 10 to 21 and compute sum of all elements, sum of each column and sum of each row of a given array.
5. Create a 3x4 array and find the missing data in the array.
6. Calculate round, floor, ceiling, truncated and round (to the given number of decimals) of the input, elementwise of an array.
7. Find the maximum and minimum value, median, Weighted average, mean, standard deviation, variance, covariance matrix, of a given flattened array, minimum and maximum value along the second axis
8. Create a structured array from given student name, height, class and their data types. Now sort by class, then height if class are equal.

```
import numpy as np
from termcolor import colored

class color:
   BOLD = '\033[1m'
   END = '\033[0m'

print("12002040701067")
def Seperator():
  print("----------------------------------------------------------------------")

#1
array1 = np.arange(30,71,2)
print(color.BOLD+"1.Array of all the even integers from 30 to 70 : "+color.END)
print(array1)
Seperator()

#2
array2 = np.zeros(10)
print(color.BOLD+"2.An array of 10 zeros : "+color.END)
print(array2)
array3 = np.ones(10)
print(color.BOLD+"  An array of 10 ones : "+color.END)
print(array3)
array4 = np.ones(10)*5
print(color.BOLD+"  An array of 10 fives : "+color.END)
print(array4)
Seperator()

#3
array5 = np.linspace(10, 49, 5, retstep=True)
print(color.BOLD+"3.Length 10 with values evenly distributed between 5 & 50 : "+color.END)
print(array5)
array6 = np.linspace(5.1, 45.1, 5, retstep=True)
print(color.BOLD+"  Length 10 with values evenly distributed between 5 & 50 : "+color.END)
print(array6)
Seperator()
Seperator()

#4
array7 = np.arange(10,22).reshape((3, 4))
print(color.BOLD+"4.A 3x4 matrix filled with values from 10 to 21 : "+color.END)
print(array7)
print(color.BOLD+"  Sum of all elements : "+color.END)
print(np.sum(array7))
print(color.BOLD+"  Sum of each column : "+color.END)
print(np.sum(array7, axis=0))
print(color.BOLD+"  Sum of each row : "+color.END)
print(np.sum(array7, axis=1))
Seperator()

#5
array8 = np.array([[3, 2, np.nan, 1],
              [10, 12, 10, 9],
              [5, np.nan, 1, np.nan]])
print(color.BOLD+"5.Original array : "+color.END)
print(array8)
print(color.BOLD+"  Find the missing data of the said array : "+color.END)
print(np.isnan(array8))
Seperator()

#6
array9 = np.array([3.1, 3.5, 4.5, 2.9, -3.1, -3.5, -5.9])
print(color.BOLD+"6.Original array : "+color.END)
print(array9)
r1 = np.around(array9)
print(color.BOLD+"  Around : "+color.END, r1)
r2 = np.floor(array9)
print(color.BOLD+"  Floor  : "+color.END,r2)
r3 = np.ceil(array9)
print(color.BOLD+"  Ceil   : "+color.END,r3)
r4 = np.trunc(array9)
print(color.BOLD+"  Trunc  : "+color.END,r4)
r5 = [round(elem) for elem in array9]
print(color.BOLD+"  Round  : "+color.END,r5)
Seperator()

#7
array10 = np.array([[1,4,7], [2,5,8],[3,6,9]])
print(color.BOLD+"7.Original matrix-array : "+color.END)
print(array10)
b = array10.flatten('F')
print(color.BOLD+"  Original array : "+color.END)
print(b)
print(color.BOLD+"  Maximum value            : "+color.END,max(b))
print(color.BOLD+"  Minimum value            : "+color.END,min(b))
print(color.BOLD+"  Median value             : "+color.END,np.median(b))
print(color.BOLD+"  Average value            : "+color.END,np.average(b))
print(color.BOLD+"  Mean value               : "+color.END,np.mean(b))
print(color.BOLD+"  Standard-Deviation value : "+color.END,np.std(b))
print(color.BOLD+"  Variance value           : "+color.END,np.var(b))
print(color.BOLD+"  Covariance matrix value  : "+color.END,np.cov(b))
print(color.BOLD+"  Maximum value along the second axis :"+color.END)
print(np.amax(array10, 1))
print(color.BOLD+"  Minimum value along the second axis :"+color.END)
print(np.amin(array10, 1))
print(np.amin(array10, 1))
Seperator()

#8
data_type = [('name', 'S15'), ('class', int), ('height', float)]
students_details = [('James', 5, 48.5), ('Nail', 6, 52.5),
                    ('Paul', 5, 42.10), ('Pit', 5, 40.11)]
# create a structured array
students = np.array(students_details, dtype=data_type)
print(color.BOLD+"8.Original array : "+color.END)
print(students)
print(color.BOLD+"  Sort by height : "+color.END)
print(np.sort(students, order='height'))
```

```
12002040701067
1.Array of all the even integers from 30 to 70 :
[30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70]
----------------------------------------------------------------------
2.An array of 10 zeros :
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  An array of 10 ones :
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
  An array of 10 fives :
[5. 5. 5. 5. 5. 5. 5. 5. 5. 5.]
----------------------------------------------------------------------
3.Length 10 with values evenly distributed between 5 & 50 :
(array([10.  , 19.75, 29.5 , 39.25, 49.  ]), 9.75)
  Length 10 with values evenly distributed between 5 & 50 :
(array([ 5.1, 15.1, 25.1, 35.1, 45.1]), 10.0)
----------------------------------------------------------------------
4.A 3x4 matrix filled with values from 10 to 21 :
[[10 11 12 13]
 [14 15 16 17]
 [18 19 20 21]]
  Sum of all elements :
186
  Sum of each column :
[42 45 48 51]
  Sum of each row :
[46 62 78]
----------------------------------------------------------------------
5.Original array :
[[ 3.  2. nan  1.]
 [10. 12. 10.  9.]
 [ 5. nan  1. nan]]
  Find the missing data of the said array :
[[False False  True False]
 [False False False False]
 [False  True False  True]]
----------------------------------------------------------------------
6.Original array :
[ 3.1  3.5  4.5  2.9 -3.1 -3.5 -5.9]
  Around :  [ 3.  4.  4.  3. -3. -4. -6.]
  Floor  :  [ 3.  3.  4.  2. -4. -4. -6.]
  Ceil   :  [ 4.  4.  5.  3. -3. -3. -5.]
  Trunc  :  [ 3.  3.  4.  2. -3. -3. -5.]
  Round  :  [3, 4, 4, 3, -3, -4, -6]
----------------------------------------------------------------------
7.Original matrix-array :
[[1 4 7]
 [2 5 8]
 [3 6 9]]
```

```
  Original array :
[1 2 3 4 5 6 7 8 9]
  Maximum value           :  9
  Minimum value           :  1
  Median value            :  5.0
  Average value           :  5.0
  Mean value              :  5.0
  Standard-Deviation value :  2.581988897471611
  Variance value          :  6.666666666666667
  Covariance matrix value  :  7.5
  Maximum value along the second axis :
[7 8 9]
  Minimum value along the second axis :
[1 2 3]
----------------------------------------------------------------------
8.Original array :
[(b'James', 5, 48.5 ) (b'Nail', 6, 52.5 ) (b'Paul', 5, 42.1 )
 (b'Pit', 5, 40.11)]
  Sort by height :
[(b'Pit', 5, 40.11) (b'Paul', 5, 42.1 ) (b'James', 5, 48.5 )
 (b'Nail', 6, 52.5 )]
```

✓ 0s completed at 8:31 PM