

Name:  
Number:

Object Oriented Programming II  
2023-2024 Spring  
Midterm

26.04.2024

**100 Minutes**

---

1) Assume that we have two non-zero vectors (i.e.  $v1$ , and  $v2$ ) with  $x$  and  $y$  coordinates. Write a Python program to transform the second vector to create a set of orthogonal vectors (40 pts). Your program must perform the following steps:

- i) The program must include **main** function. In the main function, define two **empty** lists for vectors. Then, call **inititalizeVectors** function.
- ii) The program must include **inititalizeVectors** function that receives the lists. Coordinates must be in interval  $[2, 8]$ . In the main function, call **inititalizeVectors** function two times and print the vectors.
- iii) The program must include **dotProduct** function that receives the vectors and returns the dot product of the vectors.

$$dot = v1x * v2x + v1y * v2y$$

- iv) In the main function, call **dotProduct** function and print the dot product of two vectors.
- v) The program must include **orthogonalization** function that receives the vectors. In the function, implement the following formulas to transform the second vector to be orthogonal to first vector.

$$v2xOrth = v2x - v1x * \left( \frac{dot(v1, v2)}{dot(v1, v1)} \right)$$
$$v2yOrth = v2y - v1y * \left( \frac{dot(v1, v2)}{dot(v1, v1)} \right)$$

- vi) In main function, print the vectors after orthogonalization process.
- vii) The program must include **check** function that receives the  $v1$  and normalized  $v2$  vectors and return True, if the dot product is almost equal to zero (**Hint: Use  $dot(..) < 1e-6$** ) which means that the process is correct. Otherwise something is wrong and returns False. In the main, call **check** function and print the situation of the process.

The output of the program must be as follows: **Hint: If you use `random.seed(10)`, you will get the same results.**

```
Before Coordinates for V1 [2, 9]
Before Coordinates for V2 [8, 2]
Dot Product: 34

After Coordinates for V1 [2, 9]
After Coordinates for V2 [7.2, -1.6]

The process is correct
```

Save your source file as **151220xxxxxx\_BxxxxKxxxxx\_q1.py**.

2) In this question, you will write a python program to determine the points inner and outer of a boundary of a cube (60pts). **Hint: If you use `np.random.seed(50)`, you will get the same results.** The program must include the following steps:

- i) The program must include **generateBoundaries** function. In the function, generate 3 points for maximum boundaries of  $x$ ,  $y$ , and  $z$  axis in interval  $[16, 19]$  and store maximum boundaries in a numpy array (i.e `max_bound`). Also, generate 3 points for minimum boundaries of  $x$ ,  $y$ , and  $z$  axis in interval  $[2, 5]$  and store minimum boundaries in a numpy array (i.e `min_bound`). The function will return `max_bound` and `min_bound` numpy arrays (5 pts). Output for `max_bound` and `min_bound` numpy arrays are given in Figure 1.

max_bound	Array of int32	(3,)	[16 16 17]
min_bound	Array of int32	(3,)	[3 4 2]

Fig. 1 Output for max\_bound and min\_bound numpy arrays

ii) The program must include ***generatePointCloud*** function. The function generates 15 points with x, y, and z coordinates, which will be integer values between 0 and 20 and store these values in a **numpy array**. Then, return the numpy array (10 pts). Output for numpy array is given in Figure 2.

pc - NumPy object array

	0	1	2
0	6	5	6
1	13	5	2
2	7	15	4
3	14	3	6
4	11	17	10
5	9	0	6
6	19	2	15
7	9	3	19
8	19	2	12
9	0	3	2
10	10	16	14
11	19	0	0
12	11	7	19
13	14	13	7
14	4	4	0

Fig. 2 Output for numpy array

df - DataFrame

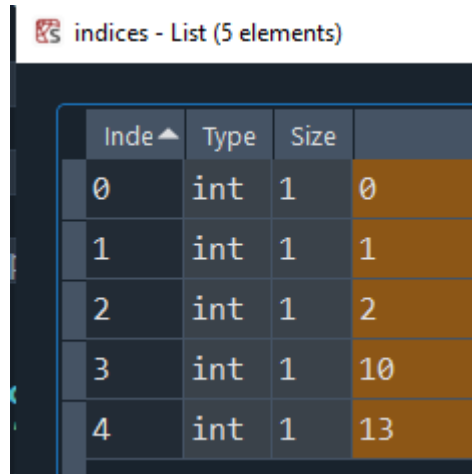
Index	axis_1	axis_2	axis_3
0	6	5	6
1	13	5	2
2	7	15	4
3	14	3	6
4	11	17	10
5	9	0	6
6	19	2	15
7	9	3	19
8	19	2	12
9	0	3	2
10	10	16	14
11	19	0	0
12	11	7	19
13	14	13	7
14	4	4	0

Fig. 3 Output for DataFrame

iii) In main function, save the numpy array to a file with .csv extension. The filename must be point\_cloud\_yourname\_yoursurname.csv. The values in the file must be separated with the asterisks character (“\*”) (5 pts).

iv) The program must include ***convertPCToDataFrame*** function. The function receives the numpy array and returns a DataFrame. In the function, convert the numpy array to a DataFrame (**Hint: You can define Series for each column and add the columns to the DataFrame**). The column headers of the DataFrame must be axis\_1, axis\_2, and axis\_3 for x, y, and z axes. (10 pts). Output for DataFrame is given in Figure 3.

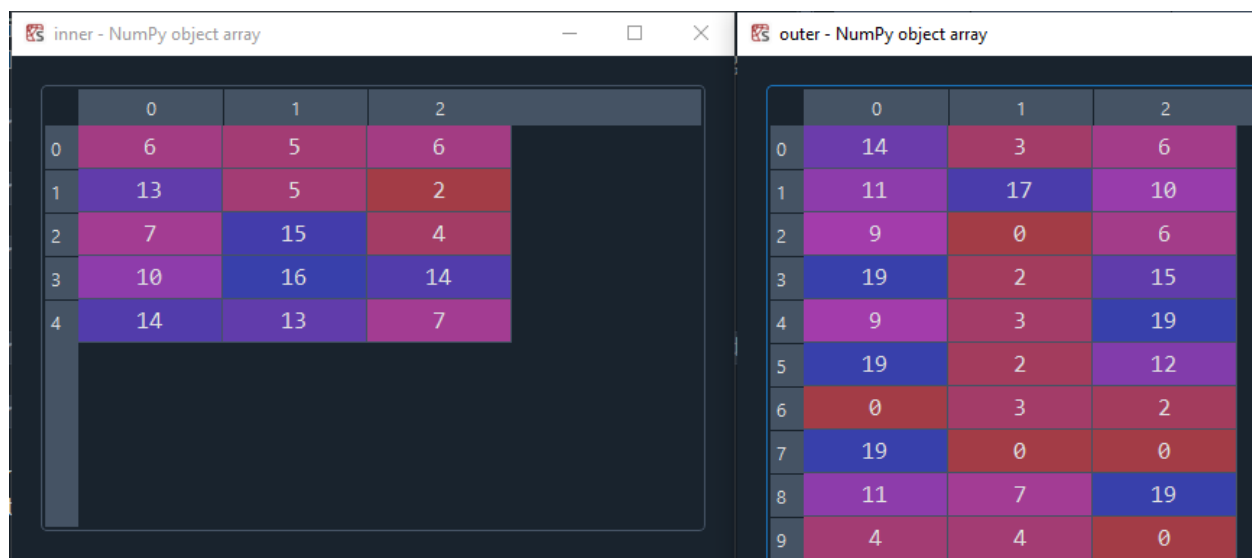
v) The program must include *findIndicesOfInnerPoints* function. The function receives the DataFrame and the maximum and minimum boundaries of the cube for x, y, and z axis. In the function, first define an empty list. Then, determine the indices of the points that are situated inner of the boundaries and push the points to list. **In the function, use pandas library to filter the points in the point cloud. To implement conditions use & instead of and keyword. You can use index data attribute of the DataFrame.** The function must return the list (25 pts). Output for the list is given in Figure 4.



Inde	Type	Size	
0	int	1	0
1	int	1	1
2	int	1	2
3	int	1	10
4	int	1	13

Fig. 4 Output for the list corresponding to DataFrame given in Fig. 3

vi) The program must include *findPoints* function. The function receives the numpy array of point cloud and indices list. The points corresponding to indices will be assigned to a new numpy array (i.e inner) and the remaining points will also assign to another numpy array (i.e outer). **You can use delete method of the numpy to obtain remaining points.** In the function, write inner and outer numpy arrays to files with .csv extension. The filenames must be inner\_points\_yourname\_yoursurname.csv and outer\_points\_yourname\_yoursurname.csv. The values in the files must be separated with the asterisks character ("\*"). (20 pts). Output after inner and outer numpy arrays are given in Figure 5.



	0	1	2
0	6	5	6
1	13	5	2
2	7	15	4
3	10	16	14
4	14	13	7

	0	1	2
0	14	3	6
1	11	17	10
2	9	0	6
3	19	2	15
4	9	3	19
5	19	2	12
6	0	3	2
7	19	0	0
8	11	7	19
9	4	4	0

Fig. 5 Output for inner and outer numpy arrays

vii) The program must include ***plotFilteredPoints*** function. The function receives numpy array of point cloud, the filenames of inner and outer and plots the inner and outer points via matplotlib.pyplot library. In the function, read the files to a numpy arrays. Then, add the following statements to your function. Use **scatter** method to plot the points. Output is given Figure 6. In the figure, inner and outer points are shown with red and green colors, respectively. Print indices to the figure using **text** method with numpy array of point cloud. Also, add axis labels (15 pts).

```
fig = plt.figure() // add the statement to your code
ax = fig.add_subplot(111, projection='3d') // add the statement to your code
```

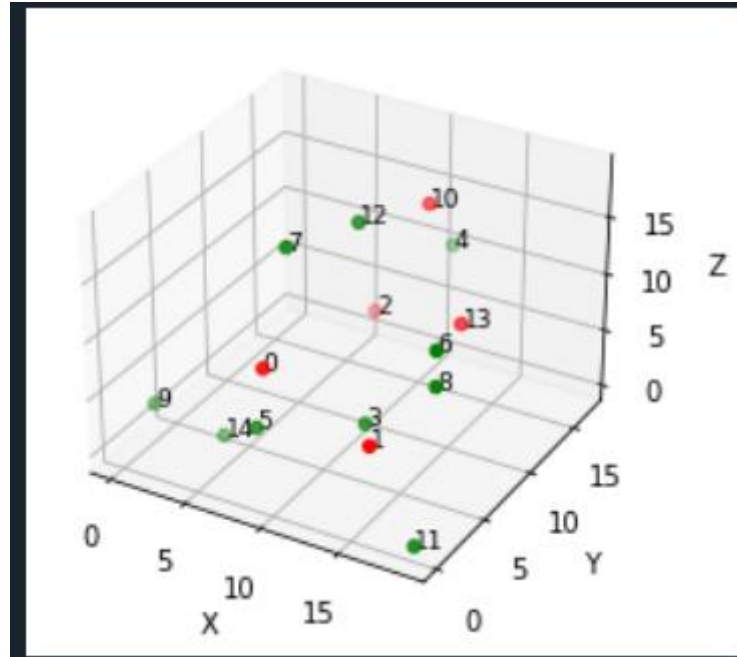


Fig. 6 Output for plotFilteredPoints function

viii) The program must include ***main*** function. In the function, call the functions ***generateBoundaries***, ***generatePointCloud***, ***convertPCToDataFrame***, ***findIndicesOfInnerPoints***, ***findPoints***, ***plotFilteredPoints*** (10 pts).

Create a folder named 151220xxxxxx\_BxxxxKxxxxx\_xx in D.

Save your source file as 151220xxxxxx\_BxxxxKxxxxx\_q2.py. Upload the file, point\_cloud\_yourname\_yoursurname.csv and extracted\_points\_yourname\_yoursurname.csv and remaining\_points\_yourname\_yoursurname.csv to DYS.

Good Luck!