**SCHOOL OF COMPUTING (SOC)**

**IT8701 Introduction to Programming for Data Science**

**Practical 2 Submission Worksheet (Graded as part of CA3)**

|  |
| --- |
| **Instructoins:**   1. Submit this at Polymall “Lab Submissions->Lab 2 Submission” folder 2. Name your file “YourLecturer-YourStudentID-YourName.docx” |

|  |  |
| --- | --- |
| **Name** | Muhammad Iylia Bin Mohd Hutta |
| **Student ID** | P7474841 |
| **Class** | |  | | --- | | NSDDA1/CE/2220/4 | |

# Section 6 Tasks

### Task 1: Concatenate arrays

|  |
| --- |
| *import* numpy *as* np  *# (a) write code to concatenate the following arrays on axis = 0*  a = np.array([[1,2,3],[4,5,6],[7,8,9]])  b = np.full((3,3),1.5)  c = np.arange(0,15).reshape(5,3)  print(a.shape)  print(b.shape)  print(c.shape)  print(np.concatenate((a,b,c)))  *# (b) write code to concatenate the following arrays on axis = 1*  a = np.array([[1,2,3,4],[4,5,6,7],[7,8,9,10],[11,12,13,14]])  b = np.random.randint(100,200,(4,6))  c = np.arange(0,40).reshape(4,10)  print(a.shape)  print(b.shape)  print(c.shape)  print(np.concatenate((a,b,c), *axis*=1)) |

|  |
| --- |
|  |

# Section 8 Tasks

### Task 1: Basic Sorting

|  |
| --- |
| *import* random  *# (a) write code to sort the following 1d-array in-place using the sort method*  arr\_1 = np.random.randint(100,200,10)  print('\*\*Before sorting\*\*')  print(arr\_1)  arr\_1.sort()  print('\n\*\*After sorting\*\*')  print(arr\_1)  *# (b) write code to sort the following 2-d array in-place by columns*  arr\_2 = np.random.randint(1,20,(3,5))  print('\*\*Before sorting\*\*')  print(arr\_2)  arr\_2.sort(*axis*=0)  print('\n\*\*After sorting\*\*')  print(arr\_2)  *# (c) write code to sort the following 2d-array by rows without affecting the original*  arr\_3 = np.random.randint(100,200,(2,5))  print('\*\*Before sorting - original array\*\*')  print(arr\_3)  arr\_3c = np.copy(arr\_3)  arr\_3c.sort(*axis*=1)  print('\n\*\*After calling sort method - original array\*\*')  print(arr\_3)  print('\nAfter calling sort method - copy of sorted array\*\*')  print(arr\_3c) |

|  |
| --- |
| Calendar  Description automatically generated  Text  Description automatically generated |

# Section 9 Tasks

### Task 2: Boolean indexing

|  |
| --- |
| *# (a) write code to return all even numbers in array a*  print(a)  print()  *# a\_even = a % 2 == 0*  *# a = a[a\_even]*  *# print(a)*  print(a[a%2==0])  *# (b) write code to return all the numbers that are greater than 150 in array b*  print(b)  print()  *# b\_over\_150 = b > 150*  *# b = b[b\_over\_150]*  *# print(b)*  print(b[b>150]) |

|  |
| --- |
| Text  Description automatically generated with low confidence  Text  Description automatically generated |

# Section 11 Tasks

### Task 1: sum,mean

|  |
| --- |
| a = np.array((np.arange(0,10),  np.arange(10,20),  np.arange(20,30),  np.arange(30,40)))  print("Contents of array a")  print(a)  print()  *# (a) What is the sum of all the numbers in array a?*  print("\*\*\* Sum of all numbers in a \*\*\*")  print(a.sum())  print()  *# (b) What is their mean?*  print("\*\*\* Mean of all numbers in a \*\*\*")  print(a.mean())  print()  *# (c) What is the sum of all the numbers in each row?*  print("\*\*\* Sum of all numbers in each row \*\*\*")  print(f"Row 1 sum = {a.sum(*axis*=1)[0]}")  print(f"Row 2 sum = {a.sum(*axis*=1)[1]}")  print(f"Row 3 sum = {a.sum(*axis*=1)[2]}")  print(f"Row 4 sum = {a.sum(*axis*=1)[3]}") |

|  |
| --- |
|  |

# Section 12 Tasks

### Task 1: Load numpy array from textfile and save it

|  |
| --- |
| *import* numpy *as* np  *### Read the csv file with the loadtxt() function*  fname = "/Users/iylia/Library/Mobile Documents/com~apple~CloudDocs/SP/Intro to Programming for DS/Lab 2/Lab2\_Dataset/singapore-residents-by-ethnic-group-and-sex-end-june-annual.csv"  data = np.loadtxt(fname,*skiprows*=1,*dtype*=[('year','i8'),('level\_1','U50'),('value','i8')],*delimiter*=",")  *### Print out total rows of data in the file*  print(f"There are altogether {len(data)} rows of data in the file {fname}")  print()  *### Print out the number of years of data captured*  data\_years = data['year'] *# Just extract the year column*  years = np.unique(data\_years) *# Get the unique values in this column*  print(f"There are {len(years)} years of data captured from 1960 to 2016")  print()  *### Extract only the rows with “Total Residents" in the “level\_1” column*  keyword = 'Total Residents'  column\_to\_search = data['level\_1']  out = [i *for* i, v *in* enumerate(column\_to\_search) *if* keyword in v]  data\_totalresidents = data[out]  *### Print out the year which has the highest total number of residents*  max = data\_totalresidents['value'].max()  argmax = data\_totalresidents['value'].argmax()  print(f"Year with the highest total number of residents: {str(data\_totalresidents[argmax]['year'])}")  print(f"Population Count: {str(max)}")  print()  *### Print out the year which has the lowest total number of residents*  min = data\_totalresidents['value'].min()  argmin = data\_totalresidents['value'].argmin()  print(f"Year with the highest total number of residents: {str(data\_totalresidents[argmin]['year'])}")  print(f"Population Count: {str(min)}")  print() |

|  |
| --- |
|  |