

CT - 2

Que. No. 1 a)

Assume the two arrays given below:

[1 2 3]

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

Using NumPy, write a code for the following operations:

- i) Creation of the above two NumPy arrays.
- ii) Apply `intersect1d()` and `cumsum()` on them.

▼ Sample Answer:

#i) Creation of the given two NumPy arrays

```
import numpy as np
```

```
a = np.array([1,2,3])  
print(a)
```

```
[1 2 3]
```

```
b = np.arange(1, 10).reshape(3,3)  
print(b)
```

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

#ii)

```
#Applying intersect1d() on them  
c = np.intersect1d(a,b)  
print(c)
```

```
[1 2 3]
```

#Applying `cumsum()` on them

```
x = np.cumsum(a)  
y = np.cumsum(b)
```

```
print(x)  
print(y)
```

```
[1 3 6]
```

CT 2

Que. No. 1 b)

What is the use of Boolean Indexing? Explain various functions available for NumPy array creation.

▼ Sample Answer:

▼ Boolean Indexing

- Boolean Indexing is used to selection of elements from an array using the index values based on conditions.
- It is used for filtering the desired element values.
- It has some boolean expression as the index.
- Those elements which satisfy that Boolean expression are resulted as output.
- **For Example:**

```
import numpy as np

a = np.array([10, 40, 80, 50, 100])

print(a[a>50])

[ 80 100]
```

▼ Various Functions for NumPy Array Creations are:

1. NumPy Array Creation using List and Tuples	
Function	Description
array()	Converts input data (list, tuple, array) to a ndarray <i>(Input can be lists, lists of tuples, tuples, tuples of tuples, tuples of lists and arrays.)</i>
2. NumPy Array Creation using Built-in Functions / Intrinsic NumPy Array Creation	
Function	Description
arange()	arange(start, stop, step) - the arange() returns a ndarray. (Similar to range())
ones() ones_like()	Creates a new array of all 1s ones_like takes already existing array and creates a 1s array of the same shape and data type
zeros() zeros_like()	Create a new array of all 0s zeros_like takes already existing array and creates a 0s array of the same shape and data type
empty() empty_like()	Creates new arrays by removing the array values and allocates random values. empty_like takes already existing array and converts it into an empty array.
full() full_like()	Creates an array with all values set to the indicated "full value" full_like takes already existing array and converts it to a filled array with the indicated value.
eye()	Create any M × N identity matrix (1s on the diagonal and 0s elsewhere)
identity()	Create a square N × N identity matrix of given array size.
3. NumPy Array Creation using Random Functions	
Function	Description
rand()	Creates an array of specific shape with random values
randint()	Create an array filled with random integers values

CT 2

Que. No. 1 c)

Using suitable code, create a 2D and 3D NumPy array and perform arithmetic operations between them.

▼ Sample Answer:

▼ Arithmetic with NumPy Arrays

- There are specific functions in NumPy for performing arithmetic operations - addition, subtraction, multiplication, divisions and remainder.

Function	Description
add()	Add corresponding elements in arrays
subtract()	Subtract elements in second array from first array
multiply()	Multiply array elements
divide()	Divide first array elements with second
mod() / remainder()	Element-wise modulus (remainder of division)

```
import numpy as np

#Creation of 2D array
a = np.array([[2,2], [2,2]])

#Creation of 3D array
b= np.array([ [[1,1],[1,1]] , [[1,1],[1,1]] ])

print("The 2D array is:\n", a)

print("The 3D array is:\n", b)

print("Addition of a and b is:\n",np.add(a,b))

print("Multiplication of a and b is:\n",np.multiply(a,b))

print("Subtraction of a and b is:\n",np.subtract(a,b))

print("Division of a by b is:\n",np.divide(a,b))

print("Reaminder of a and b is:\n",np.remainder(a,b))
```

```
The 2D array is:
[[2 2]
 [2 2]]
The 3D array is:
[[[1 1]
  [1 1]]
 [[1 1]
  [1 1]]]
Addition of a and b is:
[[[3 3]
  [3 3]]
 [[3 3]
  [3 3]]]
Multiplication of a and b is:
[[[2 2]
  [2 2]]
 [[2 2]
  [2 2]]]
Subtraction of a and b is:
[[[1 1]
  [1 1]]
 [[1 1]
  [1 1]]]
Division of a by b is:
[[[2. 2.]
  [2. 2.]]
 [[2. 2.]
  [2. 2.]]]
Reaminder of a and b is:
```

```
[[[0 0]
  [0 0]]
 [[0 0]
  [0 0]]]
```

CT 2

Que. No. 1 d)

Write a short note on:

- 1. String Manipulation Functions
- 2. Plotting using Pandas

▼ Sample Answer:

▼ 1) String Manipulation Functions

- To clean and preprocess string so that it can be analyzed and used by the algorithms Pandas library has its own string manipulation functions.
- Pandas provides a set of built-in string functions which make it easy to operate on string data. These are called String Manipulation Functions.

S. No.	Function	Description
1	lower()	Converts strings to lower case.
2	upper()	Converts strings to upper case.
3	islower()	Checks whether all characters in each string in the Given column are in lower case or not. Returns Boolean
4	isupper()	Checks whether all characters in each string in the Given column are in upper case or not. Returns Boolean.
5	isnumeric()	Checks whether all characters in each string in the Given column are numeric. Returns Boolean.
6	len()	Computes length of the string.
7	split()	Splits each string with the given pattern.
8	startswith(pattern)	Returns true if the element in the Given column starts with the pattern.
9	endswith(pattern)	Returns true if the element in the Given column ends with the pattern.
10	find(pattern)	Returns the first position of the first occurrence of the pattern.
11	findall(pattern)	Returns a list of all occurrence of the pattern
12	contains(pattern)	Returns a Boolean value True for each element if the substring contains in the element, else False.

For Example:

```
#Consider the following DataFrame
```

```
import pandas as pd
a = {'name': ['Michael Smith', 'Ana Kors', 'Sean Bill', 'Carl Jonson', 'Bob Evan'],
     'age': [19, 19, 17, 18, 20],
     }

df = pd.DataFrame(a)
df
```

	name	age
0	Michael Smith	19
1	Ana Kors	19
2	Sean Bill	17
3	Carl Jonson	18
4	Bob Evan	20



```
#Applying String Manipulation Functions
```

```
df.name.str.upper()
```

```
0    MICHAEL SMITH
1         ANA KORS
2        SEAN BILL
3     CARL JONSON
4        BOB EVAN
Name: name, dtype: object
```

```
df.name.str.lower()
```

```
0    michael smith
1         ana kors
2        sean bill
3     carl jonson
4        bob evan
Name: name, dtype: object
```

```
df.name.str.len()
```

```
0    13
1     8
2     9
3    11
4     8
Name: name, dtype: int64
```

▼ 2) Plotting using Pandas

Pandas is a data analysis tool, but it also provides great options for data visualization.

Data visualization is the representation of data through use of common graphs, charts, plots, etc.

Pandas uses the `plot()` method to create graphs.

Syntax:

```
df.plot()
```

Parameters:

- **data** : Series or DataFrame
- **x and y parameters** specify the values that you want on the x and y column.
- **figsize** specifies the size of the figure object.
- **title** to be used for the plot.
- **kind**: the kind of plot to produce.

Different Plotting methods

In df.plot(), the kind parameter has the following options to implement plotting. These are:

1. Line Plot

kind = 'line' is used for plotting graph in the line format.

2. Bar plot

kind = 'bar' plots categorical data with rectangular bars. The lengths of the bars are proportional to the values that they represent.

3. Scatter plot or Point plot

kind = 'scatter' is used to plot the points that represents correlations between two variables.

4. 'hist' for histogram

kind = 'hist' is used to show frequency distributions.

5. 'kde' or 'density' for density plots

kind = 'density' is used to generate a **Kernel Density Estimate (KDE)** plot using Gaussian kernels (i.e. Probability of given data points along Gaussian curve).

6. 'pie' for pie plots

kind = 'pie' , a pie plot is a proportional representation of numerical data.

For Example:

```
#Consider the following DataFrame

import pandas as pd
a = {'name': ['Michael Smith', 'Ana Kors', 'Sean Bill', 'Carl Jonson', 'Bob Evan'],
      'age': [19, 19, 17, 18, 50],
      }

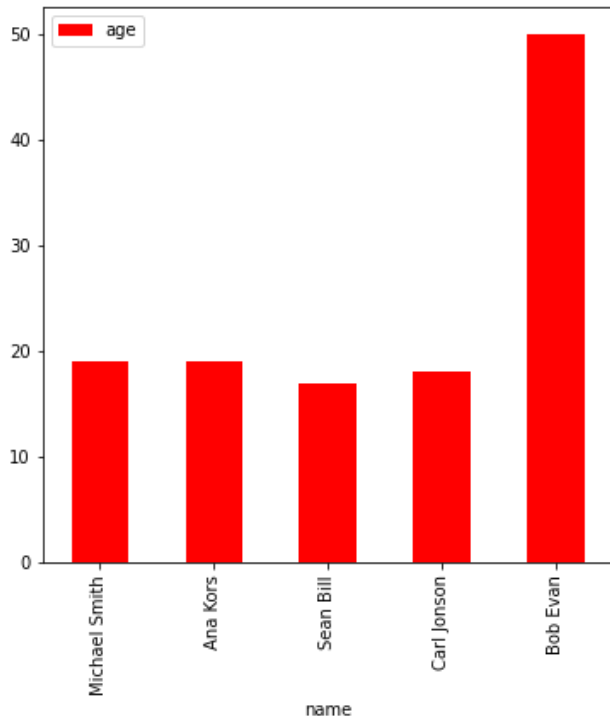
df = pd.DataFrame(a)
df
```

	name	age
0	Michael Smith	19
1	Ana Kors	19
2	Sean Bill	17
3	Carl Jonson	18
4	Bob Evan	50



```
df.plot(kind='bar' , x='name', y='age', color='red', figsize=(6,6))
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f0e4411c990>



CT 2

Que. No. 2 c)

Assume the data given below:

	Name	Score	Attempts	Qualify
a	James	35.0	3	Yes
b	Emily	19.0	2	No
c	Michael	38.0	3	Yes
d	Mathew	20.5	1	Yes
e	Laura	13.5	1	No
f	Kevin	NaN	2	No
g	Jonas	36.0	1	Yes

Using Pandas, write a code for the following operations:

- Creating a DataFrame for these data.
- Change the new column name 'Score' to 'CT marks'
- Drop the row with 'NaN' value.
- Sort the 'Score' column in descending order.

Sample Answer:

```
#i) Creating a DataFrame for these data
```

```
import pandas as pd
import numpy as np
```



```
a = pd.DataFrame({'Name': ['James', 'Emily', 'Michael', 'Mathew', 'Laura', 'Kevin', 'Jonas'],
'Score': [35.0, 19, 38, 20.5, 13.5, np.nan, 36],
'Attempts': [3, 2, 3, 1, 1, 2, 1],
'Qualify': ['yes', 'no', 'yes', 'yes', 'no', 'no', 'yes']},
index = ['a', 'b', 'c', 'd', 'e', 'f', 'g'])
```

a

	Name	Score	Attempts	Qualify
a	James	35.0	3	yes
b	Emily	19.0	2	no
c	Michael	38.0	3	yes
d	Mathew	20.5	1	yes
e	Laura	13.5	1	no
f	Kevin	NaN	2	no
g	Jonas	36.0	1	yes

```
#ii) To change the new column name ‘Score’ to ‘CT marks’
```

```
a.rename(columns={'Score':'CT marks'})
```

	Name	CT marks	Attempts	Qualify
a	James	35.0	3	yes
b	Emily	19.0	2	no
c	Michael	38.0	3	yes
d	Mathew	20.5	1	yes
e	Laura	13.5	1	no
f	Kevin	NaN	2	no
g	Jonas	36.0	1	yes

```
# iii) To drop the row with ‘NaN’ value
```

```
a.dropna(inplace = True)
```

a

	Name	Score	Attempts	Qualify
a	James	35.0	3	yes
b	Emily	19.0	2	no
c	Michael	38.0	3	yes
d	Mathew	20.5	1	yes
e	Laura	13.5	1	no
g	Jonas	36.0	1	yes

```
# iv) To sort the ‘Score’ column in descending order.
```

```
a.sort_values('Score',ascending=False)
```

	Name	Score	Attempts	Qualify
c	Michael	38.0	3	yes
g	Jonas	36.0	1	yes
a	James	35.0	3	yes
d	Mathew	20.5	1	yes



CT 2

Que. No. 2 d)

Explain `rank()`. Write a code using Pandas to add five new columns to the data given in the above question and store the values obtained by five different ranking methods in these new columns.

Sample Answer:

```
#We have
```

```
#DataFrame for the given data
```

```
import pandas as pd
import numpy as np
```

```
x = pd.DataFrame({'Name': [ 'James', 'Emily', 'Michael', 'Mathew', 'Laura', 'Kevin', 'Jonas'],
'Score': [35.0, 19, 38, 20.5, 13.5, np.nan, 36],
'Attempts': [3, 2, 3, 1, 1, 2, 1],
'Qualify': ['yes', 'no', 'yes', 'yes', 'no', 'no', 'yes']},
index = ['a', 'b', 'c', 'd', 'e', 'f', 'g'])
```

```
x
```

	Name	Score	Attempts	Qualify
a	James	35.0	3	yes
b	Emily	19.0	2	no
c	Michael	38.0	3	yes
d	Mathew	20.5	1	yes
e	Laura	13.5	1	no
f	Kevin	NaN	2	no
g	Jonas	36.0	1	yes



```
#Lets sort the DataFrame according to the Name column in ascending order
```

```
x.sort_values('Name',ascending=True, inplace = True)
```

```
x
```

	Name	Score	Attempts	Qualify
--	------	-------	----------	---------



b	Emily	19.0	2	no
----------	-------	------	---	----

```
#Now, applying the five ranking methods in the DataFrame
```

```
# Adding five new columns to store the ranks
```

```
x['dense_rank'] = x['Name'].rank(method='dense')
```

```
x['first_rank'] = x['Name'].rank(method='first')
```

```
x['min_rank'] = x['Name'].rank(method='min')
```

```
x['max_rank'] = x['Name'].rank(method='max')
```

```
x['average_rank'] = x['Name'].rank(method='average')
```

```
x
```

	Name	Score	Attempts	Qualify	dense_rank	first_rank	min_rank	max_rank	average_rank
--	------	-------	----------	---------	------------	------------	----------	----------	--------------



b	Emily	19.0	2	no	1.0	1.0	1.0	1.0	1.0
a	James	35.0	3	yes	2.0	2.0	2.0	2.0	2.0
g	Jonas	36.0	1	yes	3.0	3.0	3.0	3.0	3.0
f	Kevin	NaN	2	no	4.0	4.0	4.0	4.0	4.0
e	Laura	13.5	1	no	5.0	5.0	5.0	5.0	5.0
d	Mathew	20.5	1	yes	6.0	6.0	6.0	6.0	6.0
c	Michael	38.0	3	yes	7.0	7.0	7.0	7.0	7.0

Assignment 2

Que. No. 3)

Write a code to create the 0-Dimensional, 1-dimensional, 2-dimensional, 3-dimensional and 4-dimensional arrays using NumPy.

▼ Sample Answer:

```
import numpy as np
#Creating a 0-D array with value 10

a = np.array(10)

print(a)
```

```
10
```

```
#Creating a 1-D array containing the values 1,2,3,4,5

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

print(b)
```

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

```
#Create a 2-D array containing two arrays with the values 1,2,3 and 4,5,6:
```

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

```
print(c)
```

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

```
#Create a 3-D array (with two 2-D arrays) both containing two arrays with the values 1,2,3 and 4,5,6:
```

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

```
print(d)
```

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

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

```
#Create an array with 4 dimensions
```

```
e = np.array([1, 2], ndmin=4)
```

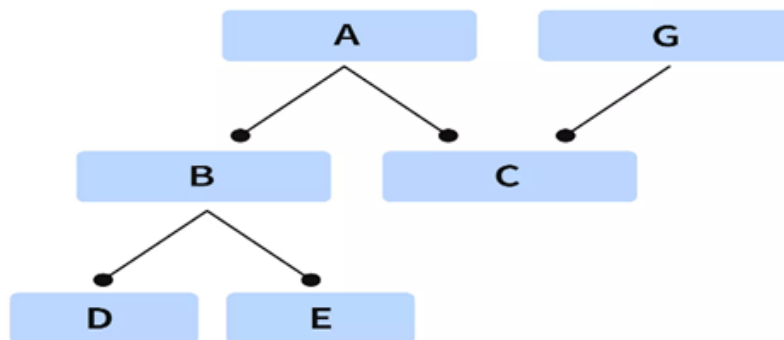
```
print(e)
```

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

▼ Assignment 2

Que. No. 4)

Write a program to implement the following inheritance:



▼ Sample Answer:

```
#Creation of Classes
```

```
class A:                                #Parent Class 1
    def A(self):
        print("This is Function A.")
```

```
class G:                                #Parent Class 2
    def G(self):
        print("This is Function G.")
```

```

class B(A):
    #Single Inheritance
    def B(self):
        print("This is Function B.")

class C(A, G):
    #Multiple Inheritance
    def C(self):
        print("This is Function C.")

class D(B):
    #Multilevel Inheritance
    def D(self):
        print("This is Function D.")

class E(B):
    #Multilevel Inheritance
    def E(self):
        print("This is Function E.")

```

#Creation of Object for each class

```
obj1 = A()
```

```
obj2 = B()
```

```
obj3 = C()
```

```
obj4 = D()
```

```
obj5 = E()
```

```
obj6 = G()
```

#obj1 and obj2 can access function A

```
obj1.A()
```

```
obj2.A()
```

#obj3 can access function G and B

```
obj3.G()
```

```
obj4.B()
```

#obj5 can access function B

```
obj5.B()
```

#obj6 can access function G

```
obj6.G()
```

```
This is Function A.
```

```
This is Function A.
```

```
This is Function G.
```

```
This is Function B.
```

```
This is Function B.
```

```
This is Function G.
```

Assignment 2

Que. No. 6)

Create a DataFrame with 4 columns: Name, ID, Age, Salary, and 5 rows. Write a code using Pandas to:

a) Sort By Rows in descending order.

b) Apply any ranking method

c) Display records with ID>2022, Age>25, and Salary>50000.

d) Add new column to the existing DataFrame.

e) Rename a Column.


▼ Sample Answer:

#Creating a DataFrame with 4columns: Name, ID, Age, Salary, and 5 rows

```
import pandas as pd
d1 = {
    "Name":["Kam","Sam","Scott","Ann","John"],
    "ID"  :[2080,2090,2015,2020,2095],
    "Age"  :[20,21,22,25,30],
    "Salary":[9000,8500,4000,2000,9500]
}
index_labels=['r1','r2','r3','r4','r5']


d1 = pd.DataFrame(d1, index=index_labels)

d1
```

	Name	ID	Age	Salary	
r1	Kam	2080	20	9000	
r2	Sam	2090	21	8500	
r3	Scott	2015	22	4000	
r4	Ann	2020	25	2000	
r5	John	2095	30	9500	

#a) Sort By Rows in descending order.


```
d1.sort_index(ascending = False)
```

	Name	ID	Age	Salary	
r5	John	2095	30	9500	
r4	Ann	2020	25	2000	
r3	Scott	2015	22	4000	
r2	Sam	2090	21	8500	
r1	Kam	2080	20	9000	

Apply any ranking method

```
d1['first_rank'] = d1['Salary'].rank(method='first')

d1
```

	Name	ID	Age	Salary	first_rank	
r1	Kam	2080	20	9000	4.0	
r2	Sam	2090	21	8500	3.0	
r3	Scott	2015	22	4000	2.0	
r4	Ann	2020	25	2000	1.0	
r5	John	2095	30	9500	5.0	

#c) Display records with ID>2022, Age>25, and Salary>50000

```
a=d1.where((d1.ID > 2022) & (d1.Age > 25) & (d1.Salary >5000), other ='-')

print(a)
```

	Name	ID	Age	Salary	first_rank
r1	-	-	-	-	-
r2	-	-	-	-	-
r3	-	-	-	-	-
r4	-	-	-	-	-
r5	John	2095	30	9500	5.0

```
#d) Add new column to the existing DataFrame.
```

```
d1['New_Column'] = ['A' , "B", "C", "D", "E"]

d1
```

	Name	ID	Age	Salary	first_rank	New_Column
r1	Kam	2080	20	9000	4.0	A
r2	Sam	2090	21	8500	3.0	B
r3	Scott	2015	22	4000	2.0	C
r4	Ann	2020	25	2000	1.0	D
r5	John	2095	30	9500	5.0	E

```
#e) Renaming Columns 'ID' and 'Age' to 'UID' and 'Years' respectively
```

```
a = d1.rename(columns = {'ID':'UID', 'Age':'Years'})

print(a)
```

	Name	UID	Years	Salary	first_rank	New_Column
r1	Kam	2080	20	9000	4.0	A
r2	Sam	2090	21	8500	3.0	B
r3	Scott	2015	22	4000	2.0	C
r4	Ann	2020	25	2000	1.0	D
r5	John	2095	30	9500	5.0	E