Sri Siddhartha Institute of Technology, Tumakuru

(A Constituent college of Sri Siddhartha Academy of Higher Education)

INFORMATION SCIENCE AND ENGINEERING

Data Processing Laboratory

LAB MANUAL (22IS304)

(III SEMESTER)

(2023-2024)

PREPARED BY

Ms. SEVANTHI M
4th SEM
Dept. of ISE
SSIT, Tumakuru - 05

Mr. SHARAN M G
4th SEM
Dept. of ISE
SSIT,Tumakuru-05

Syllabus for the Academic Year – 2023–2024
Department: Information Science and Engineering
Semester: 3

Subject Name: Data Processing Laboratory
Subject Code: 22IS304

L-T-P-C: 3-2-0-4

Course Objectives:

Sl. No	Description
1	Acquire the programming skills in core python.
2	Understand the functionalities available in Python libraries.
3	Familiarize rich data structures of Python to work with structured data in fast, easy and expressive way.
4	Learn data cleaning and preparation tools for data analysis.

Note: Implement the following using Python Programming Language:

1. Write a program to convert temperature to and from Celsius to Fahrenheit.

```
print("Options are \n")
print("1.Convert temperatures from Celsius to Fahrenheit \n")
print("2.Convert temperatures from Fahrenheit to Celsius \n")
opt=int(input("Choose any Option(1 or 2) : "))
if opt == 1:
   print("Convert temperatures from Celsius to Fahrenheit \n")
   cel = float(input("Enter Temperature in Celsius: "))
   fahr = (cel*9/5)+32
   print("Temperature in Fahrenheit =",fahr)
elif opt == 2:
   print("Convert temperatures from Fahrenheit to Celsius \n")
   fahr = float(input("Enter Temperature in Fahrenheit: "))
   cel=(fahr-32)*5/9;
   print("Temperature in Celsius =",cel)
else:
   print("Invalid Option")
```

OUTPUT:

```
Options are
1.Convert temperatures from Celsius to Fahrenheit
2.Convert temperatures from Fahrenheit to Celsius

Choose any Option(1 or 2) :
1
Convert temperatures from Celsius to Fahrenheit
Enter Temperature in Celsius:
33
Temperature in Fahrenheit = 91.4

Options are
```

```
1.Convert temperatures from Celsius to Fahrenheit
2.Convert temperatures from Fahrenheit to Celsius

Choose any Option(1 or 2) :
2
Convert temperatures from Fahrenheit to Celsius
Enter Temperature in Fahrenheit:
33
Temperature in Celsius = 0.5555555555555556
```

2. Write a script named copyfile.py. This script should prompt the user for the names of two text files and copy the contents of the first file to the second file.

file1.txt

```
This is python program
welcome to python
```

copyfile.py

```
file1=input("Enter First Filename : ")
file2=input("Enter Second Filename : ")
fn1 = open(file1, 'r')
fn2 = open(file2, 'w')
cont = fn1.readlines()
#type(cont)
for i in range(0, len(cont)):
    fn2.write(cont[i])
fn2.close()
print("Content of first file copied to second file ")
fn2 = open(file2, 'r')
cont1 = fn2.read()
print("Content of Second file :")
print(cont1)
fn1.close()
fn2.close()
```

OUTPUT:

```
Enter first filename : file1.txt
Enter second filename : file2.txt
Content of the first file copied to second file
Content of second file :
This is python program
welcome to python
```

3. Write a program to create, append and remove lists in python.

```
pets = ['cat', 'dog', 'rat', 'pig', 'tiger']
snakes=['python','anaconda','fish','cobra','mamba']
print("Pets are :",pets)
print("Snakes are :",snakes)
animals=pets+snakes
pets.append("lion")
print("list after apppending",pets)
print("Animals are :",animals)
```

```
snakes.remove("fish")
print("updated Snakes are :",snakes)
```

OUTPUT:

```
Pets are : ['cat', 'dog', 'rat', 'pig', 'tiger']
Snakes are : ['python', 'anaconda', 'fish', 'cobra', 'mamba']
list after apppending ['cat', 'dog', 'rat', 'pig', 'tiger', 'lion']
Animals are : ['cat', 'dog', 'rat', 'pig', 'tiger', 'python', 'anaconda', 'fish', 'cobra', 'mamba']
updated Snakes are : ['python', 'anaconda', 'cobra', 'mamba']
```

4. Python Program to Count Occurrences of an element in a list.

```
def count_occurrence(list, n):
    count=0
    for i in list:
        if(i==n):
            count=count+1
    return count
li=[]
n=int(input("Enter size of list "))
for i in range(0,n):
    e=int(input("Enter element of list "))
    li.append(e)
print("Original list: ",li)
x=int(input("Enter element to be checked list: "))
print(x," has occurred ",count_occurrence(li, x),"times")
```

OUTPUT:

```
Enter size of list
5
Enter element of list
2
Enter element of list
3
Enter element of list
2
Enter element of list
4
Enter element of list
5

Original list:  [2, 3, 2, 4, 5]
Enter element to be checked list:
4
    4    has occurred 1 times
```

5. Write a program to get a list of even numbers from a given list of numbers. (Use only comprehensions).

```
  x=[10,13,51,500,53]

 [i   for i in x   if i%2==0]
```

6. Write a program to generate an infinite number of even numbers (Use generator).

Logic 1: prints all the even numbers and converts in to list and gives the output.

```
def make():
  x=int(input("enter the value of x"))
  for x in range(x):
    if x%2==0:
      yield x

gen=make()
print(list(gen))
```

Logic 2: prints all the even numbers one at a time using next() method.

```
def make():
  x=int(input("enter the value of x"))
  for x in range(x):
    if x%2==0:
      yield x

gen=make()
print(next(gen))
```

7. Write a python program as a function which takes as parameter a tuple of string (s, s1) and which returns the index of the first occurrence of s1 found within the string s. The function must returns -1 if s1 is not found within the string s. Example if s = "Python Programming" and s1 = "thon" , the function returns the index 2.

```
def Find(s , s1):
    n = len(s)
    m = len(s1)
    k = -1
    for i in range(0 , n):
        if s[i:i+m] == s1:
            k = i
            break
    return k
s = "Python Programming"
s1 = "thon"
print(Find(s , s1))      # display 2
print(Find(s , 'thons')) # display -1
```

OUTPUT:

```
2
-1
```

8. Write a program to read text file data and create a dictionary of all keywords in the text file. The program should count how many times each word is repeated inside the text file and then find the keyword with a highest repeated number. The program should display both the keywords dictionary and the most repeated word.

Logic 1:

```
handle = open("Egypt.txt")
text = handle.read()
words = text.split()
counts = dict()
for word in words:
    counts[word] = counts.get(word,0) + 1
print (counts)
bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count
print ("\n bigword and bigcount")
print (bigword, bigcount)
```

Logic 2:

```
handle = open("Egypt.txt")
text = handle.read()
words = text.split()
counts = {}
for word in words:
    counts[word] = counts.get(word,0) + 1
print (counts)
bigcount = 0
for word,count in counts.items():
    if count > bigcount:
        bigword = word
        bigcount = count
print ("\n bigword and bigcount")
print (bigword, bigcount)
```

OUTPUT:

```
{'hi': 2, 'hm': 2, 'gm': 4, 'gmg': 1, 'ji': 1, 'jij': 1, 'jjj': 1, 'AAA': 1}
 bigword and bigcount
 gm 4
```

9. Using a numpy module create an array and check the following:
a) type of an array b) axis of an array c) shape of an array d) type of elements in an array.

```
import numpy as np
arr=np.array([[1,2,3],[4,2,5]])
```

```
print("Array is of type:",type(arr))
print("no.of dimensions:",arr.ndim)
print("Shape of array:",arr.shape)
print("Size of array:",arr.size)
print("Array stores elements of type:",arr.dtype)
```

OUTPUT:

```
Array is of type: <class 'numpy.ndarray'>
no.of dimensions: 2
Shape of array: (2, 3)
Size of array: 6
Array stores elements of type: int64
```

10. Using a numpy module create array and check the following:
a) List with type float b) 3*4 array with all zeros c) From tuple d) Random values

a) List with type float

```
import numpy as np
npArray = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9] , dtype=float)
print('Contents of the Array : ', npArray)
print('Type of the Array : ', npArray.dtype)
```

OUTPUT:

```
Contents of the Array :  [1. 2. 3. 4. 5. 6. 7. 8. 9.]
Type of the Array :  float64
```

b) 3*4 array with all zeros

```
import numpy as np
a = np.zeros([3, 4], dtype=int)
print("\nMatrix a : \n", a)
```

OUTPUT:

```
Matrix a :
 [[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]
```

c) From tuple

```
import numpy as np
npArray = np.array( (11,22,33,44,55,66,77,88 ) )
print(npArray)
```

OUTPUT:

```
[11 22 33 44 55 66 77 88]
```

d)Random values

```
import numpy as np
# if the shape is not mentioned the output will just be a random integer in the given range
rand_int1 = np.random.randint(5,10)
```

```
print("First array", rand_int1)
rand_int2 = np.random.randint(10,90,(4,5)) # random numpy array of shape (4,5)
print("Second array ", rand_int2)
```

OUTPUT:

```
First array : 9
Second array : [[52 87 31 28 70]
 [82 75 41 36 46]
 [41 77 73 21 11]
 [45 43 38 46 71]]
```

11. Using a numpy module create array and check the following:
a)Reshape 3X4 array to 2X2X3 array
b)Sequence of integers from 0 to 30 with steps of 5
c)Flatten array
d)Constant value array of complex type.

a)Reshape 3X4 array to 2X2X3 array

```
import numpy as np
arr = np.array([[1, 2, 3, 4],
                [5, 2, 4, 2],
                [1, 2, 0, 1]])
newarr = arr.reshape(2, 2, 3)
print ("\nOriginal array:\n", arr)
print ("Reshaped array:\n", newarr)
```

OUTPUT:

```
Original array:
 [[1 2 3 4]
 [5 2 4 2]
 [1 2 0 1]]
Reshaped array:
 [[[1 2 3]
  [4 5 2]]
 [[4 2 1]
  [2 0 1]]]
```

b) Sequence of integers from 0 to 30 with steps of 5

```
import numpy as np
f = np.arange(0, 30, 5)
print ("\nA sequential array with steps of 5:\n", f)
```

OUTPUT:

```
A sequential array with steps of 5:
[ 0  5 10 15 20 25]
```

c) Flatten array

```
import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6],[7, 8, 9]])
flarr = arr.flatten()
print ("\nOriginal array:\n", arr)
```

```
print ("Fattened array:\n", flarr)
```

OUTPUT:

```
Original array:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
Fattened array:
 [1 2 3 4 5 6 7 8 9]
```

d)Constant value array of complex type

```
import numpy as np
d = np.full((3, 3), 5, dtype = 'complex')
print ("\nAn array initialized with all 5s."
"Array type is complex:\n", d)
```

OUTPUT:

```
An array initialized with all 5s.Array type is complex:
 [[5.+0.j 5.+0.j 5.+0.j]
 [5.+0.j 5.+0.j 5.+0.j]
 [5.+0.j 5.+0.j 5.+0.j]]
```

12. Implement the following using numpy module

- **i) Creation of Arrays**
- **ii) Demonstrate indexing in numpy array**
- **iii) Demonstrate basic operations on single array**

i)Creation of Arrays

```
# Creating a rank 1 Array
arr = np.array([1, 2, 3])
print("Array with Rank 1: \n",arr)
# Creating a rank 2 Array
arr = np.array([[1, 2, 3],
                [4, 5, 6]])
print("Array with Rank 2: \n", arr)
 # Creating an array from tuple
arr = np.array((1, 3, 2))
print("\nArray created using "
      "passed tuple:\n", arr)
```

OUTPUT:

```
Array with Rank 1:
 [1 2 3]
Array with Rank 2:
 [[1 2 3]
 [4 5 6]]
Array created using passed tuple:
 [1 3 2]
```

ii) Demonstrate indexing in numpy array

```
import numpy as np
# Initial Array
arr = np.array([[-1, 2, 0, 4],
                [4, -0.5, 6, 0],
                [2.6, 0, 7, 8],
                [3, -7, 4, 2.0]])
print("Initial Array: ")
print(arr)
# Printing a range of Array
# with the use of slicing method
sliced_arr = arr[:2, ::2]
print ("Array with first 2 rows and"
    " alternate columns(0 and 2):\n", sliced_arr)
# Printing elements at
# specific Indices
Index_arr = arr[[1, 1, 0, 3],
                [3, 2, 1, 0]]
print ("\nElements at indices (1, 3), "
    "(1, 2), (0, 1), (3, 0):\n", Index_arr)
```

OUTPUT:

```
Initial Array:
[[-1.   2.   0.   4. ]
 [ 4.  -0.5  6.   0. ]
 [ 2.6  0.   7.   8. ]
 [ 3.  -7.   4.   2. ]]
Array with first 2 rows and alternate columns(0 and 2):
 [[-1.  0.]
 [ 4.  6.]]
Elements at indices (1, 3), (1, 2), (0, 1), (3, 0):
 [0. 6. 2. 3.]
```

iii) demonstrate basic operations on single array

```
import numpy as np
        # Defining Array 1
a = np.array([[1, 2],
                                    [3, 4]])
        # Defining Array 2
b = np.array([[4, 3],
                                  [2, 1]])
            # Adding 1 to every element
print ("Adding 1 to every element:", a + 1)
        # Subtracting 2 from each element
print ("\nSubtracting 2 from each element:", b - 2)
        # sum of array elements
# Performing Unary operations
print ("\nSum of all array "
        "elements: ", a.sum())
        # Adding two arrays
# Performing Binary operations
print ("\nArray sum:\n", a + b)
```

OUTPUT:

```
Adding 1 to every element: [[2 3]
                            [4 5]]
Subtracting 2 from each element: [[ 2  1]
                                  [ 0 -1]]
Sum of all array elements:  10
Array sum:
                [[5 5]
                [5 5]]
```

13.Using numpy module implement the following

i) Replace items that satisfy a condition without affecting the original array

ii) Get the positions where elements of two arrays match

iii) Compute the row wise counts of all possible values in an array

i)Replace items that satisfy a condition without affecting the original array

```
import numpy as np
arr = np.arange(10)
print(arr)
out = np.where(arr%2==1,-1,arr)
print(out)
```

OUTPUT:

```
[0 1 2 3 4 5 6 7 8 9]
[ 0 -1  2 -1  4 -1  6 -1  8 -1]
```

ii)Get the positions where elements of two arrays match

```
import numpy as np
a = np.array([1,2,3,2,3,4,3,4,5,6])
b = np.array([7,2,10,2,7,4,9,4,9,8])
position=np.where(a == b)
print(position)
```

OUTPUT:

```
(array([1, 3, 5, 7]),)
```

iii) Compute the row wise counts of all possible values in an array

```
import numpy as np
np.random.seed(100)
arr = np.random.randint(1,11,size=(6, 10))
print(arr)

def counts_of_all_values_rowwise(arr2d):
  num_counts_array = [np.unique(row, return_counts=True) for row in arr2d]
  return([[int(b[a==i]) if i in a else 0 for i in np.unique(arr2d)] for a, b in num_counts_array])
counts_of_all_values_rowwise(arr)
```

OUTPUT:

```
[[ 9 9 4 8 8 1 5 3 6 3]
[ 3 3 2 1 9 5 1 10 7 3]
[ 5 2 6 4 5 5 4 8 2 2]
```

```
 [ 8  8  1  3 10 10  4  3  6  9]
 [ 2  1  8  7  3  1  9  3  6  2]
 [ 9  2  6  5  3  9  4  6  1 10]]

[[1, 0, 2, 1, 1, 1, 0, 2, 2, 0],
 [2, 1, 3, 0, 1, 0, 1, 0, 1, 1],
 [0, 3, 0, 2, 3, 1, 0, 1, 0, 0],
 [1, 0, 2, 1, 0, 1, 0, 2, 1, 2],
 [2, 2, 2, 0, 0, 1, 1, 1, 1, 0],
 [1, 1, 1, 1, 1, 2, 0, 0, 2, 1]]
```

14.Write a python code to read a csv file using pandas module and print the first and last five lines of a file.

```
import pandas as pd
diamonds=pd.read_csv('first.csv')
print("First 5 rows:")
print(diamonds.head())
print(" Last 5 lines:")
print(diamonds.tail())
```

OUTPUT:

First 5 rows:

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

last 5 rows:

	carat	cut	color	clarity	depth	table	price	x	y	z
53935	0.72	Ideal	D	SI1	60.8	57.0	2757	5.75	5.76	3.50
53936	0.72	Good	D	SI1	63.1	55.0	2757	5.69	5.75	3.61
53937	0.70	Very Good	D	SI1	62.8	60.0	2757	5.66	5.68	3.56
53938	0.86	Premium	H	SI2	61.0	58.0	2757	6.15	6.12	3.74
53939	0.75	Ideal	D	SI2	62.2	55.0	2757	5.83	5.87	3.64

15. Write a Pandas program

a) To create and display a DataFrame from a specified dictionary data which has the index labels.

b) To select the specified columns and rows from a given DataFrame.

c) To rename columns of a given DataFrame.

d) To drop a list of rows from a specified DataFrame.

a)

```
import pandas as pd
import numpy as np
exam_data  = {'name': ['a','b','c', 'd', 'e', 'f','g','h','i','j'],
        'score': [12.5, 9, 16.5, np.nan, 9,20,14.5,np.nan,8,19],
        'attempts': [1, 3, np.nan, 3, 2, 3, 1, np.nan, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
```

```
df = pd.DataFrame(exam_data , index=labels)
print(df)
```

output:

	name	score	attempts	qualify
a	a	12.5	1.0	yes
b	b	9.0	3.0	no
c	c	16.5	NaN	yes
d	d	NaN	3.0	no
e	e	9.0	2.0	no
f	f	20.0	3.0	yes
g	g	14.5	1.0	yes
h	h	NaN	NaN	no
i	i	8.0	2.0	no
j	j	19.0	1.0	yes

b)

```
import pandas as pd
import numpy as np
exam_data  = {'name': ['a','b','c', 'd', 'e', 'f','g','h','i','j'],
        'score': [12.5, 9, 16.5, np.nan, 9,20,14.5,np.nan,8,19],
        'attempts': [1, 3, np.nan, 3, 2, 3, 1, np.nan, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
print(df)
print(df.iloc[[1, 3, 5, 6], [1, 3]])
```

output:

```
   name  score  attempts qualify
a    a   12.5       1.0     yes
b    b    9.0       3.0      no
c    c   16.5       NaN     yes
d    d    NaN       3.0      no
e    e    9.0       2.0      no
f    f   20.0       3.0     yes
g    g   14.5       1.0     yes
h    h    NaN       NaN      no
i    i    8.0       2.0      no
j    j   19.0       1.0     yes
   score qualify
b    9.0      no
d    NaN      no
f   20.0     yes
g   14.5     yes
```

---

c)

```
import pandas as pd
import numpy as np
exam_data  = {'name': ['a','b','c', 'd', 'e', 'f','g','h','i','j'],
      'score': [12.5, 9, 16.5, np.nan, 9,20,14.5,np.nan,8,19],
      'attempts': [1, 3, np.nan, 3, 2, 3, 1, np.nan, 2, 1],
      'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
print("before renaming\n",df)

print("\nafter renaming\n")
df=df.rename(columns={"name":'hi',"score":"gm","attempts":"how","qualify":"are"})
print(df)
```

output:

```
before renaming
   name  score  attempts qualify
a    a   12.5       1.0     yes
b    b    9.0       3.0      no
c    c   16.5       NaN     yes
d    d    NaN       3.0      no
e    e    9.0       2.0      no
f    f   20.0       3.0     yes
g    g   14.5       1.0     yes
h    h    NaN       NaN      no
i    i    8.0       2.0      no
j    j   19.0       1.0     yes

after renaming

  hi    gm  how  are
a  a  12.5  1.0  yes
b  b   9.0  3.0   no
c  c  16.5  NaN  yes
d  d   NaN  3.0   no
e  e   9.0  2.0   no
f  f  20.0  3.0  yes
g  g  14.5  1.0  yes
h  h   NaN  NaN   no
i  i   8.0  2.0   no
j  j  19.0  1.0  yes
```

d)

```
import pandas as pd
import numpy as np
exam_data  = {'name': ['a','b','c', 'd', 'e', 'f','g','h','i','j'],
        'score': [12.5, 9, 16.5, np.nan, 9,20,14.5,np.nan,8,19],
        'attempts': [1, 3, np.nan, 3, 2, 3, 1, np.nan, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
print(df)
print("\nafter droping list of rows\n")
df=df.drop(['a','d','c','b'])
print(df)
```

output:

```
  name  score  attempts qualify
a    a   12.5       1.0     yes
b    b    9.0       3.0      no
c    c   16.5       NaN     yes
d    d    NaN       3.0      no
e    e    9.0       2.0      no
f    f   20.0       3.0     yes
g    g   14.5       1.0     yes
h    h    NaN       NaN      no
i    i    8.0       2.0      no
j    j   19.0       1.0     yes

after droping list of rows

  name  score  attempts qualify
e    e    9.0       2.0      no
f    f   20.0       3.0     yes
g    g   14.5       1.0     yes
h    h    NaN       NaN      no
i    i    8.0       2.0      no
j    j   19.0       1.0     yes
```

16. Write a Pandas program
a) To reset index in a given DataFrame.
b) To detect missing values of a given Data Frame. Display True or False.
c) To replace NaNs with median or mean of the specified columns in a given Data Frame.

a)

```
import pandas as pd
import numpy as np
exam_data  = {'name': ['a','b','c', 'd', 'e', 'f','g','h','i','j'],
      'score': [12.5, 9, 16.5, np.nan, 9,20,14.5,np.nan,8,19],
      'attempts': [1, 3, np.nan, 3, 2, 3, 1, np.nan, 2, 1],
      'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

df = pd.DataFrame(exam_data)
print(df)
print("\nafter droping list of rows\n")
df=df.drop([0,4,6,2])
print(df)
df=df.reset_index()
print(df)
```

output:

```
  name  score  attempts qualify
0    a   12.5       1.0     yes
1    b    9.0       3.0      no
2    c   16.5       NaN     yes
3    d    NaN       3.0      no
4    e    9.0       2.0      no
5    f   20.0       3.0     yes
6    g   14.5       1.0     yes
7    h    NaN       NaN      no
8    i    8.0       2.0      no
9    j   19.0       1.0     yes

after droping list of rows

  name  score  attempts qualify
1    b    9.0       3.0      no
3    d    NaN       3.0      no
5    f   20.0       3.0     yes
7    h    NaN       NaN      no
8    i    8.0       2.0      no
9    j   19.0       1.0     yes
after reindexing
   index name  score  attempts qualify
0      1    b    9.0       3.0      no
1      3    d    NaN       3.0      no
2      5    f   20.0       3.0     yes
3      7    h    NaN       NaN      no
4      8    i    8.0       2.0      no
5      9    j   19.0       1.0     yes
```

b)

```
 import pandas as pd
import numpy as np
exam_data  = {'name': ['a','b','c', 'd', 'e', 'f','g','h','i','j'],
      'score': [12.5, 9, 16.5, np.nan, 9,20,14.5,np.nan,8,19],
      'attempts': [1, 3, np.nan, 3, 2, 3, 1, np.nan, 2, 1],
      'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
print(df)
print("\nafter missing values are detected\n")
df=df.isna()
print(df)
```

output:

```
  name  score  attempts qualify
a    a   12.5       1.0     yes
b    b    9.0       3.0      no
c    c   16.5       NaN     yes
d    d    NaN       3.0      no
e    e    9.0       2.0      no
f    f   20.0       3.0     yes
g    g   14.5       1.0     yes
h    h    NaN       NaN      no
i    i    8.0       2.0      no
j    j   19.0       1.0     yes

after missing values are detected

    name  score  attempts  qualify
a  False  False     False    False
b  False  False     False    False
c  False  False      True    False
d  False   True     False    False
e  False  False     False    False
f  False  False     False    False
g  False  False     False    False
h  False   True      True    False
i  False  False     False    False
j  False  False     False    False
```

c)

```
import pandas as pd
import numpy as np
exam_data  = {'name': ['a','b','c', 'd', 'e', 'f','g','h','i','j'],
        'score': [12.5, 9, 16.5, np.nan, 9,20,14.5,np.nan,8,19],
        'attempts': [1, 3, np.nan, 3, 2, 3, 1, np.nan, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
df = pd.DataFrame(exam_data , index=labels)
print(df)
print("Using median in score to replace NaN:")
df['score'].fillna(df['score'].median(), inplace=True)
print(df)
print("Using mean to replace NaN:")
df['attempts'].fillna(int(df['attempts'].mean()), inplace=True)
print(df)
```

output:

```
  name  score  attempts qualify
a    a   12.5       1.0     yes
b    b    9.0       3.0      no
c    c   16.5       NaN     yes
d    d    NaN       3.0      no
e    e    9.0       2.0      no
f    f   20.0       3.0     yes
g    g   14.5       1.0     yes
h    h    NaN       NaN      no
i    i    8.0       2.0      no
j    j   19.0       1.0     yes
Using median in score to replace NaN:
  name  score  attempts qualify
a    a   12.5       1.0     yes
b    b    9.0       3.0      no
c    c   16.5       NaN     yes
d    d   13.5       3.0      no
e    e    9.0       2.0      no
f    f   20.0       3.0     yes
g    g   14.5       1.0     yes
h    h   13.5       NaN      no
i    i    8.0       2.0      no
j    j   19.0       1.0     yes
Using mean to replace NaN:
  name  score  attempts qualify
a    a   12.5       1.0     yes
b    b    9.0       3.0      no
c    c   16.5       2.0     yes
d    d   13.5       3.0      no
e    e    9.0       2.0      no
f    f   20.0       3.0     yes
g    g   14.5       1.0     yes
h    h   13.5       2.0      no
i    i    8.0       2.0      no
j    j   19.0       1.0     yes
```

Course Outcomes:

Sl. No.	Description
1	Develop programs for the given problem statement in the real world.
2	Implement the programs on object oriented concepts.
3	Demonstrate the usage of NumPy module for numerical data analysis.
4	Apply data manipulation tools available in pandas for data cleaning and analysis.

Pattern for practical exam conduction:
In Semester End Practical Examination, students are allowed to pick one program from the lot of 3 cycles.