1) create 2 d array (4\*3) with following values:

[[10,20,30,40],[50,60,70,80],[90,100,110,120]]

now using array slicing concept display following values

a) display 50 60 70 80

b) display 100 and 110

c) display 10,20,30,40 and 90,100,110,120

d) display 50 60 90 and 100

*import* numpy *as* np  
arr = np.arange(10,121,10).reshape(3,4)  
*print*(arr)  
  
*print*('a) ',arr[1,:])  
*print*('b) ',arr[2, 1:3])  
*print*('c) ',arr[::2])  
*print*('d) ',arr[1:3, :2])

2)Create 3 empty lists

Prnnos,names and qualifications

Accept at least 10 Prnnos , names and qualifications and store them in the above lists.

Now create a dictionary to store

prn as key and Prnnos list as value

name as key andnames list as value

qualification as key and qualifications list as value

using the above dictionary create a dataframe and display it without the index.

*import* pandas *as* pd  
prnnos = *list*(*int*(i) *for* i *in input*("Enter 10 numbers comma separated: ").split(','))  
names = *list*(*input*("Enter names comma separated: ").split(','))  
qualifications = *list*(*input*("Enter qualification comma separated: ").split(','))  
  
records = {  
 "prn" : prnnos,  
 "name": names,  
 "qualification":qualifications  
}  
  
df = pd.DataFrame(records)  
*print*(df.to\_string(index=*False*))

3) accept from user 5 values and create a Series and perform the following operations:

a) display their sum

accept one more value for arithmetic operation on the Series

b) add the value to the above Series

c) subtract the value to the above Series

d) multiply the value to the above Series

[ Note:- for every display index of the Series should not be displayed ]

*import* pandas *as* pd  
  
s = pd.Series([*int*(i) *for* i *in input*("Enter 5 numbers: ").split(',')])  
n = *int*(*input*("Enter one more number: "))

*print*(s.to\_string(index=*False*))

*print*("a) sum: ", s.sum())

*print*("b) value added to elements of series :\n",(s+n).to\_string(index=*False*))

*print*("c) value subtracted from elements of series :\n",(s-n).to\_string(index=*False*))

*print*("d) value multiplied with elements of series :\n",(s\*n).to\_string(index=*False*))

4) create a class "Student" with rollno,name and address as instance members.

create 2 instances of Student class by passing rollno,name and address.

now print both the instances and make sure when u print the instances it should display their details (rollno,name and address)

5) Given a list of words, create a new list using list comprehension where each element is the length of the corresponding word.

Input:

words = ["cricket", "Movie", "celebrations", "fun", "powerful"]

Expected Output:

[7, 5, 12, 3, 8]

6) Take a string and use list comprehension to remove all vowels, returning the result as a list of remaining characters.

Input:

text = "List comprehensions make Python concise"

Expected Output:

['L', 's', 't', ' ', 'c', 'm', 'p', 'r', 'h', 'n', 's', 'n', 's', ' ', 'm', 'k', ' ', 'P', 'y', 't', 'h', 'n', ' ', 'c', 'n', 'c', 's']

7) define “outer” as outer function and inside it “inner” as inner function.

Outer function will accept some number from the caller and inner function will return

1 if the number passed is positive

-1 if the number passed is negative

0 if the number passed is 0

Invoke outer function by passing some number and make sure when u call inner function , it will return 1 or -1 or 0.

8) print the following pattern:

11  
 12 12  
 13 13 13  
 14 14 14 14  
15 15 15 15 15

*for* i *in range*(5):  
 *for* j *in range*(5,i,-1):  
 *print*("\_", end=' ')  
  
 *for* j *in range*(i+1):  
 *print*(11+i, end=' ')  
 *print*()

9) Write a program that contains a function that has one parameter, n, representing

an integer greater than 0. The function should return n! (n factorial). Then write a main

function that calls this function with the values 1 through 20, one at a time, printing the

returned results. This is what your output should look like:

1 1

2 2

3 6

4 24

5 120

6 720

7 5040

8 40320

9 362880

10 3628800

10) Write a Python program that:

Defines a user-defined exception class InvalidMarksError.

Accepts a student’s marks (as input).

If the marks are not in the range 0 to 100, the program should raise the InvalidMarksError exception with a proper error message.

The caller program should handle the exception using try-except.

Use a finally block to print "Program execution completed" regardless of whether an exception occurred or not.

Example

Case 1 (Valid Input):

Enter marks: 85

Valid Marks: 85

Program execution completed

Case 2 (Invalid Input):

Enter marks: 150

Error: Marks should be between 0 and 100

Program execution completed