Rajalakshmi Engineering College

Name: BEVIN ABRAHAM V B
Email: 240701076@rajalakshmi.edu.in
Roll no:
Phone: 6383017582
Branch: REC
Department: I CSE FA
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 6_MCQ

Attempt : 1
Total Mark : 20
Marks Obtained : 18

Section 1 : MCQ

1. What is the output of the following code?

```
try:
    x = 1 / 0
except ZeroDivisionError:
    print("Caught division by zero error")
finally:
    print("Executed")
```

Answer

Caught division by zero errorExecuted

Status : Correct ***Marks : 1/1***

2. What is the difference between r+ and w+ modes?

Answer

in r+ the pointer is initially placed at the beginning of the file and the pointer is at the end for w+

Status : Correct ***Marks : 1/1***

3. How do you create a user-defined exception in Python?

Answer

By creating a new class that inherits from the Exception class

Status : Correct ***Marks : 1/1***

4. What is the correct way to raise an exception in Python?

Answer

raise Exception()

Status : Correct ***Marks : 1/1***

5. What is the output of the following code?

```
class MyError(Exception):
    pass

try:
    raise MyError("Something went wrong")
except MyError as e:
    print(e)
```

Answer

Something went wrong

Status : Correct ***Marks : 1/1***

6. What happens if no arguments are passed to the seek function?

Answer

error

Status : Wrong ***Marks : 0/1***

7. Which of the following is true about the finally block in Python?

Answer

The finally block is always executed, regardless of whether an exception occurs or not

Status : Correct ***Marks : 1/1***

8. What is the default value of reference_point in the following code?

file_object.seek(offset [,reference_point])

Answer

0

Status : Correct ***Marks : 1/1***

9. What is the output of the following code?

```
try:
    x = "hello" + 5
except TypeError:
    print("Type Error occurred")
finally:
    print("This will always execute")
```

Answer

Type Error occurredThis will always execute

Status : Correct ***Marks : 1/1***

10. Fill the code to in order to read file from the current position.

Assuming exp.txt file has following 3 lines, consider current file position is beginning of 2nd line

Meri,25

John,21

Raj,20

Ouptput:

['John,21\n','Raj,20\n']

```
f = open("exp.txt", "w+")
_____(1)
print _____(2)
```

Answer

1) f.seek(0, 1)2) f.readlines()

Status : Correct ***Marks : 1/1***

11. What happens if an exception is not caught in the except clause?

Answer

The program will display a traceback error and stop execution

Status : Correct ***Marks : 1/1***

12. What will be the output of the following Python code?

```
f = None
for i in range (5):
   with open("data.txt", "w") as f:
      if i > 2:
         break
print(f.closed)
```

Answer

True

Status : Correct ***Marks : 1/1***

13. What will be the output of the following Python code?

```
# Predefined lines to simulate the file content
lines = [
    "This is 1st line",
    "This is 2nd line",
    "This is 3rd line",
    "This is 4th line",
    "This is 5th line"
]

print("Name of the file: foo.txt")

# Print the first 5 lines from the predefined list
for index in range(5):
    line = lines[index]
    print("Line No %d - %s" % (index + 1, line.strip()))
```

Answer

Displays Output

Status : Correct ***Marks : 1/1***

14. Match the following:

a) f.seek(5,1) i) Move file pointer five characters behind from the current position

b) f.seek(-5,1) ii) Move file pointer to the end of a file

c) f.seek(0,2) iii) Move file pointer five characters ahead from the current position

d) f.seek(0) iv) Move file pointer to the beginning of a file

Answer

a-iii, b-i, c-ii, d-iv

Status : Correct ***Marks : 1/1***

15. Which clause is used to clean up resources, such as closing files in Python?

Answer

finally

Status : Correct ***Marks : 1/1***

16. Fill in the code in order to get the following output:

Output:

Name of the file: ex.txt

```
fo = open(_____(1), "wb")
print("Name of the file: ",_____)(2)
```

Answer

1) "ex.txt"2) fo.name()

Status : Wrong ***Marks : 0/1***

17. Which of the following is true about

fp.seek(10,1)

Answer

Move file pointer ten characters ahead from the current position

Status : Correct ***Marks : 1/1***

18. Fill in the blanks in the following code of writing data in binary files.

```
import _____ (1)
rec=[]
while True:
```

```
    rn=int(input("Enter"))
    nm=input("Enter")
    temp=[rn, nm]
    rec.append(temp)
    ch=input("Enter choice (y/N)")
    if ch.upper=="N":
        break
f.open("stud.dat","_____________")(2)
_____________.dump(rec,f)(3)
_____________.close()(4)
```

Answer

(pickle,wb,pickle,f)

Status : Correct ***Marks : 1/1***

19. How do you rename a file?

Answer

os.rename(existing_name, new_name)

Status : Correct ***Marks : 1/1***

20. What is the purpose of the except clause in Python?

Answer

To handle exceptions during code execution

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: BEVIN ABRAHAM V B
Email: 240701076@rajalakshmi.edu.in
Roll no:
Phone: 6383017582
Branch: REC
Department: I CSE FA
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 6_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

Alex is creating an account and needs to set up a password. The program prompts Alex to enter their name, mobile number, chosen username, and desired password. Password validation criteria include:

Length between 10 and 20 characters.At least one digit.At least one special character from !@#$%^&* set. Display "Valid Password" if criteria are met; otherwise, raise an exception with an appropriate error message.

Input Format

The first line of the input consists of the name as a string.

The second line of the input consists of the mobile number as a string.

The third line of the input consists of the username as a string.

The fourth line of the input consists of the password as a string.

Output Format

If the password is valid (meets all the criteria), it will print "Valid Password"

If the password is weak (fails any one or more criteria), it will print an error message accordingly.

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: John
9874563210
john
john1#nhoj
Output: Valid Password

Answer

```
name=input()
mobile=input()
username=input()
password=input()

s_c="!@#$%^&*"
if len(password)<10 or len(password)>20:
   print("Should be a minimum of 10 characters and a maximum of 20 characters")
elif not any(char.isdigit() for  char in password):
   print("Should contain at least one digit")
elif not any(char in s_c for char in password):
   print("It should contain at leasstt one special character")
else:
   print("Valid Password")
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Alice is developing a program called "Name Sorter" that helps users organize and sort names alphabetically.

The program takes names as input from the user, saves them in a file, and then displays the names in sorted order.

File Name: sorted_names.txt.

Input Format

The input consists of multiple lines, each containing a name represented as a string.

To end the input and proceed with sorting, the user can enter 'q'.

Output Format

The output displays the names in alphabetical order, each name on a new line.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: Alice Smith
John Doe
Emma Johnson
q
Output: Alice Smith
Emma Johnson
John Doe

Answer

```
names=[]
while True:
    name=input()
    if name=='q':
        break
    names.append(name)
```

```
names.sort()
file=open("Sorted_name.txt","w")
for name in names:
   file.write(name + "\n")
file.close()

for name in names:
   print(name)
```

Status : Correct ***Marks : 10/10***

3. Problem Statement

Write a program to read the Register Number and Mobile Number of a student. Create user-defined exception and handle the following:

If the Register Number does not contain exactly 9 characters in the specified format(2 numbers followed by 3 characters followed by 4 numbers) or if the Mobile Number does not contain exactly 10 characters, throw an IllegalArgumentException. If the Mobile Number contains any character other than a digit, raise a NumberFormatException.If the Register Number contains any character other than digits and alphabets, throw a NoSuchElementException.If they are valid, print the message ‘valid’ or else print an Invalid message.

Input Format

The first line of the input consists of a string representing the Register number.

The second line of the input consists of a string representing the Mobile number.

Output Format

The output should display any one of the following messages:

If both numbers are valid, print "Valid".

If an exception is raised, print "Invalid with exception message: ", followed by the specific exception message.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 19ABC1001
9949596920

Output: Valid

Answer

```
register_number=input()
phone_number=input()
try:
    if len(register_number)!=9:
        raise ValueError("Invalid with exception message: Register Number should have exactly 9 characters.")
    if not(register_number[2:5].isalpha() and register_number[5:].isdigit()):
        raise ValueError("Invalid with exception message: Register Number should have the format: 2numbers, 3 characters, and 4 numbers.")
    if len(phone_number)!=10:
        raise ValueError("Invalid with exception message: Mobile Number should have exactly 10 characters.")
    if not(phone_number.isdigit()):
        raise ValueError("Invalid with exception message: Mobile Number should only contain digits.")
    if not register_number.isalnum():
        raise ValueError("Invalid with exception message: Registor Number should only contain alphanumeric.")
    print("Valid")
except ValueError as e:
    print(e)
```

Status : Correct ***Marks : 10/10***

4. Problem Statement

A shopkeeper is recording the daily sales of an item for N days, where the price of the item remains the same for all days. Write a program to

calculate the total sales for each day and save them in a file named sales.txt that can store the data for a maximum of 30 days. Then, read the file and display the total earnings for each day.

Note: Total Earnings for each day = Number of Items sold in that day × Price of the item.

Input Format

The first line of input consists of an integer N, representing the number of days.

The second line of input consists of N space-separated integers representing the number of items sold each day.

The third line of input consists of an integer M, representing the price of the item that is common for all N days.

Output Format

If the number of days entered exceeds 30 (N > 30), the output prints "Exceeding limit!" and terminates.

Otherwise, the code reads the contents of the file and displays the total earnings for each day on separate lines.

Contents of the file: The total earnings for N days, with each day's earnings appearing on a separate line.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
5 10 5 0
20
Output: 100
200
100

0

Answer

```
n=int(input())
if n>30:
    print("Exceeding limit!")
    exit()
items_sold=list(map(int,input().split()))
m=int(input())
total_earnings=[]
for i in range(n):
    total_earnings.append(items_sold[i]*m)

file=open("sales.txt","w")
for earnings in total_earnings:
    file.write(str(earnings)+ "\n")
file.close()

file=open("sales.txt","r")
lines=file.readlines()
file.close()
for line in lines:
    print(line.strip())
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: BEVIN ABRAHAM V B
Email: 240701076@rajalakshmi.edu.in
Roll no:
Phone: 6383017582
Branch: REC
Department: I CSE FA
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 6_PAH

Attempt : 1
Total Mark : 30
Marks Obtained : 22.5

Section 1 : Coding

1. Problem Statement

Reeta is playing with numbers. Reeta wants to have a file containing a list of numbers, and she needs to find the average of those numbers. Write a program to read the numbers from the file, calculate the average, and display it.

File Name: user_input.txt

Input Format

The input file will contain a single line of space-separated numbers (as a string).

These numbers may be integers or decimals.

Output Format

If all inputs are valid numbers, the output should print: "Average of the numbers is: X.XX" (where X.XX is the computed average rounded to two decimal places)

If the input contains invalid data, print: "Invalid data in the input."

Refer to the sample output for format specifications.

Sample Test Case

Input: 1 2 3 4 5

Output: Average of the numbers is: 3.00

Answer

```
def avg_file(filename):
    try:
        with open(filename,'r')as f:
            l=f.readline().strip()
            n=l.split()
            total,count=0,0
            for num in n:
                try:
                    total+=float(num)
                    count+=1
                except VlueError:
                    print("Invalid data in the input.")
                    return
            if count>0:
                avg=total/count
                print(f"Average of the numbers is: {avg:.2f}")
            else:
                print("Invalid data in the input.")
    except FileNotFoundError:
        print("File not found.")
a=input()
with open("user_input.txt","w")as w:
    w.write(a)
avg_file('user_input.txt')
```

Status : Partially correct ***Marks : 5/10***

2. Problem Statement

John is a data analyst who often works with text files. He needs a program that can analyze the contents of a text file and count the number of times a specific character appears in the file.

John wants a simple program that allows him to specify a file and a character to count within that file.

Input Format

The first line of input consists of the file's name to be analyzed.

The second line of the input consists of the string they want to write within the file.

The third line of the input consists of a character to count within the file.

Output Format

If the character is found, the output displays "The character 'X' appears {Y} times in the file." where X is the character and Y i the count,

If the character does not appear in the file, the output displays "Character not found."

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: test.txt
This is a test file to check the character count.
e
Output: The character 'e' appears 5 times in the file.

Answer

```
name=input()
```

```
content=input()
c=input()
with open(name,"w")as f:
    f.write(content)
with open(name,"r")as f:
    r=f.read()
    r=r.lower()
    count=r.count(c)
    if count!=0:
        print(f"The character '{c}' appears{count} times in the file.")
    else:
        print("Character not found in the file.")
```

Status : Correct ***Marks : 10/10***

3. Problem Statement

Peter manages a student database and needs a program to add students. For each student, Alex inputs their ID and name. The program checks for duplicate IDs and ensures the database isn't full.

If a duplicate or a full database is detected, an appropriate error message is displayed. Otherwise, the student is added, and a confirmation message is shown. The database has a maximum capacity of 30 students, and each student must have a unique ID.

Input Format

The first line contains an integer n, representing the number of students to be added to the school database.

The next n lines each contain two space-separated values, representing the student's ID (integer) and the student's name (string).

Output Format

The output will depend on the actions performed in the code.

If a student is added to the database, the output will display: "Student with ID [ID number] added to the database."

If there is an exception due to a duplicate student ID, the output will display: "Exception caught. Error: Student ID already exists."

If there is an exception due to the database being full, the output will display: "Exception caught. Error: Student database is full."

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 3
16 Sam
87 Sabari
43 Dani

Output: Student with ID 16 added to the database.
Student with ID 87 added to the database.
Student with ID 43 added to the database.

Answer

```
MAX_CAPACITY=30
database={}
count=0

n=int(input())
for i in range(n):
    stu_input=input().strip().split()
    stu_id=int(stu_input[0])
    stu_name=stu_input[1]
    if count>=MAX_CAPACITY:
        print("Exception caught. Error:Student database is full.")
        continue
    if stu_id in database:
        print("Exception caught. Error:Student ID already exists.")
        break
    else:
        database[stu_id]=stu_name
        count+=1
```

```
    print(f"Student with ID {stu_id} added to the database.")
```

Status : Partially correct ***Marks : 7.5/10***

Rajalakshmi Engineering College

Name: BEVIN ABRAHAM V B
Email: 240701076@rajalakshmi.edu.in
Roll no:
Phone: 6383017582
Branch: REC
Department: I CSE FA
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 6_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Write a program that calculates the average of a list of integers. The program prompts the user to enter the length of the list (n) and each element of the list. It performs error handling to ensure that the length of the list is a non-negative integer and that each input element is a numeric value.

Input Format

The first line of the input is an integer n, representing the length of the list as a positive integer.

The second line of the input consists of an element of the list as an integer, separated by a new line.

Output Format

If the length of the list is not a positive integer or zero, the output displays "Error: The length of the list must be a non-negative integer."

If a non-numeric value is entered for the length of the list, the output displays "Error: You must enter a numeric value."

If a non-numeric value is entered for a list element, the output displays "Error: You must enter a numeric value."

If the inputs are valid, the program calculates and prints the average of the provided list of integers with two decimal places: "The average is: [average]".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: -2
1
2
Output: Error: The length of the list must be a non-negative integer.

Answer

```
try:
   k=int(input())
   if k<=0:
      raise Exception
   else:
      l=[]
      for i in range(k):
         n=input().strip()
         if(not n.isdigit()):
            raise ValueError
         l.append(int(n))
      t=sum(l)
      m=t/k
      print(f"The average is:{m:.2f}")
except ValueError:
```

```
    print("Error: You must enter a numeric value.")
except Exception:
    print("Error: The length of the list must be a non-negative integer.")
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

A retail store requires a program to calculate the total cost of purchasing a product based on its price and quantity. The program performs validation to ensure valid inputs and handles specific error conditions using exceptions:

Price Validation: If the price is zero or less, raise a ValueError with the message: "Invalid Price".Quantity Validation: If the quantity is zero or less, raise a ValueError with the message: "Invalid Quantity".Cost Threshold: If the total cost exceeds 1000, raise RuntimeError with the message: "Excessive Cost".

Input Format

The first line of input consists of a double value, representing the price of a product.

The second line consists of an integer, representing the quantity of the product.

Output Format

If the calculation is successful, print the total cost rounded to one decimal place.

If the price is zero or less prints "Invalid Price".

If the quantity is zero or less prints "Invalid Quantity".

If the total cost exceeds 1000, prints "Excessive Cost".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 20.0
5
Output: 100.0

Answer

```
n=float(input())
m=int(input())
try:
    if(n<=0):
        print("Invalid Price")
    elif(m<=0):
        print("Invalid Quantity")
    elif(m*n>1000):
        print("Excessive Cost")
    else:
        print(m*n)
except valueerror:
    pass
except runtimeerror:
    pass
```

Status : Correct ***Marks : 10/10***

3. Problem Statement

Sophie enjoys playing with words and wants to count the number of words in a sentence. She inputs a sentence, saves it to a file, and then reads it from the file to count the words.

Write a program to determine the number of words in the input sentence.

File Name: sentence_file.txt

Input Format

The input consists of a single line of text containing words separated by spaces.

Output Format

The output displays the count of words in the sentence.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: Four Words In This Sentence
Output: 5

Answer

```
s=input()
with open("sentence_flie.txt","w")as f:
    f.write(s)
with open("sentence_flie.txt","r")as f:
    c=f.read()
    print(len(c.split()))
```

Status : Correct ***Marks : 10/10***

4. Problem Statement

Tara is a content manager who needs to perform case conversions for various pieces of text and save the results in a structured manner.

She requires a program to take a user's input string, save it in a file, and then retrieve and display the string in both upper-case and lower-case versions. Help her achieve this task efficiently.

File Name: text_file.txt

Input Format

The input consists of a single line containing a string provided by the user.

Output Format

The first line displays the original string read from the file in the format: "Original String: {original_string}".

The second line displays the upper-case version of the original string in the format: "Upper-Case String: {upper_case_string}".

The third line displays the lower-case version of the original string in the format: "Lower-Case String: {lower_case_string}".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: #SpecialSymBoLs1234

Output: Original String: #SpecialSymBoLs1234
Upper-Case String: #SPECIALSYMBOLS1234
Lower-Case String: #specialsymbols1234

Answer

```
s=input()
with open("text_file.txt","w")as f:
    f.write(s)
with open("text_file.txt","r")as f:
    c=f.read()
    print("Original String:",c)
    print("Upper-Case String:",c.upper())
    print("Lower-Case String:",c.lower())
```

Status : Correct ***Marks : 10/10***

5. Problem Statement

In a voting system, a person must be at least 18 years old to be eligible to vote. If a user enters an age below 18, the system should raise a user-defined exception indicating that they are not eligible to vote.

Input Format

The input contains a positive integer representing age.

Output Format

If the age is less than 18, the output displays "Not eligible to vote".

Otherwise, the output displays "Eligible to vote".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 18

Output: Eligible to vote

Answer

```
n=int(input())
try:
    if(n>=18):
        print("Eligible to vote")
    else:
        raise ValueError
except ValueError:
    print("Not eligible to vote")
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: BEVIN ABRAHAM V B
Email: 240701076@rajalakshmi.edu.in
Roll no:
Phone: 6383017582
Branch: REC
Department: I CSE FA
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 7_PAH

Attempt : 1
Total Mark : 50
Marks Obtained : 8

Section 1 : Coding

1. Problem Statement

Arjun is a data scientist working on an image processing task. He needs to normalize the pixel values of a grayscale image matrix to scale between 0 and 1. The input image data is provided as a matrix of integers.

Help him to implement the task using the numpy package.

Formula:

To normalize each pixel value in the image matrix:

normalized_pixel = (pixel - min_pixel) / (max_pixel - min_pixel)

where min_pixel and max_pixel are the minimum and maximum pixel values in the image matrix, respectively. If all pixel values are the same, the normalized image matrix should be filled with zeros.

Input Format

The first line of input consists of an integer value, rows, representing the number of rows in the image matrix.

The second line of input consists of an integer value, cols, representing the number of columns in the image matrix.

The next rows lines each consist of cols integer values separated by a space, representing the pixel values of the image matrix.

Output Format

The output prints: normalized_image

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 2
3
1 2 3
4 5 6
Output: [[0. 0.2 0.4]
 [0.6 0.8 1.]]

Answer

```
import numpy as np
a=int(input())
b=int(input())
m=[]
for _ in range(a):
    d=list(map(int,input().split()))
    m.append(d)
arr=np.array(m)
m_a=np.max(arr)
mi_a=np.min(arr)
if m_a==mi_a:
    o=np.zeros(a,b,dtype=float)
else:
    o=(arr-mi_a)/(m_a-mi_a)
```

print(o)

Status : Partially correct ***Marks : 8/10***

2. Problem Statement

A company conducted a customer satisfaction survey where each respondent provides their RespondentID and an optional textual Feedback. Sometimes, respondents submit their ID without any feedback or with empty feedback.

Your task is to process the survey responses using pandas to replace any missing or empty feedback with the phrase "No Response". Finally, print the cleaned survey responses exactly as shown in the sample output.

Input Format

The first line contains an integer n, the number of survey responses.

Each of the next n lines contains:

A RespondentID (a single alphanumeric string without spaces),

Followed optionally by a Feedback string, which may be empty or missing.

If no feedback is provided after the RespondentID, treat it as missing.

Output Format

Print the line:

Survey Responses with Missing Feedback Filled:

Then print the cleaned survey data as a table with two columns: RespondentID and Feedback.

The table should have the headers exactly as:

RespondentID Feedback

Print each respondent's data on a new line, aligned to match the output produced by pandas.DataFrame.to_string(index=False).

For any missing or empty feedback, print "No Response" in the Feedback column.

Maintain the spacing and alignment exactly as shown in the sample outputs.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
101 Great service
102
103 Loved it
104

Output: Survey Responses with Missing Feedback Filled:

```
RespondentID      Feedback
         101 Great service
         102   No Response
         103      Loved it
         104   No Response
```

Answer

```
import panda as pd
def process_survey_responses():
    n=int(input())
    data=[]
    for i in range(n):
        line=input().strip()
        parts=line.split(",1)
        res_id=parts[0]
        feedback=parts[1]if len(parts)>1 else ""
        data.append((res_id,feedback))
    df=pd.DataFrame(data,columns=['RespondentId','Feedback'])
```

```
    df['Feedback']=df['Feedback'].replace('','No Response'')
    print("Survey Responses with Missing Feedback Filled:")
    print(df.to_string(index=False))
process_survey_responses()
```

Status : Wrong ***Marks : 0/10***

3. Problem Statement

You're analyzing the daily returns of a set of financial assets over a period of time. Each day is represented as a row in a 2D array, where each column represents the return of a specific asset on that day.

Your task is to identify which days had all positive returns across every asset using numpy, and output a boolean array indicating these days.

Input Format

The first line of input consists of two integer values, rows and cols, separated by a space.

Each of the next rows lines consists of cols float values representing the returns of the assets for that day.

Output Format

The first line of output prints: "Days where all asset returns were positive:"

The second line of output prints: the boolean array positive_days, indicating True for days where all asset returns were positive and False otherwise.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3 4
0.01 0.02 0.03 0.04
0.05 0.06 0.07 0.08
-0.01 0.02 0.03 0.04

Output: Days where all asset returns were positive:
[True True False]

Answer

-

Status : Skipped ***Marks : 0/10***

4. Problem Statement

Arjun manages a busy customer service center and wants to analyze the distribution of customer wait times to improve service efficiency. He decides to group the wait times into intervals of 5 minutes each and count how many customers fall into each interval bucket.

Help him implement this bucketing and counting task using NumPy.

Bucketing Logic:

Divide the wait times into intervals (buckets) of size 5 minutes, e.g.:

[0–5), [5–10), [10–15), …

Use NumPy’s digitize function to determine which bucket each wait time falls into.

Count the number of wait times in each bucket and generate bucket labels.

Input Format

The first line contains an integer n, the number of customer wait times recorded.

The second line contains n space-separated floating-point numbers representing the wait times (in minutes).

Output Format

The first line of output is the text:

Wait Time Buckets and Counts:

Each subsequent line prints the bucket range and the number of wait times in that bucket, formatted as:

<bucket_range>: <count>

where <bucket_range> is the lower and upper bound of the bucket (inclusive lower bound, exclusive upper bound), for example:

0-5: 3

5-10: 2

10-15: 1

The output uses the default string formatting of Python's print() function (no extra spaces, no special formatting beyond the specified lines).

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 10
2.0 3.0 7.0 8.0 12.0 14.0 18.0 19.0 21.0 25.0
Output: Wait Time Buckets and Counts:
0-5: 2
5-10: 2
10-15: 2
15-20: 2
20-25: 1

Answer

-

Status : Skipped ***Marks : 0/10***

5. Problem Statement

A software development company wants to classify its employees based on their years of service at the company. They want to categorize

employees into three experience levels: Junior (less than 3 years), Mid (3 to 6 years, inclusive), and Senior (more than 6 years).

Experience Level Classification:

Junior: Years at Company < 3

Mid: 3 ≤ Years at Company < 6

Senior: Years at Company > 5

You need to create a Python program using the pandas library that reads employee data, processes it into a DataFrame, and adds a new column "Experience Level" to display the appropriate classification for each employee.

Input Format

First line: an integer n representing the number of employees.

Next n lines: each line has a string Name and a floating-point number Years at Company (space-separated).

Output Format

First line: "Employee Data with Experience Level:"

The employee data table printed with no index column, and with columns: Name, Years at Company, Experience Level.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
Alice 2
Bob 4
Charlie 7
Diana 3
Evan 6

Output: Employee Data with Experience Level:
Name Years at Company Experience Level

Alice	2.0	Junior
Bob	4.0	Mid
Charlie	7.0	Senior
Diana	3.0	Mid
Evan	6.0	Senior

Answer

```
# You are using Python
```

Status : Wrong ***Marks : 0/10***

Rajalakshmi Engineering College

Name: BEVIN ABRAHAM V B
Email: 240701076@rajalakshmi.edu.in
Roll no:
Phone: 6383017582
Branch: REC
Department: I CSE FA
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 7_COD

Attempt : 1
Total Mark : 50
Marks Obtained : 39

Section 1 : Coding

1. Problem Statement

A company tracks the monthly sales data of various products. You are given a table where each row represents a product and each column represents its monthly sales in sequential months.

Your task is to compute the cumulative monthly sales for each product using numpy, where the cumulative sales for a month is the total sales from month 1 up to that month.

Input Format

The first line of input consists of two integer values, products and months, separated by a space.

Each of the next products lines consists of months integer values representing the monthly sales data of a product.

Output Format

The first line of output prints: "Cumulative Monthly Sales:"

The second line of output prints: the 2D numpy array cumulative_array that contains the cumulative sales data for each product.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 2 4
10 20 30 40
5 15 25 35

Output: Cumulative Monthly Sales:
[[10 30 60 100]
 [5 20 45 80]]

Answer

```
import numpy as np
p,m=map(int,input().split())
sd=[list(map(int,input().split()))for i in range(p)]
sa=np.array(sd)
ca=np.cumsum(sa,axis=1)
print("Cumulative Monthly Sales:")
print(ca)
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Sita works as a sales analyst and needs to analyze monthly sales data for different cities. She receives lists of cities, months, and corresponding sales values and wants to create a pandas DataFrame using a MultiIndex of cities and months.

Help her to implement this task and calculate total sales for each city.

Input Format

The first line of input consists of an integer value, n, representing the number of records.

The second line of input consists of n space-separated city names.

The third line of input consists of n space-separated month names.

The fourth line of input consists of n space-separated float values representing sales for each city-month combination.

Output Format

The first line of output prints: "Monthly Sales Data with MultiIndex:"

The next lines print the DataFrame with MultiIndex (City, Month) and their corresponding sales values.

The following line prints: "\nTotal Sales Per City:"

The final lines print the total sales per city, computed by grouping the sales data on city names.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
NYC NYC LA LA
Jan Feb Jan Feb
100 200 300 400

Output: Monthly Sales Data with MultiIndex:
```
          Sales
City Month
NYC  Jan    100.0
     Feb    200.0
LA   Jan    300.0
     Feb    400.0

Total Sales Per City:
      Sales
City
```

LA 700.0
NYC 300.0

Answer

```
import pandas as pd
n=int(input())
c=input().split()

m=input().split()
s=list(map(float,input().split()))
index=pd.MultiIndex.from_tuples(zip(c,m),names=["city","Month"])
df=pd.DataFrame(s,index=index,columns=["Sales"])
print("Monthly Sales Data with MultiIndex:")
print(df)
ts=df.groupby(level="city").sum()
print("\n Total Sales Per City:")
print(ts)
```

Status : Partially correct ***Marks : 9/10***

3. Problem Statement

Sita is analyzing her company's daily sales data to find all sales values that are multiples of 5 and exceed 100. She wants to filter these specific sales values from the list.

Help her to implement the task using the numpy package.

Formula:

To filter sales values:

Select all values s from sales such that (s % 5 == 0) and (s > 100)

Input Format

The first line of input consists of an integer value, n, representing the number of sales entries.

The second line of input consists of n floating-point values, sales, separated by spaces, representing daily sales figures.

Output Format

The output prints: filtered_sales

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
50.0 100.0 105.0 150.0 99.0
Output: [105. 150.]

Answer

```
import numpy as np
a=int(input())
b=map(float,input().split())
l=[]
for i in b:
    if(i%5==0) and (i>100):
        l.append(i)
arr=np.array(l)
print(arr)
```

Status : Correct ***Marks : 10/10***

4. Problem Statement

Alex is a data scientist analyzing the relationship between two financial indicators over time. He has collected two time series datasets representing daily values of these indicators over several months. Alex wants to understand how these two indicators correlate at different time lags to identify possible leading or lagging behaviors.

Your task is to help Alex compute the cross-correlation of these two time series using numpy, so he can analyze the similarity between the two signals at various time shifts.

Input Format

The first line of input consists of space-separated float values representing the first time series, array1.

The second line of input consists of space-separated float values representing the second time series, array2.

Output Format

The first line of output prints: "Cross-correlation of the two time series:"

The second line of output prints: the 1D numpy array cross_corr representing the cross-correlation of array1 and array2 across different lags.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1.0 2.0 3.0
4.0 5.0 6.0
Output: Cross-correlation of the two time series:
[6. 17. 32. 23. 12.]

Answer

```
import numpy as np
a1=np.array(list(map(float,input().split())))
a2=np.array(list(map(float,input().split())))
c=np.correlate(a1,a2,mode='full')
print("Cross-correlation of the two time series:")
print(c)
```

Status : Correct ***Marks : 10/10***

5. Problem Statement

Rekha works in hospital data management and receives patient records with missing or incomplete data. She needs to clean the records by performing the following tasks:

Calculate the mean of the available Age values.Replace any missing (NaN)

values in the Age column with this mean age.Remove any rows where the Diagnosis value is missing (NaN).Reset the DataFrame index after removing these rows.

Implement this data cleaning task using the pandas package.

Input Format

The first line of input contains an integer n representing the number of patient records.

The second line contains the CSV header — comma-separated column names (e.g., "Name,Age,Diagnosis,Gender").

The next n lines each contain one patient record in comma-separated format.

Output Format

The first line of output is the text:

Cleaned Hospital Records:

The next lines print the cleaned pandas DataFrame (as produced by print(cleaned_df)).

This will include the updated values of the Age column (with missing ages filled by the mean age), and any rows with missing Diagnosis removed.

The DataFrame will be displayed using the default pandas print() representation.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
PatientID,Name,Age,Diagnosis
1,John Doe,45,Flu
2,Jane Smith,,Cold
3,Bob Lee,50,
4,Alice Green,38,Fever
5,Tom Brown,,Infection

Output: Cleaned Hospital Records:

	PatientID	Name	Age	Diagnosis
0	1	John Doe	45.000000	Flu
1	2	Jane Smith	44.333333	Cold
2	4	Alice Green	38.000000	Fever
3	5	Tom Brown	44.333333	Infection

Answer

```
import panda as pd
import numpy as np
n=int(input())
columns=input().split()

data=[input().split(',')for i in range(n)]
df=pd.DataFrame(data,columns=columns)
df['Age']=pd.to_numeric(df['PatientID'],errors='coerce')
df.replace('',np.nan,inplace=True)
mean_age=df['Age'].mean()
df['Age']=df['Age'].fillna(mean_age)
cleaned_df=df.dropna(subset=['Diagnosis']).reset_index(drop=True)
print("Cleaned HOspital Records:")
print(cleaned_df)
```

Status : Wrong ***Marks : 0/10***