Rajalakshmi Engineering College

Name: talapaneni sujitha
Email: 240701551@rajalakshmi.edu.in
Roll no: 2116240701551
Phone: 9445179891
Branch: REC
Department: I CSE FF
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 : 19

Section 1 : MCQ

1. How do you rename a file?

Answer

os.rename(existing_name, new_name)

Status : Correct *Marks : 1/1*

2. 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*

3. 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***

4. 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***

5. 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***

6. 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*

7. 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*

8. 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*

9. 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***

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 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***

12. 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***

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

Answer

file position remains unchanged

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

15. 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***

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

Answer

finally

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

17. 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***

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

Answer

raise Exception()

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

19. 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***

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: talapaneni sujitha
Email: 240701551@rajalakshmi.edu.in
Roll no: 2116240701551
Phone: 9445179891
Branch: REC
Department: I CSE FF
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

```
# You are using Python
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

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

```
# You are using Python
s=input()
with open("sentence_file.txt","w") as f:
  f.write(s)
with open("sentence_file.txt","r") as f:
  c=f.read()
```

```
print(len(c.split()))
```

Status : Correct **_Marks : 10/10_**

3. 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

```
# You are using Python
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***

4. 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

```
# You are using Python
k=int(input())
try:
   if(k>=18):
        print("Eligible to vote")
    else:
        raise ValueError
except ValueError:
        print("Not eligible to vote")
```

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

5. 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

```
# You are using Python
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***

Rajalakshmi Engineering College

Name: talapaneni sujitha
Email: 240701551@rajalakshmi.edu.in
Roll no: 2116240701551
Phone: 9445179891
Branch: REC
Department: I CSE FF
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 : 28.5

Section 1 : Coding

1. 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

```
# You are using Python
# Define the maximum capacity of the student database
MAX_CAPACITY = 30

# Initialize an empty dictionary to store student IDs and names
student_database = {}

# Read the number of students to be added
num_students_to_add = int(input())
```

```
# Loop through the number of students specified
for _ in range(num_students_to_add):
    try:
        # Read the student ID and name from a single line
        student_info = input().split()

        student_id = int(student_info[0])
        student_name = " ".join(student_info[1:])

        # Check if the database is full
        # This condition means we are about to add the (MAX_CAPACITY + 1)th
student
        # or more, so we should block and print error.
        if len(student_database) >= MAX_CAPACITY:
            # Print the exact error message as specified
            print("Exception caught. Error: Student database is full.")
            # Break the loop because no more students can be added to a full
database
            break

        # Check for duplicate student ID
        if student_id in student_database:
            # Print the exact error message as specified
            print("Exception caught. Error: Student ID already exists.")
        else:
            # Add the student to the database
            student_database[student_id] = student_name
            # Print the exact success message as specified using an f-string
            print(f"Student with ID {student_id} added to the database.")

    except ValueError:
        # This block is for robustness, though constraints imply valid input.
        pass
    except IndexError:
        # This block is for robustness, though constraints imply valid input.
        pass
```

Status : Partially correct ***Marks : 8.5/10***

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

```
# Read the file name
file_name = input()
```

```
# Read the string to write into the file
file_content = input()

# Read the character to count
char_to_count = input()

# Convert the character to count to lowercase (or uppercase) for case-
insensitive comparison
# This is crucial for the expected output of Testcase 2
char_to_count_lower = char_to_count.lower()

# Create and write to the file
with open(file_name, 'w') as f:
   f.write(file_content)

# Read the content of the file and count the character
char_count = 0
with open(file_name, 'r') as f:
   content = f.read()
   for char in content:
      # Convert each character from the file to lowercase before comparison
      if char.lower() == char_to_count_lower:
         char_count += 1

# Display the result
if char_count > 0:
   print(f"The character '{char_to_count}' appears {char_count} times in the file.")
else:
   print("Character not found in the file.")# You are using Python
```

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

3. 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

```
# You are using Python
def calculate_average(numbers):
    valid_numbers = []

    for num in numbers:
        try:
            valid_numbers.append(float(num))
        except ValueError:
            print("Invalid data in the input.")
            return

    average = sum(valid_numbers) / len(valid_numbers)
    print(f"Average of the numbers is: {average:.2f}")

input_data = input()
numbers = input_data.strip().split()
calculate_average(numbers)
```

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

Rajalakshmi Engineering College

Name: talapaneni sujitha
Email: 240701551@rajalakshmi.edu.in
Roll no: 2116240701551
Phone: 9445179891
Branch: REC
Department: I CSE FF
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

Bob, a data analyst, requires a program to automate the process of analyzing character frequency in a given text. This program should allow the user to input a string, calculate the frequency of each character within the text, save these character frequencies to a file named "char_frequency.txt," and display the results.

Input Format

The input consists of the string.

Output Format

The first line prints "Character Frequencies:".

The following lines print the character frequency in the format: "X: Y" where X is the character and Y is the count.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: aaabbbccc

```
Output: Character Frequencies:
a: 3
b: 3
c: 3
```

Answer

```
# You are using Python
# 1. Read the input string
input_string = input()

# 2. Calculate character frequencies
# Use a dictionary to store character counts
char_counts = {}
for char in input_string:
    # If the character is already in the dictionary, increment its count
    if char in char_counts:
        char_counts[char] += 1
    # If it's a new character, add it to the dictionary with a count of 1
    else:
        char_counts[char] = 1

# Prepare the output content for both console and file
output_lines = ["Character Frequencies:"]
for char, count in char_counts.items():
    output_lines.append(f"{char}: {count}")

# Convert the list of output lines to a single string with newlines
output_content = "\n".join(output_lines)

# 3. Print the frequencies to the console
print(output_content)

# 4. Save the frequencies to a file named "char_frequency.txt"
```

```
file_name = "char_frequency.txt"
with open(file_name, 'w') as f:
    f.write(output_content)
```

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

2. Problem Statement

Write a program to obtain the start time and end time for the stage event show. If the user enters a different format other than specified, an exception occurs and the program is interrupted. To avoid that, handle the exception and prompt the user to enter the right format as specified.

Start time and end time should be in the format 'YYYY-MM-DD HH:MM:SS'If the input is in the above format, print the start time and end time.If the input does not follow the above format, print "Event time is not in the format "

Input Format

The first line of input consists of the start time of the event.

The second line of the input consists of the end time of the event.

Output Format

If the input is in the given format, print the start time and end time.

If the input does not follow the given format, print "Event time is not in the format".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2022-01-12 06:10:00
2022-02-12 10:10:12

Output: 2022-01-12 06:10:00
2022-02-12 10:10:12

Answer

```
# You are using Python
from datetime import datetime

# Define the expected datetime format
DATE_TIME_FORMAT = "%Y-%m-%d %H:%M:%S"

# Flag to track if any input is in the wrong format
is_format_correct = True

# Read and parse start time
start_time_str = input()
try:
    # Attempt to parse the start time string
    datetime.strptime(start_time_str, DATE_TIME_FORMAT)
except ValueError:
    # If parsing fails, set the flag to False
    is_format_correct = False

# Read and parse end time
end_time_str = input()
try:
    # Attempt to parse the end time string
    datetime.strptime(end_time_str, DATE_TIME_FORMAT)
except ValueError:
    # If parsing fails, set the flag to False
    is_format_correct = False

# Determine output based on the flag
if is_format_correct:
    print(start_time_str)
    print(end_time_str)
else:
    print("Event time is not in the format")
```

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

3. Problem Statement

Implement a program that checks whether a set of three input values can form the sides of a valid triangle. The program defines a function is_valid_triangle that takes three side lengths as arguments and raises a ValueError if any side length is not a positive value. It then checks whether the sum of any two sides is greater than the third side to determine the validity of the triangle.

Input Format

The first line of input consists of an integer A, representing side1.

The second line of input consists of an integer B, representing side2.

The third line of input consists of an integer C, representing side3.

Output Format

The output prints either "It's a valid triangle" if the input side lengths form a valid triangle,

or "It's not a valid triangle" if they do not.

If there is a ValueError, it should print "ValueError: <error_message>".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
4
5
Output: It's a valid triangle

Answer

```
# You are using Python
def is_valid_triangle(side1, side2, side3):
    """
    Checks if three given side lengths can form a valid triangle.
    Raises ValueError if any side length is not a positive value.
    """
    # Check if all side lengths are positive
    if side1 <= 0 or side2 <= 0 or side3 <= 0:
        raise ValueError("Side lengths must be positive")

    # Check the triangle inequality theorem
    # The sum of any two sides must be greater than the third side
    if (side1 + side2 > side3) and \
       (side1 + side3 > side2) and \
       (side2 + side3 > side1):
        return True
    else:
        return False

# Main part of the program
if __name__ == "__main__":
    try:
        # Read the three side lengths from input
        A = int(input())
        B = int(input())
        C = int(input())

        # Call the function and print the result
        if is_valid_triangle(A, B, C):
            print("It's a valid triangle")
        else:
            print("It's not a valid triangle")

    except ValueError as e:
        # Catch ValueError raised by is_valid_triangle or by int(input())
        # The problem statement implies ValueError for non-positive sides from the
function
        # and not for non-integer inputs. Assuming valid integer inputs.
        print(f"ValueError: {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

```
# You are using Python
# Define the maximum allowed days
MAX_DAYS = 30
FILE_NAME = "sales.txt"

# 1. Read Inputs
N = int(input())
items_sold_str = input().split()
M = int(input())

# Convert items sold to a list of integers
items_sold_per_day = [int(item) for item in items_sold_str]

# 2. Check for Limit
if N > MAX_DAYS:
    print("Exceeding limit!")
else:
    # 3. Calculate Daily Earnings and Write to File
    daily_earnings = []
    for items in items_sold_per_day:
        earnings = items * M
        daily_earnings.append(str(earnings)) # Convert to string for writing to file

    # Write daily earnings to sales.txt
    # 'w' mode truncates the file if it exists or creates it if it doesn't
    with open(FILE_NAME, 'w') as file:
        file.write("\n".join(daily_earnings))

    # 4. Read from File and Display
    # 'r' mode opens the file for reading
    with open(FILE_NAME, 'r') as file:
        # Read all lines from the file
        content = file.read()
        # Print the content directly, which already contains newlines
        print(content.strip()) # .strip() removes any trailing newline if present, to
match exact output
```

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