Class XI Computer Science
Practical File Check List Term II 2023-24

Programs Based on Strings

1) Write a program in python to print alternate characters from the string accepted from the user
2) Write a program in python to accept a string from the user and print the words which are palindromes
3) Write a program to print all the digits from the string accepted from the user.
4) Write a Python program to accept a string from the user and store it in a variable STR. count the number of words.
5) Write a program to assign a multi-line string into a variable Lines and display the number of words in each line.
6) Write a program to accept a string from the user and print alternate words
7) Write a program to accept a string from the user and print only those words starting with a vowel.
8) Write a program to accept a string from the user and print alternate words which are starting with a digit
9) Write a program to accept a string from the user and print alternate words which are 5 characters in length.
10) Write a Python program to accept a string from the user and read a random word from the list of words.
11) Write a program to accept a string from the user and print only those words which are starting with 'a' and ending with 'e'.
12) Write a program to accept a string from the user and print only those words which have vowels in it.
13) Write a program to accept a string from the user and print the longest and the shortest word.
14) Write a Python program to count the frequency of words in a string Count and print the number of times the following words appear in the string accepted from the user
    a) "and" b) "is"
15) Write a program to copy the contents of the string STR1 into STR2. While copying the content append "#" after each and every character.
16) Write a program to copy the contents of the string STR1 into STR2. While copying the content remove the words starting with lower case.

Programs Based on Lists

1) Write a Menu Driven program to perform the following operations on a list. The program should execute as long as the user wants.
   Menu.
   1. Create a List
   2. Append a random 4-digit integer into the list
   3. Extend the list
   4. Insert an element into a list at a particular position
   5. Remove an element from the list based on the user's choice.
      a. The last element
      b. The element at a particular position
      c. The first occurrence of the element
   6. Print the largest element and smallest element in the list
   7. Find the second largest and second smallest element in the list.
   8. Sort the list in ascending or descending order as per user's choice (Do not change original list)
   9. Reverse the elements of a list (Do not change original list)
   10. Clear the list
   11. Exit
2) Write a program to accept values into a list and search for a particular element from a list of numbers (Linear Search)
3) Write a program to accept a list of numbers, find and print the sum and the average
4) Write a program to input two lists and create a third list that contains the elements from both the lists and sort it
5) Write a program to accept a list of numbers from the user (between 1-15) and replace all the entries in the list which are greater than 10 by **100**
6) Write a program to accept a string from the user and convert it to a list
7) Write a program to accept two lists of same size and form another list which has the sum of the corresponding elements of both the lists
   Eg: list1= [2,3,4,5,6]
   list2= [2,2,1,1,1]
   list3= [4,5,5,6,7]
8) Write a program that rotates the elements of a list so that the elements at the first index moves to the second index, the second to third and last to first.
   List1 [] = [1, 2, 3, 4, 5, 6] Output: [6, 1, 2, 3, 4, 5]
9) Write a program to accept a list of numbers and then put even elements into one list and odd elements into another.
10) Write a program to read a list of words. Find and print the longest word in the list with its length

Programs Based on Tuples

11) Write a program to convert a list to tuple and replicate it 3 times
12) Write a program in python to remove an item from a tuple
13) Write a program to find the index of each element of a tuple
14) Write a program to accept the details (Id, Name, Age, DOB, Salary) of Five employees into a tuple Employee. Unpack the details of first employee into variables Id, Name, Age, DOB and Salary
15) Write a program to reverse a tuple
16) Write a program to accept five strings into a tuple and find the length of the shortest string
17) Accept marks of five students in three subjects into a tuple. For each find the and display the following
    a. Maximum marks out of three subjects
    b. Minimum marks out of three subjects
    c. Sum and average of the marks in three subjects

Programs Based on Dictionaries

18) Write a program to accept a sentence/ paragraph from the user. Create a dictionary that stores each word and its Count of its frequency in the given sentence /paragraph as key-value pairs respectively.
19) Repeatedly ask the user to enter a team name and how many games the team has won and how many they have lost. Store this information in a dictionary where the keys are the team names and the values are the list of the form [wins, losses]
    a. Using the dictionary, created above, allow the user to enter a team name and print out the team's winning percentage
    b. Using the dictionary, create a list whose entries are the number of wins of each team
    c. Using the dictionary, create a list of all those teams that have only winning record (i.e. zero losses)
20) Write a program to repeatedly ask the user to enter product names and prices. Store all of these in a dictionary whose keys are the product names and whose values are the prices.
    ****when the user is done entering products and prices, allow them to repeatedly enter a product name and print the corresponding price or print a message product does not exist.
21) Given the dictionary X={'11A':"Uzma",'11B':"Sonia",'11C':"Doyel",'11D':"Sreeja"}
    Create a dictionary with the opposite mapping, i.e. write a program to create the dictionary as:
    Inverted_X= {"Uzma":'11A',"Sonia":'11B',"Doyel":'11C',"Sreeja":'11D'}
22) Given two dictionaries say D1 and D2. Write a program that lists the overlapping keys of the two dictionaries i.e. if a key of D1 is also present in D2, then list it
23) Create a dictionary whose keys are the names of the month and whose values are the number of days in the month
    a. Printout all of the keys in alphabetical order
    b. Print out the key –value pair sorted by the no of days in a month
    c. Print out all the months with 31 days

Programs based on Strings:

Program 1:

Write a program in python to print alternate characters from the string accepted from the user

Source Code:

```
def print_alternate_characters(input_string):
    result = ""
    for i in range(0, len(input_string), 2):
        result += input_string[i] # Add the character at index i to the result string
    print("Alternate characters:", result)

# Get input from the user
user_input = input("Enter a string: ")

# Call the function to print alternate characters
print_alternate_characters(user_input)
```

Output:

```
Enter a string: this is a sample string
Alternate characters: ti sasml tig
```

Program 2:

1) Write a program in python to accept a string from the user and print the words which are palindromes

Source Code:

```python
def is_palindrome(word):
    return word == "".join(reversed(word))

def print_palindrome_words(input_string):
    words = input_string.split()
    palindrome_words = [word for word in words if is_palindrome(word)]
    print("Palindrome words:", palindrome_words)

# Get input from the user
user_input = input("Enter a string: ")

# Call the function to print palindrome words
print_palindrome_words(user_input)
```

Output:

```
Enter a string: racecar
Palindrome words: racecar
```

Program 3:

Write a program to print all the digits from the string accepted from the user.

Source Code:

```python
#user input
in_str = str(input("Enter a string: "))
# print the list of digits in the string
print(list(char for char in in_str if char.isdigit()))
```

Output:

```
Enter a string: this is a string 12 thi 123
['1', '2', '1', '2', '3']
```

Program 4:

1) Write a Python program to accept a string from the user and store it in a variable STR. count the number of words.

Source Code:

```
#user input
input_str = str(input("Enter a string: "))
#count no of words
print(len(input_str.split()))
```

Output:

```
Enter a string: this is a string
4
```

Program 5:

Write a program to assign a multi-line string into a variable Lines and display the number of words in each line.

Source Code:

```
Lines = """This is a multi-line string.
It spans across several lines.
Each line contains different words."""

# Split the lines into a list of words and count the number of words in each line
for i, line in enumerate(Lines.split('\n'), 1):
    word_count = len(line.split())
    print(f"Line {i}: {word_count} words")
```

Output:

```
Line 1: 5 words
Line 2: 5 words
Line 3: 5 words
```

Program 6:

Write a program to accept a string from the user and print alternate words

Source Code:

```
user_input = input("Enter a string: ")
print("Alternate words:", ', '.join(user_input.split()[::2]))
```

Output:

```
Enter a string: this is a string
Alternate words: this, a
```

Program 7:

Write a program to accept a string from the user and print only those words starting with a vowel

Source Code:

```
def print_words_starting_with_vowel(input_string):
    words = input_string.split()
    vowel_words = [word for word in words if word[0].lower() in 'aeiou']
    print("Words starting with a vowel:", ', '.join(vowel_words))

# Get input from the user
user_input = input("Enter a string: ")

# Call the function to print words starting with a vowel
print_words_starting_with_vowel(user_input)
```

Output:

```
Enter a string: this is a string and an apple
Words starting with a vowel: is, a, and, an, apple
```

Program 8:

Write a program to accept a string from the user and print alternate words which are starting with a digit

Source Code:

```
def print_alternate_words_starting_with_digit(input_string):
    words = input_string.split()
    selected_words = [word for i, word in enumerate(words) if i % 2 == 0 and word[0].isdigit()]
    print("Alternate words starting with a digit:", ', '.join(selected_words))

# Get input from the user
user_input = input("Enter a string: ")

# Call the function to print alternate words starting with a digit
print_alternate_words_starting_with_digit(user_input)
```

Output:

```
Enter a string: this is a string 12 thi 123
Alternate words starting with a digit: 12, 123
```

Program 9:

Write a program to accept a string from the user and print alternate words which are 5 characters in length

Source Code:

```
def print_alternate_words_of_length(input_string, target_length=5):
    words = input_string.split()
    selected_words = [word for i, word in enumerate(words) if i % 2 == 0 and len(word) == target_length]
    print(f"Alternate words of length {target_length}:", ', '.join(selected_words))

# Get input from the user
user_input = input("Enter a string: ")

# Call the function to print alternate words of length 5
print_alternate_words_of_length(user_input, target_length=5)
```

Output:

```
Enter a string: this is a thing thing thing
Alternate words of length 5: thing
```

Program 10:

Write a Python program to accept a string from the user and read a random word from the list of words

Source Code:

```
import random
user_input = input("Enter a string: ")
word_list = user_input.split()
print(f"You entered: {user_input}")
print(f"Random word from the list: {random.choice(word_list)}")
```

Output:

```
Enter a string: thhis is a string
You entered: thhis is a string
Random word from the list: is
```

Program 11:

Write a program to accept a string from the user and print only those words which are starting with ‘a’ and ending with ‘e’.

Source Code:

```
def main():
    user_input = input("Enter a string: ")

    filtered_words = [word for word in user_input.split() if word.startswith('a') and word.endswith('e')]

    if filtered_words:
        print("Words starting with 'a' and ending with 'e':", *filtered_words)
    else:
        print("No matching words found.")

if __name__ == "__main__":
    main()
```

Output:

```
Enter a string: thhis is a string
No matching words found.
```

Program 12:

Write a program to accept a string from the user and print only those words which have vowels in it.

Source Code:

```
def main():
    user_input = input("Enter a string: ")

    words_with_vowels = [word for word in user_input.split() if any(char in "aeiouAEIOU" for char in word)]

    print("Words with vowels:", *words_with_vowels) if words_with_vowels else print("No words with vowels found.")

if __name__ == "__main__":
    main()
```

Output:

```
Enter a string: thhis is a string
Words with vowels: thhis is a string
```

Program 13:

Write a program to accept a string from the user and print the longest and the shortest word

Source Code:

```
user_input = input("Enter a string: ")
words = user_input.split()
if words:
    longest = max(words, key=len)
    shortest = min(words, key=len)
    print(f"Longest word: {longest}, shortest word: {shortest}")
else:
    print("No words entered.")
```

Output:

```
Enter a string: thhis is a string
Longest word: string, shortest word: a
```

Program 14:

Write a Python program to count the frequency of words in a string Count and print the number of times the following words appear in the string accepted from the user

a) “and” b) “is”

Source Code:

```
#count frequency of words
user_input = input("Enter a string: ")
words = user_input.split()
word_freq = {}
for word in words:
    word_freq[word] = word_freq.get(word, 0) + 1
print("Word frequency:", word_freq)
word_and_count = word_freq.get("and", 0)
word_is_count = word_freq.get("is", 0)
print(f"Frequency of 'and': {word_and_count}, frequency of 'is': {word_is_count}")
```

Output:

```
Enter a string: thhis is a string
Word frequency: {'thhis': 1, 'is': 1, 'a': 1, 'string': 1}
Frequency of 'and': 0, frequency of 'is': 1
```

Program 15:

Write a program to copy the contents of the string STR1 into STR2. While copying the content append “#” after each and every character.

Source Code:

```
str1 = str(input("Enter a string: "))
str2 = ''
for i in str1:
    str2 += i + '#'
print(str2)
```

Output:

```
Enter a string: thhis is a string
t#h#h#i#s# #i#s# #a# #s#t#r#i#n#g#
```

Program 16:

Write a program to copy the contents of the string STR1 into STR2. While copying the content remove the words starting with lower case

Source Code:

```
str1 = input("Enter a string (STR1): ")
str2 = ' '.join(word for word in str1.split() if not word[0].islower())
print(f"STR2 after removing words starting with lowercase: {str2}")
```

Output:

```
Enter a string (STR1): thhis is a string
STR2 after removing words starting with lowercase:
```

Programs Based on Lists

Program 1:

Write a Menu Driven program to perform the following operations on a list. The program should execute as long as the user wants.

Menu.

1) Create a List
2) Append a random 4-digit integer into the list
3) Extend the list
4) Insert an element into a list at a particular position
5) Remove an element from the list based on the user's choice.
    a. The last element
    b. The element at a particular position
    c. The first occurrence of the element
6) Print the largest element and smallest element in the list
7) Find the second largest and second smallest element in the list.
8) Sort the list in ascending or descending order as per user's choice (Do not change original list)
9) Reverse the elements of a list (Do not change original list)
10) Clear the list
11) Exit

Source Code:

```
import random      Debugging Bug, last week • Add files via upload

# Function to create a new list
def create_list():
    global my_list
    my_list = []
    print("New list created.")

# Function to append a random 4-digit integer to the list
def append_random():
    random_num = random.randint(1000, 9999)
    my_list.append(random_num)
    print(f"Random 4-digit number {random_num} appended to the list.")

# Function to extend the list
def extend_list():
    extension = input("Enter elements to extend the list (comma-separated): ").split(',')
    extension = [int(elem.strip()) for elem in extension]
    my_list.extend(extension)
    print("List extended.")

# Function to insert an element at a particular position in the list
def insert_element():
    pos = int(input("Enter position to insert the element: "))
    elem = int(input("Enter the element to insert: "))
    my_list.insert(pos, elem)
    print(f"Element {elem} inserted at position {pos}.")

# Function to remove elements from the list
def remove_element():
    print("Select an option to remove an element:")
    print("1. Remove last element")
    print("2. Remove element at a particular position")
    print("3. Remove the first occurrence of an element")
    choice = input("Enter your choice: ")
```

```
    if choice == '1':
        if len(my_list) > 0:
            removed_element = my_list.pop()
            print(f"Removed last element: {removed_element}")
        else:
            print("List is empty.")
    elif choice == '2':
        pos = int(input("Enter position to remove the element: "))
        if 0 <= pos < len(my_list):
            removed_element = my_list.pop(pos)
            print(f"Removed element at position {pos}: {removed_element}")
        else:
            print("Invalid position.")
    elif choice == '3':
        elem = int(input("Enter the element to remove: "))
        if elem in my_list:
            my_list.remove(elem)
            print(f"Removed first occurrence of element {elem}.")
        else:
            print("Element not found in the list.")
    else:
        print("Invalid choice.")

# Function to print largest and smallest elements in the list
def print_largest_smallest():
    if len(my_list) > 0:
        print(f"Largest element: {max(my_list)}")
        print(f"Smallest element: {min(my_list)}")
    else:
        print("List is empty.")

# Function to find second largest and second smallest elements in the list
def find_second_largest_smallest():
    if len(my_list) > 1:
        sorted_list = sorted(set(my_list))
        if len(sorted_list) > 1:
            print(f"Second largest element: {sorted_list[-2]}")
            print(f"Second smallest element: {sorted_list[1]}")
        else:
            print("Not enough elements in the list.")
    else:
        print("Not enough elements in the list.")

# Function to sort the list in ascending or descending order
def sort_list():
    print("Select sorting order:")
    print("1. Sort in ascending order")
    print("2. Sort in descending order")
    choice = input("Enter your choice: ")
```

```
    if choice == '1':
        sorted_asc = sorted(my_list)
        print(f"Sorted list in ascending order: {sorted_asc}")
    elif choice == '2':
        sorted_desc = sorted(my_list, reverse=True)
        print(f"Sorted list in descending order: {sorted_desc}")
    else:
        print("Invalid choice.")

# Function to reverse the elements of the list
def reverse_list():
    reversed_list = my_list[::-1]
    print(f"Reversed list: {reversed_list}")

# Function to clear the list
def clear_list():
    global my_list
    my_list = []
    print("List cleared.")

# Main program
my_list = []

while True:
    print("\nMenu:")
    print("a. Create a List")
    print("b. Append a random 4 digit integer into the list")
    print("c. Extend the list")
    print("d. Insert an element into a list at a particular position")
    print("e. Remove an element from the list")
    print("f. Print the largest element and smallest element in the list")
    print("g. Find the second largest and second smallest element in the list")
    print("h. Sort the list in ascending or descending order")
    print("i. Reverse the elements of a list")
    print("j. Clear the list")
    print("k. Exit")

    choice = input("Enter your choice: ").lower()
```

```
    if choice == 'a':
        create_list()
    elif choice == 'b':
        append_random()
    elif choice == 'c':
        extend_list()
    elif choice == 'd':
        insert_element()
    elif choice == 'e':
        remove_element()
    elif choice == 'f':
        print_largest_smallest()
    elif choice == 'g':
        find_second_largest_smallest()
    elif choice == 'h':
        sort_list()
    elif choice == 'i':
        reverse_list()
    elif choice == 'j':
        clear_list()
    elif choice == 'k':
        print("Exiting the program. Goodbye!")
        break
    else:
        print("Invalid choice. Please enter a valid option.")
```

Program 2:

Write a program to accept values into a list and search for a particular element from a list of numbers (Linear Search)

Source Code:

```
# Function for linear search
def linear_search(arr, target):
    for i in range(len(arr)):
        if arr[i] == target:
            return i  # Return the index if the element is found
    return -1  # Return -1 if the element is not found

# Accepting values into a list
num_list = []
n = int(input("Enter the number of elements in the list: "))

print("Enter the elements of the list:")
for i in range(n):
    num = int(input())
    num_list.append(num)

# Accept the element to search
element_to_find = int(input("Enter the element to search: "))

# Perform linear search
result = linear_search(num_list, element_to_find)

if result != -1:
    print(f"Element {element_to_find} found at index {result}.")
else:
    print(f"Element {element_to_find} not found in the list.")
```

Output:

```
Enter the number of elements in the list: 4
Enter the elements of the list:
12
12
45
23
Enter the element to search: 23
Element 23 found at index 3.
```

Program 3:

Write a program to accept a list of numbers, find and print the sum and the average

Source Code:

```
# Function to calculate sum and average of a list of numbers
def calculate_sum_average(num_list):
    list_sum = sum(num_list)
    list_average = list_sum / len(num_list)
    return list_sum, list_average

# Accepting a list of numbers from the user
num_list = []
n = int(input("Enter the number of elements in the list: "))

print("Enter the elements of the list:")
for i in range(n):
    num = float(input())
    num_list.append(num)

# Calculate sum and average
sum_of_list, average_of_list = calculate_sum_average(num_list)

print(f"The sum of the numbers in the list is: {sum_of_list}")
print(f"The average of the numbers in the list is: {average_of_list:.2f}")
```

Output:

```
Enter the number of elements in the list: 4
Enter the elements of the list:
123
3434
4545
23
The sum of the numbers in the list is: 8125.0
The average of the numbers in the list is: 2031.25
```

Program 4:

Write a program to input two lists and create a third list that contains the elements from both the lists and sort it

Source Code:

```
# Function to merge two lists and sort the merged list
def merge_and_sort_lists(list1, list2):
    merged_list = list1 + list2
    sorted_list = sorted(merged_list)
    return sorted_list

# Accepting the first list from the user
list1 = []
n1 = int(input("Enter the number of elements in the first list: "))

print("Enter the elements of the first list:")
for i in range(n1):
    elem = int(input())
    list1.append(elem)

# Accepting the second list from the user
list2 = []
n2 = int(input("Enter the number of elements in the second list: "))

print("Enter the elements of the second list:")
for i in range(n2):
    elem = int(input())
    list2.append(elem)

# Merge and sort the lists
sorted_merged_list = merge_and_sort_lists(list1, list2)

print(f"Merged and sorted list: {sorted_merged_list}")
```

Output:

```
Enter the number of elements in the first list: 2
Enter the elements of the first list:
23
54
Enter the number of elements in the second list: 4
Enter the elements of the second list:
3423
12
34
14
Merged and sorted list: [12, 14, 23, 34, 54, 3423]
```

Program 5:

Write a program to accept a list of numbers from the user (between 1-15) and replace all the entries in the list which are greater than 10 by 100

Source Code:

```
# Function to replace elements greater than 10 with 100
def replace_greater_than_10(num_list):
    for i in range(len(num_list)):
        if num_list[i] > 10:
            num_list[i] = 100

# Accepting a list of numbers from the user
num_list = []
n = int(input("Enter the number of elements in the list (between 1-15): "))

if n < 1 or n > 15:
    print("Invalid input! Number of elements should be between 1 and 15.")
else:
    print("Enter the elements of the list:")
    for i in range(n):
        num = float(input())
        num_list.append(num)

    # Replace elements greater than 10 with 100
    replace_greater_than_10(num_list)

    print("List after replacing elements greater than 10 with 100:")
    print(num_list)
```

Output:

```
Enter the number of elements in the list (between 1-15): 4
Enter the elements of the list:
1
234
10
343
List after replacing elements greater than 10 with 100:
[1.0, 100, 10.0, 100]
```

Program 6:

Write a program to accept a string from the user and convert it to a list

Source Code:

```
user_string = input("Enter a string: ")
char_list = list(user_string)
print(char_list)
```

Output:

```
Enter a string: this is a string
['t', 'h', 'i', 's', ' ', 'i', 's', ' ', 'a', ' ', 's', 't', 'r', 'i', 'n', 'g']
```

Program 7:

Write a program to accept two lists of same size and form another list which has the sum of the corresponding elements of both the lists

Source Code:

```
# Function to form a list with the sum of corresponding elements from two lists
def sum_corresponding_elements(list1, list2):
    if len(list1) != len(list2):
        return None  # If lists are not of the same size, return None

    result_list = [list1[i] + list2[i] for i in range(len(list1))]
    return result_list

# Accepting the first list from the user
list1 = []
n = int(input("Enter the number of elements in the lists: "))

print("Enter the elements of the first list:")
for i in range(n):
    elem = int(input())
    list1.append(elem)

# Accepting the second list from the user
list2 = []
print("Enter the elements of the second list:")
for i in range(n):
    elem = int(input())
    list2.append(elem)

# Forming the list with the sum of corresponding elements
result = sum_corresponding_elements(list1, list2)

if result is None:
    print("Lists are not of the same size.")
else:
    print(f"The resulting list with the sum of corresponding elements is: {result}")
```

Output:

```
Enter the number of elements in the lists: 2
Enter the elements of the first list:
12
34
Enter the elements of the second list:
123
343
The resulting list with the sum of corresponding elements is: [135, 377]
```

Program 8:

Write a program that rotates the elements of a list so that the elements at the first index moves to the second index, The second to third and last to first

Source Code:

```
# Function to rotate the elements of a list
def rotate_list_elements(my_list):
    if len(my_list) <= 1:
        return my_list  # Return the list as it is if it has 0 or 1 element

    last_element = my_list[-1]  # Store the last element before rotation
    for i in range(len(my_list) - 1, 0, -1):
        my_list[i] = my_list[i - 1]  # Shift elements to the right

    my_list[0] = last_element  # Place the last element at the first index
    return my_list

# Accepting a list of elements from the user
num_list = []
n = int(input("Enter the number of elements in the list: "))

print("Enter the elements of the list:")
for i in range(n):
    num = int(input())
    num_list.append(num)

# Rotate the elements of the list
rotated_list = rotate_list_elements(num_list)

print(f"The list after rotating its elements: {rotated_list}")
```

Output:

```
Enter the number of elements in the list: 3
Enter the elements of the list:
343
45
12
The list after rotating its elements: [12, 343, 45]
```

Program 9:

Write a program to accept a list of numbers and then put even elements into one list and odd elements into another.

Source Code:

```python
def separate_even_odd(input_list):
    even_list = []
    odd_list = []

    for number in input_list:
        if number % 2 == 0:
            even_list.append(number)
        else:
            odd_list.append(number)

    return even_list, odd_list

# Input: List of numbers
input_numbers = [int(x) for x in input("Enter a list of numbers separated by space: ").split()]

# Separating even and odd elements
even_elements, odd_elements = separate_even_odd(input_numbers)
      You, 7 hours ago • more questions
# Displaying the results
print("Even elements:", even_elements)
print("Odd elements:", odd_elements)
```

Output:

```
Enter a list of numbers separated by space: 45 234 454 12 45
Even elements: [234, 454, 12]
Odd elements: [45, 45]
```

Program 10:

Write a program to read a list of words. Find and print the longest word in the list with its length

Source Code:

```python
def find_longest_word(word_list):
    if not word_list:
        return None, 0

    longest_word = max(word_list, key=len)
    length_of_longest_word = len(longest_word)

    return longest_word, length_of_longest_word

# Input: List of words
word_list = input("Enter a list of words separated by space: ").split()

# Finding the longest word and its length
longest_word, length_of_longest_word = find_longest_word(word_list)

# Displaying the result
if longest_word:
    print("Longest word:", longest_word)
    print("Length of the longest word:", length_of_longest_word)
else:
    print("No words entered.")
```

Output:

```
Enter a list of words separated by space: this is a word
Longest word: this
Length of the longest word: 4
```

Programs on Tuples:

Program 1:

Write a program to convert a list to tuple and replicate it 3 times

Source Code:

```
replicated_tuple = tuple(input("Enter a list of elements separated by space: ").split()) * 3
print(replicated_tuple)
```

Output:

```
Enter a list of elements separated by space: this is a list
('this', 'is', 'a', 'list', 'this', 'is', 'a', 'list', 'this', 'is', 'a', 'list')
```

Program 2:

Write a program in python to remove an item from a tuple

Source Code:

```
def remove_item_from_tuple(input_tuple, item_to_remove):
    # Check if the item is present in the tuple
    if item_to_remove in input_tuple:
        # Create a new tuple excluding the specified item
        new_tuple = tuple(element for element in input_tuple if element != item_to_remove)
        return new_tuple
    else:
        print(f"{item_to_remove} not found in the tuple.")
        return input_tuple

# Input: Tuple of elements
input_tuple = tuple(input("Enter a tuple of elements separated by space: ").split())

# Input: Item to be removed
item_to_remove = input("Enter the item to remove from the tuple: ")

# Remove the item from the tuple
result_tuple = remove_item_from_tuple(input_tuple, item_to_remove)

# Displaying the result
print("Original Tuple:", input_tuple)
print("Tuple after removing", item_to_remove + ":", result_tuple)
```

Output:

```
Enter a tuple of elements separated by space: this is a bunch of elements
Enter the item to remove from the tuple: this
Original Tuple: ('this', 'is', 'a', 'bunch', 'of', 'elements')
Tuple after removing this: ('is', 'a', 'bunch', 'of', 'elements')
```

Program 3:

Write a program to find the index of each element of a tuple

Source Code:

```
input_tuple = tuple(input("Enter a tuple of elements separated by space: ").split())
element_indices = {element: index for index, element in enumerate(input_tuple)}
print("Element indices:", {element: index for index, element in enumerate(input_tuple)})
```

Output:

```
Enter a tuple of elements separated by space: this is a bunch of elements
Element indices: {'this': 0, 'is': 1, 'a': 2, 'bunch': 3, 'of': 4, 'elements': 5}
```

Program 4:

Write a program to accept the details (Id, Name, Age, DOB, Salary) of Five employees into a tuple Employee. Unpack the details of first employee into variables Id, Name, Age, DOB and Salary

Source Code:

```
employee_details = []

for _ in range(5):
    employee_id = input("Enter employee ID: ")
    employee_name = input("Enter employee name: ")
    employee_age = input("Enter employee age: ")
    employee_dob = input("Enter employee DOB: ")
    employee_salary = input("Enter employee salary: ")

    employee = (employee_id, employee_name, employee_age, employee_dob, employee_salary)
    employee_details.append(employee)

# Unpacking details of the first employee
first_employee_id, first_employee_name, first_employee_age, first_employee_dob, first_employee_salary = employee_details[0]

# Displaying the details of the first employee
print("\nDetails of the first employee:")
print("Employee ID:", first_employee_id)
print("Employee Name:", first_employee_name)
print("Employee Age:", first_employee_age)
print("Employee DOB:", first_employee_dob)
print("Employee Salary:", first_employee_salary)
```

Output:

```
Enter employee ID: 1
Enter employee name: margasm
Enter employee age: 69
Enter employee DOB: 12/2/1969
Enter employee salary: 69,000
Enter employee ID: 2
Enter employee name: suresh
Enter employee age: 56
Enter employee DOB: 12/3/1978
Enter employee salary: 70,000
Enter employee ID:
Enter employee name:
Enter employee age:
Enter employee DOB:
Enter employee salary:
Enter employee ID:
Enter employee name:
Enter employee age:
Enter employee DOB:
Enter employee salary:
Enter employee ID:
Enter employee name:
Enter employee age:
Enter employee DOB:
Enter employee salary:

Details of the first employee:
Employee ID: 1
Employee Name: margasm
Employee Age: 69
Employee DOB: 12/2/1969
Employee Salary: 69,000
```

Program 5:

Write a program to reverse a tuple

Source Code:

```
input_tuple = tuple(input("Enter a tuple of elements separated by space: ").split())
print("Original Tuple:", input_tuple, "\nReversed Tuple:", input_tuple[::-1])
```

Output:

```
Enter a tuple of elements separated by space: this is a bunch of elements
Original Tuple: ('this', 'is', 'a', 'bunch', 'of', 'elements')
Reversed Tuple: ('elements', 'of', 'bunch', 'a', 'is', 'this')
```

Program 6:

Write a program to accept five strings into a tuple and find the length of the shortest string

Source Code:

```
# Accepting five strings into a tuple
string_tuple = tuple(input("Enter five strings separated by space: ").split())

# Finding the length of the shortest string
shortest_length = min(len(s) for s in string_tuple)

# Displaying the result
print("Strings:", string_tuple)
print("Length of the shortest string:", shortest_length)
```

Output:

```
Enter five strings separated by space: this is a bunch of
Strings: ('this', 'is', 'a', 'bunch', 'of')
Length of the shortest string: 1
```

Program 7:

Accept marks of five students in three subjects into a tuple. For each find the and display the following

a. Maximum marks out of three subjects
b. Minimum marks out of three subjects
c. Sum and average of the marks in three subjects

Source Code:

```python
# Function to calculate maximum, minimum, sum, and average marks
def analyze_marks(student_marks):
    max_marks = max(student_marks)
    min_marks = min(student_marks)
    total_marks = sum(student_marks)
    average_marks = total_marks / len(student_marks)

    return max_marks, min_marks, total_marks, average_marks

# Accepting marks for five students into a list of tuples
student_marks_list = []

for i in range(5):
    student_marks = tuple(map(int, input(f"Enter marks for student {i + 1} in three subjects separated by space: ").split()))
    student_marks_list.append(student_marks)

# Analyzing and displaying results for each student
for i, student_marks in enumerate(student_marks_list, 1):
    max_marks, min_marks, total_marks, average_marks = analyze_marks(student_marks)

    print(f"\nStudent {i} - Marks Analysis:")
    print("Maximum Marks:", max_marks)
    print("Minimum Marks:", min_marks)
    print("Total Marks:", total_marks)
    print("Average Marks:", average_marks)
```

Output:

```
Enter marks for student 1 in three subjects separated by space: 34 45 56
Enter marks for student 2 in three subjects separated by space: 23 45 56
Enter marks for student 3 in three subjects separated by space: 34 34 54
Enter marks for student 4 in three subjects separated by space: 23 45 66
Enter marks for student 5 in three subjects separated by space: 33 44 55

Student 1 - Marks Analysis:
Maximum Marks: 56
Minimum Marks: 34
Total Marks: 135
Average Marks: 45.0

Student 2 - Marks Analysis:
Maximum Marks: 56
Minimum Marks: 23
Total Marks: 124
Average Marks: 41.333333333333336

Student 3 - Marks Analysis:
Maximum Marks: 54
Minimum Marks: 34
Total Marks: 122
Average Marks: 40.666666666666664

Student 4 - Marks Analysis:
Maximum Marks: 66
Minimum Marks: 23
Total Marks: 134
Average Marks: 44.666666666666664

Student 5 - Marks Analysis:
Maximum Marks: 55
Minimum Marks: 33
Total Marks: 132
Average Marks: 44.0
```

Programs on Dictionaries:

Program 1:

Write a program to accept a sentence/ paragraph from the user. Create a dictionary that stores each word and its Count of its frequency in the given sentence /paragraph as key-value pairs respectively.

Source Code:

```
# Accepting a sentence or paragraph from the user
input_text = input("Enter a sentence or paragraph: ")

# Removing punctuation and converting to lowercase
import string
input_text = input_text.translate(str.maketrans("", "", string.punctuation)).lower()

# Splitting the text into words
words = input_text.split()      You, 8 hours ago • more questions

# Creating a dictionary to store word frequencies
word_frequency = {}
print(word_frequency)
# Counting the frequency of each word
for word in words:
    if word in word_frequency:
        word_frequency[word] += 1
    else:
        word_frequency[word] = 1

# Displaying the result
print("\nWord Frequency Dictionary:")
for word, frequency in word_frequency.items():
    print(f"{word}: {frequency}")
```

Output:

```
Enter a sentence or paragraph: this is a bunch of letters
{}

Word Frequency Dictionary:
this: 1
is: 1
a: 1
bunch: 1
of: 1
letters: 1
```

Program 2:

Repeatedly ask the user to enter a team name and how many games the team has won and how many they have lost. Store this information in a dictionary where the keys are the team names and the values are the list of the form [wins, losses]
a. Using the dictionary, created above, allow the user to enter a team name and print out the team's winning percentage
b. Using the dictionary, create a list whose entries are the number of wins of each team
c. Using the dictionary, create a list of all those teams that have only winning record (i.e. zero losses)

Source Code:

```
team_records = {}  # Dictionary to store team records

while True:
    team_name = input("Enter a team name (or 'exit' to stop): ")

    if team_name.lower() == 'exit':
        break

    try:
        wins = int(input(f"Enter the number of wins for {team_name}: "))
        losses = int(input(f"Enter the number of losses for {team_name}: "))

        team_records[team_name] = [wins, losses]
    except ValueError:
        print("Invalid input. Please enter a valid number.")

# a. Print the winning percentage for a given team
selected_team = input("Enter a team name to check its winning percentage: ")
if selected_team in team_records:
    wins, losses = team_records[selected_team]
    winning_percentage = (wins / (wins + losses)) * 100 if wins + losses != 0 else 0
    print(f"{selected_team}'s Winning Percentage: {winning_percentage:.2f}%")
else:
    print(f"Team '{selected_team}' not found.")

# b. Create a list of the number of wins for each team
wins_list = [team_records[team][0] for team in team_records]

# c. Create a list of teams with only winning records
winning_teams = [team for team, record in team_records.items() if record[1] == 0]

# Displaying the results
print("\nList of Wins for Each Team:", wins_list)
print("Teams with Only Winning Records:", winning_teams)
```

Output:

```
Enter a team name (or 'exit' to stop): team1
Enter the number of wins for team1: 24
Enter the number of losses for team1: 0
Enter a team name (or 'exit' to stop): team2
Enter the number of wins for team2: 44
Enter the number of losses for team2: 45
Enter a team name (or 'exit' to stop): exit
Enter a team name to check its winning percentage: team1
team1's Winning Percentage: 100.00%

List of Wins for Each Team: [24, 44]
Teams with Only Winning Records: ['team1']
```

Program 3:

Write a program to repeatedly ask the user to enter product names and prices. Store all of these in a dictionary whose keys are the product names and whose values are the prices. ****when the user is done entering products and prices, allow them to repeatedly enter a product name and print the corresponding price or print a message product does not exist

Source Code:

```python
# Dictionary to store product names and prices
products = {}

# Repeatedly ask the user to enter product names and prices
while True:
    product_name = input("Enter a product name (or 'done' to stop): ")

    if product_name.lower() == 'done':
        break

    try:
        product_price = float(input(f"Enter the price for {product_name}: "))
        products[product_name] = product_price
    except ValueError:
        print("Invalid input. Please enter a valid price.")

# Allow the user to check prices for entered products
while True:
    query_product = input("Enter a product name to check its price (or 'exit' to stop): ")

    if query_product.lower() == 'exit':
        break

    if query_product in products:
        print(f"The price for {query_product} is ${products[query_product]:.2f}")
    else:
        print(f"Product '{query_product}' does not exist.")
```

Output:

```
Enter a product name (or 'done' to stop): banana
Enter the price for banana: 23
Enter a product name (or 'done' to stop): bottle
Enter the price for bottle: 34
Enter a product name (or 'done' to stop): cap
Enter the price for cap: 34
Enter a product name (or 'done' to stop): oats
Enter the price for oats: 43
Enter a product name (or 'done' to stop): done
Enter a product name to check its price (or 'exit' to stop): cap
The price for cap is $34.00
```

Program 4:

Given the dictionary X={'11A':"Uzma",'11B':"Sonia",'11C':"Doyel",'11D':"Sreeja"} Create a dictionary with the opposite mapping, i.e. write a program to create the dictionary as: Inverted_X= {"Uzma":'11A',"Sonia":'11B',"Doyel":'11C',"Sreeja":'11D'}

Source Code:

```
X = {'11A': "Uzma", '11B': "Sonia", '11C': "Doyel", '11D': "Sreeja"}
Inverted_X = {value: key for key, value in X.items()}
print("Original Dictionary:", X, "\nInverted Dictionary:", Inverted_X)
```

Output:

```
Original Dictionary: {'11A': 'Uzma', '11B': 'Sonia', '11C': 'Doyel', '11D': 'Sreeja'}
Inverted Dictionary: {'Uzma': '11A', 'Sonia': '11B', 'Doyel': '11C', 'Sreeja': '11D'}
```

Program 5:

Given two dictionaries say D1 and D2. Write a program that lists the overlapping keys of the two dictionaries i.e. if a key of D1 is also present in D2, then list it

Source Code:

```
# Sample dictionaries
D1 = {'a': 1, 'b': 2, 'c': 3}
D2 = {'b': 4, 'c': 5, 'd': 6}

# Find overlapping keys
overlapping_keys = set(D1.keys()).intersection(D2.keys())

# Display the result
print("Overlapping Keys:", list(overlapping_keys))
```

Output:

```
Overlapping Keys: ['c', 'b']
```

Program 6:

Create a dictionary whose keys are the names of the month and whose values are the number of days in the month
a. Printout all of the keys in alphabetical order
b. Print out the key –value pair sorted by the no of days in a month
c. Print out all the months with 31 days

Source Code:

```
# Lists for month names and corresponding number of days
month_names = ['January', 'February', 'March', 'April', 'May', 'June',
               'July', 'August', 'September', 'October', 'November', 'December']

days_in_month = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

# Create the dictionary using the two lists
months_days = dict(zip(month_names, days_in_month))

# a. Print all keys in alphabetical order
print("a. All months in alphabetical order:")
for month in sorted(months_days.keys()):
    print(month)

# b. Print key-value pairs sorted by the number of days
print("\nb. Months and their respective days sorted by the number of days:")
sorted_by_days = sorted(months_days.items(), key=lambda x: x[1])
for month, days in sorted_by_days:
    print(f"{month}: {days} days")

# c. Print all months with 31 days
print("\nc. Months with 31 days:")
for month, days in months_days.items():
    if days == 31:
        print(month)
```

Output:

```
a. All months in alphabetical order:
April
August
December
February
January
July
June
March
May
November
October
September

b. Months and their respective days sorted by the number of days:
February: 28 days
April: 30 days
June: 30 days
September: 30 days
November: 30 days
January: 31 days
March: 31 days
May: 31 days
July: 31 days
August: 31 days
October: 31 days
December: 31 days

c. Months with 31 days:
January
March
May
July
August
October
December
```