

A.Yuvan Charan

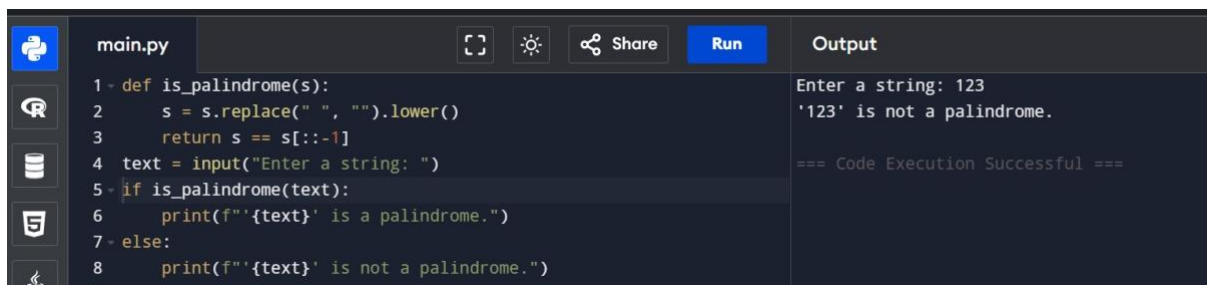
192424383

SLOT-B

PYTHON PROGRAMMING FOR BLOCK CHAIN PROJECTS

CSA0815

1. Write a program to check whether a string is a palindrome or not



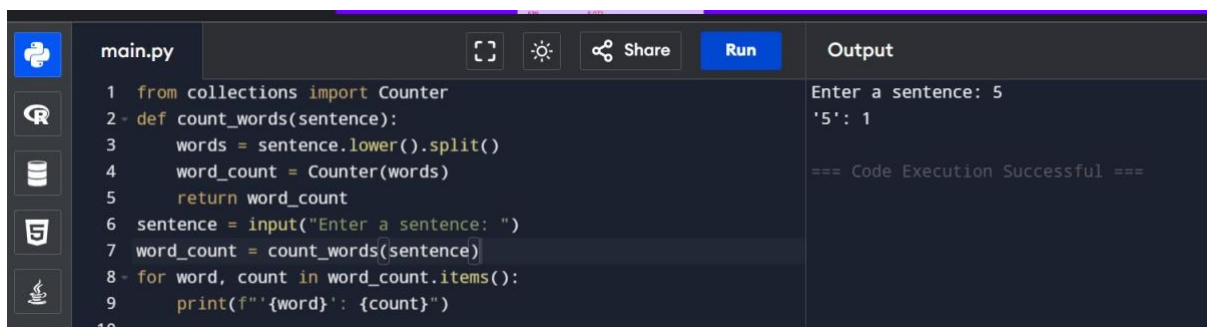
```
main.py
1 def is_palindrome(s):
2     s = s.replace(" ", "").lower()
3     return s == s[::-1]
4 text = input("Enter a string: ")
5 if is_palindrome(text):
6     print(f'{text} is a palindrome.')
7 else:
8     print(f'{text} is not a palindrome.')
9
```

Output

```
Enter a string: 123
'123' is not a palindrome.

=== Code Execution Successful ===
```

2. Write a program to count the occurrences of each word in a given sentence



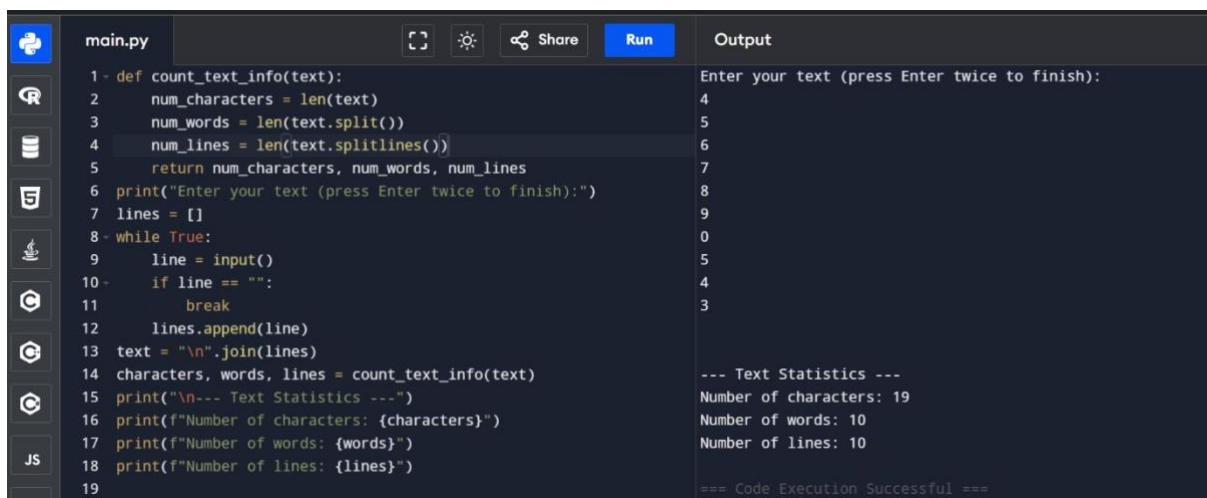
```
main.py
1 from collections import Counter
2 def count_words(sentence):
3     words = sentence.lower().split()
4     word_count = Counter(words)
5     return word_count
6 sentence = input("Enter a sentence: ")
7 word_count = count_words(sentence)
8 for word, count in word_count.items():
9     print(f'{word}: {count}')
10
```

Output

```
Enter a sentence: 5
'5': 1

=== Code Execution Successful ===
```

3. Write a program to count number of characters, words, and lines in the given string



```
main.py
1 def count_text_info(text):
2     num_characters = len(text)
3     num_words = len(text.split())
4     num_lines = len(text.splitlines())
5     return num_characters, num_words, num_lines
6 print("Enter your text (press Enter twice to finish):")
7 lines = []
8 while True:
9     line = input()
10    if line == "":
11        break
12    lines.append(line)
13 text = "\n".join(lines)
14 characters, words, lines = count_text_info(text)
15 print("\n--- Text Statistics ---")
16 print(f"Number of characters: {characters}")
17 print(f"Number of words: {words}")
18 print(f"Number of lines: {lines}")
19
```

Output

```
Enter your text (press Enter twice to finish):
4
5
6
7
8
9
0
5
4
3

--- Text Statistics ---
Number of characters: 19
Number of words: 10
Number of lines: 10

=== Code Execution Successful ===
```

4. Maximum Number of Words Found in Sentence A sentence is a list of words that are separated by a single space with no leading or trailing

spaces. You are given a list of strings sentences, where each sentences[i] represents a single sentence. Write a python program to return the maximum number of words that appear in a single sentence

### Test Cases:

1. Input: sentences = ["alice and bob love apple", "i think so too", "this is great thanks very much"]

Output: 6

main.py	Output
<pre>1 sentences = ["alice and bob love apple", "i think so too", "this   is great thanks very much"] 2 max_words = max(len(sentence.split()) for sentence in sentences) 3 4 print("Maximum number of words in a sentence:", max_words) 5</pre>	<pre>Maximum number of words in a sentence: 6 === Code Execution Successful ===</pre>

2. Input: sentences = ["please wait", "continue to fight", "continue to win"]

Output: 3

main.py	Output
<pre>1 sentences = ["please wait", "continue to fight", "continue to   win"] 2 max_words = max(len(sentence.split()) for sentence in sentences) 3 4 print("Maximum number of words in a sentence:", max_words) 5</pre>	<pre>Maximum number of words in a sentence: 3 === Code Execution Successful ===</pre>

3. ["the heads", "of", "two", "sorted linked lists"]

The screenshot shows a Python IDE with a file named `main.py`. The code defines a `ListNode` class and a `mergeTwoLists` function. The `ListNode` class has an `__init__` method that takes `val` and `next` as arguments. The `mergeTwoLists` function takes two lists, `list1` and `list2`, and returns a merged list. The output shows "Code Execution Successful".

```

1 class ListNode:
2     def __init__(self, val=0, next=None):
3         self.val = val
4         self.next = next
5 def mergeTwoLists(list1, list2):
6     dummy = ListNode()
7     tail = dummy
8     while list1 and list2:
9         if list1.val < list2.val:
10            tail.next = list1
11            list1 = list1.next
12        else:
13            tail.next = list2
14            list2 = list2.next
15        tail = tail.next
16    tail.next = list1 if list1 else list2
17
18    return dummy.next

```

Output: `=== Code Execution Successful ===`

4. ["python", "is", "an object-oriented programming language"]

The screenshot shows a Python IDE with a file named `main.py`. The code creates a list `words` containing the strings "python", "is", and "an object-oriented programming language". It then joins these words into a single string `sentence` and prints it. The output shows "python is an object-oriented programming language" and "Code Execution Successful".

```

1 words = ["python", "is", "an object-oriented programming
2     language"]
3 sentence = " ".join(words)
4 print(sentence)

```

Output: `python is an object-oriented programming language`  
`=== Code Execution Successful ===`

5. ["python", "is", "an interactive language"]

The screenshot shows a Python IDE with a file named `main.py`. The code creates a list `phrases` containing the strings "python", "is", and "an interactive language". It then joins these phrases into a single string `sentence` and prints it. The output shows "python is an interactive language" and "Code Execution Successful".

```

1 phrases = ["python", "is", "an interactive language"]
2 sentence = " ".join(phrases)
3 print(sentence)
4

```

Output: `python is an interactive language`  
`=== Code Execution Successful ===`

5. Given an integer `x` as numeric data type. Write a python program to return true if `x` is palindrome integer.

An integer is a palindrome when it reads the same backward as forward.

For example, 121 is a palindrome while 123 is not.

Test cases:

1.Input: `x = 121`

Output: true

main.py		Output
<pre> 1- def is_palindrome(x): 2-     if x &lt; 0: 3-         return False 4-     return str(x) == str(x[::-1]) 5- x = int(input("Enter an integer: ")) 6- if is_palindrome(x): 7-     print(f"{x} is a palindrome.") 8- else: 9-     print(f"{x} is not a palindrome.") </pre>	<pre> Enter an integer: 123 123 is not a palindrome.  === Code Execution Successful === </pre>	

6. Write a python program to print the numbers from M to N by skipping K numbers in between? Get the input using list data type.

main.py		Output
<pre> 1 input_values = [50, 100, 7] 2 M, N, K = input_values 3 print(f"Numbers from {M} to {N} skipping {K} numbers in between:") 4 for i in range(M, N + 1, K + 1): 5     print(i, end=" ") 6 </pre>	<pre> Numbers from 50 to 100 skipping 7 numbers in between: 50 58 66 74 82 90 98  === Code Execution Successful === </pre>	

7. Most years have 365 days. However, the time required for the Earth to orbit the Sun is actually slightly more than that. As a result, an extra day, February 29, is included in some years to correct for this difference. Such years are referred to as leap years.

The rules for determining whether or not a year is a leap year follow:

Any year that is divisible by 400 is a leap year.

Sample Input:


Enter Date: 1947

main.py		Output
<pre> 1 year = int(input("Enter Date: ")) 2 if (year % 400 == 0) or (year % 4 == 0 and year % 100 != 0): 3     print(f"{year} is a Leap Year.") 4 else: 5     print(f"{year} is Not a Leap Year.") 6 </pre>	<pre> Enter Date: 1947 1947 is Not a Leap Year.  === Code Execution Successful === </pre>	




8. Write a python program to print the following pattern.

Sample Input:

Number of rows: 5



main.py

 Share

Run

```
1 num_rows = int(input("Number of rows: "))
2 for i in range(1, num_rows + 1):
3     for j in range(1, i + 1):
4         print(j, end=" ")
5     print()
6
```

Output

Number of rows: 5  
1  
1 2  
1 2 3  
1 2 3 4  
1 2 3 4 5  
  
=== Code Execution Successful ===