1:- write a code reverse python code

```python
def reverse_string(s):
    return s[::-1]
print(reverse_string("hello"))
```

    olleh

write a code to count the number of vowels in a string

```python
def count_vowels(s):
    vowels = "aeiouAEIOU"
    return sum(1 for char in s if char in vowels)

# Example usage:
print(count_vowels("hello"))  # Output: 2
```

    2

write a code to check if a given string in a palindrome or not

```python
def is_palindrome(s):
    return s == s[::-1]

# Example usage:
print(is_palindrome("radar"))  # Output: True
print(is_palindrome("hello"))  # Output: False
```

    True
    False

write a code to check if two given strings are anagrams of each other

```python
def are_anagrams(str1, str2):
    return sorted(str1) == sorted(str2)
```

```
# Example usage:
print(are_anagrams("listen", "silent"))  # Output: True
print(are_anagrams("hello", "world"))   # Output: False
```

```
True
False
```

write a code to find all occurence of a given substring within another string

```python
def find_all_occurrences(s, sub):
    start = 0
    while True:
        start = s.find(sub, start)
        if start == -1:
            return
        yield start
        start += len(sub)

# Example usage:
print(list(find_all_occurrences("banana", "ana")))  # Output: [1, 3]
```

```
[1]
```

Write a code to perform basic string compression using the counts of repeated character.

```python
def compress_string(s):
    if not s:
        return ""
    compressed = []
    count = 1
    for i in range(1, len(s)):
        if s[i] == s[i - 1]:
            count += 1
        else:
            compressed.append(s[i - 1] + str(count))
            count = 1
    compressed.append(s[-1] + str(count))
    return ''.join(compressed)
```

```
# Example usage:
print(compress_string("aaabbccc"))  # Output: "a3b2c3"
```

    a3b2c3

Write a code to determine if a string


```python
def has_unique_characters(s):
    return len(s) == len(set(s))

# Example usage:
print(has_unique_characters("abcdef"))  # Output: True
print(has_unique_characters("aabbcc"))  # Output: False
```

    True
    False

write a code to convert a given string to uppercase or lowercase


```python
def to_uppercase(s):
    return s.upper()

def to_lowercase(s):
    return s.lower()

# Example usage:
print(to_uppercase("hello"))  # Output: "HELLO"
print(to_lowercase("HELLO"))  # Output: "hello"
```

    HELLO
    hello

write a code to count the number of words in a string


```python
def count_words(s):
    return len(s.split())
```

```
# Example usage:
print(count_words("hello world"))  # Output: 2
```

    2

write code to concatenate two string without the + operator


```python
def concatenate_strings(str1, str2):
    return f"{str1}{str2}"

# Example usage:
print(concatenate_strings("hello", "world"))
```

    helloworld

write a code to remove all occurence of a specific element from a list


```python
def remove_element(lst, element):
    return [x for x in lst if x != element]

# Example usage:
print(remove_element([1, 2, 2, 3, 4], 2))  # Output: [1, 3, 4]
```

    [1, 3, 4]


```python

```



```python
def second_largest(nums):
    first, second = float('-inf'), float('-inf')
    for n in nums:
        if n > first:
            second = first
            first = n
        elif first > n > second:
```

```
            second = n
    return second

# Example usage:
print(second_largest([1, 2, 3, 4]))  # Output: 3
```

    3


```python


```



```python
def count_occurrences(lst):
    counts = {}
    for item in lst:
        if item in counts:
            counts[item] += 1
        else:
            counts[item] = 1
    return counts

# Example usage:
print(count_occurrences([1, 2, 2, 3, 3, 3]))  # Output: {1: 1, 2: 2, 3: 3}
```

    {1: 1, 2: 2, 3: 3}


```python
def reverse_list(lst):
    left, right = 0, len(lst) - 1
    while left < right:
        lst[left], lst[right] = lst[right], lst[left]
        left += 1
        right -= 1

# Example usage:
lst = [1, 2, 3, 4]
reverse_list(lst)
print(lst)  # Output: [4, 3, 2, 1]
```

```
    [4, 3, 2, 1]
```

```python
```

```python
def remove_duplicates(lst):
    seen = set()
    result = []
    for item in lst:
        if item not in seen:
            seen.add(item)
            result.append(item)
    return result

# Example usage:
print(remove_duplicates([1, 2, 2, 3, 4, 4, 5]))
```

```
    [1, 2, 3, 4, 5]
```

```python
```

```python
def is_sorted(lst):
    return lst == sorted(lst) or lst == sorted(lst, reverse=True)

# Example usage:
print(is_sorted([1, 2, 3, 4]))  # Output: True
print(is_sorted([4, 3, 2, 1]))  # Output: True
print(is_sorted([1, 3, 2, 4]))  # Output: False
```

```
    True
    True
    False
```

```python
```

```python
def merge_sorted_lists(lst1, lst2):
    result = []
    i = j = 0
    while i < len(lst1) and j < len(lst2):
        if lst1[i] < lst2[j]:
            result.append(lst1[i])
            i += 1
        else:
            result.append(lst2[j])
            j += 1
    result.extend(lst1[i:])
    result.extend(lst2[j:])
    return result

# Example usage:
print(merge_sorted_lists([1, 3, 5], [2, 4, 6]))  # Output: [1, 2, 3, 4, 5, 6]
```

    [1, 2, 3, 4, 5, 6]

```python

```

```python
def list_intersection(lst1, lst2):
    return list(set(lst1) & set(lst2))

# Example usage:
print(list_intersection([1, 2, 3], [2, 3, 4]))  # Output: [2, 3]
```

    [2, 3]

```python

```

```python
def difference_of_sets():
    set1 = set(input("Enter elements of the first set separated by spaces:
").split())
```

```
    set2 = set(input("Enter elements of the second set separated by spaces:
").split())
    difference = set1 - set2
    print("Elements in the first set but not in the second set:", difference)

# Example usage:
difference_of_sets()
```

    Enter elements of the first set separated by spaces: 45
    Enter elements of the second set separated by spaces: 78
    Elements in the first set but not in the second set: {'45'}

```python

```

```python
def elements_in_range(tup, start, end):
    return tup[start:end+1]

# Example usage:
sample_tuple = (1, 2, 3, 4, 5, 6)
print(elements_in_range(sample_tuple, 2, 4))  # Output: (3, 4, 5)
```

    (3, 4, 5)

```python
def union_of_sets():
    set1 = set(input("Enter elements of the first set separated by spaces:
").split())
    set2 = set(input("Enter elements of the second set separated by spaces:
").split())
    union = set1 | set2
    print("Union of the two sets:", union)

# Example usage:
union_of_sets()
```

    Enter elements of the first set separated by spaces: 44
    Enter elements of the second set separated by spaces: 44
    Union of the two sets: {'44'}

```python

```


```python
def min_max_values(tup):
    return min(tup), max(tup)

# Example usage:
sample_tuple = (5, 1, 8, 3, 2)
print(min_max_values(sample_tuple))  # Output: (1, 8)
```

    (1, 8)


```python

```


```python
def set_operations():
    set1 = set(map(int, input("Enter elements of the first set separated by spaces: ").split()))
    set2 = set(map(int, input("Enter elements of the second set separated by spaces: ").split()))
    print("Union:", set1 | set2)
    print("Intersection:", set1 & set2)
    print("Difference (set1 - set2):", set1 - set2)
    print("Difference (set2 - set1):", set2 - set1)

# Example usage:
set_operations()
```

    Enter elements of the first set separated by spaces: 4
    Enter elements of the second set separated by spaces: 4
    Union: {4}
    Intersection: {4}
    Difference (set1 - set2): set()
    Difference (set2 - set1): set()

```python
def count_occurrences(tup, element):
    return tup.count(element)

# Example usage:
sample_tuple = (1, 2, 3, 2, 4, 2)
print(count_occurrences(sample_tuple, 2))  #
```

    3


```python

```



```python
def symmetric_difference_of_sets():
    set1 = set(input("Enter elements of the first set separated by spaces:
").split())
    set2 = set(input("Enter elements of the second set separated by spaces:
").split())
    symmetric_diff = set1 ^ set2
    print("Symmetric difference of the two sets:", symmetric_diff)

# Example usage:
symmetric_difference_of_sets()
```

    Enter elements of the first set separated by spaces: 6
    Enter elements of the second set separated by spaces: 8
    Symmetric difference of the two sets: {'8', '6'}


```python

```



```python
def word_frequencies(words):
    freq_dict = {}
    for word in words:
        if word in freq_dict:
            freq_dict[word] += 1
        else:
```

```
            freq_dict[word] = 1
    return freq_dict

# Example usage:
words_list = ["apple", "banana", "apple", "orange", "banana", "banana"]
print(word_frequencies(words_list))  # Output: {'apple': 2, 'banana': 3,
'orange': 1}
```

```
    {'apple': 2, 'banana': 3, 'orange': 1}
```

```python

```

```python
def merge_dictionaries(dict1, dict2):
    merged_dict = dict1.copy()
    for key, value in dict2.items():
        if key in merged_dict:
            merged_dict[key] += value
        else:
            merged_dict[key] = value
    return merged_dict

# Example usage:
dict1 = {'a': 1, 'b': 2, 'c': 3}
dict2 = {'b': 3, 'c': 4, 'd': 5}
print(merge_dictionaries(dict1, dict2))  # Output: {'a': 1, 'b': 5, 'c': 7,
'd': 5}
```

```
    {'a': 1, 'b': 5, 'c': 7, 'd': 5}
```

```python
def access_nested_dict(nested_dict, keys):
    current = nested_dict
    for key in keys:
        if key in current:
            current = current[key]
        else:
            return None
    return current
```

```
# Example usage:
nested_dict = {'a': {'b': {'c': 42}}}
keys = ['a', 'b', 'c']
print(access_nested_dict(nested_dict, keys))  # Output: 42
keys = ['a', 'b', 'd']
print(access_nested_dict(nested_dict, keys))  # Output: None
```

```
    42
    None
```

```python
def invert_dictionary(d):
    inverted = {}
    for key, value in d.items():
        if value in inverted:
            inverted[value].append(key)
        else:
            inverted[value] = [key]
    return inverted

# Example usage:
sample_dict = {'a': 1, 'b': 2, 'c': 1}
print(invert_dictionary(sample_dict))  # Output: {1: ['a', 'c'], 2: ['b']}
```

```
    {1: ['a', 'c'], 2: ['b']}
```

```python
```