```python
#1 Write a code to reverse a string
s1 = "train"
str = s1[::-1]
print(str)

#2 Write a code to count the number of vowels in a string
s2 = "train"
h = 0
for i in s2:
    if i == 'a' or i == 'e' or i == 'i' or i == 'o' or i == 'u':
        h = h+1
print(h)

#3 Write a code to check if a given string is a palindrome or not
s3 = "ueanaeu"
print(s3 == s3[::-1])

#4 Write a code to check if two given strings are anagrams of each other
s4a = "bored"
s4b = "robed"
print(sorted(s4a) == sorted(s4b))


#5 Write a code to find all occurrences of a given substring within another string
s5a = "anghjantnjanjgh"
s5b = "an"
occurance_index = []
#for i in range(len(s5a)):
for i in range(len(s5a)):
    if s5a[i:i+len(s5b)] == s5b:
        occurance_index.append(i)
print(occurance_index)

#6 Write a code to perform basic string compression using the counts of repeated characters
s6a = "aabbccccaad"
s6ans = ""
c = 1
for i in range(1,len(s6a)):
    if s6a[i] != s6a[i-1]:
        s6ans = s6ans+s6a[i-1]
        s6ans = s6ans+f"{c}"
        c = 1
    else:
        c = c+1
s6ans = s6ans+s6a[len(s6a)-1]
s6ans = s6ans+f"{c}"
print(s6ans)

#7 Write a code to determine if a string has all unique characters
is_unique = True
s7 = "abcdef12@@"
for i in range(0,len(s7)-1):
    if s7[i] == s7[i+1]:
        is_unique = False
print(is_unique)

#8 Write a code to convert a given string to uppercase or lowercase
s8a = "anshuman"
s8b = "ANSHUMAN"
print(s8a.upper())
print(s8a.lower())

#9 Write a code to count the number of words in a string
s9 = "hi, my name is anshuman mishra"
print(len(s9.split(" ")))

#10 Write a code to concatenate two strings without using the + operator
s10a = "Hi my name is"
s10b = "Anshuman mishra"
print(f"{s10a} {s10b}")

#11 Write a code to remove all occurrences of a specific element from a list
l1 = [1,2,3,2,4,5,6,4,2,2,6,7,8]
rem = 2
i = 0
while i<len(l1):
    if l1[i] == rem:
        l1.pop(i)
    else:
        i = i+1
print(l1)

#12 Implement a code to find the second largest number in a given list of integers
l2 = [2,1,3,56,7,5,4,33,6,6,67,8,9]
max1 = -99999
max2 = -99999
for i in l2:
    if i>max1:
        max1 = i
for i in l2:
    if i>max2 and i<max1:
        max2 = i
print(max2)

#13 Create a code to count the occurrences of each element in a list and return a dictionary with elements as keys and their counts as values
l3 = [1,2,4,5,4,2,3,4,5,1,4,5,3,4,1,2,5]
l3_sort = sorted(l3)
l3_element = []
l3_count = []
c = 1
for i in range(1,len(l3_sort)):
    if l3_sort[i] != l3_sort[i-1]:
        l3_element.append(l3_sort[i-1])
        l3_count.append(c)
        c = 1
    else:
        c = c+1
l3_element.append(l3_sort[i])
l3_count.append(c)
dict_count = {}
for ele, ct in zip(l3_element,l3_count):
    dict_count[ele] = ct
print(dict_count)

#14 Write a code to reverse a list in-place without using any built-in reverse functions
l4 = [1,2,3,4,5]
print(l4[::-1])

#15 Implement a code to find and remove duplicates from a list while preserving the original order of elements
l5 = [1,2,1,3,4,5,1,4,6,6,7,8]
```

```python
print(l5.count(1))
i = len(l5)-1
while i>0:
    if l5.count(l5[i])>1:
        l5.pop(i)
    else:
        i = i-1
print(l5)

#16 Create a code to check if a given list is sorted (either in ascending or descending order) or not
l6 = [6,5,4,3,2,1,]
if l6 == sorted(l6) or l6[::-1] == sorted(l6):
    print("List is sorted in asccending or descending order")
else:
    print("List is not sorted")

#17 Write a code to merge two sorted lists into a single sorted list
l7a = [1,3,5,7,9]
l7b = [2,4,6,8]
l7_sorted = []
i = 0
j = 0
while i<len(l7a) and j<len(l7b):
    if l7a[i]<=l7b[j]:
        l7_sorted.append(l7a[i])
        i = i+1
    else:
        l7_sorted.append(l7b[j])
        j = j+1
if i<len(l7a):
    while i<len(l7a):
        l7_sorted.append(l7a[i])
        i = i+1
else:
    while j<len(l7b):
        l7_sorted.append(l7b[j])
        j = j+1
print(l7_sorted)

#18 Implement a code to find the intersection of two given lists
l8a = [1,2,3,4,5]
l8b = [3,4,5,6,7,8]
l8_intersection = []
for i in l8a:
    if l8b.count(i) != 0:
        l8_intersection.append(i)
print(l8_intersection)

#19 Create a code to find the union of two lists without duplicates
l9a = [1,2,3,4,5]
l9b = [3,4,5,6,7,8]
l9_union = l9a
for i in l9b:
    if l9a.count(i) == 0:
        l9_union.append(i)
print(l9_union)

#20 Write a code to shuffle a given list randomly without using any built-in shuffle functions
import random
l10 = [1,2,3,4,4,5,6,7,9]
for i in range(len(l10)):
    rand = random.randint(0,len(l10)-1)
    temp = l10[i]
    l10[i] = l10[rand]
    l10[rand] = temp
print(l10)

#21 Write a code that takes two tuples as input and returns a new tuple containing elements that are common to both input tuples
t1 = (1,2,3,4,5)
t2 = (4,5,6,7,8,9)
def intersection_of_tuple(t1a = (),t1b = ()):
    t1_new_list = []
    for i in range(len(t1a)):
        if t1b.count(t1a[i])>0:
            t1_new_list.append(t1a[i])
    t1_new_tuple = tuple(t1_new_list)
    return t1_new_tuple
print(intersection_of_tuple(t1,t2))

#22 Create a code that prompts the user to enter two sets of integers separated by commas. Then, print the intersection of these two sets
# s1 = input("Enter first set of integers seperated by commas:")
# s2 = input("Enter second set of integers seperated by commas:")
# st1 = {'dummy1'}
# st2 = {'dummy2'}
# st1.update(s1.split(','))
# st2.update(s2.split(','))
# print(st1&st2)

#23 Write a code to concatenate two tuples. The function should take two tuples as input and return a new tuple containing elements from both input tuples.
def concetenated_tuples(t1 = (),t2 = ()):
    return t1+t2
t3a = (1,2,3)
t3b = (3,4,5)
print(concetenated_tuples(t3a,t3b))

#24 Develop a code that prompts the user to input two sets of strings. Then, print the elements that are present in the first set but not in the second set
# s1 = input("Enter first set of spaced strings:")
# s2 = input("Enter second set of spaced strings:")
# st1 = {'dummy1'}
# st2 = {'dummy2'}
# st1.update(s1.split(' '))
# st2.update(s2.split(' '))
# st_new = st1-st2
# st_new.discard('dummy1')
# st_new.discard('dummy2')
# print(st_new)

#25 Create a code that takes a tuple and two integers as input. The function should return a new tuple containing elements from the original tuple within the specified range of indices
def specified_tuple(t = (),a = 0,b = 0):
    return t[a:b]
t5 = (1,2,3,4,5,6)
print(specified_tuple(t5,1,4))

#26 Write a code that prompts the user to input two sets of characters. Then, print the union of these two sets
# s1 = input("Enter first set of characters seperated by commas:")
# s2 = input("Enter second set of chracters seperated by commas:")
# st1 = {'dummy1'}
# st2 = {'dummy2'}
# st1.update(s1.split(','))
# st2.update(s2.split(','))
```

```
# st_new = st1|st2
# st_new.discard('dummy1')
# st_new.discard('dummy2')
# print(st_new)

#27 Develop a code that takes a tuple of integers as input. The function should return the maximum and minimum values from the tuple using tuple unpacking
def max_min_tuple(t = ()):
    a = 99999
    b = -99999
    for i in t:
        if i<a:
            a = i
        if i>b:
            b = i
    return [a,b]
print(max_min_tuple((1,32,-13,-12,56,78,9,0,5)))

#28 Create a code that defines two sets of integers. Then, print the union, intersection, and difference of these two sets
set1 = {1,2,3,4,5,6}
set2 = {4,5,6,7,8,9}
#Intersection
print(set1&set2)
#Union
print(set1|set2)
#Difference
print(set1-set2)
print(set2-set1)

#29 Write a code that takes a tuple and an element as input. The function should return the count of occurrences of the given element in the tuple
def occurance_tuple(t = (),a = 0):
    return list(t).count(a)
print(occurance_tuple((1,2,3,1,2,4,4,5,6,4,7,6,8,9),4))

#30 Develop a code that prompts the user to input two sets of strings. Then, print the symmetric difference of these two sets
# s1 = input("Enter first set of spaced strings:")
# s2 = input("Enter second set of spaced strings:")
# st1 = {'dummy1'}
# st2 = {'dummy2'}
# st1.update(s1.split(' '))
# st2.update(s2.split(' '))
# st_new = st1^st2
# st_new.discard('dummy1')
# st_new.discard('dummy2')
# print(st_new)

#31 Write a code that takes a list of words as input and returns a dictionary where the keys are unique words and the values are the frequencies of those words in the input list
def string_freq(list_words = []):
    list_words_sort = sorted(list_words)
    list_word_count = []
    list_counts = []
    c = 1
    for i in range(1,len(list_words_sort)):
        if list_words_sort[i] != list_words_sort[i-1]:
            list_word_count.append(list_words_sort[i-1])
            list_counts.append(c)
            c = 1
        else:
            c = c+1
    list_word_count.append(list_words_sort[i])
    list_counts.append(c)
    dict_word_count = {}
    for wrd, ct in zip(list_word_count,list_counts):
        dict_word_count[wrd] = ct
    return dict_word_count
listofword = ["hi","hi","anshuman","namaste","hi","anshuman"]
print(string_freq(listofword))

#32 Write a code that takes two dictionaries as input and merges them into a single dictionary. If there are common keys, the values should be added together
def merge_dict(d1 = {},d2 = {}):
    merged_keys = []
    merged_value = []
    for i in range(len(list(d1.keys()))):
        if list(d2.keys()).count(list(d1.keys())[i]) != 0:
            merged_keys.append(list(d1.keys())[i])
            merged_value.append(list(d1.values())[i]+list(d2.values())[list(d2.keys()).index(list(d1.keys())[i])])
        else:
            merged_keys.append(list(d1.keys())[i])
            merged_value.append(list(d1.values())[i])
    for i in range(len(list(d2.keys()))):
        if list(d1.keys()).count(list(d2.keys())[i]) == 0:
            merged_keys.append(list(d2.keys())[i])
            merged_value.append(list(d2.values())[i])
    mrg_dict = {}
    for key, value in zip(merged_keys,merged_value):
        mrg_dict[key] = value
    return mrg_dict
dict1 = {'a':5,'b':3,'c':2}
dict2 = {'b':2,'c':2,'d':8}
print(merge_dict(dict1,dict2))

#33 Write a code to access a value in a nested dictionary. The function should take the dictionary and a list of keys as input, and return the corresponding value. If any of the keys d
def check_value(dct1 = {},listt_key = []):
    dct_key = []
    dct_val = []
    for i in range(len(list(dct1.keys()))):
        dct_key.append(list(dct1.keys())[i])
        dct_val.append(list(dct1.values())[i])
        if type(list(dct1.values())[i]) == dict:
            dct2 = list(dct1.values())[i]
            for j in range(len(list(dct2.keys()))):
                dct_key.append(list(dct2.keys())[j])
                dct_val.append(list(dct2.values())[j])
    ans_list = []
    for i in listt_key:
        if (i in dct_key) == True:
            ans_list.append(dct_val[dct_key.index(i)])
        else:
            ans_list.append("None")
    return ans_list
print(check_value({'a':1,'b':4,'c':8,'d':{'e':3,'f':4},'g':5},['a','d','e','g','h']))

#34 Write a code that takes a dictionary as input and returns a sorted version of it based on the values. You can choose whether to sort in ascending or descending order
def sorted_dct(dct3 = {}):
    lst_sort = sorted(list(dct3.values()))
    key_sort = []
    for i in range(len(lst_sort)):
        key_sort.append(list(dct3.keys())[list(dct3.values()).index(lst_sort[i])])
    dct_sort = {}
    for k, v in zip(key_sort,lst_sort):
        dct_sort[k] = v
```

```
    return dct_sort
print(sorted_dct({'a':3,'b':1,'c':2}))

#35 Write a code that inverts a dictionary, swapping keys and values. Ensure that the inverted dictionary correctly handles cases where multiple keys have the same value by storing the
def sorted_rev(dct4 = {}):
    val_rev = []
    key_rev = []
    f = len(list(dct4.keys()))-1
    while f>=0:
        val_rev.append(list(dct4.values())[f])
        key_rev.append(list(dct4.keys())[f])
        f = f - 1
    dct_rev = {}
    for ke, va in zip(key_rev,val_rev):
        dct_rev[ke] = va
    return dct_rev
print(sorted_rev({'a':3,'b':1,'c':2}))
```