```
#1)write a code to reverse a string
input_string = "hello world"
reversed_string = input_string[::-1]
print(reversed_string)
```

```
dlrow olleh
```

```
#2)write a code to count number of vowels in string
input_string = "hello world"
vowels = "aeiouAEIOU"
count = 0
for char in input_string:
    if char in vowels:
        count += 1
print(count)
```

```
3
```

```
#3)write a code to check if a given string is a palindrome or not
input_string="a man,a plan,a canal,panama"
if is_palindrome(input_string):
  print(f'"{input_string}" is a palindrome')
else:
  print(f'"{input_string}" is not a palindrome
```

```
  File "<ipython-input-3-e8fc15351d09>", line 6
    print(f'"{input_string}" is not a palindrome
          ^
SyntaxError: unterminated string literal (detected at line 6)
```

```
#4)write a code to check if two given strings are anagrams of each other
def is_anagram(str1, str2):
  str1 = str1.replace(" ", "").lower()
  str2 = str2.replace(" ", "").lower()
  return sorted(str1) == sorted(str2)
```

```
#5)write a code to find all occurences of a given sub string within another string
def find_all_occurrences(string, substring):
  start=0
  occurrences=[]
  while True:
    start=string.find(substring,start)
    if start==-1:
      break
    occurrences.append(start)
    start+=1
  return occurrences
```

```
#6)write a code to perform basic string compression using the counts of repeated characters
 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)
```

```
  File "<ipython-input-2-f5542b95c403>", line 2
    def compress_string(s):
    ^
IndentationError: unexpected indent
```

```
#7)write a code to determine if a string has all unique characters
def has_all_unique_characters(s):
  seen=set()
  for char in s:
    if char in seen:
      return False# duplicate character found
      seen.add(char)
      return True # all character are unique
```

```
input_string="abcdefg"
if has_all_unique_characters(input_string):
  print(f'"{input_string}" has all unique characters')
else:
  print(f'"{input_string}" does not have all unique characters.'
```

```
  File "<ipython-input-5-b06b58a88bac>", line 5
    print(f'"{input_string}" does not have all unique characters.'
                                                                  ^
SyntaxError: incomplete input
```

```
#8)write a code to convert a given string to uppercase or lowe case
input_string="hello,world!"
uppercase_string=input_string.upper()
lowercase_string=input_string.lower()
print(uppercase_string)
print(lowercase_string)
```

```
HELLO,WORLD!
hello,world!
```

```
#9)write a code to count numbers number of words in string
input_string="hello,world!"
word_count=len(input_string.split())
print(word_count)
```

```
1
```

```
#10)write a code to concatenate two strings without using the + operator
  # example
string1="hello"
string2="world"
result=concatenate_strings(string1,string2)
print(result)
```

```
helloworld
```

```
#11)write a code to remove all occurences of a specific elements from a list
my_list=[1,2,3,4,2,5,2]
element_to_remove=2
my_list=[x for x in my_list if x!=element_to_remove]
print(my_list)
```

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

```
#12)implement a code to find the second largest number in a given list of integers
def second_largest(input_list)
unique_numbers=list(set(input_list))
unique_numbers.sort()
#check if there are at least two unique numbers
if len(unique_numbers)<2:
  return none
  return unique_numbers[-2]
```

```
  File "<ipython-input-15-f2c88a3c5501>", line 2
    def second_largest(input_list)
                                  ^
SyntaxError: expected ':'
```

```
#example usage
numbers=[3,1,4,4,2,5]
result=second_largest(numbers)
print(result)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-18-272a9673bd8f> in <cell line: 3>()
      1 #example usage
      2 numbers=[3,1,4,4,2,5]
----> 3 result=second_largest(numbers)
      4 print(result)

NameError: name 'second_largest' is not defined
```

```
#13)create a code........
#using loop
def count_occurences(input_list):
  count_dict={}

  for item in input_list:
    if item in count_dict:
      count_dict[item]+=1
    else:
      count_dict[item]=1
      return count_dict


#examples
my_list=[1,2,3,2,4,1,5]
result=count_occurences(my_list)
print(result)
```

```
{1: 1}
```

```
#14)write a code to reverse a list in-place withount using any built-in reverse function
def reverse_list(input_list):
  left=0
  right=len(input_list)-1

  while left<right:
    input_list[left],input_list[right]=input_list[right],input_list[left]
    left+=1
    right-=1




#example
my_list=[1,2,3,4,5]
reverse_list(my_list)
print(my_list)
```

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

```
#15)implement a code to find and remove duplicates from a list while preserving the original order of element
def remove_duplicates(input_list):
  seen=set()
  result=[]

  for item in input_list:
    if item not in seen:
      seen.add(item)
      result.append(item)
      result.append(item)

      return result



#examples
my_list=[1,2,3,2,4,3,5,1]
result=remove_duplicates(my_list)
print(result)
```

```
[1, 1]
```

```
#16)create a code to check if a given list is sorted list(either in ascending or descending order) or not
def is_sorted(input_list):
  if not input_list:
    return True

    ascending=all(input_list[i]<=input_list[i+1]for i in range(len(input_list)-1))
    descending=all(input_list[i]>=input_list[i+1]for i in range(len(input_list)-1))
    return ascending or descending


#examples usage
list1=[1,2,3,4,5]
list2=[5,4,3,2,1]
list3=[1,3,2,4,5]
list4=[]

print(is_sorted(list1))
print(is_sorted(list2))
print(is_sorted(list3))
print(is_sorted(list4))
```

```
None
None
None
True
```

```
#17)write a code to merge two sorted lists into a single sorted list
def merge_sorted_lists(list1,list2):
  merged_list=[]
  i,j=0,0
  #compare elements from both list and merge them
  while i<len(list1)andj<len(list2):
    if list[i]<=list2[j]:
      merged_list.append(list1[i])
      i+=1
    else:
      merged_list.append(list2[j])
      j+=1

      while i<len(list1):
        merged_list.appens(list2[j])
        j+=1
        return merged_list
        #example usage
        list1=[1,3,5,7]
        list2=[2,4,6,8]
        result=merge_sorted_lists(list1,list2)
        print(result)
```

```
  File "<ipython-input-33-dfc3b7565a93>", line 6
    while i<len(list1)andj<len(list2):
                     ^
SyntaxError: invalid syntax
```

```
#18)implement a code to find the intersection of two given list
def intersection(list1,list2):
  set1=set(list1)
  set2=set(list2)
  common_elements=set1.intersection(set2)
  return list(common_elements)



#example usage
list1=[1,2,3,4,5]
list2=[4,5,6,7,8]
result=intersection(list1,list2)
print(result)
```

```
[4, 5]
```

```
#19)create a code to find the union of two lists without duplicate
def union(list1,list2):
  combined_set=set(list1)|set(list2)
  return list(combined_set)
#example usage
list1=[1,2,3,4,5]
list2=[4,5,6,7,8]
result=union(list1,list2)
print(result)
```

```
[1, 2, 3, 4, 5, 6, 7, 8]
```

```
#20)write a code to shuffle a given  list randomly without using built-in shuffle function
def shuffle_list(input_list):
  shuffled_list=input_list[:]
  for i in range (len(shuffled_list)-1,0,-1):
    j=random.randint(0,i)
    shuffled_list[i],shuffled_list[j]=shuffled_list[j],shuffled_list[i]
    return shuffled_list
```

```
#21)write a code that takes two tuples as input and returns a new tuple containing element that are common to both input tuples
def common_elements(tuple1,tuple2):
  common_set=set(tuple1)&set(tuple2)
  return tuple(common_set)
  #example
   tuple1=(1,2,3,4,5)
   tuple2=(4,5,6,7,8)
   result=common_elements(tuple1,tuple2)
   print(result)
```

```
  File "<ipython-input-42-8316a56cddb6>", line 6
    tuple1=(1,2,3,4,5)
    ^
IndentationError: unexpected indent
```

```
#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
#function to get integer sets from user input
def get_integer_set(prompt):
  user_input=input(prompt)
  return set(map(int,user_input.split(',')))
#get two sets from the user
set1=get_integer_set("Enter the first set of integers separated by commas:")
set2=get_integer_set("Enter the second set of integers separated by commas:")
#calculate the intersection
intersection=set1.intersection(set2)
#print the intersection
print("Intersection of the two sets:",intersection)
```

```
Enter the first set of integers separated by commas:1,2,3
Enter the second set of integers separated by commas:4,5,8
Intersection of the two sets: set()
```

```
#23)write a code to concatenate two tuples the functions should take two tuples as input and return a new tuple containing elements from
def concatenate_tuples(tuple1,tuple2):
  """
  concatenate two tuples and return a new tuple

  parameters:
  tuple1(tuple):the first tuple.
  tuple2(tuple):the second tuple.

  returns:
  tuple:A new tuple containing elements from both input tuples.
  """
  return tuple1+tuple2
 #example
 tuple_a=(1,2,3)
 tuple_b=(4,5,6,7)


 result=concatenate_tuples(tuple_a,tuple_b)
 print(result)
```

```
  File "<tokenize>", line 15
    tuple_a=(1,2,3)
    ^
IndentationError: unindent does not match any outer indentation level
```

```
#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
#function to get a set of strings
def get_input_set(prompt):
  return set(input(prompt).split())
#get two sets from user input
set1=get_input_set("Enter the first set of strings separated by commas:")
set2=get_input_set("Enter the second set of strings separated by commas:")
#find the elements in set1 but not in set2
difference=set1-set2
#print the difference
print("Elements in set1 but not in set2:",difference)
```

```
Enter the first set of strings separated by commas:1,5,6
Enter the second set of strings separated by commas:5,9,10
Elements in set1 but not in set2: {'1,5,6'}
```

```
#25)create a code that takes a tuple and two integers as input the function should return a new tuple containing elements from the origi
def slice_tuple(original_tuple,start_index,end_index):
  #validate the indices
  if start_index<0 or end_index>=len(original_tuple)or start_index>end_index:
    #return a new tuple
    return original_tuple[start_index:end_index]
#example usage
my_tuple=(1,2,3,4,5)
start=1
end=4
result=slice_tuple(my_tuple,start,end)
print(result)
```

```
None
```

```
#26)write a code that prompts the user to input two sets of character then print the union of these two sets
def main():
  #prompt user for first set of characters
  set1_input=input("enter the first set of characters (without spaces):")
  set1=set(set1_input)
  set2_input=input("enter the second set of characters (without spaces):")
  set2=set(set2_input)
  union_set=set1.union(set2)
  print("the union of the two sets is:",''.join(union_set))
  if __name__=="__main__":
    main()
```

```
#27)develop a code that takes a tuple of integers as input. the function should return the maximum and minimum values from the tuple usi
def find_max_min(input_tuple):
  if not input_tuple:
    raise valueerror("the input tuple cannot be empty")
    max_value=max(input_tuple)
    min_value=min(input_tuple)
    return max_value,min_value
    #example usage
my_tuple=(1,2,3,4,5)
max_value,min_value=find_max_min(my_tuple)
print("maximum value:",max_value)
print("minimum value:",min_value)
```

```
#28)create a code that defines two sets of integers then print the union intersection and difference of these two sets
def main():
  set_a={1,2,3,4,5}
  set_b={4,5,6,7,8}
  #calculate the union
  union_set=set_a.union(set_b)
  #calculate intersection
  intersection_set=set_a.intersection(set_b)
  #calculate the difference
  difference_set_b_a=set_b.difference(set_a)
  print("SetA:",set_a)
  print("SetB:",set_b)
  print("union:",union_set)
  print("intersection:",intersection_set)
  print("difference (A-B):",difference_set_a_b)
  print("difference (B-A):",difference_set_b_a)
  if __name__=="__main__":
    main()
```

```
#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
def count_occurrences(input_tuple,element):
  #count the occurrences of the element in the tuple
  count=input_tuple.count(element)
  return count
  #example
  my_tuple=(1,2,3,2,4,2,5)
  element_to_count=2
  result=count_occurrences(my_tuple,element_to_count)

  print(f"the element {element_to_count} occurs {result} times in the tuple.")
```

```
#30)develop a code that prompts the user to input two sets of string then print the symmetric difference of these two sets
def main():
  #prompt user for the first set of strings
  set_input=input("enter the first set of strings (comma-separated):")
  set1=set(item.strip()for item in set1_input.split(',') )
  #prompt user for the second set of strings
  set2_input=input("enter the second set of strings (comma-separated):")
  set2=set(item.strip()for item in set2_input.split(','))
  #calculate the symmetric difference
  symmetric_difference=set1.symmetric_difference(set2)
  #print the result
  print("the symmetric difference of the two set is:",symmetric_difference)
  if __name__=="__main__":
    main()



#31)
def word_frequencies(word_list):
  frequency_dict={}
  for word in word_list:
    if word in frequency_dict:
      frequency_dict[word]+=1
      else:
        frequecy_dict[word]=1
return frequency_dict



#example
input_words=input("enter a list of words (separated by spaces):").split()
result=word_frequencies(input_words)
print("word frequencies:,"result)
```

```
  File "<ipython-input-9-597b1e440d26>", line 7
    else:
    ^
SyntaxError: invalid syntax
```

```
#32)write a code that takes two dictionaries as input and merges them into a single dictionary if there are common keys the values shoul
def merge_dictionaries(dict,dict2):
  #create a nbew dictionary to hold the merged result
  merged_dict=dict1.copy()
  #iterate through the second dictionary
  for key,value in dict2.items():
    if key in merged_dict:
      #if the key exists,add the values
      merged_dict[key]+=value

 #example
dict1={'a':1,'b':2,'c':3}
dict2={'b':4,'c':5,'d':6}
result=merge_dictionaries(dict1,dict2)
print("merged dictionary:",result)
```

```
merged dictionary: None
```

```
#33)write a code to acess a value in a nested dictionary the function should take the dictionary and a list of keys as input and return
def acess_nested_dict(nested_dict,keys):
  current_value=nested_dict
  for key in keys:
    if key in current_value:
      current_value=current_value[key]
      else:
        return None
        return current_value
#example
nested_dect={'a':{'b':1},'c':{'d':2}}
keys=['a','b']
result=acess_nested_dict(nested_dict,keys)
print(result
```

```
  File "<ipython-input-11-1599d00087e2>", line 7
    else:
    ^
SyntaxError: invalid syntax
```

Next steps: Fix error

```
#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
def sort_dictionary_by_value(input_dict,ascending=True):
  sorted_dict=dict(sorted(input_dict.items(),key=lambda item:item[1],reverse=not ascending))
  return sorted_dict
#example
input_dictionary={'apple':5,'banana':2,'orange':8,'grape':3}
#sort in ascending order
sorted_ascending=sort_dictionary_by_value(input_dictionary,ascending=True)
print("sorted dictionary (ascending):",sorted_ascending)
#sort in descending order
sorted_descending=sort_dictionary_by_value(input_dictionary,ascending=False)
print("sorted dictionary (descending):",sorted_descending)
```

```
sorted dictionary (ascending): {'banana': 2, 'grape': 3, 'apple': 5, 'orange': 8}
sorted dictionary (descending): {'orange': 8, 'apple': 5, 'grape': 3, 'banana': 2}
```

```
#35)write a code that inverts a dictionary swapping keys and values ensure that the inverted dictionary correctly handles cases where mu
def invert_dictionary(input_dict):
  inverted_dict={}

  for key,value in input_dict.items():
    if value in inverted_dict:
      inverted_dict[value].append(key)
    else:
      inverted_dict[value]=[key]
      return inverted_dict


#example
original_dictionary={'a':1,'b':2,'c':1,'d':3}
inverted_dictionary=invert_dictionary(original_dictionary)
print("inverted dictionary:",inverted_dictionary)
```

Start coding or generate with AI.