```
#1. Write a program that defines a function count_lower_upper() that accepts a string an
#calculates the number of uppercase and lowercase alphabets in it. It should return thes
# Call this function for some sample string.

# Define the function to count lowercase and uppercase letters
def count_lower_upper(input_string):
    result = {"lowercase": 0, "uppercase": 0}

    for char in input_string:
        if char.islower():
            result["lowercase"] += 1
        elif char.isupper():
            result["uppercase"] += 1

    return result

# Sample string to test the function
sample_string = "Hello World! This is Python."

# Call the function and print the result
result = count_lower_upper(sample_string)
print("Count of lowercase and uppercase letters:", result)
```

```
Count of lowercase and uppercase letters: {'lowercase': 18, 'uppercase': 4}
```

```
#2. Write a program that defines a function compute() that calculates
#the value of n + nn + nnn + nnnn, where n is digit received by the function.
#test the function for digits 4 to 7.
# Define the function compute
def compute(n):
    # Calculate the value of n + nn + nnn + nnnn
    result = n + int(str(n)*2) + int(str(n)*3) + int(str(n)*4)
    return result

# Test the function for digits 4 to 7
for i in range(4, 8):
    print(f"Result for {i}: {compute(i)}")
```

```
Result for 4: 4936
Result for 5: 6170
Result for 6: 7404
Result for 7: 8638
```

```
#3. Write a program that defines a function create_array() to create and return a 3D arr
#whose dimensions are passed to the function. Also initialize each element of this
#aray to a value passed to the function. e.g. create_array(3,4,5,n) where first three
#arguments are 3D array dimensions and 4th value is for initialing each value of the 3D
import numpy as np

# Define the function to create and initialize a 3D array
def create_array(x, y, z, value):
    # Create a 3D array with dimensions (x, y, z) and initialize all elements to the giv
    array = np.full((x, y, z), value)
    return array
```

```
# Test the function with dimensions (3, 4, 5) and initializing each element to 'n'
result = create_array(3, 4, 5, 'n')

# Print the resulting 3D array
print("3D Array:")
print(result)
```

```
3D Array:
[[['n' 'n' 'n' 'n' 'n']
  ['n' 'n' 'n' 'n' 'n']
  ['n' 'n' 'n' 'n' 'n']
  ['n' 'n' 'n' 'n' 'n']]

 [['n' 'n' 'n' 'n' 'n']
  ['n' 'n' 'n' 'n' 'n']
  ['n' 'n' 'n' 'n' 'n']
  ['n' 'n' 'n' 'n' 'n']]

 [['n' 'n' 'n' 'n' 'n']
  ['n' 'n' 'n' 'n' 'n']
  ['n' 'n' 'n' 'n' 'n']
  ['n' 'n' 'n' 'n' 'n']]]
```

```
#4. Write a program that defines a function sum_avg() to accept marks of five subjects
#and calculates total and average. It should return  directly both values.
# Define the function sum_avg to calculate total and average
def sum_avg(marks):
    total = sum(marks)  # Calculate the total sum of marks
    average = total / len(marks)  # Calculate the average
    return total, average

# Sample input: marks of five subjects
marks = [85, 90, 78, 92, 88]

# Call the function and get the total and average
total_marks, average_marks = sum_avg(marks)

# Print the results
print(f"Total Marks: {total_marks}")
print(f"Average Marks: {average_marks:.2f}")
```

```
Total Marks: 433
Average Marks: 86.60
```

```
#5. Pangram is a sentence that uses every letter of the alphabet. Write a program to
#check whether a given string is pangram or not, through a user-defined function
#ispangram(). Test the function with “The quick brown fox jumps over the lazy
#dog” or “Crazy Fredrick bought many very exquisite opal jewels”. Hint: use set()
#o convert the string into a set of characters present in the string and use <= to check
# whether alphaset is a subset of the given string.

# Define the function to check if the string is a pangram
def ispangram(s):
    # Create a set of all lowercase letters in the alphabet
    alphabet_set = set("abcdefghijklmnopqrstuvwxyz")

    # Convert the string to lowercase and create a set of characters present in the stri
    s_set = set(s.lower())
```

```
    # Check if the alphabet set is a subset of the set of characters from the string
    return alphabet_set <= s_set

# Test the function with sample sentences
test_strings = [
    "The quick brown fox jumps over the lazy dog",
    "Crazy Fredrick bought many very exquisite opal jewels"
]

for sentence in test_strings:
    if ispangram(sentence):
        print(f"'{sentence}' is a pangram.")
    else:
        print(f"'{sentence}' is not a pangram.")
```

```
'The quick brown fox jumps over the lazy dog' is a pangram.
'Crazy Fredrick bought many very exquisite opal jewels' is a pangram.
```

```
#6. Write a function to create and return a list containing tuples of the form
 #(x,x2,x3) for all x between 1 and given ending value (both inclusive).
 # Define the function to create the list of tuples (x, x², x³)
def create_tuples(end_value):
    result = []
    for x in range(1, end_value + 1):
        result.append((x, x**2, x**3))  # Create the tuple (x, x², x³) and add to the li
    return result

# Test the function with an ending value
end_value = 5  # You can change this to test with other values
tuples_list = create_tuples(end_value)

# Print the resulting list of tuples
print("List of tuples (x, x², x³):", tuples_list)
```

```
List of tuples (x, x², x³): [(1, 1, 1), (2, 4, 8), (3, 9, 27), (4, 16, 64), (5, 25, 125)]
```

```
#7. A palindrome is a word or phrase that reads the same in both directions.
#Write a program that defines a function ispalindrome() which checks whether a given
#string is a palindrome or not. Ignore spaces and case mismatch while checking for
#palindrome.

# Define the function to check if the string is a palindrome
def ispalindrome(s):
    # Remove spaces and convert the string to lowercase
    cleaned_string = s.replace(" ", "").lower()

    # Check if the cleaned string is equal to its reverse
    return cleaned_string == cleaned_string[::-1]

# Test the function with some sample strings
test_strings = [
    "A man a plan a canal Panama",
    "Hello World",
    "Madam In Eden Im Adam"
```

```
]

for sentence in test_strings:
    if ispalindrome(sentence):
        print(f"'{sentence}' is a palindrome.")
    else:
        print(f"'{sentence}' is not a palindrome.")
```

```
'A man a plan a canal Panama' is a palindrome.
'Hello World' is not a palindrome.
'Madam In Eden Im Adam' is a palindrome.
```

```
#8. Write a program that defines a function convert() that receives a string containing
#a sequence of whitespace separated words and returns a string after removing all
#duplicate words and sorting them alphanumerically. Hint: use set(), list () , sorted(),

# Define the function to remove duplicates and sort words
def convert(s):
    words = s.split()                   # Split the string into individual words
    unique_words = set(words)          # Remove duplicates using set
    sorted_words = sorted(unique_words)  # Sort the words alphanumerically
    return ' '.join(sorted_words)      # Join the sorted words back into a string

# Test the function
input_str = "banana apple orange banana apple mango"
result = convert(input_str)
print("Converted string:", result)
```

```
Converted string: apple banana mango orange
```

```
#9. Write a program that defines a function count_alpha_digits() that accepts a string an
#calculates the number of alphabets and digits in it. It should return these values as a

# Define the function to count alphabets and digits
def count_alpha_digits(s):
    alpha_count = 0
    digit_count = 0

    for char in s:
        if char.isalpha():
            alpha_count += 1
        elif char.isdigit():
            digit_count += 1

    return {"Alphabets": alpha_count, "Digits": digit_count}

# Test the function
sample_input = "Hello123World456"
result = count_alpha_digits(sample_input)
print("Result:", result)
```

```
Result: {'Alphabets': 10, 'Digits': 6}
```

```
#10.    Write a program that defines a function called frequency() which computes the
#frequency of words present in a string passed to it. The frequencies should be returned
#in sorted order of words in the string.

# Define the function to compute word frequency
def frequency(s):
    words = s.split()           # Split the string into words
    freq = {}                   # Initialize an empty dictionary

    for word in words:
        if word in freq:
            freq[word] += 1     # Increment count if word already in dictionary
        else:
            freq[word] = 1      # Add new word with count 1

    # Sort the dictionary by word (alphabetically)
    sorted_freq = dict(sorted(freq.items()))

    return sorted_freq

# Test the function
input_str = "apple banana apple mango banana apple orange"
result = frequency(input_str)
print("Word Frequencies:", result)
```

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

```
#11.  Write a function create_list() that creates and returns a list which is an
#intersection of two lists passed to it.
# Define the function to get intersection of two lists
def create_list(list1, list2):
    intersection = []
    for item in list1:
        if item in list2 and item not in intersection:
            intersection.append(item)
    return intersection

# Test the function
a = [1, 2, 3, 4, 5]
b = [4, 5, 6, 7, 8]

result = create_list(a, b)
print("Intersection:", result)
```

```
Intersection: [4, 5]
```