1. Write a Python program to find sum of elements in list?

ANS: def main():

numbers = [1, 2, 3, 4, 5]

# Using the built-in sum() function

total\_sum = sum(numbers)

print("List of numbers:", numbers)

print("Sum of elements in the list:", total\_sum)

if \_\_name\_\_ == "\_\_main\_\_":

main()

When we run program,it will give output:

List of numbers: [1, 2, 3, 4, 5]

Sum of elements in the list: 15

1. Write a Python program to Multiply all numbers in the list?

ANS: To multiply all the numbers in a list, you can use a simple loop or the built-in `reduce()` function from the `functools` module. Here's how we can do it using both methods:

A). Using a loop:

def multiply\_list\_elements(numbers):

result = 1

for num in numbers:

result \*= num

return result

def main():

numbers = [1, 2, 3, 4, 5]

# Using a loop

multiplied\_result = multiply\_list\_elements(numbers)

print("List of numbers:", numbers)

print("Multiplication of elements in the list:", multiplied\_result)

if \_\_name\_\_ == "\_\_main\_\_":

main()

B). Using `reduce()` from the `functools` module (Python 3.x):

from functools import reduce

def multiply\_list\_elements(numbers):

return reduce(lambda x, y: x \* y, numbers)

def main():

numbers = [1, 2, 3, 4, 5]

# Using reduce()

multiplied\_result = multiply\_list\_elements(numbers)

print("List of numbers:", numbers)

print("Multiplication of elements in the list:", multiplied\_result)

if \_\_name\_\_ == "\_\_main\_\_":

main()

Both versions of the program will produce the same output:

List of numbers: [1, 2, 3, 4, 5]

Multiplication of elements in the list: 120

You can modify the `numbers` list to contain different elements, and the program will calculate the multiplication of those elements accordingly.

1. Write a Python program to find smallest number in a list?

ANS: def main():

numbers = [5, 2, 8, 1, 10]

# Using the built-in min() function

smallest\_number = min(numbers)

print("List of numbers:", numbers)

print("Smallest number in the list:", smallest\_number)

if \_\_name\_\_ == "\_\_main\_\_":

main()

When we run program,it will give output:

List of numbers: [5, 2, 8, 1, 10]

Smallest number in the list: 1

1. Write a Python program to find largest number in a list?

ANS: def main():

numbers = [5, 2, 8, 1, 10]

# Using the built-in max() function

largest\_number = max(numbers)

print("List of numbers:", numbers)

print("Largest number in the list:", largest\_number)

if \_\_name\_\_ == "\_\_main\_\_":

main()

When we run program,it will give output:

List of numbers: [5, 2, 8, 1, 10]

Largest number in the list: 10

1. Write a Python program to find second largest number in a list?

ANS: def find\_second\_largest(nums):

if len(nums) < 2:

return "List must have at least two elements."

largest = second\_largest = float('-inf')

for num in nums:

if num > largest:

second\_largest = largest

largest = num

elif num > second\_largest and num != largest:

second\_largest = num

if second\_largest == float('-inf'):

return "There is no second largest element in the list."

else:

return second\_largest

# Test the function

if \_\_name\_\_ == "\_\_main\_\_":

try:

num\_list = list(map(int, input("Enter space-separated numbers: ").split()))

second\_largest\_num = find\_second\_largest(num\_list)

print("The second largest number is:", second\_largest\_num)

except ValueError:

print("Invalid input. Please enter space-separated integers.")

1. Write a Python program to find N largest elements from a list?

ANS: def find\_n\_largest(nums, n):

if n <= 0:

return "N must be a positive integer."

if len(nums) < n:

return "List does not have enough elements to find N largest."

# Use a min-heap to efficiently find N largest elements

import heapq

largest\_elements = heapq.nlargest(n, nums)

return largest\_elements

# Test the function

if \_\_name\_\_ == "\_\_main\_\_":

try:

num\_list = list(map(int, input("Enter space-separated numbers: ").split()))

n = int(input("Enter the value of N: "))

n\_largest\_nums = find\_n\_largest(num\_list, n)

print(f"The {n} largest elements are:", n\_largest\_nums)

except ValueError:

print("Invalid input. Please enter space-separated integers.")

1. Write a Python program to print even numbers in a list?

ANS: def print\_even\_numbers(nums):

even\_numbers = [num for num in nums if num % 2 == 0]

return even\_numbers

# Test the function

if \_\_name\_\_ == "\_\_main\_\_":

try:

num\_list = list(map(int, input("Enter space-separated numbers: ").split()))

even\_nums = print\_even\_numbers(num\_list)

print("Even numbers in the list:", even\_nums)

except ValueError:

print("Invalid input. Please enter space-separated integers.")

1. Write a Python program to print odd numbers in a List?

ANS: def print\_odd\_numbers(nums):

odd\_numbers = [num for num in nums if num % 2 != 0]

return odd\_numbers

# Test the function

if \_\_name\_\_ == "\_\_main\_\_":

try:

num\_list = list(map(int, input("Enter space-separated numbers: ").split()))

odd\_nums = print\_odd\_numbers(num\_list)

print("Odd numbers in the list:", odd\_nums)

except ValueError:

print("Invalid input. Please enter space-separated integers.")

1. Write a Python program to Remove empty List from List?

ANS: def remove\_empty\_lists(lst):

return [sublist for sublist in lst if isinstance(sublist, list) and len(sublist) > 0]

# Test the function

if \_\_name\_\_ == "\_\_main\_\_":

try:

input\_list = eval(input("Enter a list containing sublists: "))

result\_list = remove\_empty\_lists(input\_list)

print("List with empty sublists removed:", result\_list)

except (NameError, SyntaxError):

print("Invalid input. Please enter a valid list.")

1. Write a Python program to Cloning or Copying a list?

ANS: We can create a Python program to clone or copy a list using various methods. Here are a few different ways to achieve this:

A). Using the slicing technique:

`def clone\_list(original\_list):

return original\_list[:]

# Test the function

if \_\_name\_\_ == "\_\_main\_\_":

original\_list = [1, 2, 3, 4, 5]

cloned\_list = clone\_list(original\_list)

print("Original List:", original\_list)

print("Cloned List:", cloned\_list)

B). Using the list() constructor:

```python

def clone\_list(original\_list):

return list(original\_list)

# Test the function

if \_\_name\_\_ == "\_\_main\_\_":

original\_list = [1, 2, 3, 4, 5]

cloned\_list = clone\_list(original\_list)

print("Original List:", original\_list)

print("Cloned List:", cloned\_list)

C). Using the `copy()` method:

def clone\_list(original\_list):

return original\_list.copy()

# Test the function

if \_\_name\_\_ == "\_\_main\_\_":

original\_list = [1, 2, 3, 4, 5]

cloned\_list = clone\_list(original\_list)

print("Original List:", original\_list)

print("Cloned List:", cloned\_list)

D). Using the `copy` module:

import copy

def clone\_list(original\_list):

return copy.copy(original\_list)

# Test the function

if \_\_name\_\_ == "\_\_main\_\_":

original\_list = [1, 2, 3, 4, 5]

cloned\_list = clone\_list(original\_list)

print("Original List:", original\_list)

print("Cloned List:", cloned\_list)

All of these methods create a new list that is a copy of the original list, so any changes made to one list will not affect the other.

1. Write a Python program to Count occurrences of an element in a list?

ANS: def count\_occurrences(lst, element):

return lst.count(element)

# Test the function

if \_\_name\_\_ == "\_\_main\_\_":

try:

num\_list = list(map(int, input("Enter space-separated numbers: ").split()))

element\_to\_count = int(input("Enter the element to count: "))

occurrences = count\_occurrences(num\_list, element\_to\_count)

print(f"The element {element\_to\_count} occurs {occurrences} times in the list.")

except ValueError:

print("Invalid input. Please enter space-separated integers.")