Q1. Create a python program to sort the given list of tuples based on integer value using a

lambda function.

[('Sachin Tendulkar', 34357), ('Ricky Ponting', 27483), ('Jack Kallis', 25534), ('Virat Kohli', 24936)]

data = [('Sachin Tendulkar', 34357), ('Ricky Ponting', 27483), ('Jack Kallis', 25534), ('Virat Kohli', 24936)]

sorted\_data = sorted(data, key=lambda x: x[1])

for item in sorted\_data:

print(item)

Output:

('Virat Kohli', 24936)

('Jack Kallis', 25534)

('Ricky Ponting', 27483)

('Sachin Tendulkar', 34357)

In this program, the ‘**sorted()’** function is called with the ‘**data’** list and a lambda function as the ‘**key’** parameter. The lambda function ‘**lambda x: x[1]’** returns the second element of each tuple, which is the integer value. This lambda function is used as the key to determine the sorting order.

The sorted list ‘**sorted\_data’** is then printed, showing the tuples sorted in ascending order based on the integer value.

Q2. Write a Python Program to find the squares of all the numbers in the given list of integers using

lambda and map functions.

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

squared\_numbers = list(map(lambda x: x\*\*2, numbers))

print(squared\_numbers)

Output:

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

a list of integers called ‘**numbers’**. We then use the ‘**map()’** function to apply a lambda function to each element of the ‘**numbers’** list. The lambda function takes a single argument ‘**x’** and returns its square ‘**x\*\*2’**. The ‘**map()’** function returns an iterator, so we convert it to a list using the ‘**list()’** function and assign it to the ‘**squared\_numbers’** variable. Finally, we print the ‘**squared\_numbers’** list which contains the squares of the numbers from the original list.

Q3. Write a python program to convert the given list of integers into a tuple of strings. Use map and

lambda functions

Given String: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Expected output: ('1', '2', '3', '4', '5', '6', '7', '8', '9', '10')

int\_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

str\_tuple = tuple(map(lambda x: str(x), int\_list))

print(str\_tuple)

Output:

('1', '2', '3', '4', '5', '6', '7', '8', '9', '10')

the input list ‘**int\_list’**. Then, we use ‘**map’** function with a ‘**lambda’** function to convert each integer element to a string. Finally, we convert the resulting map object to a tuple using the ‘**tuple’** function, and store it in the variable ‘**str\_tuple’**. Finally, we print the resulting tuple.

Q4. Write a python program using reduce function to compute the product of a list containing numbers

from 1 to 25.

from functools import reduce

# Define the list of numbers

numbers = list(range(1, 26))

# Define the function to multiply two numbers

def multiply(x, y):

return x \* y

# Use reduce to compute the product of the list

product = reduce(multiply, numbers)

# Print the product

print("Product:", product)

the ‘**reduce’** function from the ‘**functools’** module. It defines the list of numbers from 1 to 25 and a helper function ‘**multiply’** that multiplies two numbers. Then, it uses ‘**reduce’** with the ‘**multiply’** function and the list of numbers to compute the product. Finally, it prints the product.

Q5. Write a python program to filter the numbers in a given list that are divisible by 2 and 3 using the

filter function.

[2, 3, 6, 9, 27, 60, 90, 120, 55, 46]

def divisible\_by\_2\_and\_3(num):

return num % 2 == 0 and num % 3 == 0

numbers = [2, 3, 6, 9, 27, 60, 90, 120, 55, 46]

filtered\_numbers = list(filter(divisible\_by\_2\_and\_3, numbers))

print(filtered\_numbers)

Output:

[6, 60, 90, 120]

a function ‘**divisible\_by\_2\_and\_3’** that checks if a number is divisible by both 2 and 3. Then, we use the ‘**filter’** function to apply this function to each element in the ‘**numbers’** list. The ‘**filter’** function returns an iterator containing only the elements for which the function returns ‘**True’**. Finally, we convert the iterator to a list and print the filtered numbers.

Q6. Write a python program to find palindromes in the given list of strings using lambda and filter

function.

['python', 'php', 'aba', 'radar', 'level']

strings = ['python', 'php', 'aba', 'radar', 'level']

palindromes = list(filter(lambda x: x == x[::-1], strings))

print("Palindromes in the list:")

for palindrome in palindromes:

print(palindrome)

Output :

Palindromes in the list:

aba

radar

level

the ‘**filter()’** function to filter the strings that are palindromes. The lambda function ‘**lambda x: x == x[::-1]’** checks if a string ‘**x’** is equal to its reverse, which determines whether it is a palindrome or not. The ‘**filter()’** function returns an iterator that yields the elements for which the lambda function returns ‘**True’**. Finally, we convert the iterator to a list and print the palindromes.