**ML INTERNSHIP ASSIGNMENT**

NAME:**QUDSIYA**  COLLEGE:**CHAITANYA BHARATHI INSITUTE OF TECHNOLOGY,HYDERABAD**

**1.Write a python function which should be capable of finding the factorial of any given number as an argument.**

ANS:

def factorial(n):

if n < 0:

raise ValueError("Factorial is not defined for negative numbers")

if n == 0 or n == 1:

return 1

result = 1

for i in range(2, n + 1):

result \*= i

return result

# Example usage

print(factorial(5)) # Output: 120

**2. Luke Skywalker has family and friends. Help him remind them who is who. Given a string with a name, return the relation of that person to Luke.**

**Person Relation**

**Darth Vader father**

**Leia sister**

**Han brother in law**

**R2D2 droid**

**Example : relation\_to\_luke("Darth Vader") ➞ "Luke, I am your father."**

ANS:

def relation\_to\_luke(name):

relations = {

"Darth Vader": "Luke, I am your father.",

"Leia": "Luke, I am your sister.",

"Han": "Luke, I am your brother in law.",

"R2D2": "Luke, I am your droid."

}

return relations.get(name, "Unknown")

# Example usage

print(relation\_to\_luke("Darth Vader")) # Output: Luke, I am your father.

**3. Create a function which takes a number as its argument and return the number of digits in it. Use of len function is not allowed. For example for 5 it should return 1, for 32 it should return 2 and 123 , 3 should be returned and so on.**

ANS:

def count\_digits(num):

if num == 0:

return 1

count = 0

while num > 0:

num //= 10

count += 1

return count

# Example usage

print(count\_digits(123)) # Output: 3

**4. Write a function which takes a number as argument suppose 5 and gives results as multiplication of factorial of each positive number less than or equal to the number given. i.e !5\*!4\*!3\*!2\*!1 = 34560**.

ANS:

def factorial\_product(n):

def factorial(x):

if x == 0 or x == 1:

return 1

result = 1

for i in range(2, x + 1):

result \*= i

return result

product = 1

for i in range(1, n + 1):

product \*= factorial(i)

return product

# Example usage

print(factorial\_product(5)) # Output: 34560

**5. Write a function which takes any number of arguments from a user and return the result which should be output of a2 + b2+ c2+… if a , b ,c are numbers supplied ..i.e if 1,2,3 are supplied then result returned should be 14. But user may supply any number of inputs so make the function to adapt to that**.

ANS:

def sum\_of\_squares(\*args):

return sum(x \*\* 2 for x in args)

# Example usage

print(sum\_of\_squares(1, 2, 3)) # Output: 14

**6.Write a function which accepts 3 arguments from the user.1 .number 1, 2. Number2 and 3. An operation. The operation supported should be +, -, \*, and /. The function should return the result of given operation. For example arguments are 3,2,+ then result returned should be 5**

ANS:

def calculate(num1, num2, operation):

if operation == '+':

return num1 + num2

elif operation == '-':

return num1 - num2

elif operation == '\*':

return num1 \* num2

elif operation == '/':

if num2 == 0:

raise ValueError("Cannot divide by zero")

return num1 / num2

else:

raise ValueError("Unsupported operation")

# Example usage

print(calculate(3, 2, '+')) # Output: 5

**7. Write a function which takes an argument which should be a numeric +ve integer. Depending on the input supplied you have to print “I CAN”, “I WILL”. Suppose some one enters argument as 1 then only “I CAN” should be printed. But if some one enters 2 then first “I CAN” should be printed then “I WILL”. And if someone enters 3 then following should be printed in corresponding order: “I CAN”, “I WILL”, “I CAN” and so on for any numbers entered**.

ANS:

def print\_statements(n):

statements = ["I CAN", "I WILL"]

for i in range(n):

print(statements[i % 2])

# Example usage

print\_statements(3)

# Output:

# I CAN

# I WILL

# I CAN

**8. We have been given a list of whole numbers which represents the color of each gloves, determine how many pairs of gloves with matching colors there are. For example, there are 7 gloves with colors [1, 2, 1, 2, 1, 3, 2]. There is one pair of color 1 and one of color 2. There are three odd gloves left, one of each color. The number of pairs is 2. Create a function that returns an integer representing the number of matching pairs of gloves that are available.**

ANS:

def count\_glove\_pairs(gloves):

from collections import Counter

color\_count = Counter(gloves)

return sum(count // 2 for count in color\_count.values())

# Example usage

print(count\_glove\_pairs([1, 2, 1, 2, 1, 3, 2])) # Output: 2

**9. Write a function that returns True if two arrays, when combined, form a consecutive sequence. A consecutive sequence is a sequence without any gaps in the integers, e.g. 1, 2, 3, 4, 5 is a consecutive sequence, but 1, 2, 4, 5 is not. Notes • The input lists will have unique values. • The input lists can be in any order. Examples consecutive\_combo([7, 4, 5, 1], [2, 3, 6]) ➞ True consecutive\_combo([1, 4, 6, 5], [2, 7, 8, 9]) ➞ False consecutive\_combo([1, 4, 5, 6], [2, 3, 7, 8, 10]) ➞ False consecutive\_combo([44, 46], [45]) ➞ True**

ANS:

def consecutive\_combo(list1, list2):

combined = list1 + list2

combined.sort()

return all(combined[i] == combined[i - 1] + 1 for i in range(1, len(combined)))

# Example usage

print(consecutive\_combo([7, 4, 5, 1], [2, 3, 6])) # Output: True

**10. You work for a manufacturer, and have been asked to calculate the total profit made on the sales of a product. You are given a dictionary containing the cost price per unit (in dollars), sell price per unit (in dollars), and the starting inventory. Return the total profit made, rounded to the nearest dollar. Examples profit({ "cost\_price": 32.67, "sell\_price": 45.00, "inventory": 1200 }) ➞ 14796**

ANS:

def profit(sales\_data):

cost\_price = sales\_data["cost\_price"]

sell\_price = sales\_data["sell\_price"]

inventory = sales\_data["inventory"]

total\_profit = (sell\_price - cost\_price) \* inventory

return round(total\_profit)

# Example usage

print(profit({

"cost\_price": 32.67,

"sell\_price": 45.00,

"inventory": 1200

})) # Output: 14796