PRACTICAL-1

QUES 1: Write a program to find sum of two numbers.

CODE:

```
a= int(input("Enter the first number: "))
b= int(input("Enter the second number: "))
sum= int(a+b)
print("Sum of the two number is:",sum)
```

QUES 2: Write a program to find MAXIMUM of two numbers.

CODE:

```
a=int(input("Enter the first number: "))
b=int(input("Enter the seconnd number: "))

if(a>b):
        print("a is greater than b")
else:
        print("b is greater than a")
```

QUES 3: *Write a program to find MAXIMUM of three numbers.*

CODE:

```
a=int(input("Enter the first number: "))
b=int(input("Enter the second number: "))
c=int(input("Enter the third number: "))

if(a>b):
        if(a>c):
                print("a is the largest number")
        else:
                print("c is the largest number")
        else if(a<b):
                if(b>c):
                        print("b is the largest number")
                else:
                        print("c is the largest number")
```

QUES 4: *Write a program to calculate the factorial of number.*

CODE:

```
n=int(input("Enter the number: "))

def factorial(n):
if(n==0):
        return 1
else:
        return (n*factorial(n-1))
print("Factorial of the number",n,"is",factorial(n))
```

QUES 5 : *Check if the number is Armstrong number or not.*

CODE:

```
def is_armstrong(num):

num_str = str(num)
power = len(num_str)
total = 0
for digit in num_str:
total += int(digit) ** power
return total == num

number = int(input("Enter a number to check: "))
if is_armstrong(number):
        print(f"{number} is an Armstrong number.")
else:
        print(f"{number} is not an Armstrong number.")
```

PRACTICAL – 2

QUES 1: *Write a program to calculate Simple Interest.*

CODE:

```
p=float(input("Enter the Principal Amount: "))

t=float(input("Enter the time period in Years: "))
r=float(input("Enter the interest rate per annum: "))

simpleInterest=(p*r*t)/100

print("The simple interest for",t,"years at",r,"interest rate is",simpleInterest)
```

QUES 2 : Write a program to calculate Compound Interest.

CODE :

```
P=float(input("Enter the Principal Amount: "))

t=float(input("Enter the time period in Years: "))
r=float(input("Enter the interest rate per annum: "))
n=float(input("Enter the number of times amount is compunded"))

def compound_interest(P, r, n, t):
        compoundInterest= (P * (1 + r/n)**(n*t))-P
        return compoundInterest

print(compound_interest(P,r,n,t))
```

QUES 3 : *Write a program to determine whether the number is prime or not.*
CODE :

```
import math


def is_prime(n):
if n <= 1:
        return False
for i in range(2, int(math.sqrt(n)) + 1):
        if n % i == 0:
                return False
        return True

n=int(input("Enter the number: "))

print(is_prime(n))
```

QUES 4 : *Write a program to print the Fibonacci Series.*

CODE :

```
nTerms=int(input("Length of the terms: "))


def fibonacci(n):
    if(n<=1):
        return n
    else:
        return(fibonacci(n-1)+fibonacci(n-2))
if nTerms<=0:
    print("Please enter a positive integer")
else:
    print("Fibonacci sequence: ")
for i in range(nTerms):
    print(fibonacci(i))
```

QUES 5 : *Write a program to count the number of digit in a number.*

CODE :

```
n=int(input("Enter the number: "))

count=0
while(n!=0):
    n=int(n/10)
    count+=1

print(count)
```

PRACTICAL – 3

QUES 1 : *Write a program to check whether the string is a palindrome or not.*

CODE :

```
def is_palindrome(s):
    s = ''.join(c.lower() for c in s if c.isalnum())
    return s == s[::-1]


user_input = input("Enter a string to check if it's a palindrome: ")
if is_palindrome(user_input):
    print(f'"{user_input}" is a palindrome!')
else:
    print(f'"{user_input}" is not a palindrome.')
```

QUES 2 : *Write a program to calculate the number of letters and numbers in a string.*

CODE :

```
def count_letters_and_numbers(text):
    letter_count = 0
    number_count = 0
    for char in text:
        if char.isalpha():
            letter_count += 1
        elif char.isdigit():
            number_count += 1
    return {
        'letters': letter_count,
        'numbers': number_count}

user_input = input("Enter a string: ")

result = count_letters_and_numbers(user_input)
print(f"Letters: {result['letters']}")
print(f"Numbers: {result['numbers']}")
print(f"Total characters: {len(user_input)}")
print(f"Other characters: {len(user_input) - result['letters'] - result['numbers']}")
```

QUES 3 : *Write a program to check the validity of a password.*

CODE:

```
import re


def validate_password(password):
    if not password:
        return False, "Password cannot be empty"
```

```
        if len(password) < 8:
                return False, "Password must be at least 8 characters long"
        if not re.search(r'[A-Z]', password):
                return False, "Password must contain at least one uppercase letter"
        if not re.search(r'[a-z]', password):
                return False, "Password must contain at least one lowercase letter"
        if not re.search(r'[0-9]', password):
                return False, "Password must contain at least one digit"
        if not re.search(r'[@$!%*?&]', password):
                return False, "Password must contain at least one special character (@, $, !, %, *, ?, &)"
        return True, "Password is valid"


print("Password Validator")
print("==================")
print("Password must contain:")
print("- At least 8 characters")
print("- At least one uppercase letter")
print("- At least one lowercase letter")
print("- At least one digit")
print("- At least one special character (@, $, !, %, *, ?, &)")
print()
while True:
        password = input("Enter a password (or 'exit' to quit): ")
        if password.lower() == 'exit':
                print("Goodbye!")
                break
is_valid, message = validate_password(password)
if is_valid:
        print("✓ " + message)
else:
        print("✗ " + message)
print()
```

QUES 4 : *Write a program to print the multiplication table of a number.*

CODE:

```
n=int(input("Enter the number: "))

print("The Table of ",n,"is : ")

for i in range(11):
        print(n,"x",i,"=",n*i)
```

QUES 5 : *Write a program to convert months name into number of days.*

CODE:

```
def get_days_in_month(month_name):
        month_name = month_name.lower().strip()
        days_in_month = {
```

```
                "january": 31,
                "february": 28,
                "march": 31,
                "april": 30,
                "may": 31,
                "june": 30,
                "july": 31,
                "august": 31,
                "september": 30,
                "october": 31,
                "november": 30,
                "december": 31}
        if month_name not in days_in_month:
                raise ValueError(f"Invalid month name: {month_name}")
                return days_in_month[month_name]

def get_days_in_month_with_leap_year(month_name, year=None):
        days = get_days_in_month(month_name)
        if month_name.lower().strip() == "february" and year is not None:
                if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
                        return 29
        return days


try:
        test_months = ["January", "February", "april", "december"]
        print("Standard days in month:")
        for month in test_months:
        print(f"{month}: {get_days_in_month(month)} days")
        print("\nDays in February considering leap years:")
        print(f"February 2023: {get_days_in_month_with_leap_year('February', 2023)} days")
        print(f"February 2024: {get_days_in_month_with_leap_year('February', 2024)} days")
        print(f"February 2100: {get_days_in_month_with_leap_year('February', 2100)} days")
        print(f"February 2000: {get_days_in_month_with_leap_year('February', 2000)} days")
        print("\nEnter a month name to get the number of days (or 'quit' to exit):")
while True:
        user_input = input("> ")
        if user_input.lower() == "quit":
                break
        try:
                print(f"{user_input} has {get_days_in_month(user_input)} days")
        except ValueError as e:
                print(f"Error: {e}")
except Exception as e:
        print(f"An error occurred: {e}")
```

PRACTICAL - 4

QUES 1 : *Write a program to reverse a user given input.*

CODE:

```
def reverse_value(value):
    if isinstance(value, str):
        return value[::-1]
    elif isinstance(value, (int, float)):
        str_value = str(value)
        if str_value.startswith('-'):
        reversed_str = '-' + str_value[1:][::-1]
        else:
            reversed_str = str_value[::-1]
        if isinstance(value, int):
            if reversed_str.endswith('.'):
                reversed_str = reversed_str[:-1]
            return int(reversed_str)
        else:
            return float(reversed_str)
    elif isinstance(value, list):
        return value[::-1]
    else:
        return "Unsupported type. Please enter a string, number, or list."

print("Welcome to the Value Reverser!")
print("This program can reverse strings, numbers, and lists.")

while True:
    print("\nWhat would you like to reverse?")
    print("1. String")
    print("2. Number")
    print("3. List")
    print("4. Exit")
    choice = input("Enter your choice (1-4): ")
    if choice == '4':
        print("Thanks for using this. Bye!")
        break
    if choice == '1':
        value = input("Enter a string: ")
        reversed_value = reverse_value(value)
    elif choice == '2':
        try:
            value_str = input("Enter a number: ")
            if '.' in value_str:
                value = float(value_str)
            else:
                value = int(value_str)
                reversed_value = reverse_value(value)
        except ValueError:
        print("That's not a valid number. Try again.")
```

```
                continue
        elif choice == '3':
                try:
                        value_str = input("Enter a list of items separated by commas: ")
                        value = [item.strip() for item in value_str.split(',')]
                        reversed_value = reverse_value(value)
                except:
                        print("Something went wrong with the list. Try again.")
                        continue
        else:
                print("Invalid choice. Enter 1, 2, 3, or 4.")
                continue
        print(f"Original value: {value}")
        print(f"Reversed value: {reversed_value}")
```

QUES 2: *Write a program to find the factorial of number using Recursion.*

CODE:

```
n=int(input("Enter the number: "))

def factorial(n):
        if(n==0):
                return 1
        else:
                return (n*factorial(n-1))
print("Factorial of the number",n,"is",factorial(n))
```

QUES 3: *Write a program that takes a character and returns true of its a vowel.*
CODE:

```
def is_vowel(char):
        return char.lower() in "aeiou"
def count_vowels(text):
        vowel_count = 0
        for char in text:
                if is_vowel(char):
                        vowel_count += 1
        return vowel_count

def find_vowel_positions(text):
        positions = []
        for i, char in enumerate(text):
                if is_vowel(char):
                        positions.append(i)
        return positions

def analyze_text(text):
                if not text:
                        print("No text provided to analyze")
                return
        total_vowels = count_vowels(text)
        vowel_positions = find_vowel_positions(text)
```

```
print(f"Analyzed text: {text}")
print(f"Total vowels found: {total_vowels}")
print(f"Vowel positions: {vowel_positions}")
if total_vowels > 0:
        vowel_percentage = (total_vowels / len(text)) * 100
        print(f"Vowels make up {vowel_percentage:.2f}% of the text")
else:
        print("No vowels found in the text")

text = input("Enter text to analyze for vowels: ")
analyze_text(text)
```

QUES 4 : *Write a program to count the length of a string using functions.*
CODE :

```
def get_string_length(input_str):
        if not input_str:
                return 0
        length = 0
        for _ in input_str:
                length += 1
        return length


text = input("Enter a string: ")
result = get_string_length(text)
print(f"The length of the string is: {result}")
```

QUES 5: *Write a program that takes two lists and return true if they have one or more elements same*
CODE :

```
def has_common_element(list1, list2):
        for element in list1:
                if element in list2:
                        return True
        return False


print(has_common_element([1, 2, 3, 4], [5, 6, 7, 8]))
print(has_common_element([1, 2, 3, 4], [4, 5, 6, 7]))
print(has_common_element(["apple", "banana"], ["orange", "apple"]))
print(has_common_element([], [1, 2, 3]))
```

PRACTICAL – 5

QUES 1: *Write a program that uppercase the first half of a string.*
CODE:

```
def uppercase_first_half(input_string):
    half_length = len(input_string) // 2
    first_half = input_string[:half_length].upper()
    second_half = input_string[half_length:]
return first_half + second_half

input_text = input("Enter a string: ")
result = uppercase_first_half(input_text)
print(f"Result: {result}")
```

QUES 2: *Write a program to find the least frequent letter in a string.*
CODE:

```
def find_least_frequent_char(input_list):
    char_count = {}
    for item in input_list:
        for char in str(item):
            if char in char_count:
                char_count[char] += 1
            else:
                char_count[char] = 1
    if not char_count:
        return None, 0
min_frequency = min(char_count.values())
least_frequent_chars = [char for char, count in char_count.items() if count == min_frequency]
return least_frequent_chars, min_frequency
```

QUES 3: *Write a program to find the length of string without space.*
CODE:

```
def length_without_spaces(input_string):
    string_without_spaces = input_string.replace(" ", "")
    return len(string_without_spaces)

input_text = input("Enter a string: ")
result = length_without_spaces(input_text)
print(f"Length of string without spaces: {result}")
```

QUES 4: *Write a program to check whether the key is present in the dictionary or not.*
CODE:

```
def check_key_exists(dictionary, key):
    if key in dictionary:
        return True
    else:
        return False

def add_key_if_not_exists(dictionary, key, value):
    if not check_key_exists(dictionary, key):
```

```
                dictionary[key] = value
                return True
        return False

my_dict = {}
key_to_check = "example_key"
value_to_add = "example_value"

exists = check_key_exists(my_dict, key_to_check)
print(f"Key '{key_to_check}' exists: {exists}")

added = add_key_if_not_exists(my_dict, key_to_check, value_to_add)
print(f"Key '{key_to_check}' was added: {added}")
print(f"Dictionary after: {my_dict}")

exists = check_key_exists(my_dict, key_to_check)
print(f"Key '{key_to_check}' exists: {exists}")

added = add_key_if_not_exists(my_dict, key_to_check, "new_value")
print(f"Key '{key_to_check}' was added: {added}")
print(f"Dictionary after second attempt: {my_dict}")
```

QUES 5: *Write a program to print the sum of key value pair of a dictionary.*
CODE:

```
def sum_dict_key_value(dictionary):
        total_sum = 0
        for key, value in dictionary.items():
                if isinstance(key, (int, float)) and isinstance(value, (int, float)):
                        total_sum += key + value
        return total_sum

dictionary = {1: 10, 2: 20, 3: 30, "test": 40, 5: "string"}
result = sum_dict_key_value(dictionary)
print(f"Sum of numeric keys and values: {result}")
```

PRACTICAL – 6

QUES 1: *Write a program to replace dictionary values from another dictionary.*

CODE:

```
def replace_dict_values(target_dict, source_dict):
    for key in target_dict:
        if key in source_dict:
            target_dict[key] = source_dict[key]
    return target_dict

target = {'a': 1, 'b': 2, 'c': 3, 'd': 4}
source = {'a': 10, 'c': 30, 'e': 50}

result = replace_dict_values(target, source)
print(result)
```

QUES 2: *Write a program to delete a list of keys in a given list.*

CODE:

```
def delete_keys_from_list(original_list, keys_to_delete):
    result = [item for item in original_list if item not in keys_to_delete]
    return result

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

result_list = delete_keys_from_list(my_list, keys_to_delete)

print(f"Original list: {my_list}")
print(f"Keys to delete: {keys_to_delete}")
print(f"Result after deletion: {result_list}")
```

QUES 3: *Write a program to check if dictionary is empty or not.*

CODE:

```
def is_dictionary_empty(dictionary):
    if not dictionary:
        return True
    else:
        return False

test_dict1 = {}
test_dict2 = {"key": "value"}

print(f"Is test_dict1 empty? {is_dictionary_empty(test_dict1)}")
print(f"Is test_dict2 empty? {is_dictionary_empty(test_dict2)}")
```

QUES 4: *Write a program to get keys with MAXIMUM and MINIMUM values.*

CODE:

```
def get_max_min_keys(dictionary):
```

```
    if not dictionary:
        return None, None
    max_key = min_key = next(iter(dictionary))
    max_value = min_value = dictionary[max_key]
    for key, value in dictionary.items():
        if value > max_value:
            max_key = key
            max_value = value
        if value < min_value:
            min_key = key
            min_value = value
    return max_key, min_key

sample_dict = {
        'a': 5,
        'b': 9,
        'c': 2,
        'd': 14,
        'e': 1}

max_key, min_key = get_max_min_keys(sample_dict)
print(f"Key with maximum value: {max_key}")
print(f"Key with minimum value: {min_key}")
```

QUES 5: *Write a program to get a key for the value.*

CODE:

```
def get_key_for_value(dictionary, value):

    for key, val in dictionary.items():

        if val == value:

            return key

    return None

my_dict = {
    "apple": 5,
    "banana": 10,
    "orange": 15,
    "grape": 20,
    "mango": 15}

search_value = 15
result = get_key_for_value(my_dict, search_value)

print(f"Key for value {search_value}: {result}")

def get_all_keys_for_value(dictionary, value):
```

```
    return [key for key, val in dictionary.items() if val == value]

all_keys = get_all_keys_for_value(my_dict, search_value)
print(f"All keys for value {search_value}: {all_keys}")
```