Banarsidas Chandiwala Institute of Information Technology

Affiliated to

Guru Govind Singh Indraprastha University, New Delhi

MASTER OF COMPUTER APPLICATION

Python Lab (MCA-166)

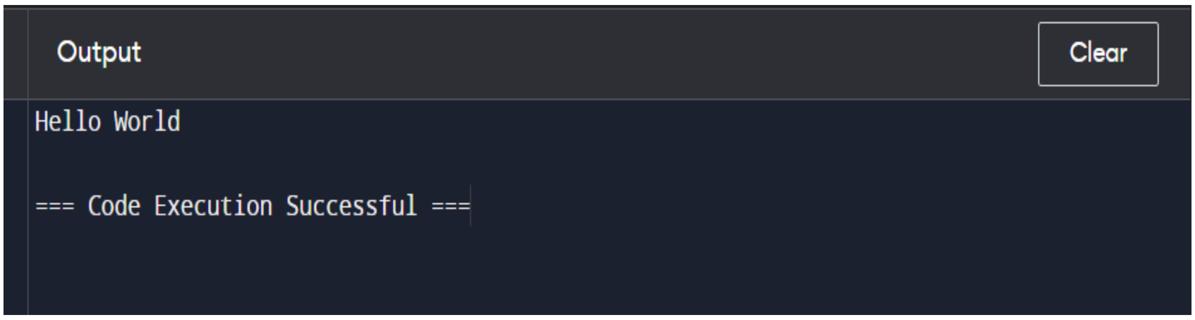
Submitted by:	Submitted to:
Nandan Shah	Mr. Alok Mishra
Roll no: 02311104423	Asst. Professor
MCA 2nd Sem	
Batch:2023-25	

INDEX

17	Consider the list lst=[9,8,7,6,5,4,3]. Write the Python program which performs the following operation: A. Insert element 10 at beginning of the list. B. Insert element 2 at end of the list. C. Delete the element at index position 5. D. Print all elements in reverse order	22	
18	Write a Python program which will return the sum of the numbers in the array, returning 0 for an empty array. Except the number 13 is very unlucky, so it does not count and number that come immediately after 13 also do not count. Example : [1, 2, 3, 4] = 10 , [1, 2, 3, 4, 13] = 10 , [13, 1, 2, 3, 13] = 5	23	
19	Write a Python program to Check Whether a String is Palindrome or Not.	24	
20	Write a Python program to Illustrate Different Set Operations.	25	
21	Write a python program to do the following : A. To sum all the items in a list. B. To multiplies all the items in a list. C. To get the largest number from a list. D. To get the smallest number from a list E. To remove duplicates from a list F. To check a list is empty or not G. To select an item randomly from a list. H. To clone or copy a list I. To find the second smallest number in a list. J. To find the second largest number in a list K. To get unique values from a list. L. To remove the K'th element from a given list, print the new list. M. To insert an element at a specified position into a given list	26-28	
22	Write a Python program to sort a dictionary by value.	29	
23	Write a Python program to add a key to a dictionary. Sample Dictionary: {0: 100, 1: 200} Expected Result: {0: 100, 1: 200, 2: 300} Write a Python program to print a dictionary where the keys are numbers	30	

1. **Write a program to print “Hello World”.**

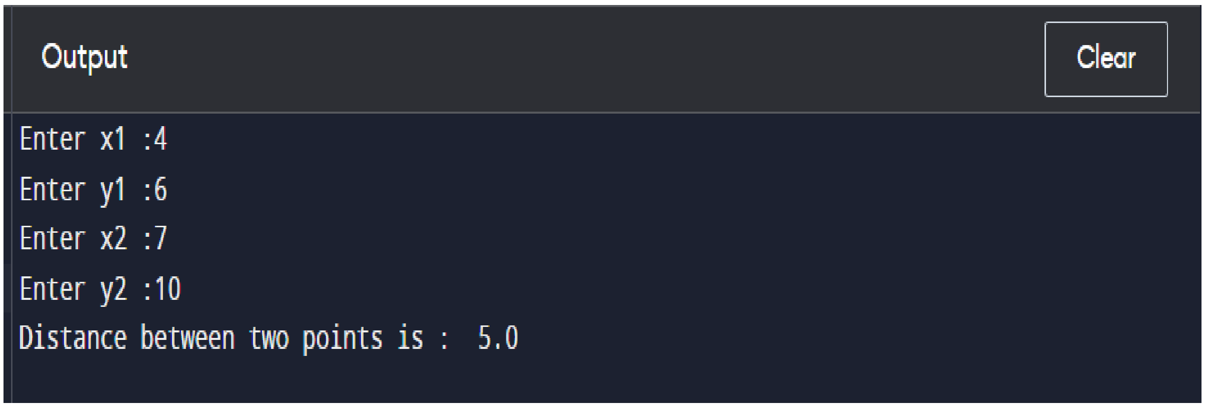
```
Print(“Hello World”)
```



```
Hello World

=== Code Execution Successful ===
```

2. Write a program to compute distance between two points taking input from the user.

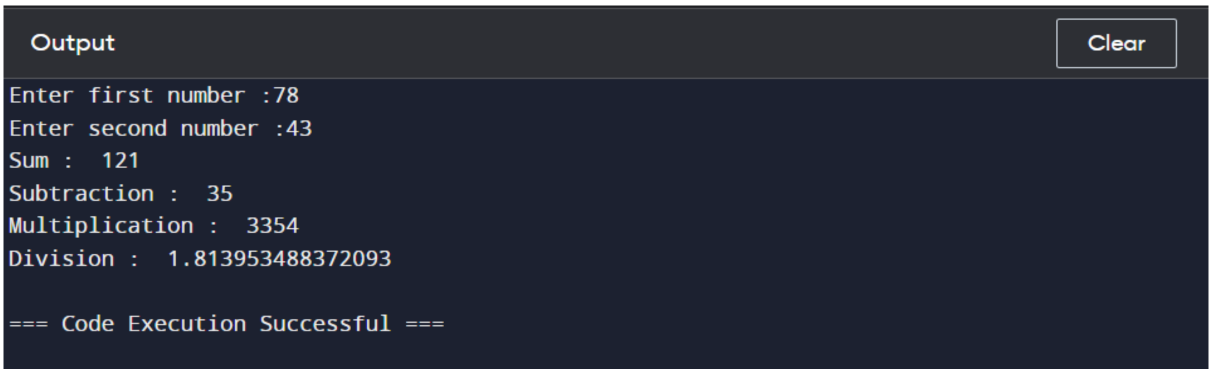
```
x1=int(input("Enter x1 :"))
y1=int(input("Enter y1 :"))

x2=int(input("Enter x2 :"))
y2=int(input("Enter y2 :"))

d=((x2-x1)**2+(y2-y1)**2) ** 0.5
print("Distance between two points is : ",d)
```



```
Enter x1 :4
Enter y1 :6
Enter x2 :7
Enter y2 :10
Distance between two points is :  5.0
```

3. Write a python Program for addition, Subtraction, multiplication, division.

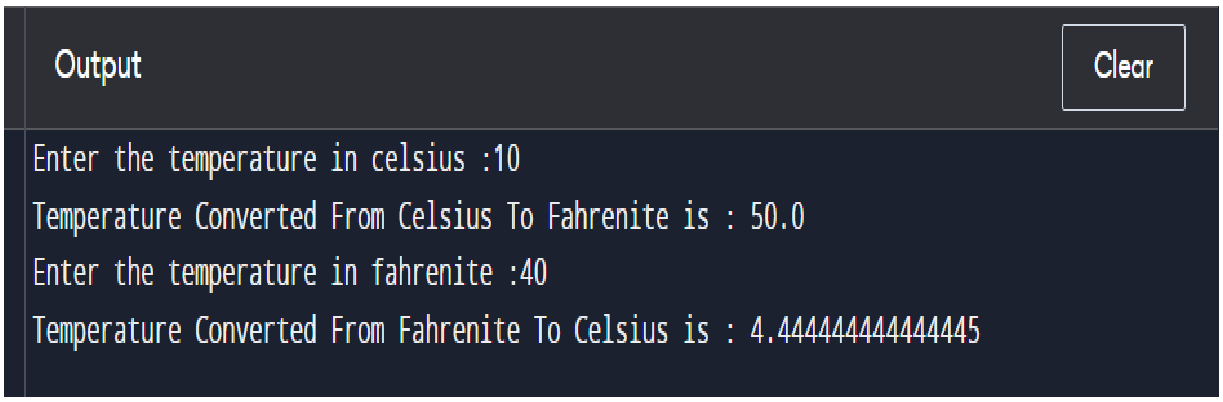
```
a=eval(input("Enter first number :"))
b=eval(input("Enter second number :"))

sum1=a+b;
sub=a-b;
mul=a*b;
div=a/b

print("Sum : ",sum1)
print("Subtraction : ",sub)
print("Multiplication : ",mul)
print("Division : ",div)
```



```
Enter first number :78
Enter second number :43
Sum :  121
Subtraction :  35
Multiplication :  3354
Division :  1.813953488372093

=== Code Execution Successful ===
```

4. Write a python program for converting Temperature to and from Celsius and Fahrenheit.

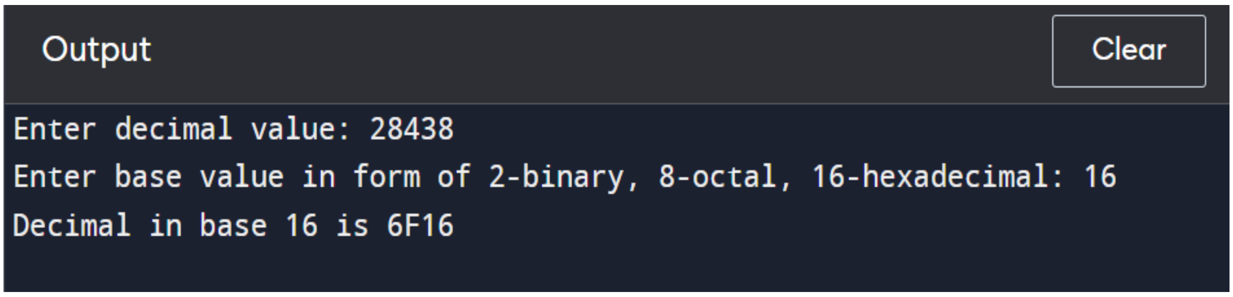
```
C=eval(input("Enter the temperature in Celsius :"))
F = (C * 9/5) + 32;
print("Temperature Converted From Celsius To Fahrenite is :",F)

f=eval(input("Enter the temperature in fahrenite :"))
c=(5/9) * (f – 32)
print("Temperature Converted From Fahrenite To Celsius is :",c)
```



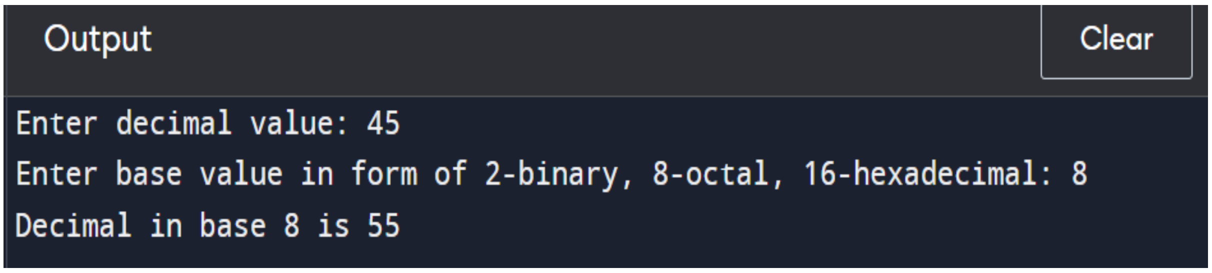
```
Enter the temperature in celsius :10
Temperature Converted From Celsius To Fahrenite is : 50.0
Enter the temperature in fahrenite :40
Temperature Converted From Fahrenite To Celsius is : 4.444444444444445
```

5. Write a python program to Convert Decimal to Binary, Octal and Hexadecimal.

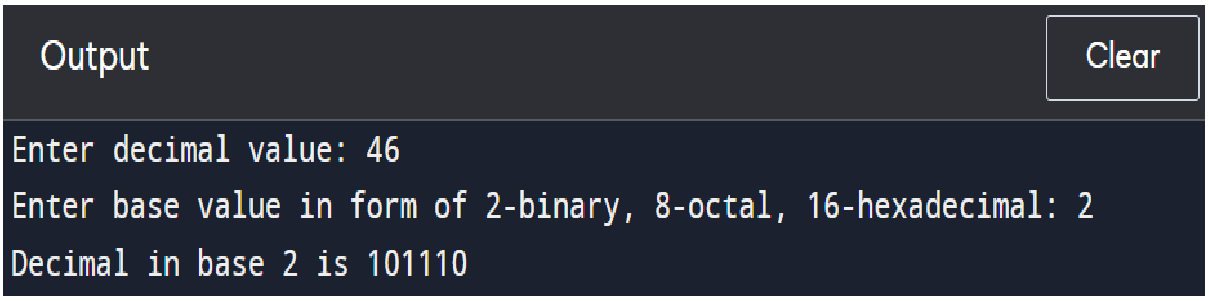
```
def convert(decimal, base):
   result = " "
   remainder = [0,1,2,3,4,5,6,7,8,9,"A","B","C","D","E","F"]
   while decimal>0:
       result = str(remainder[decimal%base])+ result
       decimal = decimal//base
   print("Decimal in base", base, "is", result[:-1])

decimal = int(input("Enter decimal value: "))
base = int(input("Enter base value in form of 2-binary, 8-octal, 16-hexadecimal: "))

if 2<=base<=16:
   convert(decimal, base)
else:
   print("invalid base")
```



```
Enter decimal value: 28438
Enter base value in form of 2-binary, 8-octal, 16-hexadecimal: 16
Decimal in base 16 is 6F16
```



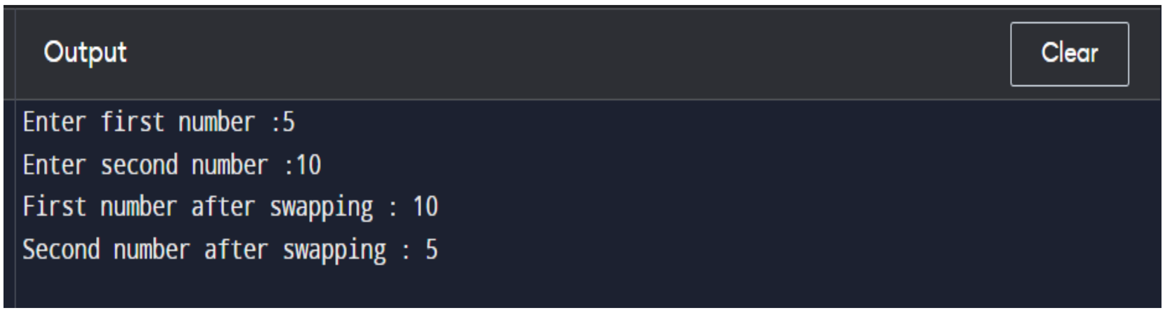
```
Enter decimal value: 45
Enter base value in form of 2-binary, 8-octal, 16-hexadecimal: 8
Decimal in base 8 is 55
```



```
Enter decimal value: 46
Enter base value in form of 2-binary, 8-octal, 16-hexadecimal: 2
Decimal in base 2 is 101110
```

6. **Write a Python program to swap values of Two Variables without using third variable.**

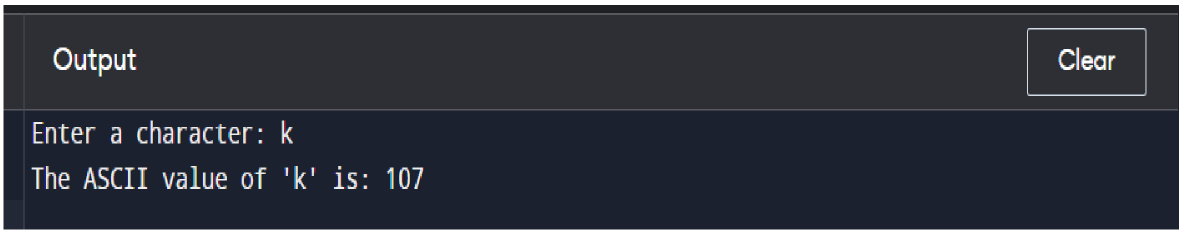
```
a=eval(input("Enter first number :"))
b=eval(input("Enter second number :"))

a=a+b
b=a-b
a=a-b

print("First number after swapping :",a)
print("Second number after swapping :",b)
```



```
Enter first number :5
Enter second number :10
First number after swapping : 10
Second number after swapping : 5
```

7. Write a python program to Find ASCII Value of Character.

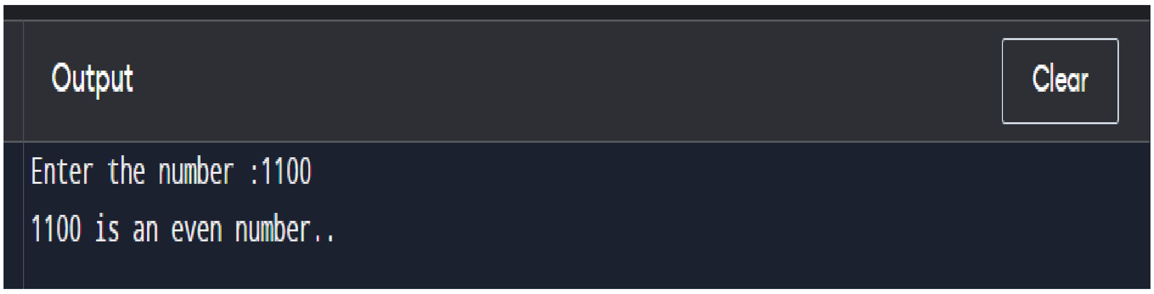
```
def find_ascii(char):
    return ord(char)

character = input(“Enter a character: “)
ascii_value = find_ascii(character)
print(“The ASCII value of ‘{character}’ is: {ascii_value}”)
```



```
Enter a character: k
The ASCII value of 'k' is: 107
```

8. Write a Python Program for checking whether the given number is an even number or not.

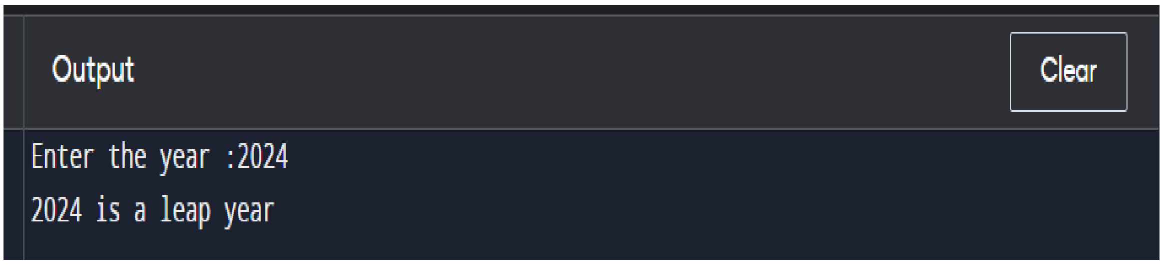
```
a=eval(input(“Enter the number :”));
if a%2==0:
    print(a, “is an even number..”)
else:
    print(a, “is not an even number ..”)
```



```
Enter the number :1100
1100 is an even number..
```

9. Write a Python Program to check leap year.

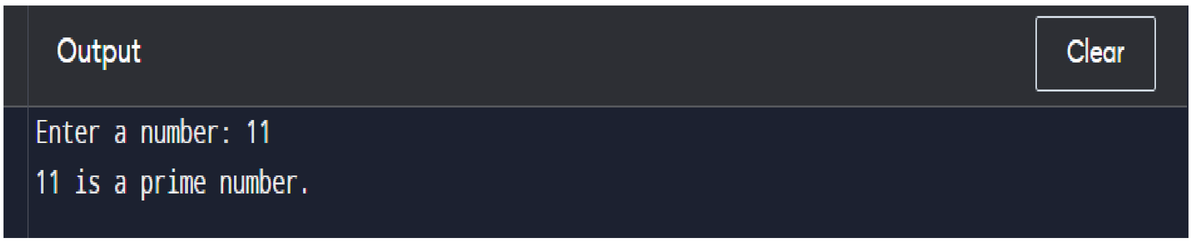
```
def leap_year(year):
    if year%4==0:
        if year%100==0 and year%400!=0:
            return False
        else:
            return True
    else:
        return False

year=int(input(“Enter the year :”))

if leap_year(year):
    print(year ,“is a leap year”)
else:
    print(year, “is not a leap year”)
```



```
Enter the year :2024
2024 is a leap year
```

10.Write a Python Program to Check Prime Number.

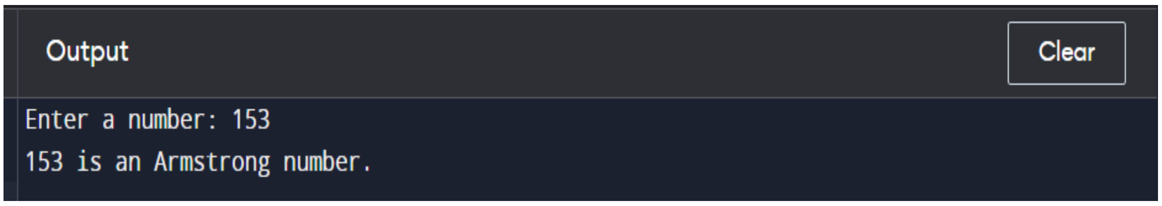
```
def is_prime(num):
    if num <= 1:
        return False
    for r in range(2, num//2):
        if num % r == 0:
            return False
    return True

number = int(input(“Enter a number: ” ))

if is_prime(number):
    print(number, “is a prime number.”)
else:
    print(number, “is not a prime number.”)
```



```
Enter a number: 11
11 is a prime number.
```

11. Write a Python program to check whether the given no is Armstrong or not.

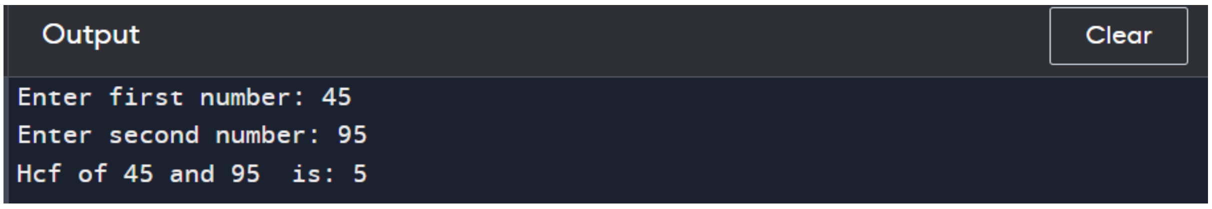
```
def is_armstrong(number):
    num_digits = len(str(number))
    sum = 0
    temp = number
    while temp > 0:
        digit = temp % 10
        sum += digit ** num_digits
        temp //= 10
    if number == sum:
        return True
    else:
        return False

number = int(input("Enter a number: "))

if is_armstrong(number):
    print(number, "is an Armstrong number.")
else:
    print(number, "is not an Armstrong number.")
```



```
Enter a number: 153
153 is an Armstrong number.
```

12.Write a Python Program to Find HCF.

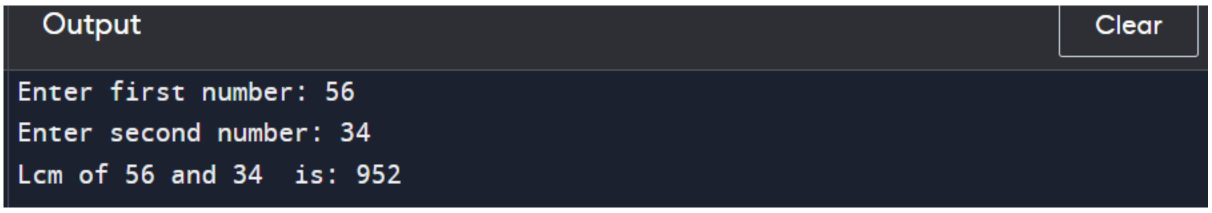
```
def hcf(num1, num2):
    result = 1
    if num1 < num2:
        smaller = num1
    else:
        smaller = num2
    for r in range(smaller+1,1,-1):
        if (num1 % r == 0) and (num2 % r == 0):
            result = r
            break
    return result

num1 =int(input("Enter first number: "))
num2 =int(input("Enter second number: "))
print("Hcf of" ,num1, "and", num2, " is:" ,hcf(num1, num2))
```



```
Enter first number: 45
Enter second number: 95
Hcf of 45 and 95  is: 5
```

13.Write a Python Program to Find LCM.

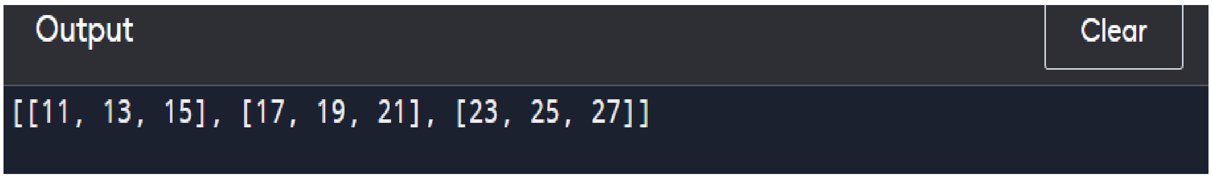
```
def lcm(num1, num2):
    result = 1
    if num1 < num2:
        smaller = num1
    else:
        smaller = num2
    for r in range(smaller+1,1,-1):
        if (num1 % r == 0) and (num2 % r == 0):
            result = r
            break
    lcm_res = (num1*num2)//result
    return lcm_res

num1 =int(input("Enter first number: "))
num2 =int(input("Enter second number: "))
print("Lcm of" ,num1, "and", num2, " is:" ,lcm(num1, num2))
```



```
Enter first number: 56
Enter second number: 34
Lcm of 56 and 34  is: 952
```

14.Write a Python Program to Add Two Matrices.

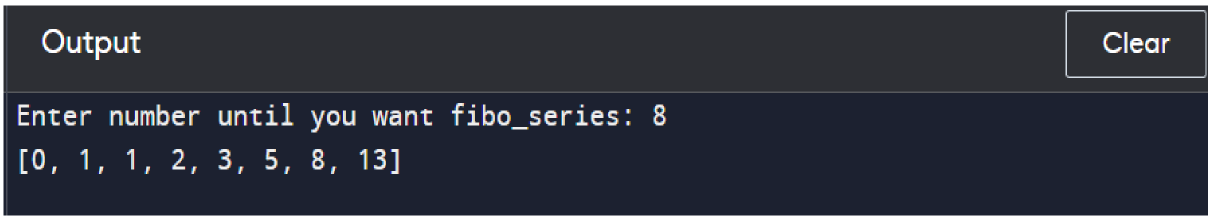
```
X = [[1,2,3], [4,5,6], [7,8,9]]

Y = [[10,11,12], [13,14,15], [16,17,18]]

result = [[0,0,0], [0,0,0], [0,0,0]]

for r in range(len(X)):
   for j in range(len(X[0])):
       result[i][j] = X[i][j] + Y[i][j]
for r in result:
   print(result)
```



```
[[11, 13, 15], [17, 19, 21], [23, 25, 27]]
```

15. Write a Python program to generate list of Fibonacci number up to n Fibonacci numbers.

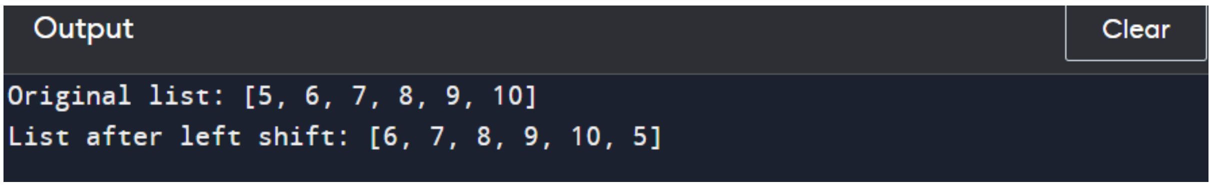
```
def fibo(num):
    a=-1
    b=1
    list = []
    for i in range(0,num):
        c=a+b
        a=b
        b=c
        list.append(c)
    return list

num= int(input("Enter number until you want fibo_series: "))
print(fibo(num))
```



```
Enter number until you want fibo_series: 8
[0, 1, 1, 2, 3, 5, 8, 13]
```

16. Write a Python program which takes a list and returns a list with the elements
"Shifted left by one position" so [1, 2, 3] yields [2, 3, 1].
Example: [1, 2, 3] → [2, 3, 1] & [11, 12, 13] → [12, 13, 11]

```
def left_shift(lst, shift_count):
    shift_count %= len(lst)
    return lst[shift_count:] + lst[:shift_count]

my_list = [5, 6, 7, 8, 9, 10]
shifted_list = left_shift(my_list, 1)

print("Original list:", my_list)
print("List after left shift:", shifted_list)
```

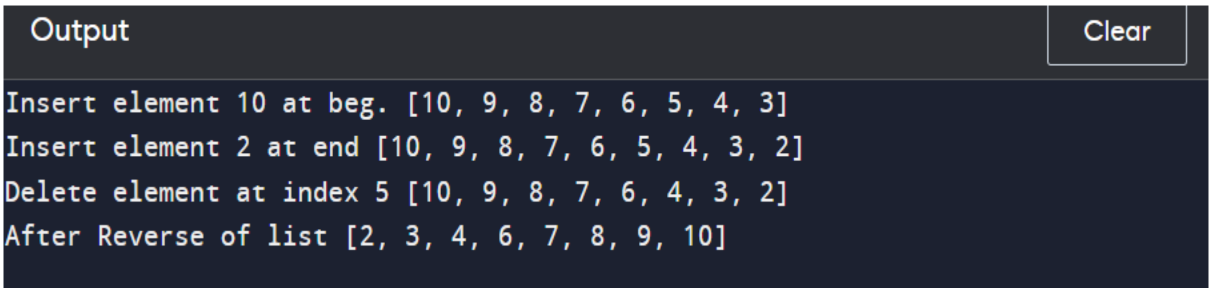


```
Original list: [5, 6, 7, 8, 9, 10]
List after left shift: [6, 7, 8, 9, 10, 5]
```

17. Consider the list lst=[9,8,7,6,5,4,3]. Write the Python program which performs the following operation:

A. Insert element 10 at beginning of the list.
B. Insert element 2 at end of the list.
C. Delete the element at index position 5.
D. Print all elements in reverse order.

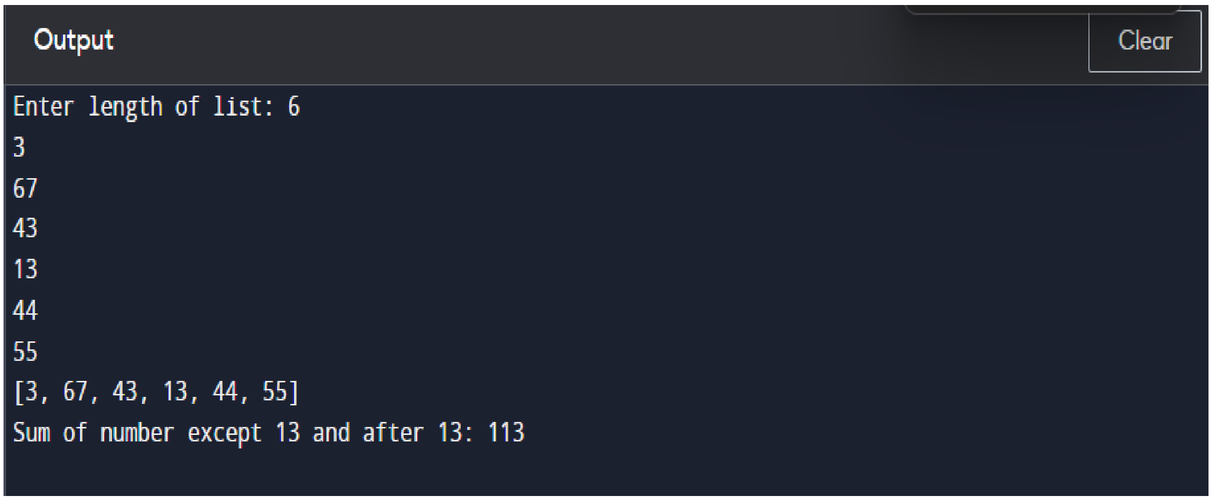
```
list = [9,8,7,6,5,4,3]

list.insert(0,10)
print("Insert element 10 at beg.", list)

list.insert(len(list)+1,2)
print("Insert element 2 at end", list)

list.pop(5)
print("Delete element at index 5", list)

list.reverse()
print("After Reverse of list", list)
```



```
Insert element 10 at beg. [10, 9, 8, 7, 6, 5, 4, 3]
Insert element 2 at end [10, 9, 8, 7, 6, 5, 4, 3, 2]
Delete element at index 5 [10, 9, 8, 7, 6, 4, 3, 2]
After Reverse of list [2, 3, 4, 6, 7, 8, 9, 10]
```

18. Write a Python program which will return the sum of the numbers in the array, returning 0 for an empty array.
Except the number 13 is very unlucky, so it does not count and number that come immediately after 13 also do not count.
Example : [1, 2, 3, 4] = 10 , [1, 2, 3, 4, 13] = 10 , [13, 1, 2, 3, 13] = 5

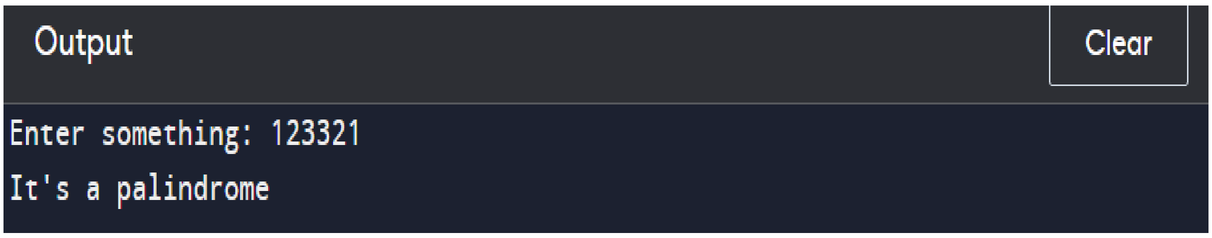
```
def sum_lucky(list):
    sum=0

    for i in range(0,n):
        if list[i] != 13:
             sum+= list[i]
        else:
            break
    return sum

list=[]
n=int(input("Enter length of list: "))
for i in range(0,n):
    list.append(int(input()))
print(list)
print("Sum of number except 13 and after 13:" , sum_lucky(list))
```



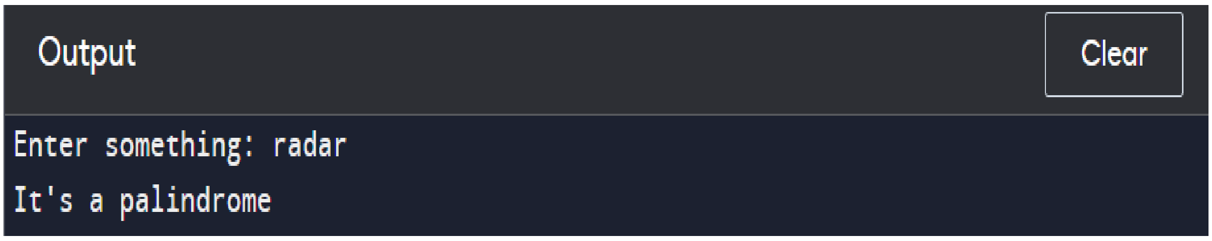
```
Enter length of list: 6
3
67
43
13
44
55
[3, 67, 43, 13, 44, 55]
Sum of number except 13 and after 13: 113
```

19. Write a Python program to Check Whether a String is Palindrome or Not.

```
def palindrome(str):
    s=len(str)
    mid = s//2
    i=0
    flag =0
    while i<mid:
        if str[i] == str[s-1]:
            flag =flag+1
            i=i+1
            s=s-1

    if flag == mid:
        print("It's a palindrome")
    else:
        print("It's not a palindrome")

str = input("Enter something: ")
palindrome(str)
```



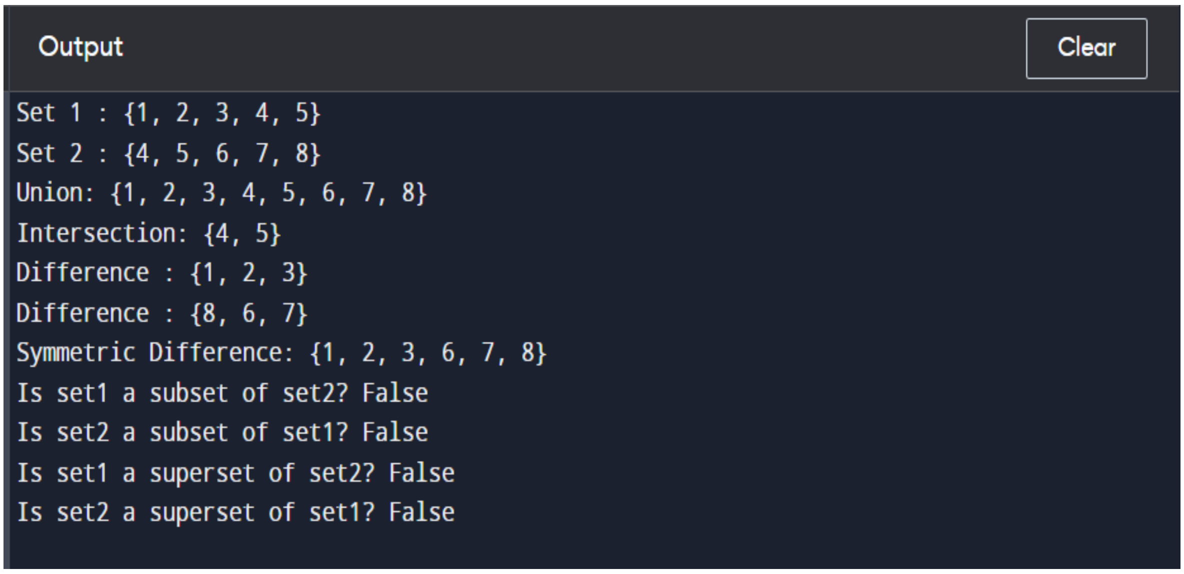
```
Enter something: 123321
It's a palindrome
```



```
Enter something: radar
It's a palindrome
```

20. Write a Python program to Illustrate Different Set Operations.

```
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}
print("Set 1 :",set1)
print("Set 2 :",set2)

union_set = set1.union(set2)
print("Union:", union_set)

intersection_set = set1.intersection(set2)
print("Intersection:", intersection_set)

difference_set1 = set1.difference(set2)
print("Difference :", difference_set1)

difference_set2 = set2.difference(set1)
print("Difference :", difference_set2)

symmetric_difference_set = set1.symmetric_difference(set2)
print("Symmetric Difference:", symmetric_difference_set)

is_subset = set1.issubset(set2)
print("Is set1 a subset of set2?", is_subset)

is_subset = set2.issubset(set1)
print("Is set2 a subset of set1?", is_subset)

is_superset = set1.issuperset(set2)
print("Is set1 a superset of set2?", is_superset)

is_superset = set2.issuperset(set1)
print("Is set2 a superset of set1?", is_superset)
```



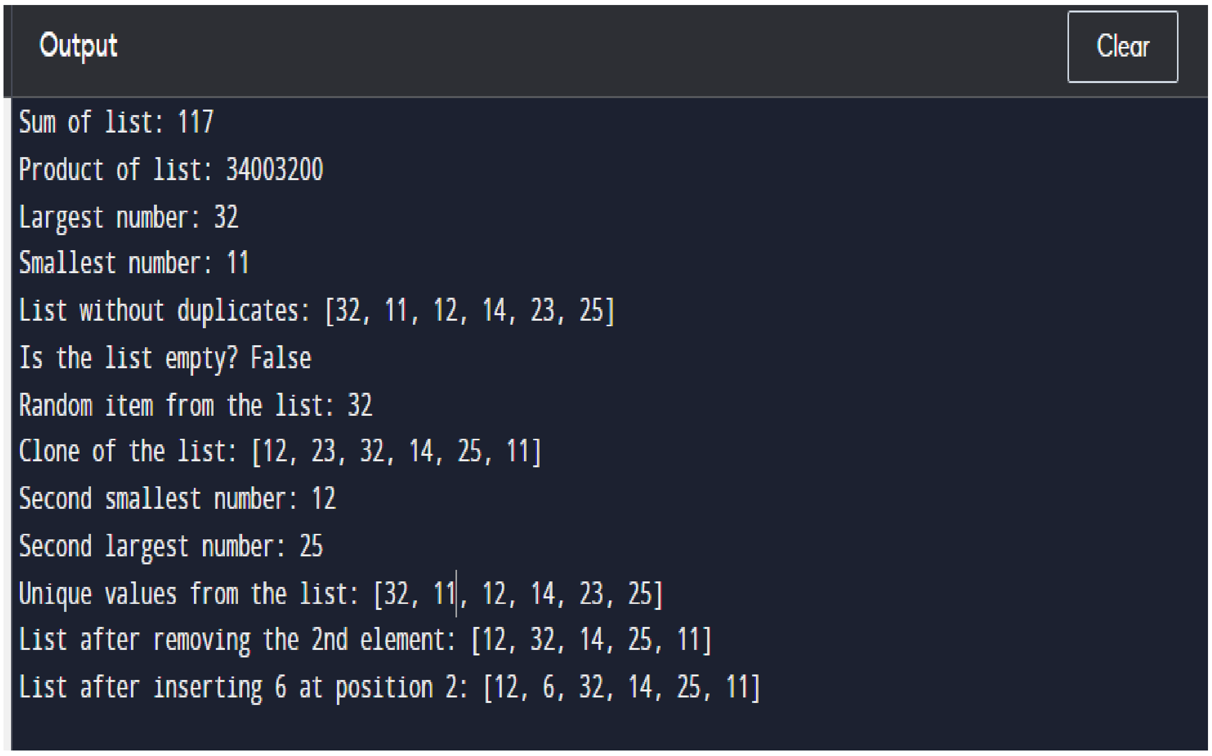
```
Set 1 : {1, 2, 3, 4, 5}
Set 2 : {4, 5, 6, 7, 8}
Union: {1, 2, 3, 4, 5, 6, 7, 8}
Intersection: {4, 5}
Difference : {1, 2, 3}
Difference : {8, 6, 7}
Symmetric Difference: {1, 2, 3, 6, 7, 8}
Is set1 a subset of set2? False
Is set2 a subset of set1? False
Is set1 a superset of set2? False
Is set2 a superset of set1? False
```

21. Write a python program to do the following.

A. To sum all the items in a list.
B. To multiplies all the items in a list.
C. To get largest number from a list.
D. To get the smallest number from a list.
E. To remove duplicates from a list.
F. To check a list is empty or not.
G. To select an item randomly from a list.
H. To clone or copy a list.
I. To find the second smallest number in a list.
J. To find the second largest number in a list.
K. To get unique values from a list.
L. To remove the K'th element from a given list, print the new list.
M. To insert an element at specified position into a given list.

```
import random

# A. Sum all the items in a list.
def sum_list(lst):
    return sum(lst)

# B. Multiply all the items in a list.
def multiply_list(lst):
    result = 1
    for item in lst:
        result *= item
    return result

# C. Get the largest number from a list.
def largest_number(lst):
    return max(lst)

# D. Get the smallest number from a list.
def smallest_number(lst):
    return min(lst)

# E. Remove duplicates from a list.
def remove_duplicates(lst):
    return list(set(lst))

# F. Check if a list is empty or not.
def is_empty(lst):
    return len(lst) == 0

# G. Select an item randomly from a list.
def random_item(lst):
    return random.choice(lst)

# H. Clone or copy a list.
def clone_list(lst):
    return lst.copy()

# I. Find the second smallest number in a list.
def second_smallest_number(lst):
    unique_sorted = sorted(set(lst))
    if len(unique_sorted) >= 2:
        return unique_sorted[1]
    else:
        return "List has less than two unique elements"

# J. Find the second largest number in a list.
def second_largest_number(lst):
    unique_sorted = sorted(set(lst))
    if len(unique_sorted) >= 2:
        return unique_sorted[-2]
    else:
        return "List has less than two unique elements"

# K. Get unique values from a list.
def unique_values(lst):
    return list(set(lst))

# L. Remove the K’th element from a given list, print the new list.
def remove_kth_element(lst, k):
    if 0 <= k < len(lst):
        del lst[k]
    return lst

# M. Insert an element at specified position into a given list.
def insert_element_at_position(lst, element, position):
    lst.insert(position, element)
    return lst

# Example usage:
my_list = [12, 23, 32, 14, 25, 11]
```

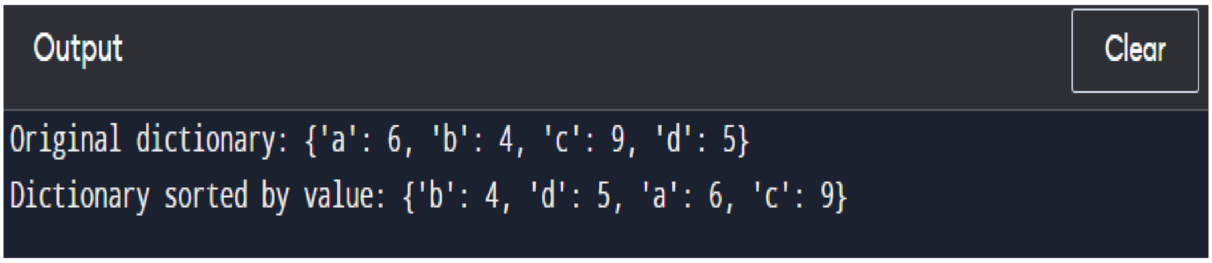
```
print("Sum of list:", sum_list(my_list))
print("Product of list:", multiply_list(my_list))
print("Largest number:", largest_number(my_list))
print("Smallest number:", smallest_number(my_list))
print("List without duplicates:", remove_duplicates(my_list))
print("Is the list empty?", is_empty(my_list))
print("Random item from the list:", random_item(my_list))
print("Clone of the list:", clone_list(my_list))
print("Second smallest number:", second_smallest_number(my_list))
print("Second largest number:", second_largest_number(my_list))
print("Unique values from the list:", unique_values(my_list))
print("List after removing the 2nd element:", remove_kth_element(my_list,
1))
print("List after inserting 6 at position 2:",
insert_element_at_position(my_list, 6, 1))
```



```
Sum of list: 117
Product of list: 34003200
Largest number: 32
Smallest number: 11
List without duplicates: [32, 11, 12, 14, 23, 25]
Is the list empty? False
Random item from the list: 32
Clone of the list: [12, 23, 32, 14, 25, 11]
Second smallest number: 12
Second largest number: 25
Unique values from the list: [32, 11, 12, 14, 23, 25]
List after removing the 2nd element: [12, 32, 14, 25, 11]
List after inserting 6 at position 2: [12, 6, 32, 14, 25, 11]
```

22.Write a Python program to sort a dictionary by value.

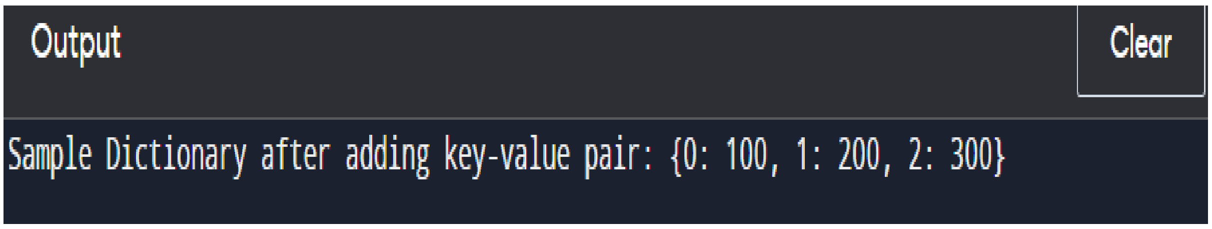
```
def sort_dict_value(dictionary):
    sorted_dict = dict(sorted(dictionary.items(), key=lambda item: item[1]))
    return sorted_dict

my_dict = {'a': 6, 'b': 4, 'c': 9, 'd':5}
sort_dict = sort_dict_value(my_dict)

print("Original dictionary:", my_dict)
print("Dictionary sorted by value:", sort_dict)
```



```
Original dictionary: {'a': 6, 'b': 4, 'c': 9, 'd': 5}
Dictionary sorted by value: {'b': 4, 'd': 5, 'a': 6, 'c': 9}
```

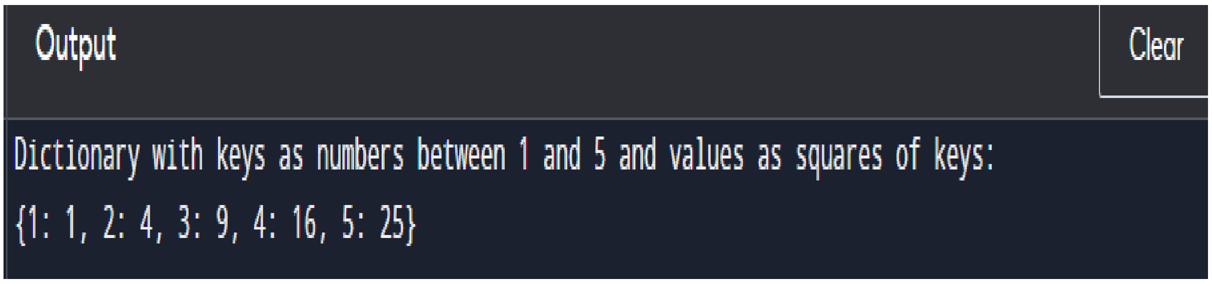
23. Write a python program to add a key to a dictionary.
Sample Dictionary: {0:100, 1:200}
Expected Result: {0:100, 1:200, 2:300}
Write a python program to print a dictionary where the keys are number between 1 and 5 (both included) and the values are square of keys.
Sample Dictionary: {1:1, 2:4, 3:9, 4:16, 5:25}

```
def add_key_dict(dictionary, key, value):
    dictionary[key] = value

sample_dict = {0: 100, 1: 200}

add_key_dict(sample_dict, 2, 300)

print("Sample Dictionary after adding key-value pair:", sample_dict)
```



```
Sample Dictionary after adding key-value pair: {0: 100, 1: 200, 2: 300}
```

```
def generate_square_dict(start, end):
    square_dict = {}
    for num in range(start, end + 1):
        square_dict[num] = num ** 2
    return square_dict


square_dict = generate_square_dict(1, 5)

print("Dictionary with keys as numbers between 1 and 5 and values as squares of keys:")
print(square_dict)
```



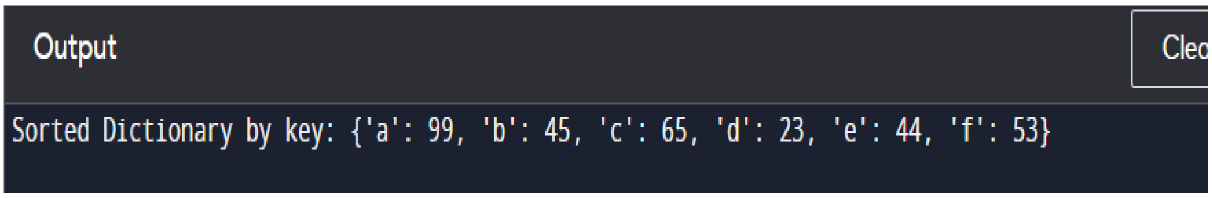
```
Dictionary with keys as numbers between 1 and 5 and values as squares of keys:
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

24. Write a python program to do the following:

A. To sort a dictionary by key.
B. To get the maximum and minimum value in a dictionary.
C. To remove duplicates from Dictionary.

# A. Sorting a dictionary by key

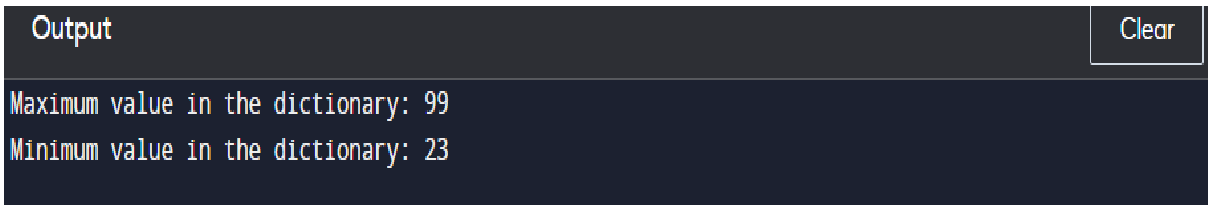
```
def sort_dict_by_key(dictionary):
    sorted_dict = dict(sorted(dictionary.items()))
    return sorted_dict

sample_dict = {'b': 45, 'a': 99, 'c': 65, 'd': 23, 'e': 44, 'f': 53}

sorted_dict = sort_dict_by_key(sample_dict)
print("Sorted Dictionary by key:", sorted_dict)
```



```
Sorted Dictionary by key: {'a': 99, 'b': 45, 'c': 65, 'd': 23, 'e': 44, 'f': 53}
```

# B. Getting the maximum and minimum value in a dictionary

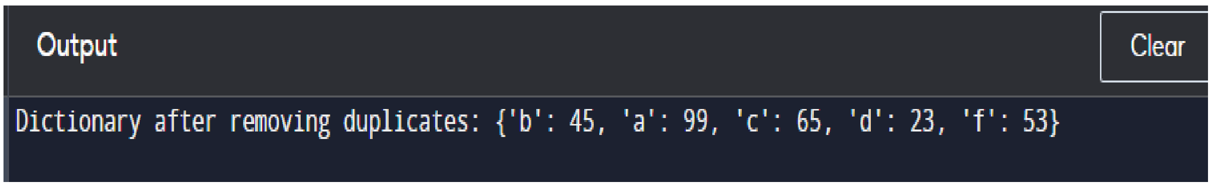
```
def get_max_min_value(dictionary):
    max_value = max(dictionary.values())
    min_value = min(dictionary.values())
    return max_value, min_value

sample_dict = {'b': 45, 'a': 99, 'c': 65, 'd': 23, 'e': 44, 'f': 53}

max_value, min_value = get_max_min_value(sample_dict)
print("Maximum value in the dictionary:", max_value)
print("Minimum value in the dictionary:", min_value)
```



```
Maximum value in the dictionary: 99
Minimum value in the dictionary: 23
```

# C. Removing duplicates from a dictionary

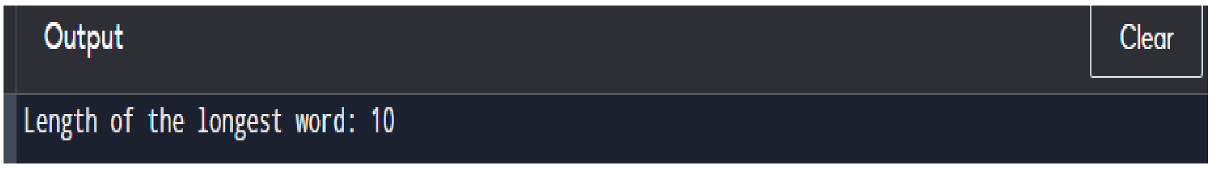
```
def remove_duplicates_from_dict(dictionary):
    unique_dict = {}
    for key, value in dictionary.items():
        if value not in unique_dict.values():
            unique_dict[key] = value
    return unique_dict

sample_dict = {'b': 45, 'a': 99, 'c': 65, 'd': 23, 'e': 45, 'f': 53}

unique_dict = remove_duplicates_from_dict(sample_dict)
print("Dictionary after removing duplicates:", unique_dict)
```



```
Dictionary after removing duplicates: {'b': 45, 'a': 99, 'c': 65, 'd': 23, 'f': 53}
```

25. Write a function find_longest_word() that takes a list of words and returns the length of the largest one.

```
def find_longest_word(words):
    if not words:
        return 0
    max_length = len(words[0])
    for word in words[1:]:
        if len(word) > max_length:
            max_length = len(word)
    return max_length

word = ["apple", "banana", "orange", "strawberry", "kiwi"]
print("Length of the longest word:", find_longest_word(word))
```

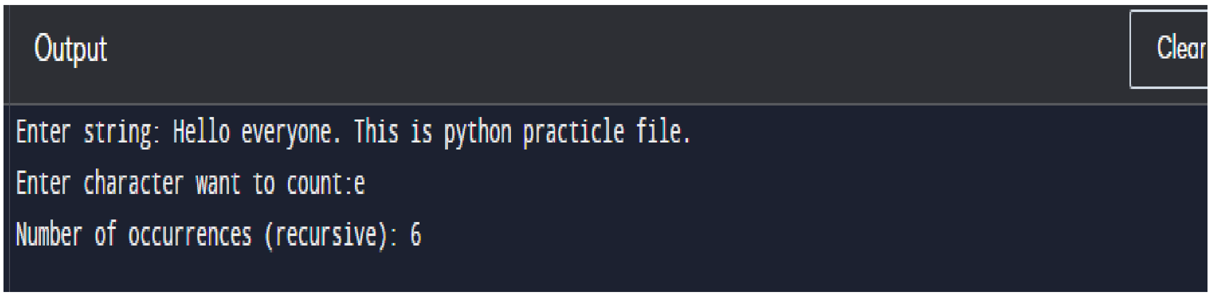


```
Length of the longest word: 10
```

26. Write a python program that counts the number of occurrences of the character in the given string using function. Provide two implementation: recursive and iterative.

<u>Using Rcursion:-</u>

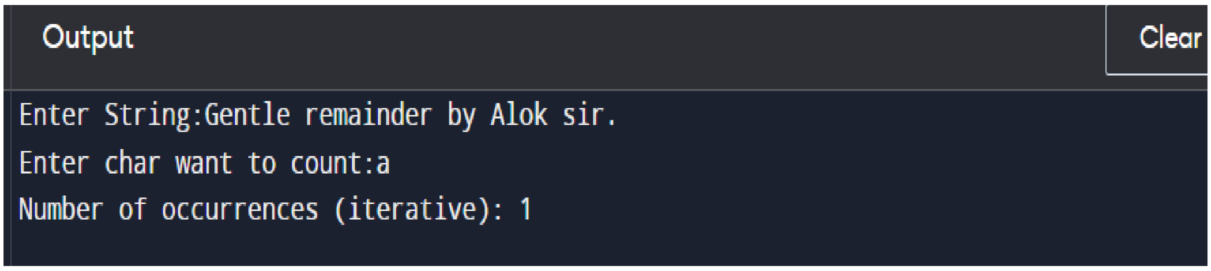
```
def recursive(string, char):
    if len(string) == 0:
        return 0
    if string[0] == char:
        return 1 + recursive(string[1:], char)
    else:
        return recursive(string[1:], char)

input_str = input("Enter string: ")
char = input("Enter character want to count:")

print("Number of occurrences (recursive):", recursive(input_str, char))
```



```
Enter string: Hello everyone. This is python practicle file.
Enter character want to count:e
Number of occurrences (recursive): 6
```

<u>Using Iteration:-</u>

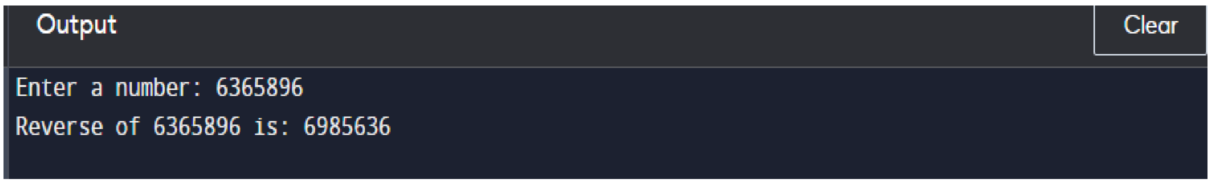
```
def iterative(string, char):
    count = 0
    for c in string:
        if c== char:
            count = count +1
    return count

input_str = input("Enter String:")
char = input("Enter char want to count:")
print("Number of occurrences (iterative):",iterative(input_str, char))
```



```
Enter String:Gentle remainder by Alok sir.
Enter char want to count:a
Number of occurrences (iterative): 1
```

27. Write a python program to find reverse of given number using user defined function.

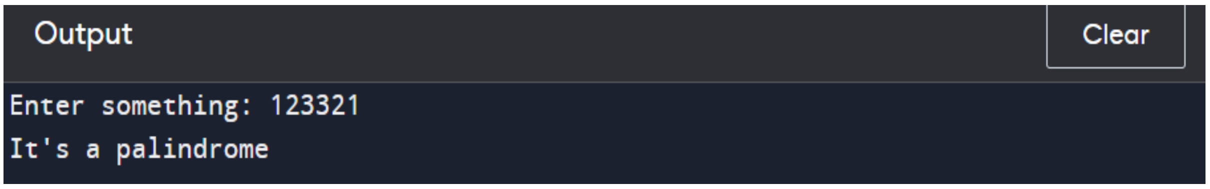
```
def reverse_number(num):
    reversed_num = 0
    while num > 0:
        digit = num % 10
        reversed_num = (reversed_num * 10) + digit
        num = num // 10
    return reversed_num
num = int(input("Enter a number: "))
print("Reverse of", num, "is:", reverse_number(num))
```



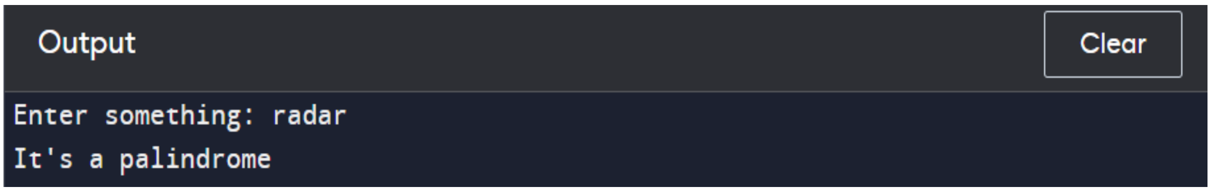
```
Enter a number: 6365896
Reverse of 6365896 is: 6985636
```

28. Write a python program to implement is palindrome() function to check given string is palindrome or no.

```
def palindrome(str):
    s=len(str)
    mid = s//2
    i=0
    flag =0
    while i<mid:
        if str[i] == str[s-1]:
            flag =flag+1
            i=i+1
            s=s-1
    if flag == mid:
        print("It's a palindrome")
    else:
        print("It's not a palindrome")
str = input("Enter something: ")
palindrome(str)
```



```
Enter something: 123321
It's a palindrome
```

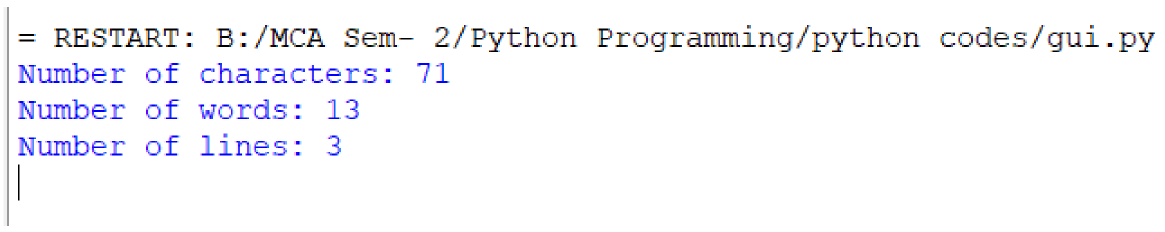


```
Enter something: radar
It's a palindrome
```

29. Write a program to compute the number of characters, words, and lines in a file.

```
def count_chars_words_lines(filename):
    try:
        with open("python.txt", "r") as file:
            lines = file.readlines()
            num_lines = len(lines)
            num_words = sum(len(line.split()) for line in lines)
            num_chars = sum(len(line) for line in lines)
        return num_chars, num_words, num_lines
    except FileNotFoundError:
        print("File not found.")
        return 0, 0, 0


num_chars, num_words, num_lines =
count_chars_words_lines("python.txt")
print("Number of characters:", num_chars)
print("Number of words:", num_words)
print("Number of lines:", num_lines)
```

Output:-

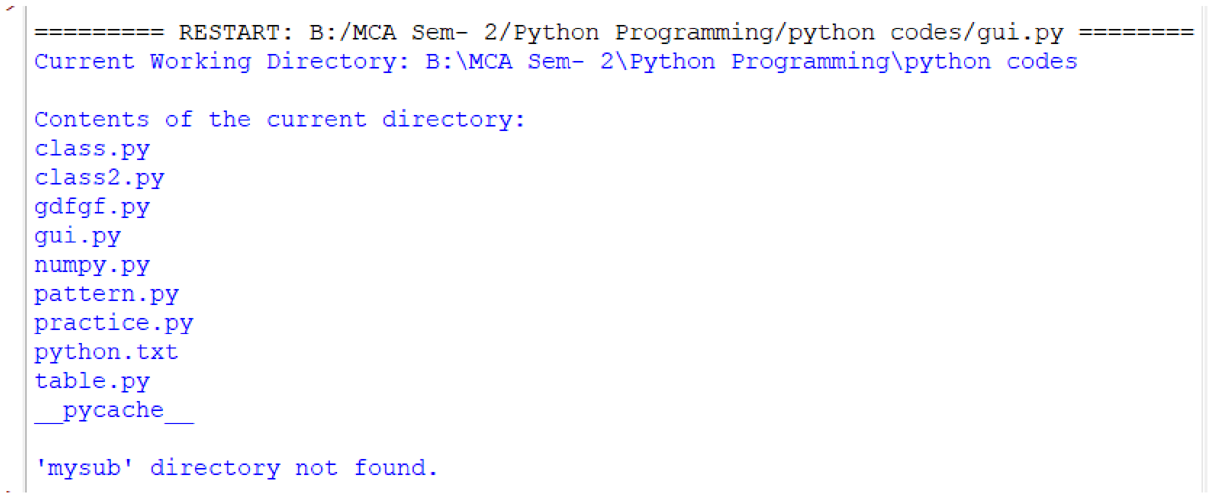


```
= RESTART: B:/MCA Sem- 2/Python Programming/python codes/gui.py
Number of characters: 71
Number of words: 13
Number of lines: 3
```

30. Write a python program to know the current working directory and to print all contents of the current directory. What changes we need to make in the program if we wish to display the contents of only 'mysub' directory available in current directory?

```
import os

# Get and print the current working directory
current_directory = os.getcwd()
print("Current Working Directory:", current_directory)

# Print all contents of the current directory
print("\nContents of the current directory:")
for item in os.listdir(current_directory):
    print(item)

# To display contents of only 'mysub' directory:
mysub_directory = os.path.join(current_directory, 'mysub')

# Check if 'mysub' directory exists
if os.path.exists(mysub_directory) and os.path.isdir(mysub_directory):
    print("\nContents of 'mysub' directory:")
    for item in os.listdir(mysub_directory):
        print(item)
else:
    print("\n'mysub' directory not found.")
```

Output:-

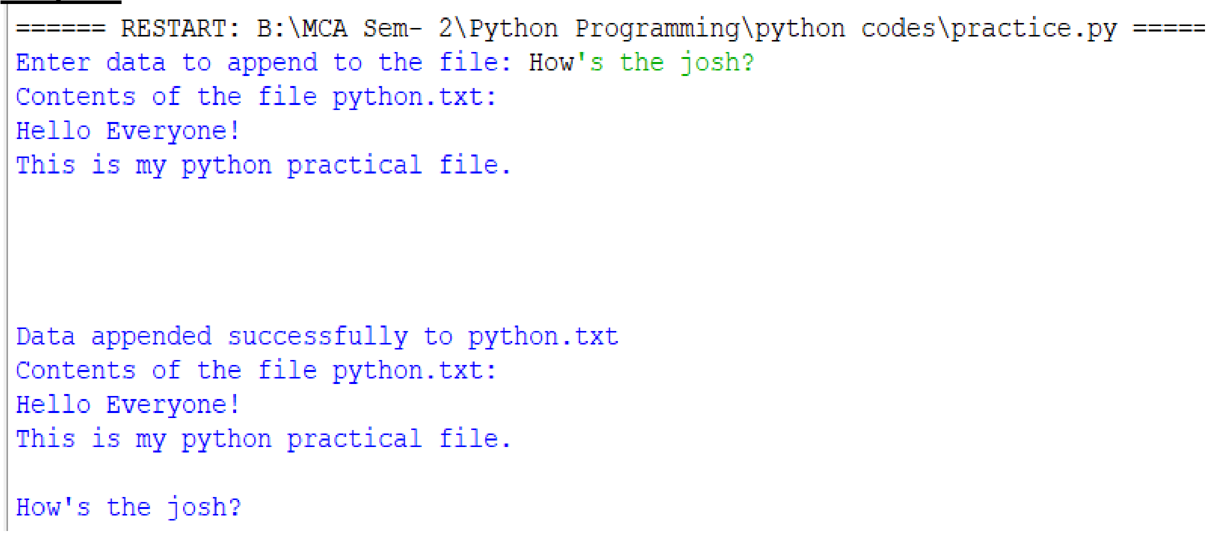


```
========= RESTART: B:/MCA Sem- 2/Python Programming/python codes/gui.py ========
Current Working Directory: B:\MCA Sem- 2\Python Programming\python codes

Contents of the current directory:
class.py
class2.py
gdfgf.py
gui.py
numpy.py
pattern.py
practice.py
python.txt
table.py
__pycache__

'mysub' directory not found.
```

31. Write a python program to append data to an existing file ‘python.py’. Read data to be appended from the user. Then display the contents of entire file.

```
def append_file(filename, data):
    try:
        with open(filename, "a") as file:
            file.write(data + "\n")
        print("Data appended successfully to", filename)
    except FileNotFoundError:
        print("File not found.")

def display_file_contents(filename):
    try:
        with open(filename, "r") as file:
            content = file.read()
            print("Contents of the file", filename + ":\n" + content)
            print("\n")
    except FileNotFoundError:
        print("File not found.")

filename = "python.txt"
data_append = input("Enter data to append to the file: ")

display_file_contents(filename)

append_file(filename, data_append)

display_file_contents(filename)
```

Output:-

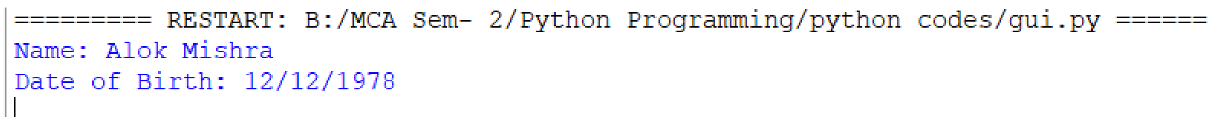


```
====== RESTART: B:\MCA Sem- 2\Python Programming\python codes\practice.py =====
Enter data to append to the file: How's the josh?
Contents of the file python.txt:
Hello Everyone!
This is my python practical file.




Data appended successfully to python.txt
Contents of the file python.txt:
Hello Everyone!
This is my python practical file.

How's the josh?
```

32. Write a python program to retrieve name and date of birth (mm-dd-yyyy) of students from a given file student.txt.

```
def student_info(filename):
    try:
        with open(filename, "r") as file:
            for line in file:
                # Split each line into name and date of birth
                name, dob = line.strip().split(",")
                print("Name:", name)
                print("Date of Birth:", dob)
    except FileNotFoundError:
        print("File not found.")

filename = "student.txt"
student_info(filename)
```

<u>Output:-</u>

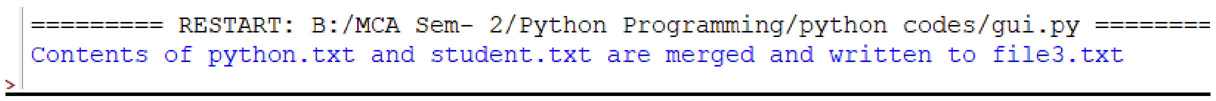


```
========= RESTART: B:/MCA Sem- 2/Python Programming/python codes/gui.py ======
Name: Alok Mishra
Date of Birth: 12/12/1978
```

33. Write a python program to read line by line from a given files file1 & file2 and write into file3.

```
def merge_files(file1, file2, file3):
    try:
        with open(file1, "r") as f1, open(file2, "r") as f2, open(file3, "w") as f3:
            for line in f1:
                f3.write(line)
            for line in f2:
                f3.write(line)
        print("Contents of", file1, "and", file2, "are merged and written to",
file3)
    except FileNotFoundError:
        print("One or more files not found.")

file1 = "python.txt"
file2 = "student.txt"
file3 = "file3.txt"

merge_files(file1, file2, file3)
```

<u>Output:-</u>

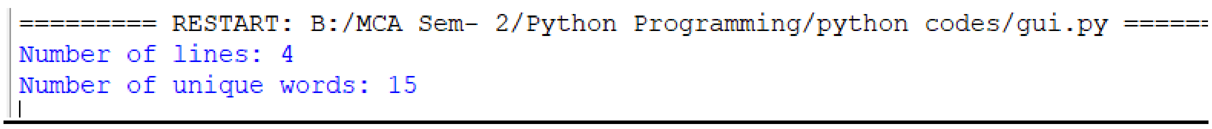


```
========= RESTART: B:/MCA Sem- 2/Python Programming/python codes/gui.py ========
Contents of python.txt and student.txt are merged and written to file3.txt
>
```

34.Read a text file in python and print no. of lines and no. of unique words.

```
def count_lines_and_unique_words(filename):
    try:
        with open(filename, "r") as file:
            lines = file.readlines()
            num_lines = len(lines)


            unique_words = set()
            for line in lines:
                words = line.split()
                unique_words.update(words)

            num_unique_words = len(unique_words)

            return num_lines, num_unique_words
    except FileNotFoundError:
        print("File not found.")
        return 0, 0


filename = "python.txt"
num_lines, num_unique_words = count_lines_and_unique_words(filename)
print("Number of lines:", num_lines)
print("Number of unique words:", num_unique_words)
```

<u>Output:-</u>

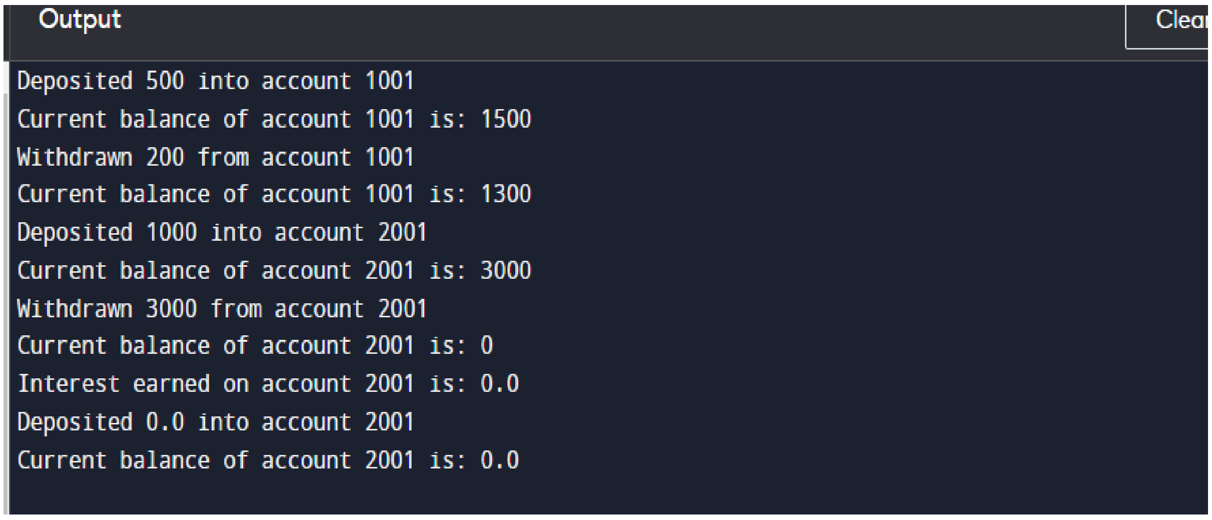


```
========= RESTART: B:/MCA Sem- 2/Python Programming/python codes/gui.py ======
Number of lines: 4
Number of unique words: 15
```

35.Write a python program to illustrate the oops concepts.

```
class BankAccount:
    def __init__(self, account_number, balance):
        self.account_number = account_number
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        print("Deposited", amount, "into account", self.account_number)
        self.display_balance()

    def withdraw(self, amount):
        if self.balance >= amount:
            self.balance -= amount
            print("Withdrawn", amount, "from account", self.account_number)
        else:
            print("Insufficient funds to withdraw", amount, "from account",
self.account_number)
        self.display_balance()

    def display_balance(self):
        print("Current balance of account", self.account_number, "is:",
self.balance)


class SavingsAccount(BankAccount):
    def __init__(self, account_number, balance, interest_rate):
        super().__init__(account_number, balance)
        self.interest_rate = interest_rate

    def calculate_interest(self):
        interest_amount = self.balance * (self.interest_rate / 100)
        print("Interest earned on account", self.account_number, "is:",
interest_amount)
        return interest_amount



account1 = BankAccount(1001, 1000)
account2 = SavingsAccount(2001, 2000, 5)
```

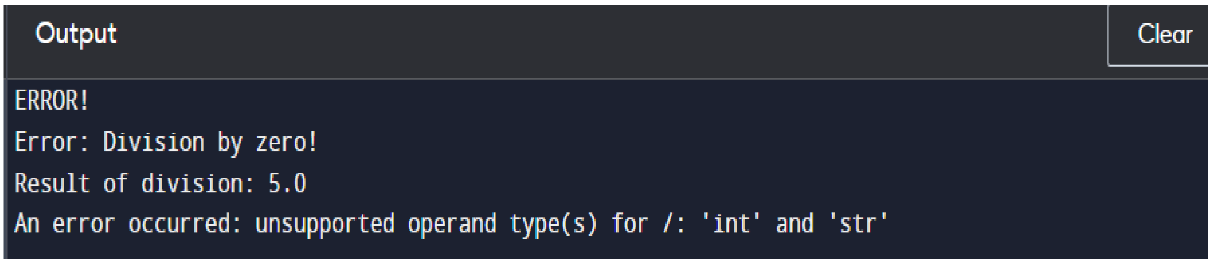
```
account1.deposit(500)
account1.withdraw(200)
account2.deposit(1000)
account2.withdraw(3000)

interest = account2.calculate_interest()
account2.deposit(interest)
```



```
Deposited 500 into account 1001
Current balance of account 1001 is: 1500
Withdrawn 200 from account 1001
Current balance of account 1001 is: 1300
Deposited 1000 into account 2001
Current balance of account 2001 is: 3000
Withdrawn 3000 from account 2001
Current balance of account 2001 is: 0
Interest earned on account 2001 is: 0.0
Deposited 0.0 into account 2001
Current balance of account 2001 is: 0.0
```

36.Write a python to illustrate the exception handling.

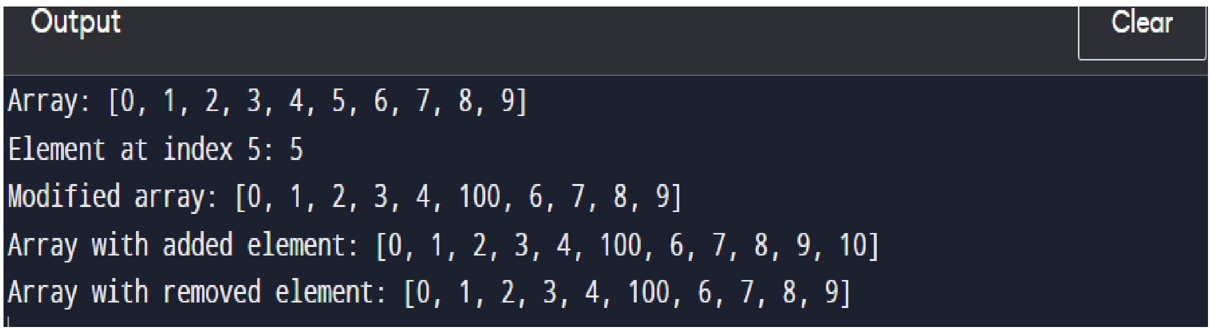
```
def divide(x, y):
    try:
        result = x / y
        print("Result of division:", result)
    except ZeroDivisionError:
        print("Error: Division by zero!")
    except Exception as e:
        print("An error occurred:", e)


divide(10, 0)

divide(10, 2)

divide(10, 'a')
```



```
ERROR!
Error: Division by zero!
Result of division: 5.0
An error occurred: unsupported operand type(s) for /: 'int' and 'str'
```

37.Write a python to create one dimensional array.

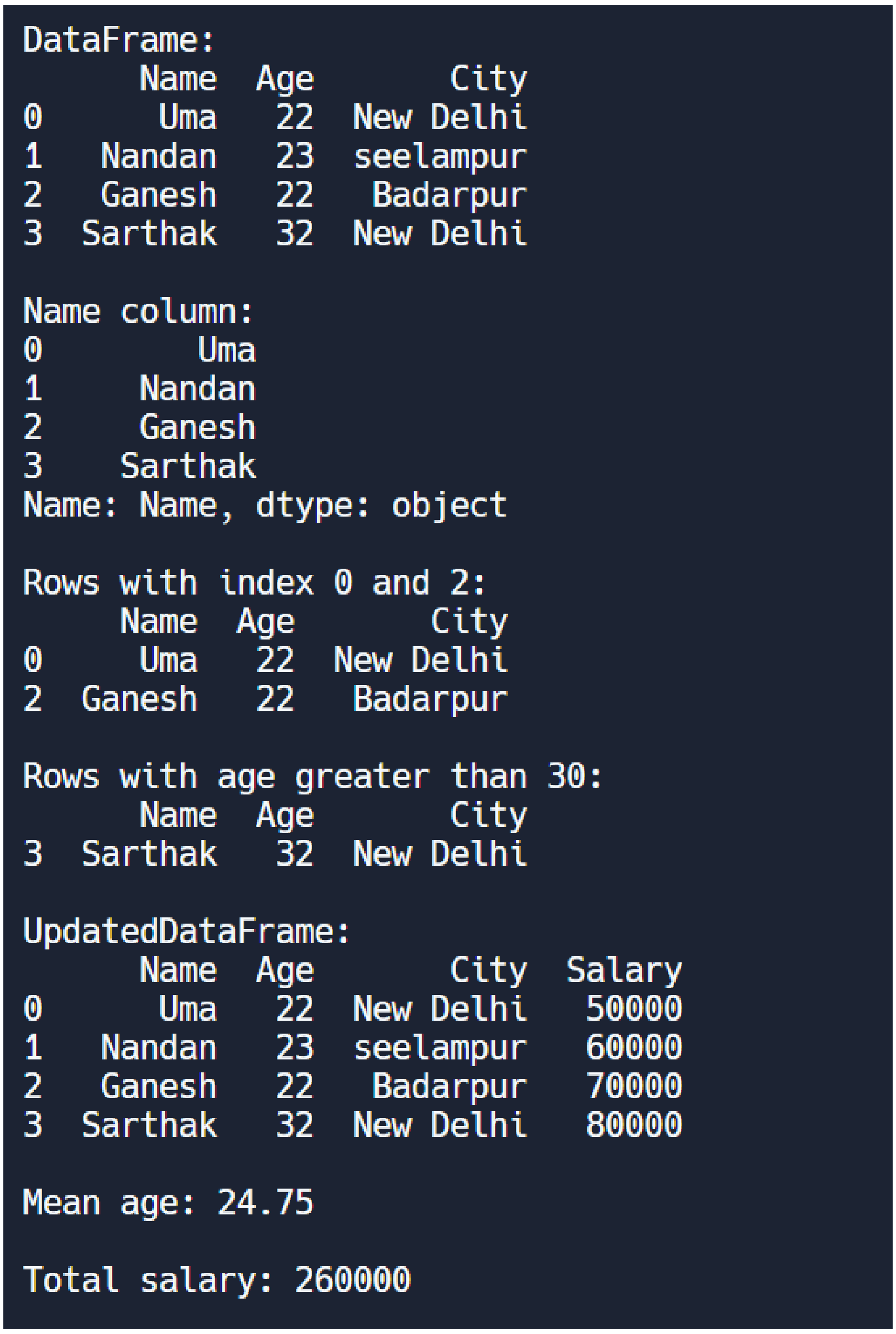
```
array = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print("Array:", array)

print("Element at index 5:",array[5])

array[5] = 100
print("Modified array:", array)

array.append(10)
print("Array with added element:", array)

array.pop()
print("Array with removed element:", array)
```



```
Array: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Element at index 5: 5
Modified array: [0, 1, 2, 3, 4, 100, 6, 7, 8, 9]
Array with added element: [0, 1, 2, 3, 4, 100, 6, 7, 8, 9, 10]
Array with removed element: [0, 1, 2, 3, 4, 100, 6, 7, 8, 9]
```

38.Write a python program to illustrate pandas library.

```
import pandas as pd

data = {
'Name': ['Uma', 'Nandan', 'Ganesh', 'Sarthak'], 'Age': [22, 23, 22, 32],
'City': ['New Delhi', 'Seelampur', 'Badarpur', 'New Delhi']
}

df = pd.DataFrame(data)

print("DataFrame:")
print(df)

print("\nName column:")
print(df['Name'])

print("\nRows with index 0 and 2:")
print(df.loc[[0, 2]])

print("\nRows with age greater than 30:")
print(df[df['Age'] > 30])

df['Salary'] = [50000, 60000, 70000, 80000]
print("\nUpdatedDataFrame:")
print(df)

mean_age = df['Age'].mean()
print("\nMean age:", mean_age)

total_salary = df['Salary'].sum()
print("\nTotal salary:", total_salary)
```

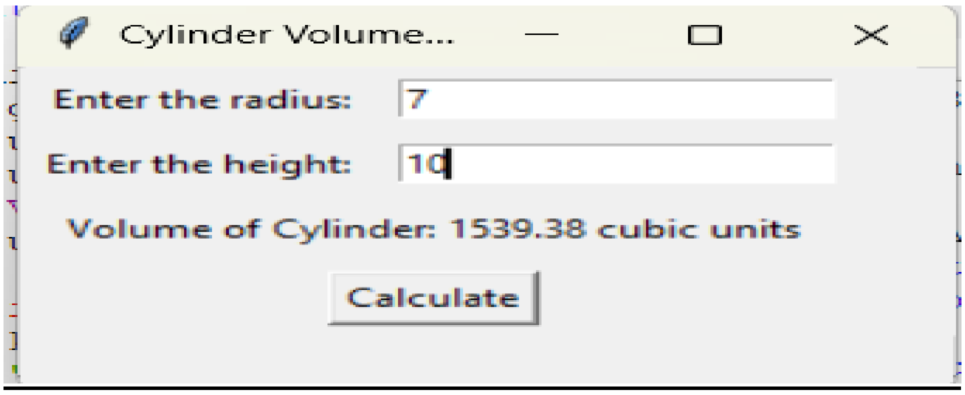


```
DataFrame:
      Name  Age       City
0      Uma   22  New Delhi
1   Nandan   23  seelampur
2   Ganesh   22   Badarpur
3  Sarthak   32  New Delhi

Name column:
0        Uma
1     Nandan
2     Ganesh
3    Sarthak
Name: Name, dtype: object

Rows with index 0 and 2:
     Name  Age       City
0     Uma   22  New Delhi
2  Ganesh   22   Badarpur

Rows with age greater than 30:
      Name  Age       City
3  Sarthak   32  New Delhi

UpdatedDataFrame:
      Name  Age       City  Salary
0      Uma   22  New Delhi   50000
1   Nandan   23  seelampur   60000
2   Ganesh   22   Badarpur   70000
3  Sarthak   32  New Delhi   80000

Mean age: 24.75

Total salary: 260000
```

39.Write a python program using Tkinter to create the GUI Interface.

```
import tkinter as tk
from math import pi

def calculate_volume():
    try:
        radius = float(radius_entry.get())
        height = float(height_entry.get())
        volume = pi * radius ** 2 * height
        result_label.config(text=f"Volume of Cylinder: {volume:.2f} cubic
units")
    except ValueError:
        result_label.config(text="Please enter valid numeric values.")

root = tk.Tk()
root.title("Cylinder Volume Calculator")
radius_label = tk.Label(root, text="Enter the radius:")
radius_label.grid(row=0, column=0, padx=5, pady=5, sticky="e")
radius_entry = tk.Entry(root)
radius_entry.grid(row=0, column=1, padx=5, pady=5)
height_label = tk.Label(root, text="Enter the height:")
height_label.grid(row=1, column=0, padx=5, pady=5, sticky="e")
height_entry = tk.Entry(root)
height_entry.grid(row=1, column=1, padx=5, pady=5)
result_label = tk.Label(root, text="")
result_label.grid(row=2, columnspan=2, padx=5, pady=5)
calculate_button = tk.Button(root, text="Calculate",
command=calculate_volume)
calculate_button.grid(row=3, columnspan=2, padx=5, pady=5)
root.mainloop()
```

<u>Output:-</u>



Enter the radius: 7

Enter the height: 10

Volume of Cylinder: 1539.38 cubic units