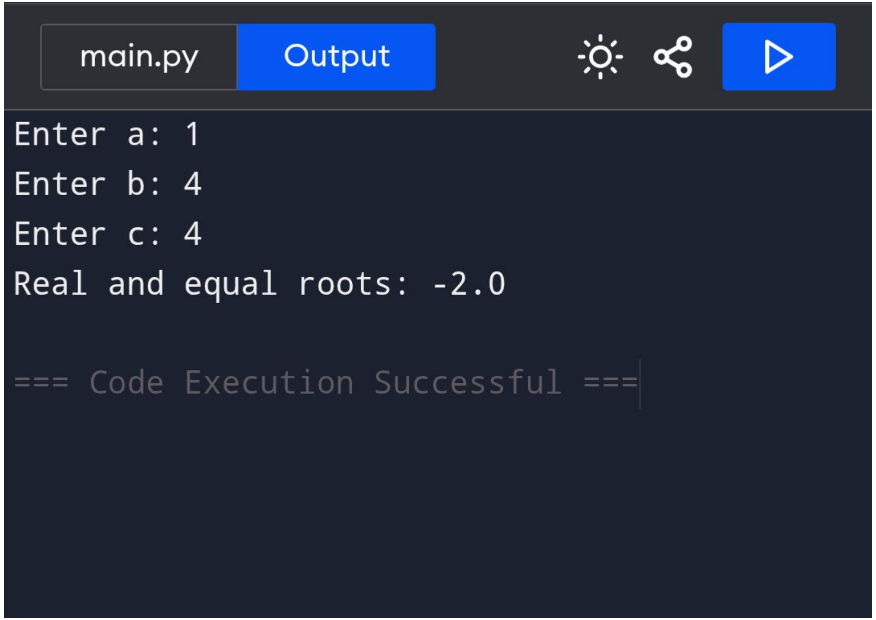
Practical file of OOPs using Python

Name: Ram Subhag Yadav
Roll no.: 25/CS(H)/131
Section: B
Course: B.Sc. (Hons.) Computer Science
College: Keshav Mahavidyalaya
Semester: 1st

Practicale(1) To finds root of quadratic equations.

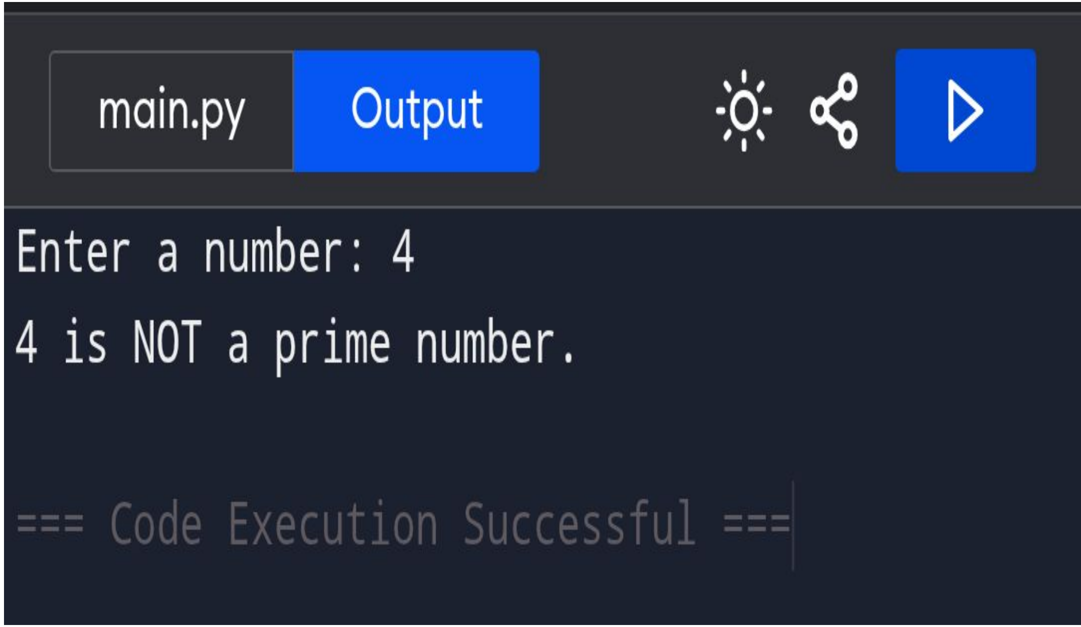
```
import math
a = float(input("Enter a: "))
b = float(input("Enter b: "))
c = float(input("Enter c: "))
d = b**2 - 4*a*c
if d > 0:
    root1 = (-b + math.sqrt(d)) / (2*a)
    root2 = (-b - math.sqrt(d)) / (2*a)
    print("Real and distinct roots:", root1, root2)
elif d == 0:
    root = -b / (2*a)
    print("Real and equal roots:", root)
else:
    print("Complex roots")
```



```
Enter a: 1
Enter b: 4
Enter c: 4
Real and equal roots: -2.0


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

Practical(2):(1)Checking prime number.

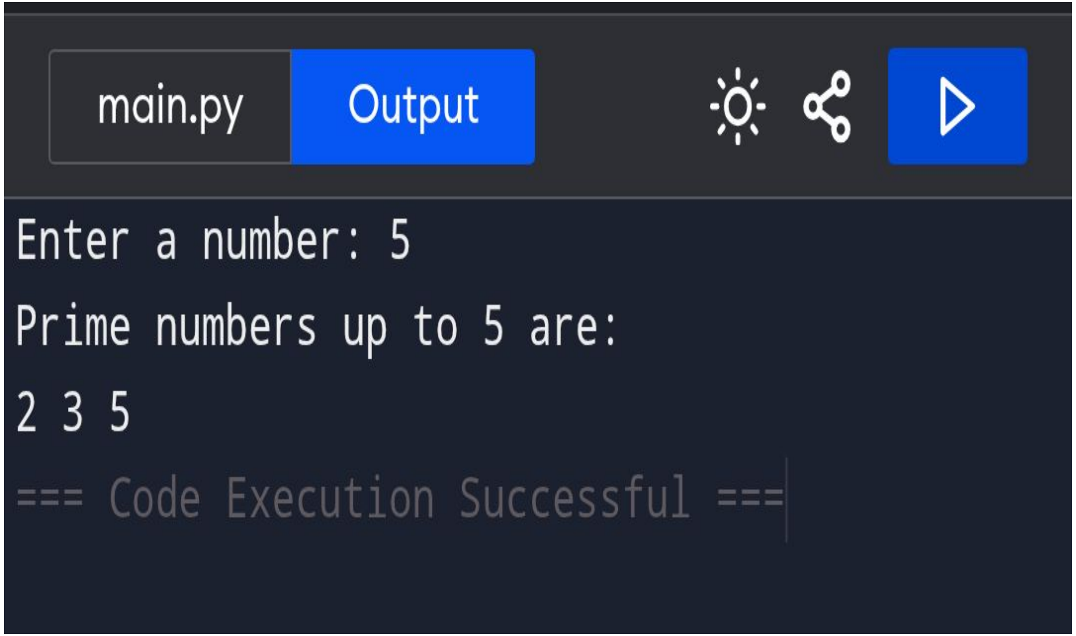
```
n = int(input("Enter a number: "))

if n <= 1:
    print(n, "is NOT a prime number.")
else:
    prime = True
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            prime = False
            break

    if prime:
        print(n, "is a prime number.")
    else:
        print(n, "is NOT a prime number.")
```



```
Enter a number: 4
4 is NOT a prime number.


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

Practical(2):(2)Generate prime no. till n.

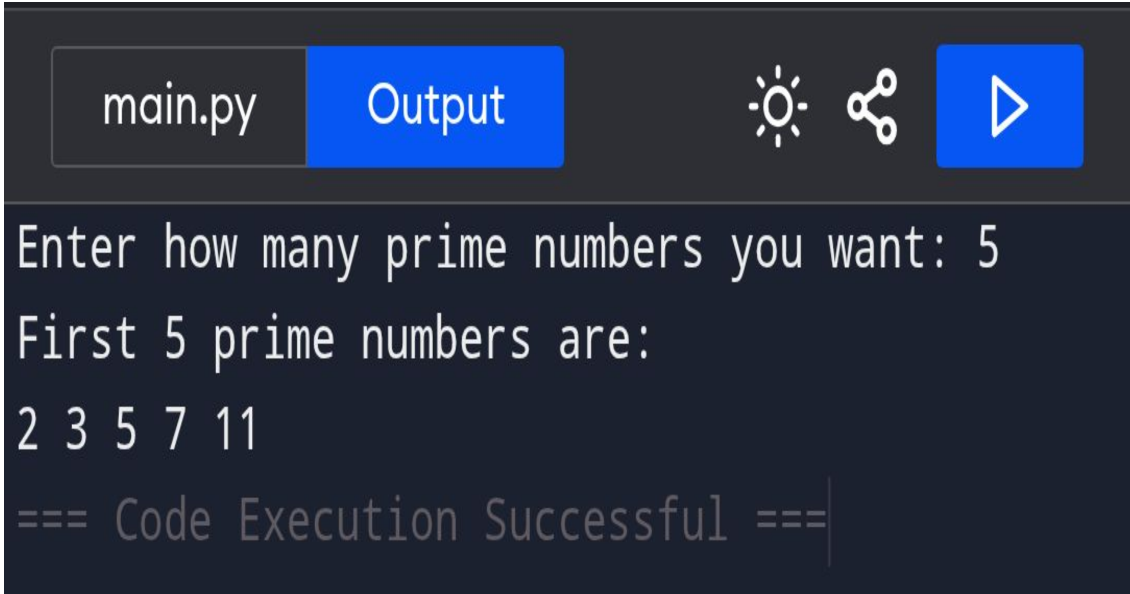
```
n = int(input("Enter a number: "))

print("Prime numbers up to", n, "are:")

for num in range(2, n + 1):
    prime = True
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            prime = False
            break
    if prime:
        print(num, end=" ")
```



```
Enter a number: 5
Prime numbers up to 5 are:
2 3 5
=== Code Execution Successful ===
```

Practical(2):(3)Generate prime no. first n.

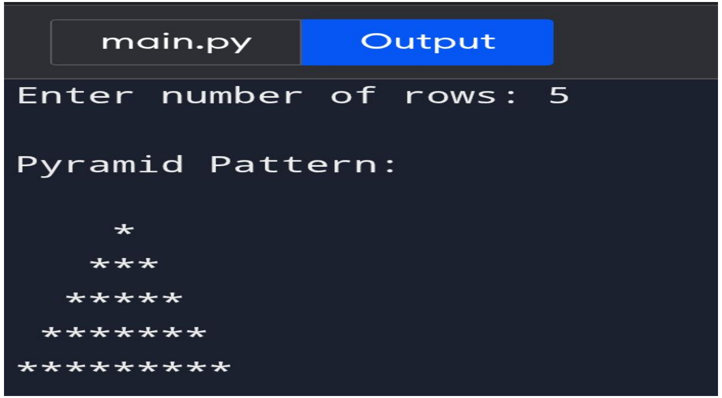
```
n = int(input("Enter how many prime numbers you want: "))

count = 0
num = 2

print("First", n, "prime numbers are:")

while count < n:
    prime = True
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            prime = False
            break

    if prime:
        print(num, end=" ")
        count += 1

    num += 1
```



```
Enter how many prime numbers you want: 5
First 5 prime numbers are:
2 3 5 7 11
=== Code Execution Successful ===
```

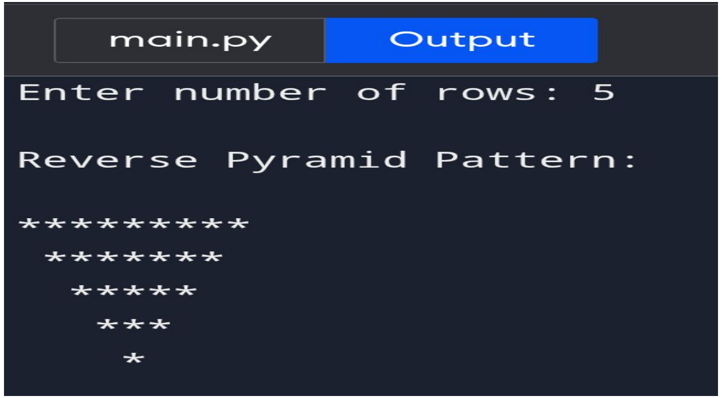
Practical(3):Pyramid Pattern

```
n = int(input("Enter number of rows: "))
print("\nPyramid Pattern:\n")
for i in range(1, n + 1):
    print(" " * (n - i) + "*" * (2 * i - 1))
```



```
Enter number of rows: 5

Pyramid Pattern:

    *
   ***
  *****
 *******
*********
```

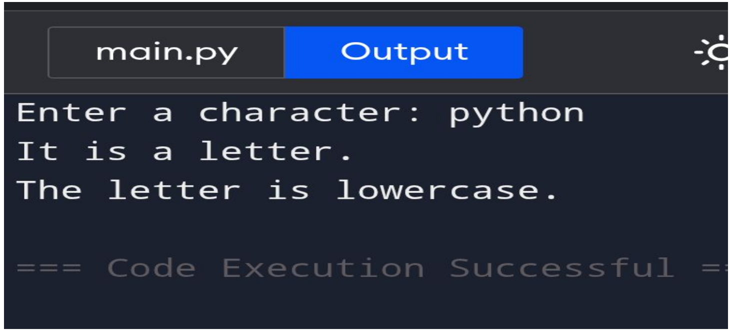
```
n = int(input("Enter number of rows: "))
print("\nReverse Pyramid Pattern:\n")
for i in range(n, 0, -1):
    print(" " * (n - i) + "*" * (2 * i - 1))
```



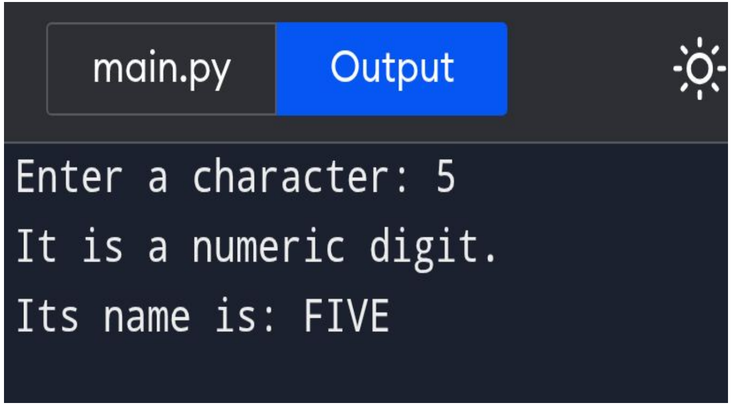
```
Enter number of rows: 5

Reverse Pyramid Pattern:

*********
 *******
  *****
   ***
    *
```

Practical(4):Checking whether alphabet, numeric or special.

```
ch = input("Enter a character: ")
print("\nCharacter Analysis:\n")
if ch.isalpha():
    print("It is a letter.")
    if ch.isupper():
        print("The letter is uppercase.")
    else:
        print("The letter is lowercase.")

elif ch.isdigit():
    print("It is a numeric digit.")
    digit_names = ["ZERO", "ONE", "TWO", "THREE", "FOUR",
                   "FIVE", "SIX", "SEVEN", "EIGHT", "NINE"]
    print("Its name is:", digit_names[int(ch)])

else:
    print("It is a special character.")
```



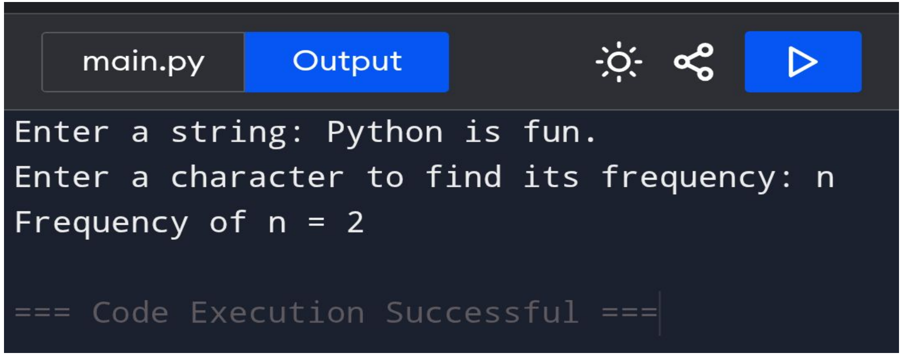
```
Enter a character: python
It is a letter.
The letter is lowercase.

=== Code Execution Successful =
```



```
Enter a character: 5
It is a numeric digit.
Its name is: FIVE
```

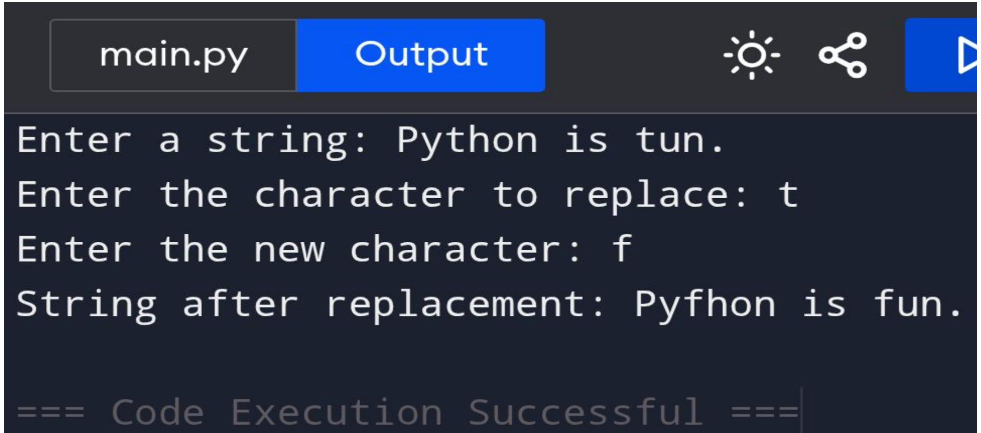
Practical(5):(1)Some operations on string like count, replace.

```
s = input("Enter a string: ")
ch = input("Enter a character to find its frequency: ")
count = 0
for c in s:
    if c == ch:
        count += 1
print("Frequency of", ch, "=", count)
```



```
Enter a string: Python is fun.
Enter a character to find its frequency: n
Frequency of n = 2

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

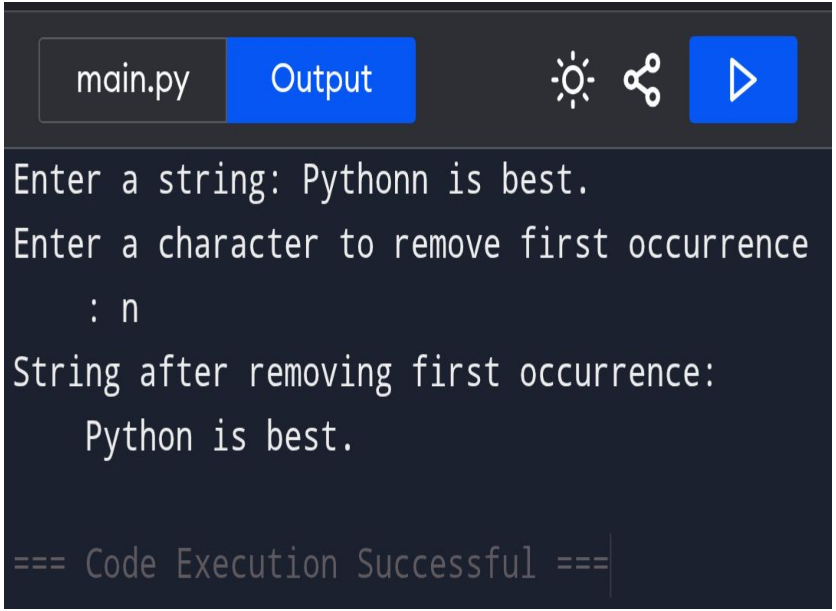
```
s = input("Enter a string: ")
old = input("Enter the character to replace: ")
new = input("Enter the new character: ")
result = ""
for c in s:
    if c == old:
        result += new
    else:
        result += c
print("String after replacement:", result)
```



```
Enter a string: Python is tun.
Enter the character to replace: t
Enter the new character: f
String after replacement: Pyfhon is fun.

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

Practical(5):(2)Removing first occurrence character.

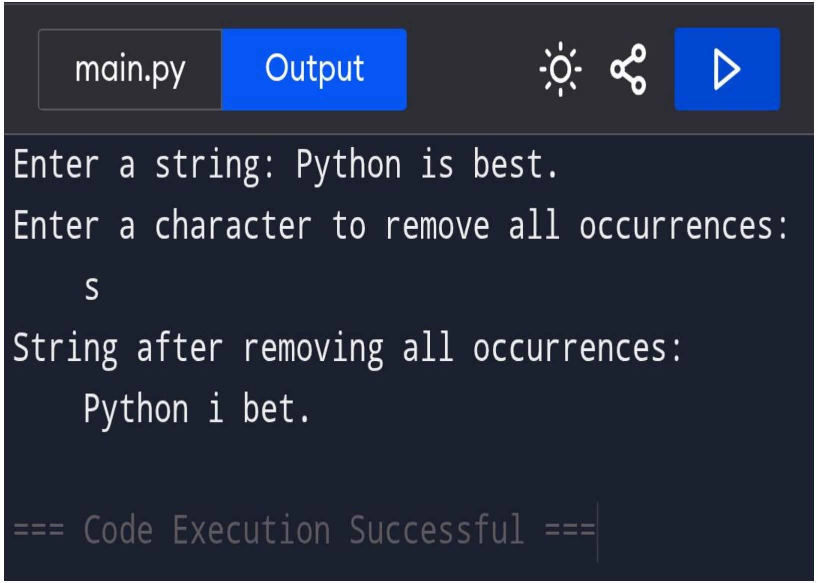
```
s = input("Enter a string: ")

ch = input("Enter a character to remove first
occurrence: ")

result = ""

removed = False

for c in s:

    if c == ch and not removed:

        removed = True

        continue

    result += c

print("String after removing first occurrence:", result)
```



```
Enter a string: Pythonn is best.
Enter a character to remove first occurrence
    : n
String after removing first occurrence:
    Python is best.

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

Practical(5):(3)Removing all occurrence.

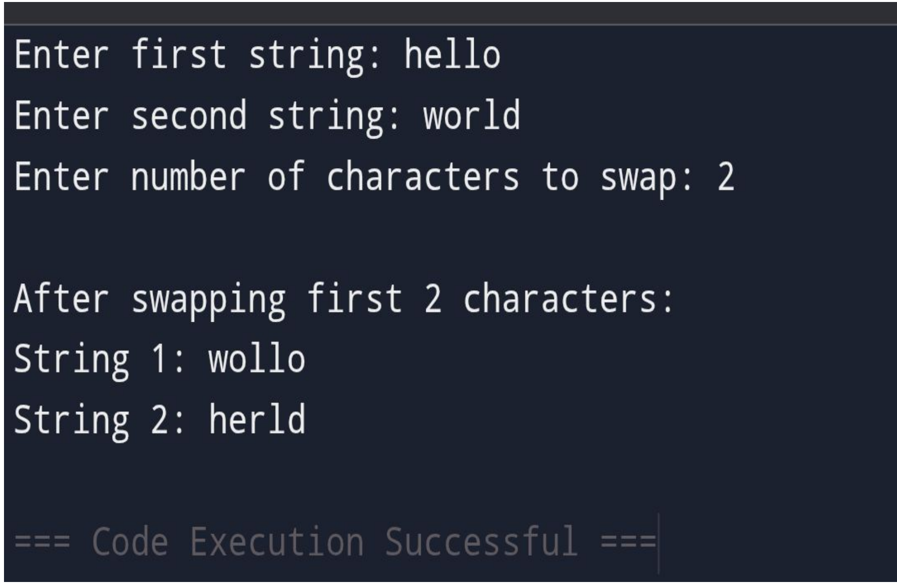
```
s = input("Enter a string: ")
ch = input("Enter a character to remove all occurrences: ")

result = ""
for c in s:
    if c != ch:
        result += c

print("String after removing all occurrences:", result)
```



```
Enter a string: Python is best.
Enter a character to remove all occurrences:
    s
String after removing all occurrences:
    Python i bet.


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

Practical(6):Swapping two characters of two strings.

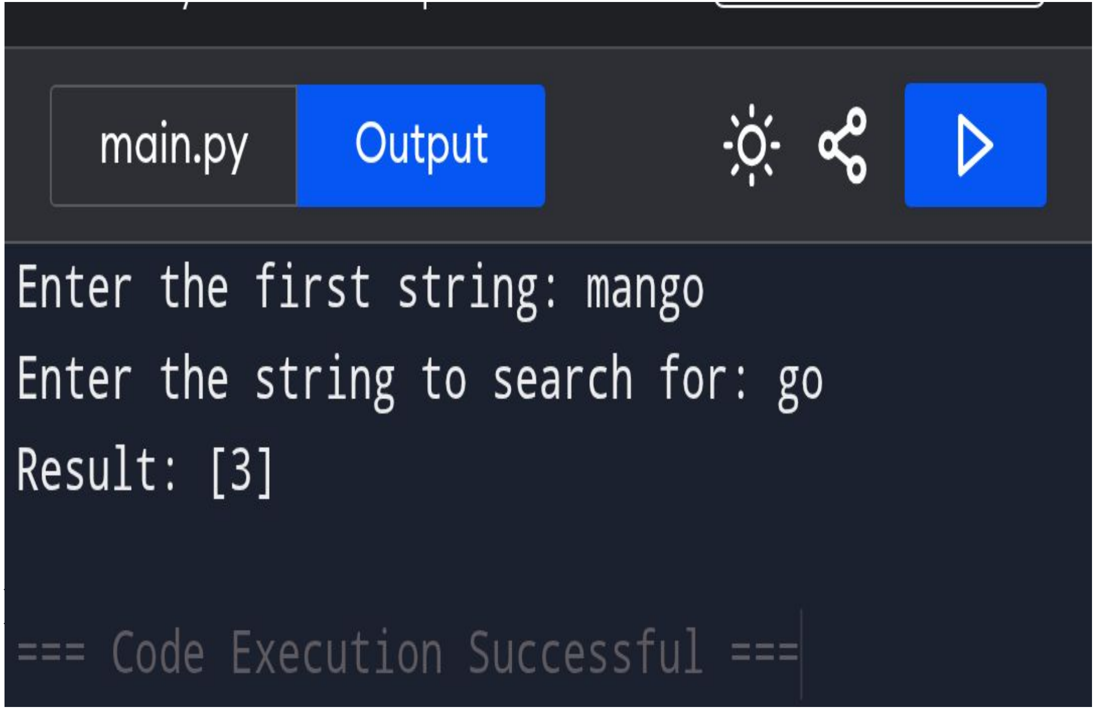
```
s1 = input("Enter first string: ")

s2 = input("Enter second string: ")

n = int(input("Enter number of characters to swap: "))


new_s1 = s2[:n] + s1[n:]

new_s2 = s1[:n] + s2[n:]


print("\nAfter swapping first", n, "characters:")

print("String 1:", new_s1)

print("String 2:", new_s2)
```



```
Enter first string: hello
Enter second string: world
Enter number of characters to swap: 2

After swapping first 2 characters:
String 1: wollo
String 2: herld


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

Practical(7):Finding index of occurrence.

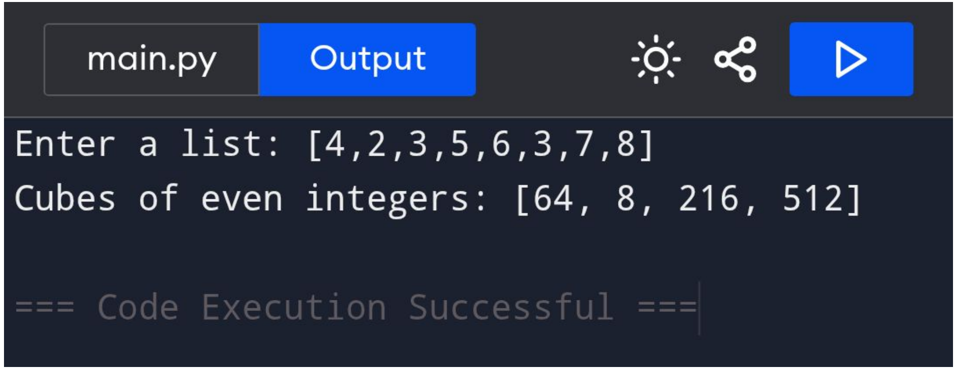
```
def find_occurrences(s1, s2):
    indices = []
    start = 0
    while True:
        pos = s1.find(s2, start)
        if pos == -1:
            break
        indices.append(pos)
        start = pos + 1
    if len(indices) == 0:
        return -1
    else:
        return indices
s1 = input("Enter the first string: ")
s2 = input("Enter the string to search for: "

result = find_occurrences(s1, s2)
print("Result:", result)
```



```
Enter the first string: mango
Enter the string to search for: go
Result: [3]


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

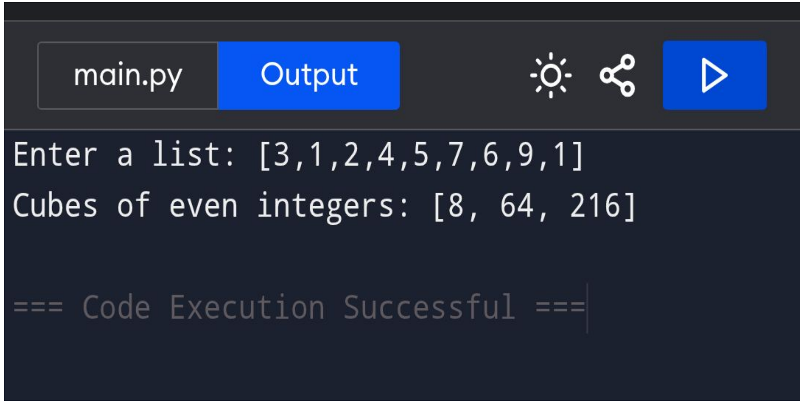
Practical(8):Print even cubes of even integer from list.

```
lst = eval(input("Enter a list: "))
result = []
for x in lst:
    if isinstance(x, int) and x % 2 == 0:
        result.append(x ** 3)
print("Cubes of even integers:", result)
```



```
Enter a list: [4,2,3,5,6,3,7,8]
Cubes of even integers: [64, 8, 216, 512]

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

```
lst = eval(input("Enter a list: "))

result = [x**3 for x in lst if isinstance(x, int) and x % 2 == 0]
print("Cubes of even integers:", result)
```



```
Enter a list: [3,1,2,4,5,7,6,9,1]
Cubes of even integers: [8, 64, 216]

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

Practical(9):(1) File handling

```
file = open("sample.txt", "r")
data = file.read()
file.close()

lines = data.split("\n")
words = data.split()
characters = len(data)

print("Total characters:", characters)
print("Total words:", len(words))
print("Total lines:", len(lines))
```

```
file = open("sample.txt", "r")
data = file.read()
file.close()

freq = {}

for ch in data:
    if ch in freq:
        freq[ch] += 1
    else:
        freq[ch] = 1

print("Character Frequency:")
print(freq)
```

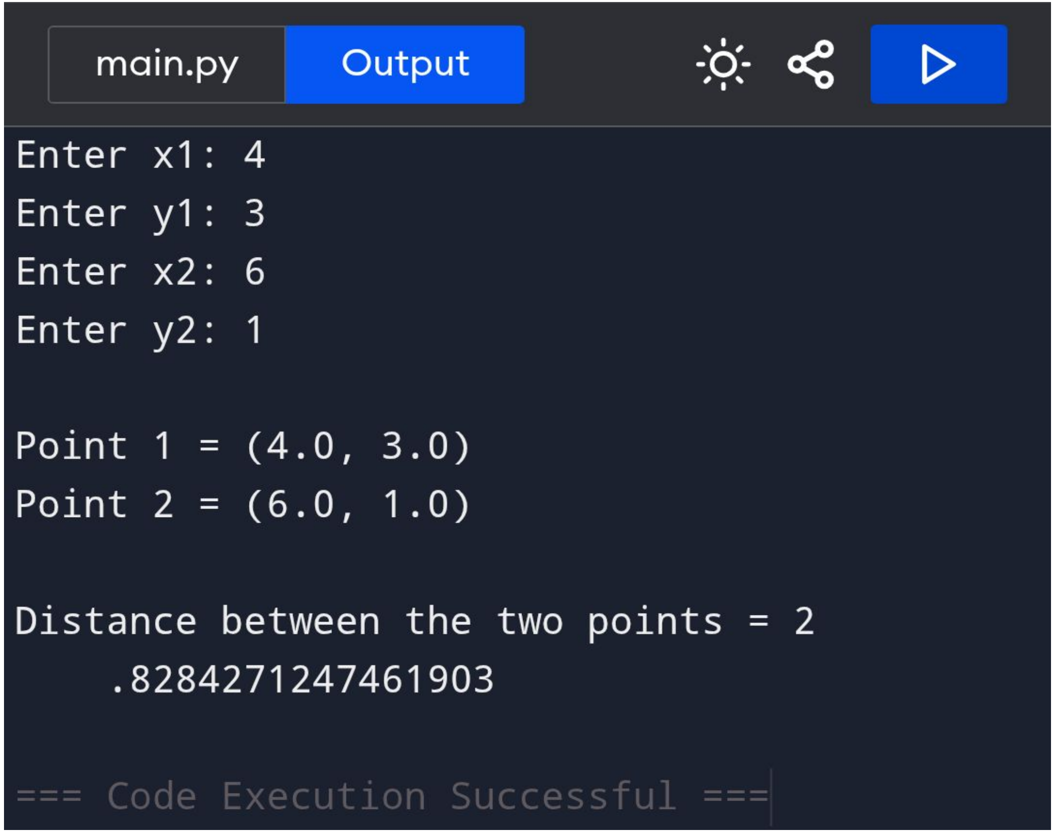
Practical(9):(2)File handling.

```
file = open("sample.txt", "r")
data = file.read()
file.close()

words = data.split()
words.reverse()

print("Words in reverse order:")
for w in words:
    print(w, end=" ")
```

```
input_file = open("sample.txt", "r")
file1 = open("File1.txt", "w")
file2 = open("File2.txt", "w")
lines = input_file.readlines()
for i in range(len(lines)):
    if (i + 1) % 2 == 0:
        file1.write(lines[i])     # even line → File1
    else:
        file2.write(lines[i])     # odd line → File2
input_file.close()
file1.close()
file2.close()
print("Even lines copied to File1.txt")
print("Odd lines copied to File2.txt")
```

Practical(10):Find distance of two coordinate using class.

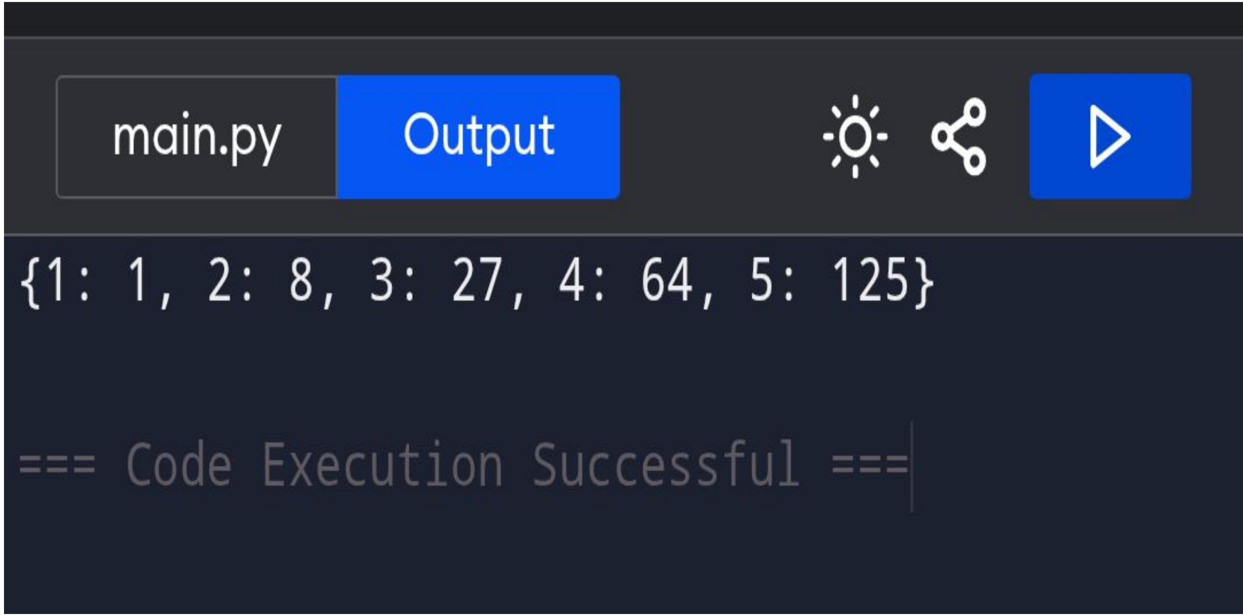
```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def __str__(self):
        return f"({self.x}, {self.y})"
    def distance(self, other):
        dx = self.x - other.x
        dy = self.y - other.y
        return math.sqrt(dx*dx + dy*dy)
x1 = float(input("Enter x1: "))
y1 = float(input("Enter y1: "))
x2 = float(input("Enter x2: "))
y2 = float(input("Enter y2: "))

p1 = Point(x1, y1)
p2 = Point(x2, y2)

print("\nPoint 1 =", p1)
print("Point 2 =", p2)
```



```
Enter x1: 4
Enter y1: 3
Enter x2: 6
Enter y2: 1

Point 1 = (4.0, 3.0)
Point 2 = (6.0, 1.0)

Distance between the two points = 2
    .8284271247461903


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

Practical(11):printing cubes in dictionary.

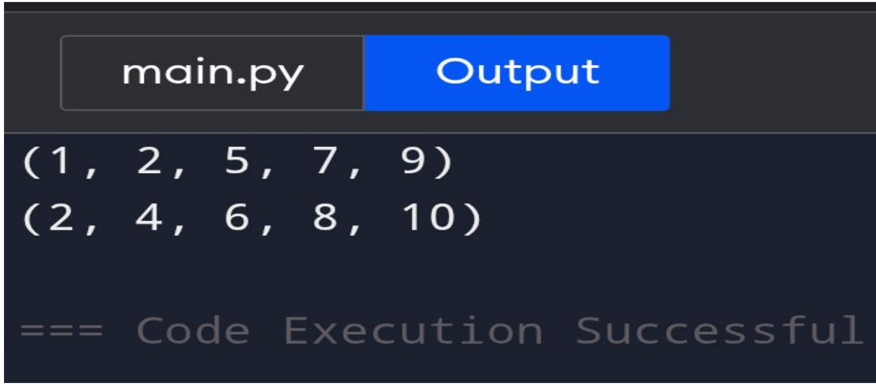
```
def cube_dictionary():
    d = {}
    for i in range(1, 6):
        d[i] = i**3
    print(d)
cube_dictionary()
```



```
{1: 1, 2: 8, 3: 27, 4: 64, 5: 125}


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

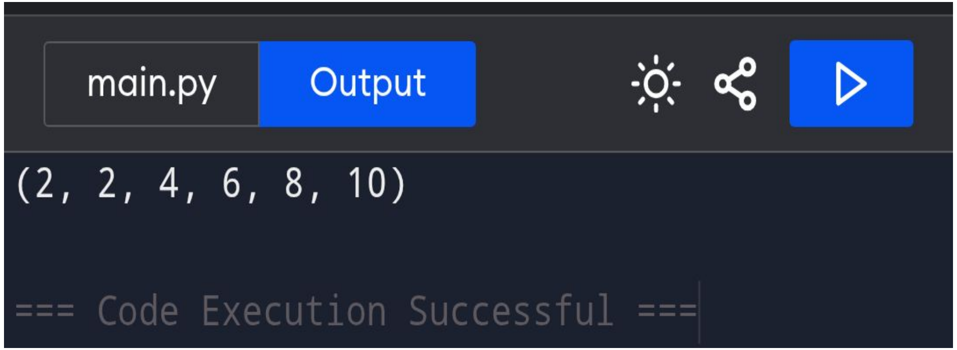
Practical(12):(1)Operations on tuple

```python
t1 = (1, 2, 5, 7, 9, 2, 4, 6, 8, 10)

mid = len(t1) // 2

print(t1[:mid])
print(t1[mid:])
```



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

=== Code Execution Successful
```

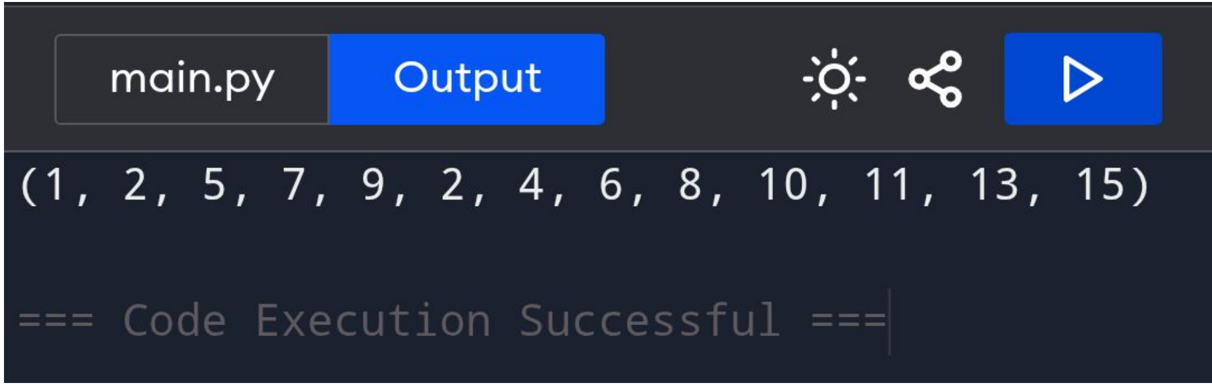
```python
t1 = (1, 2, 5, 7, 9, 2, 4, 6, 8, 10)

even_tuple = tuple(x for x in t1 if x % 2 == 0)
print(even_tuple)
```



```
(2, 2, 4, 6, 8, 10)

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

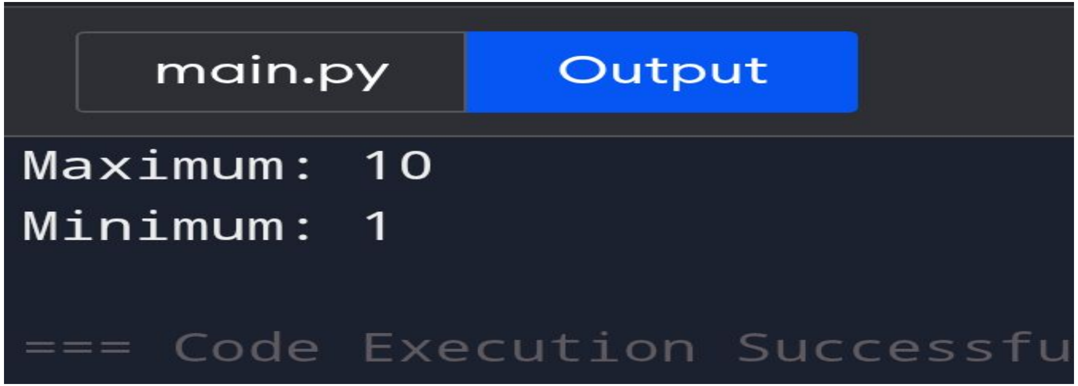
Practical(12):(2) operation on tuple

```
t1 = (1, 2, 5, 7, 9, 2, 4, 6, 8, 10)
t2 = (11, 13, 15)

t3 = t1 + t2
print(t3)
```



```
(1, 2, 5, 7, 9, 2, 4, 6, 8, 10, 11, 13, 15)


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

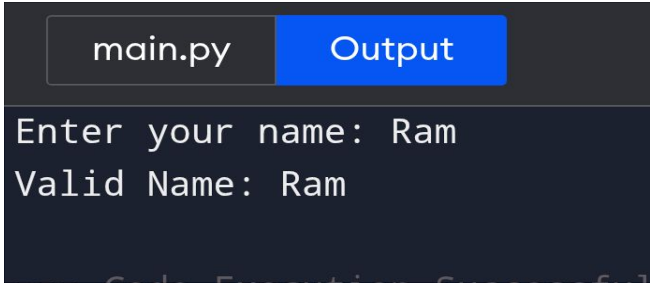
```
t1 = (1, 2, 5, 7, 9, 2, 4, 6, 8, 10)

print("Maximum:", max(t1))
print("Minimum:", min(t1))
```



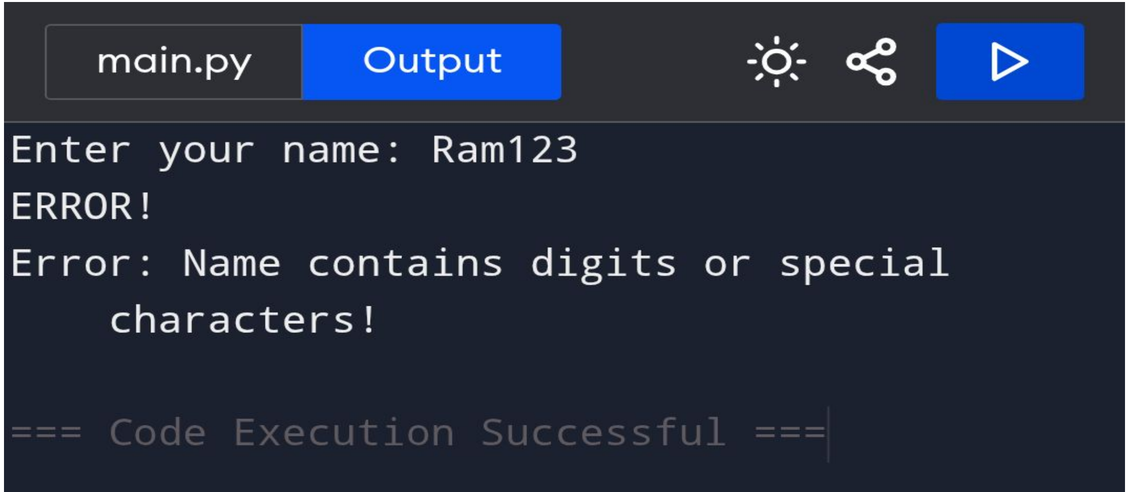
```
Maximum: 10
Minimum: 1


=== Code Execution Successfu
```

Practical(13): Raise exception if numeric present in name.

```
name = input("Enter your name: ")
try:
    if not name.isalpha():
        raise ValueError("Name contains digits or special characters!")

    print("Valid Name:", name)

except ValueError as e:
    print("Error:", e)
```



```
Enter your name: Ram
Valid Name: Ram
```



```
Enter your name: Ram123
ERROR!
Error: Name contains digits or special
    characters!

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