PYTHON PROGRAMMING
(PRACTICAL LAB FILE)

School of Computer Applications

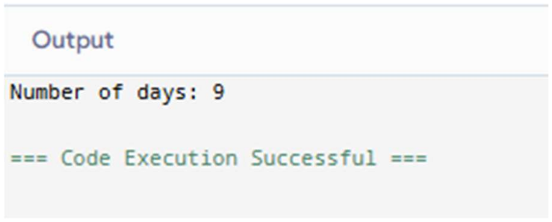
Department of Computer Applications

Submitted By	
Student Name	Laksh Chikara
Roll No	*24/SCA/BCA(AI&ML)/035*
Programme	*BCA (AI&ML)*
Semester	*3rd Semester*
Section/Group	III C
Department	*Computer Applications*
Batch	*2024-2027*
Submitted To	
Faculty Name	*Mrs. Sakshi*

Program 1: Number of days between two dates

```
from datetime import date

d1 = date(2014, 7, 2)
d2 = date(2014, 7, 11)

delta = d2 - d1
print("Number of days:", delta.days)
```

Output screenshot:



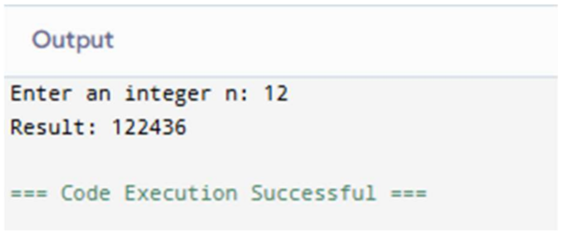
Output

```
Number of days: 9

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

Program 2: Compute n + nn + nnn

```
n = int(input("Enter an integer n: "))

n1 = n
n2 = int(str(n) * 2)
n3 = int(str(n) * 3)

result = n1 + n2 + n3
print("Result:", result)
```

Output screenshot:



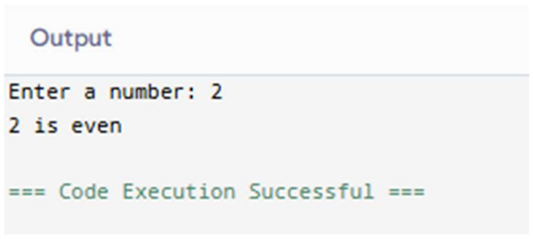
Output

```
Enter an integer n: 12
Result: 122436

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

Program 3: Check whether a number is even or odd

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

if num % 2 == 0:
    print(num, "is even")
else:
    print(num, "is odd")
```

Output screenshot:



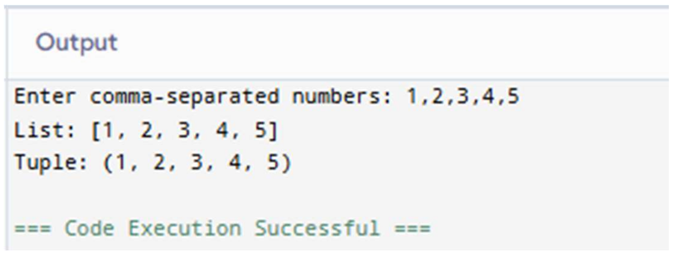
Output

```
Enter a number: 2
2 is even

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

Program 4: Generate list and tuple from comma-separated numbers

```
data = input("Enter comma-separated numbers: ")

items = data.split(",")

num_list = [int(x) for x in items]
num_tuple = tuple(num_list)

print("List:", num_list)
print("Tuple:", num_tuple)
```

Output screenshot:



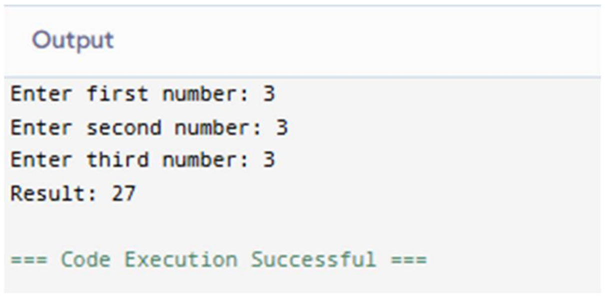
Output

```
Enter comma-separated numbers: 1,2,3,4,5
List: [1, 2, 3, 4, 5]
Tuple: (1, 2, 3, 4, 5)

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

Program 5: Sum of three numbers (thrice if equal)

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

total = a + b + c

if a == b == c:
    total *= 3

print("Result:", total)
```

Output screenshot:



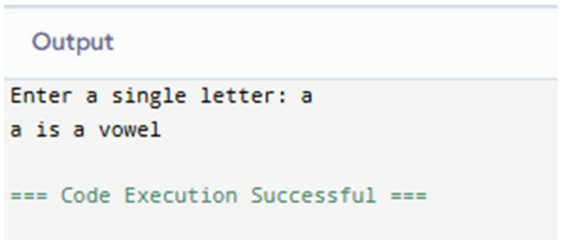
Output

```
Enter first number: 3
Enter second number: 3
Enter third number: 3
Result: 27

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

Program 6: Check whether a letter is a vowel or not

```
ch = input("Enter a single letter: ").lower()

if ch in ('a', 'e', 'i', 'o', 'u'):
    print(ch, "is a vowel")
else:
    print(ch, "is not a vowel")
```

Output screenshot:



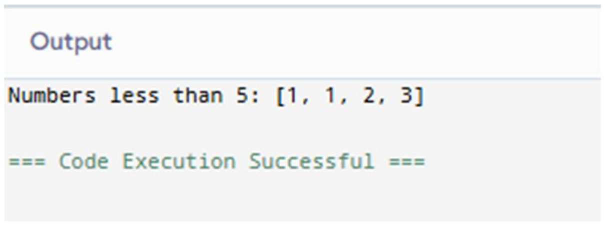
Output

```
Enter a single letter: a
a is a vowel

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

Program 7: Elements of list less than 5

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

less_than_5 = []

for num in a:
    if num < 5:
        less_than_5.append(num)

print("Numbers less than 5:", less_than_5)
```

Output screenshot:

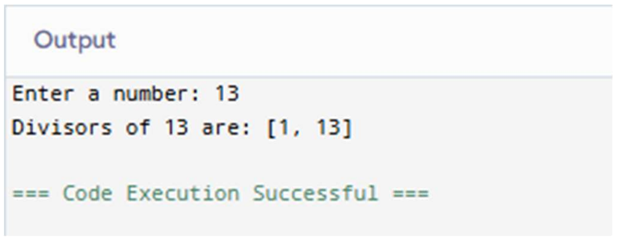


```
Output
Numbers less than 5: [1, 1, 2, 3]

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

Program 8: List all divisors of a number

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

divisors = []

for i in range(1, n + 1):
    if n % i == 0:
        divisors.append(i)

print("Divisors of", n, "are:", divisors)
```

Output screenshot:

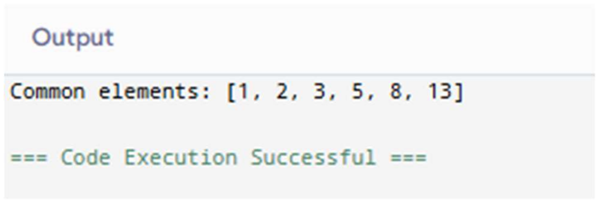


```
Output
Enter a number: 13
Divisors of 13 are: [1, 13]

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

Program 9: Common elements of two lists (without duplicates)

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

common = []

for item in a:
    if item in b and item not in common:
        common.append(item)

print("Common elements:", common)
```

Output screenshot:



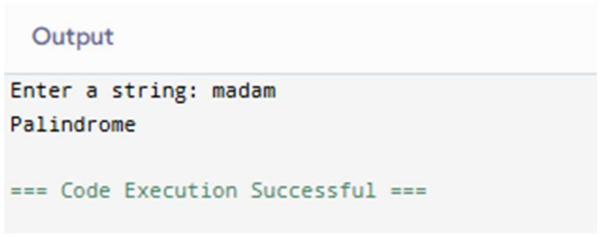
Output

```
Common elements: [1, 2, 3, 5, 8, 13]

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

Program 10: Check whether a string is a palindrome

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

if s == s[::-1]:
    print("Palindrome")
else:
    print("Not a palindrome")
```

Output screenshot:



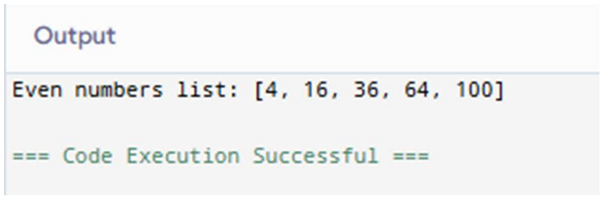
Output

```
Enter a string: madam
Palindrome

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

Program 11: New list with only even numbers

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

even_list = [x for x in a if x % 2 == 0]

print("Even numbers list:", even_list)
```

Output screenshot:



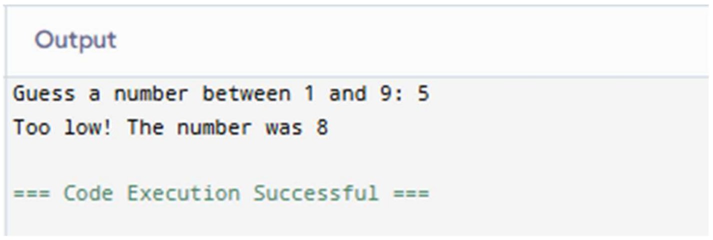
Output

```
Even numbers list: [4, 16, 36, 64, 100]

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

Program 12: Guess the number between 1 and 9

```
import random

number = random.randint(1, 9)

guess = int(input("Guess a number between 1 and 9: "))

if guess < number:
    print("Too low! The number was", number)
elif guess > number:
    print("Too high! The number was", number)
else:
    print("Exactly right!")
```

Output screenshot:



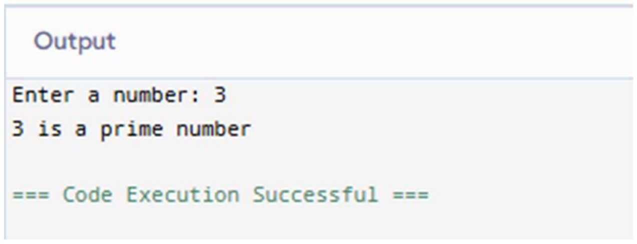
Output

```
Guess a number between 1 and 9: 5
Too low! The number was 8

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

Program 13: Check whether a number is prime or not

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

if n > 1:
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            print(n, "is not a prime number")
            break
    else:
        print(n, "is a prime number")
else:
    print(n, "is not a prime number")
```

Output screenshot:



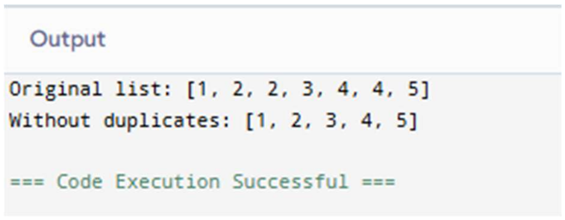
Output

```
Enter a number: 3
3 is a prime number

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

Program 14: Remove duplicates from a list using a function

```
def remove_duplicates(lst):
    result = []
    for item in lst:
        if item not in result:
            result.append(item)
    return result

data = [1, 2, 2, 3, 4, 4, 5]
print("Original list:", data)
print("Without duplicates:", remove_duplicates(data))
```

Output screenshot:



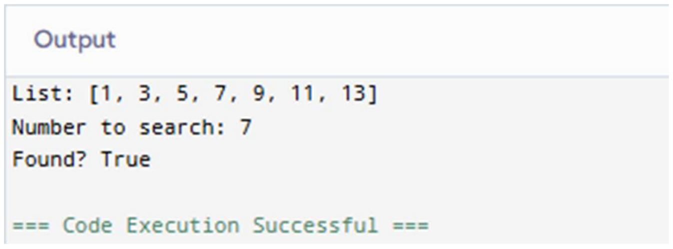
Output

```
Original list: [1, 2, 2, 3, 4, 4, 5]
Without duplicates: [1, 2, 3, 4, 5]

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

Program 15: Search an ordered list for a number (Boolean)

```
def contains_number(ordered_list, number):
    left = 0
    right = len(ordered_list) - 1

    while left <= right:
        mid = (left + right) // 2
        if ordered_list[mid] == number:
            return True
        elif ordered_list[mid] < number:
            left = mid + 1
        else:
            right = mid - 1
    return False

lst = [1, 3, 5, 7, 9, 11, 13]
n = 7

print("List:", lst)
print("Number to search:", n)
print("Found?", contains_number(lst, n))
```

Output screenshot:

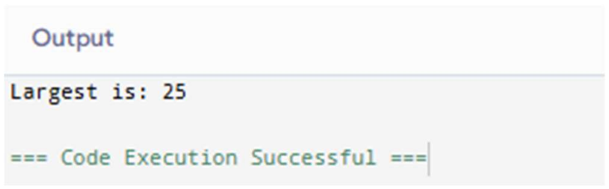


```
List: [1, 3, 5, 7, 9, 11, 13]
Number to search: 7
Found? True

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

Program 16: Largest of three numbers without max()

```
def largest_of_three(a, b, c):
    largest = a
    if b > largest:
        largest = b
    if c > largest:
        largest = c
    return largest

x = 10
y = 25
z = 15

print("Largest is:", largest_of_three(x, y, z))
```

Output screenshot:



```
Largest is: 25

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

Program 17: Read and write operations on a file

```
# Write to a file
with open("sample.txt", "w") as f:
    f.write("Hello, Python file handling!\n")
    f.write("This is a second line.\n")

# Read from the same file
with open("sample.txt", "r") as f:
    content = f.read()

print("File contents:")
print(content)
```

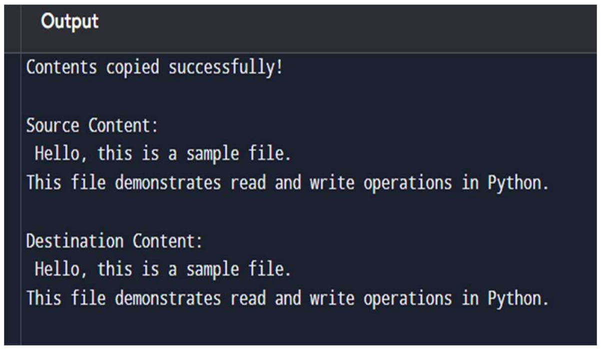
Output screenshot:

```
File contents:
Hello, Python file handling!
This is a second line.
```

Program 18: Copy contents of one file to another file

```python
source_file = "sample.txt"
destination_file = "copy_sample.txt"
with open(source_file, "r") as src:
    data = src.read()
with open(destination_file, "w") as dest:
    dest.write(data)
print(f"Contents of '{source_file}' copied to '{destination_file}'.")
```

Output screenshot:

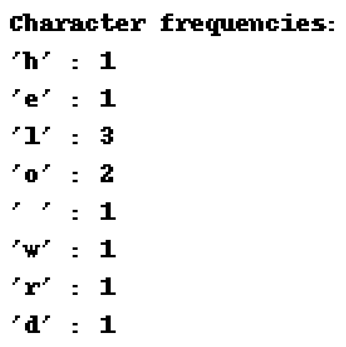


```
Contents copied successfully!

Source Content:
 Hello, this is a sample file.
This file demonstrates read and write operations in Python.

Destination Content:
 Hello, this is a sample file.
This file demonstrates read and write operations in Python.
```

Program 19: Count frequency of characters in a file

```python
filename = "text.txt"

# Create a file for demonstration
with open(filename, "w") as f:
    f.write("hello world")

freq = {}

with open(filename, "r") as f:
    text = f.read()
    for ch in text:
        freq[ch] = freq.get(ch, 0) + 1

print("Character frequencies:")
for ch, count in freq.items():
    print(repr(ch), ":", count)
```

Output screenshot:

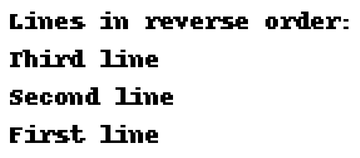


```
Character frequencies:
'h' : 1
'e' : 1
'l' : 3
'o' : 2
' ' : 1
'w' : 1
'r' : 1
'd' : 1
```

Program 20: Print each line of a file in reverse order

```
filename = "lines.txt"

# Create example file
with open(filename, "w") as f:
    f.write("First line\n")
    f.write("Second line\n")
    f.write("Third line\n")

with open(filename, "r") as f:
    lines = f.readlines()

print("Lines in reverse order:")
for line in reversed(lines):
    print(line.rstrip())
```

Output screenshot:

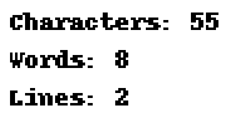


```
Lines in reverse order:
Third line
Second line
First line
```

Program 21: Count characters, words and lines in a file

```
filename = "info.txt"

# Create example file
with open(filename, "w") as f:
    f.write("Python is fun.\n")
    f.write("File handling is useful.\n")

with open(filename, "r") as f:
    text = f.read()

num_chars = len(text)
words = text.split()
num_words = len(words)
num_lines = text.count("\n")

print("Characters:", num_chars)
print("Words:", num_words)
print("Lines:", num_lines)
```

Output screenshot:

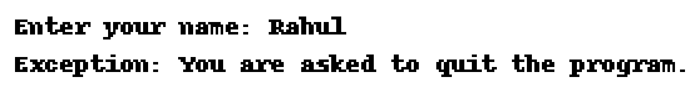


```
Characters: 55
Words: 8
Lines: 2
```

Program 22: Raise exception for specific name

```
class NameErrorException(Exception):
    pass

name = input("Enter your name: ")

try:
    if name.lower() == "rahul":
        raise NameErrorException("You are asked to quit the program.")
    else:
        print("Hello,", name)
except NameErrorException as e:
    print("Exception:", e)
```

Output screenshot:

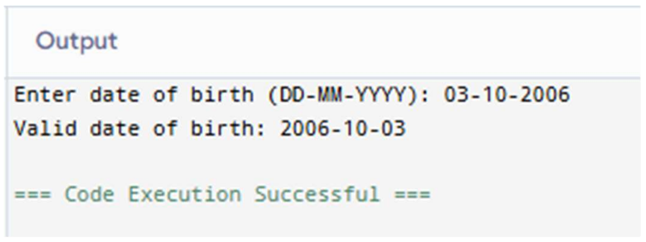


```
Enter your name: Rahul
Exception: You are asked to quit the program.
```

Program 23: Validate date of birth

```
from datetime import datetime

dob_str = input("Enter date of birth (DD-MM-YYYY): ")

try:
    dob = datetime.strptime(dob_str, "%d-%m-%Y")
    print("Valid date of birth:", dob.date())
except ValueError:
    print("Invalid date entered!")
```

Output screenshot:



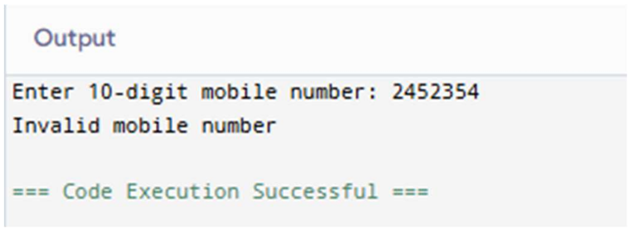
Output

```
Enter date of birth (DD-MM-YYYY): 03-10-2006
Valid date of birth: 2006-10-03

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

Program 24: Validate 10-digit mobile number using regex

```
import re

pattern = re.compile(r"^[7-9][0-9]{9}$")

mobile = input("Enter 10-digit mobile number: ")

if pattern.match(mobile):
    print("Valid mobile number")
else:
    print("Invalid mobile number")
```

Output screenshot:



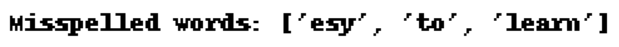
Output

```
Enter 10-digit mobile number: 2452354
Invalid mobile number

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

Program 25: Simple spell checker

```
# known_words.txt - file with correct words (one per line)
with open("known_words.txt", "w") as f:
    f.write("python\n")
    f.write("is\n")
    f.write("easy\n")

# user_file.txt - file to be checked
with open("user_file.txt", "w") as f:
    f.write("python is esy to learn")

# Load known words
with open("known_words.txt", "r") as f:
    known = set(word.strip() for word in f)

# Read user file
with open("user_file.txt", "r") as f:
    words = f.read().split()

misspelled = [w for w in words if w.lower() not in known]

print("Misspelled words:", misspelled)
```

Output screenshot:

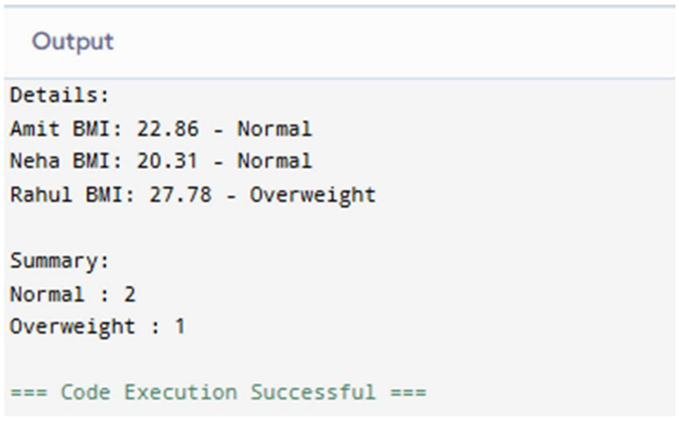


```
Misspelled words: ['esy', 'to', 'learn']
```

Program 26: BMI calculation and categorization

```
people = [
    {"name": "Amit", "age": 25, "weight": 70, "height": 1.75},
    {"name": "Neha", "age": 30, "weight": 52, "height": 1.60},
    {"name": "Rahul", "age": 35, "weight": 90, "height": 1.80},
]

def bmi_category(bmi):
    if bmi < 18.5:
        return "Underweight"
    elif bmi < 25:
        return "Normal"
    elif bmi < 30:
        return "Overweight"
    else:
        return "Obese"

summary = {}

for person in people:
    bmi = person["weight"] / (person["height"] ** 2)
    category = bmi_category(bmi)
    person["bmi"] = round(bmi, 2)
    person["category"] = category
    summary[category] = summary.get(category, 0) + 1

print("Details:")
for person in people:
    print(person["name"], "BMI:", person["bmi"], "-", person["category"])

print("\nSummary:")
for cat, count in summary.items():
    print(cat, ":", count)
```

Output screenshot:



```
Details:
Amit BMI: 22.86 - Normal
Neha BMI: 20.31 - Normal
Rahul BMI: 27.78 - Overweight

Summary:
Normal : 2
Overweight : 1

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