“Programming using Python”

College Name	**Kirori Mal College**
Paper Name	**Programming with Python**
Course Name	**B.Sc. (Hons.) Math 3nd Semester**
Student Name	**Mohd Jabir Qureshi**
College Roll number	**2332185**
Phone Number	**8077129285**

Table of Contents

Question 1 3
Question 2 3
Question 3 4
Question 4 5
Question 5 5
Question 6 6
Question 7 6
Question 8 7
Question 9 9
Question 10 9
Question 11 9
Question 12 10
Question 13 11
Question 14 11
Question 15 12
Question 16 13
Question 17 14
Question 18 15
Question 19 16
Question 20 17

Question 1→ Running instructions in Interactive interpreter and a Python Script

```
Tutorial_1.py > ...
a = 5
b = 3
print(a + b)
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS          2: Python

PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/AppData/Local/Programs/Python/Python313/python.exe "c:/
Users/Nazia/Desktop/My Python Program/Tutorial_1.py"
8
PS C:\Users\Nazia\Desktop\My Python Program>
```

Question 2→ Write a program to purposefully raise Indentation Error and Correct it

Part 1→With Indentation Error

```
Program_with_an_Indentation_Error.py > ...
def greet(Jabir):
print("Hello, " + Jabir)
    print("I am Jabir Bsc 2nd Year student!")
```

```
PROBLEMS 6   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS          2: Python

PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/AppData/Local/Programs/Python/Pyth
Users/Nazia/Desktop/My Python Program/Program_with_an_Indentation_Error.py"
  File "c:\Users\Nazia\Desktop\My Python Program\Program_with_an_Indentation_Error.py", line 2
    print("Hello, " + Jabir)
    ^^^^^
IndentationError: expected an indented block after function definition on line 1
PS C:\Users\Nazia\Desktop\My Python Program>
```

Part 2→With fixing Indentation Error

```
Indentation_Corrected_Version.py > ...
def greet(Jabir):
    print("Hello, " + Jabir)
    print("I am Jabir Bsc 2nd Year student!")
```

```
PROBLEMS 6   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS          2: Python

PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/AppData/Local/Programs/Python/Python3
Users/Nazia/Desktop/My Python Program/Indentation_Corrected_Version.py"
PS C:\Users\Nazia\Desktop\My Python Program>
```

Question 3→ Write a program to compute distance between two points taking input from the user. (Pythagorean Theorem)

Pythagorean_Theorem.py

```python
import math

# Take input for the coordinates of the first point
x1 = float(input("Enter the x-coordinate of the first point: "))
y1 = float(input("Enter the y-coordinate of the first point: "))

# Take input for the coordinates of the second point
x2 = float(input("Enter the x-coordinate of the second point: "))
y2 = float(input("Enter the y-coordinate of the second point: "))

# Calculate the distance
distance = math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)

# Display the result
print(f"The distance between the points ({x1}, {y1}) and ({x2}, {y2}) is {distance:.2f}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 2: Python

```
PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Nazia/Desktop/My Python Program/Pythagorean_Theorem.py"
Enter the x-coordinate of the first point: 10.2
Enter the y-coordinate of the first point: 10.5
Enter the x-coordinate of the second point: 10.8
Enter the y-coordinate of the second point: 15.6
The distance between the points (10.2, 10.5) and (10.8, 15.6) is 5.14
PS C:\Users\Nazia\Desktop\My Python Program>
```

Question 4→ Write a program add.py that takes 2 numbers as command line arguments and prints its sum.

Program in Visual studio Code (Name changes→ Question_3_sys_module

Question_3_sys_module.py > ...

```python
import sys

# Check if exactly 2 arguments (besides the script name) are provided
if len(sys.argv) != 3:
    print("Usage: python add.py <number1> <number2>")
    sys.exit(1)

# Convert arguments to numbers and calculate the sumS
try:
    num1 = float(sys.argv[1])
    num2 = float(sys.argv[2])
    result = num1 + num2
    print(f"The sum of {num1} and {num2} is: {result}")
except ValueError:
    print("Please enter valid numbers as arguments.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 2: Pyth

```
PS C:\Users\Nazia\Desktop\My Python Program>
```

Solution→ Program executed in Linux

nazia@zakir: /mnt/c/Users/Nazia/Desktop/My Python Program

```
nazia@zakir:/mnt/c/Users/Nazia/Desktop/My Python Program$ python3 Question_3_sys_module.py 3 5
The sum of 3.0 and 5.0 is: 8.0
nazia@zakir:/mnt/c/Users/Nazia/Desktop/My Python Program$
```

Question 5→ Write a Program for checking whether the given number is an even number or not.

Question_5_given_number_is_even_or_not.py

Question_5_given_number_is_even_or_not.py > ...

```python
# Take input from the user
number = int(input("Enter a number: "))

# Check if the number is even
if number % 2 == 0:
    print(f"{number} is an even number.")
else:
    print(f"{number} is not an even number.")
```

TERMINAL 4: Python

```
PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/AppDat
a/Local/Programs/Python/Python313/python.exe "c:/Users/Nazia/Desktop
/My Python Program/Question_5_given_number_is_even_or_not.py"
Enter a number: 15
15 is not an even number.
PS C:\Users\Nazia\Desktop\My Python Program>
```

Question 6→ Using a for loop, write a program that prints out the decimal equivalents of 1/2, 1/3.

Question_6_prints_out_the_decimal_equivalents.py

```python
# List of fractions to convert
fractions = [2, 3]

# Loop through each fraction in the list
for denom in fractions:
    decimal_value = 1 / denom
    print(f"The decimal equivalent of 1/{denom} is: {decimal_value}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 4: Python

```
PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/AppData/Local/Programs/Pyt
sers/Nazia/Desktop/My Python Program/Question_6_prints_out_the_decimal_equivalents.py"
The decimal equivalent of 1/2 is: 0.5
The decimal equivalent of 1/3 is: 0.3333333333333333
PS C:\Users\Nazia\Desktop\My Python Program>
```

Question 7→ 1/4, 1/10 Write a program using a for loop that loops over a sequence. What is the sequence?

question_7_loop_sequence.py

```python
# Sequence of denominators
sequence = [4, 10]

# Loop through each denominator in the sequence
for denom in sequence:
    decimal_value = 1 / denom
    print(f"The decimal equivalent of 1/{denom} is: {decimal_value}")
```

PROBLEMS OUTPUT TERMINAL ··· 4: Python

```
PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/AppData/Local/Programs/Python/
Python313/python.exe "c:/Users/Nazia/Desktop/My Python Program/question_7_loop_sequence.py"
The decimal equivalent of 1/4 is: 0.25
The decimal equivalent of 1/10 is: 0.1
PS C:\Users\Nazia\Desktop\My Python Program>
```

Question 8→ Write a program using a while loop that asks the user for a number, and prints a countdown from that number to zero?

Question_8_prints_a_countdown.py

```python
# Ask the user for a starting number
number = int(input("Enter a number to start the countdown: "))

# Countdown loop
while number >= 0:
    print(number)
    number -= 1  # Decrement the number by 1 10in each iteration

print("Countdown complete!")
```

PROBLEMS OUTPUT TERMINAL ··· 4: Python

```
PS C:\Users\Nazia\Desktop\My Python Program> ^C
PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/AppData/Local/Pro
Python313/python.exe "c:/Users/Nazia/Desktop/My Python Program/Question_8_print
.py"
Enter a number to start the countdown: 5
5
4
3
2
1
0
Countdown complete!
PS C:\Users\Nazia\Desktop\My Python Program>
```

Question 9→ Write a program using a while loop that asks the user for a number, and prints a countdown from that number to zero?

Part→1

Question_9_fabbo.py > ...

```python
def sum_of_primes(limit):
    sieve = [True] * limit
    sieve[0] = sieve[1] = False
    for i in range(2, int(limit**0.5) + 1):
        if sieve[i]:
            for j in range(i * i, limit, i):
                sieve[j] = False
    return sum(i for i, is_prime in enumerate(sieve) if is_prime)

# Set the limit to two million
limit = 2000000
print("Sum of all primes below two million:", sum_of_primes(limit))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Nazia\Desktop\My Python Program> ^C
PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/AppData/Local/Prog
Sum of all primes below two million: 142913828922
```

Part→ 2

Question_9_part_2.py > ...

```python
def fibonacci_sequence(n_terms):
    sequence = [1, 2]
    while len(sequence) < n_terms:
        sequence.append(sequence[-1] + sequence[-2])
    return sequence

# Get the first 10 terms of the Fibonacci sequence
print("First 10 terms of the Fibonacci sequence:", fibonacci_sequence(10))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Nazia\Desktop\My Python Program> ^C
PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/AppData/Local/Programs/Pyth
Sum of all primes below two million: 142913828922
PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/AppData/Local/Programs/Pyth
First 10 terms of the Fibonacci sequence: [1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
PS C:\Users\Nazia\Desktop\My Python Program>
```

Question 10→ By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms?

Question_10_Fabb0_4_M.py > ...

```python
# Initialize the first two terms of the Fibonacci sequence
a, b = 1, 2
even_sum = 0

# Loop until the terms exceed four million
while a <= 4000000:
    # Check if the term is even
    if a % 2 == 0:
        even_sum += a
    # Move to the next term in the sequence
    a, b = b, a + b

print("Sum of the even-valued terms:", even_sum)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/AppData/
y"
Sum of the even-valued terms: 4613732
PS C:\Users\Nazia\Desktop\My Python Program>
```

Question 11→ Write a program to count the numbers of characters in the string and store them in a dictionary data structure?

Question_11_char_count.py > ...

```python
def count_characters(string):
    # Initialize an empty dictionary to store character counts
    char_count = {}
    # Loop through each character in the string
    for char in string:
        # If the character is already in the dictionary, increment its count
        if char in char_count:
            char_count[char] += 1
        # Otherwise, add the character to the dictionary with a count of 1
        else:
            char_count[char] = 1

    return char_count

# Example usage
input_string = "My Name is Jabir"
result = count_characters(input_string)
print("Character counts:", result)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Nazia\Desktop\My Python Program> ^C
PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/AppData/Local/Programs/Python/Python313/python.exe "c
py"
Character counts: {'M': 1, 'y': 1, ' ': 3, 'N': 1, 'a': 2, 'm': 1, 'e': 1, 'i': 2, 's': 1, 'J': 1, 'b': 1, 'r': 1}
PS C:\Users\Nazia\Desktop\My Python Program>
```

Question 12→ Write a program to use split and join methods in the string and trace a birthday with a dictionary data structure?

Question_12_date_of_Birth.py > ...

```python
birthdays = {
    "Jabir": "2004-08-28",
    "Zakir": "1989-11-08",
    "Zainab": "2000-11-08",
    "Papa": "1965-07-05"
}

# Function to search and format birthday
def find_birthday(name):
    # Check if the name is in the dictionary
    if name in birthdays:
        # Use split to break the date into components
        date_parts = birthdays[name].split("-")

        # Join the date parts in a new format
        formatted_birthday = "/".join(date_parts[::-1])  # Format as DD/MM/YYYY
        return f"{name}'s birthday is on {formatted_birthday}."
    else:
        return f"No birthday found for {name}."

# Example usage
name_to_search = "Jabir"
result = find_birthday(name_to_search)
print(result)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Nazia\Desktop\My Python Program> ^C
PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/AppData/Local/Programs/Python/Py
th.py"
Jabir's birthday is on 28/08/2004.
PS C:\Users\Nazia\Desktop\My Python Program>
```

Question 13→ Write a program combining lists that combines these lists into a dictionary?

Question_13_combining_list.py

Question_13_combining_list.py > ...

```python
# Define the lists
keys = ["name", "age", "city", "occupation"]
values = ["Jabir", 19, "India", "Student"]
# Combine the lists into a dictionary using zip
combined_dict = dict(zip(keys, values))
print("Combined dictionary:", combined_dict)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Nazia\Desktop\My Python Program> ^C
PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/AppData/Local/Programs/Python/P
ist.py"
Combined dictionary: {'name': 'Jabir', 'age': 19, 'city': 'India', 'occupation': 'Student'}
PS C:\Users\Nazia\Desktop\My Python Program>
```

Question 14→ Write a program combining lists that combines these lists into a dictionary?

Question_14_84_char.py > determine_file_type

```python
import string
def count_characters(filename):
    # Initialize a dictionary to store character frequencies
    char_count = {char: 0 for char in string.printable}  # Track all printable ASCII characters

    # Read the file and count each character's frequency
    with open(filename, 'r') as file:
        content = file.read()
        for char in content:
            if char in char_count:
                char_count[char] += 1

    return char_count
```

```python
def determine_file_type(char_count):
    # Define a rough heuristic based on character frequencies
    python_keywords = ["def", "import", "self", ":", "#"]
    c_keywords = ["int", "#include", "{", "}", ";"]
    text_chars = set(string.ascii_letters + string.whitespace + ".,!?")  # Common in plain text

    # Count occurrences of Python and C indicators
    python_score = sum(char_count.get(char, 0) for char in ":#") + sum(1 for kw in python_keywords if kw in char_count)
    c_score = sum(char_count.get(char, 0) for char in "{};") + sum(1 for kw in c_keywords if kw in char_count)

    # Check if text-like characters are in higher proportion
    text_score = sum(char_count.get(char, 0) for char in text_chars)

    # Heuristic thresholds for determining the file type
    if python_score > c_score and python_score > text_score:
        return "Python program file"
    elif c_score > python_score and c_score > text_score:
        return "C program file"
    else:
        return "Text file"

# Example usage
```

```python
filename = "Question_14.txt"
char_count = count_characters(filename)
file_type = determine_file_type(char_count)
print(f"The file '{filename}' is likely a {file_type}.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Nazia\Desktop\My Python Program> ^C
PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/AppDa

The file 'Question_14.txt' is likely a C program file.
PS C:\Users\Nazia\Desktop\My Python Program>
```

Question 15→ Write a program to print each line of a file in reverse order.

Question_15_txt_file.py > ...

```python
def print_lines_in_reverse(filename):
    # Open the file in read mode
    with open(filename, 'r') as file:
        # Read each line, strip trailing whitespace, and print in reverse
        for line in file:
            print(line.strip()[::-1])

# Example usage
filename = "C:/Users/Nazia/Desktop/My Python Program/Question_15_text_file.txt"
print_lines_in_reverse(filename)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Nazia\Desktop\My Python Program> ^C
PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/AppData/Local/Programs/Python/P
"
elif nohtyp ym
PS C:\Users\Nazia\Desktop\My Python Program>
```

Question 16→ Write a program to compute the number of characters, words and lines in a file?

Question_16_Count_char_txt.py > count_file_details

```python
def count_file_details(filename):
    try:
        with open(filename, 'r') as file:
            text = file.read()
            lines = text.splitlines()
            words = text.split()

            num_chars = len(text)
            num_words = len(words)
            num_lines = len(lines)

            print(f"Characters: {num_chars}")
            print(f"Words: {num_words}")
            print(f"Lines: {num_lines}")
    except FileNotFoundError:
        print(f"File {filename} not found.")

# Example usage
filename = 'Question_15_text_file.txt'
count_file_details(filename)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Nazia\Desktop\My Python Program> ^C
PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users
txt.py"
Characters: 14
Words: 3
Lines: 1
PS C:\Users\Nazia\Desktop\My Python Program>
```

Question 17→ Write a function ball collide that takes two balls as parameters and computers if they are colliding. Your function should return a Boolean representing whether or not the balls are colliding. Hint: Represent a ball on a plane as a tuple of (x, y, r), r being the radius. If (distance between two balls centres) <= (sum of their radii) then (they are colliding)?

Question_17_Ball_collide.py > ...

```python
import math

def ball_collide(ball1, ball2):
    (x1, y1, r1) = ball1
    (x2, y2, r2) = ball2

    distance = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
    return distance <= (r1 + r2)

# Example usage:
ball1 = (0, 0, 5)
ball2 = (4, 3, 3)
print(ball_collide(ball1, ball2))  # Output should be True
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Nazia\Desktop\My Python Program> ^C
PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/AppData/Loc
e.py"
True
PS C:\Users\Nazia\Desktop\My Python Program>
```

Question 18→ Find mean, median, mode for the given set of numbers in a list

Question_18_Mean_Stats.py > ...

```python
from statistics import mean, median, mode

def calculate_statistics(numbers):
    mean_value = mean(numbers)
    median_value = median(numbers)
    mode_value = mode(numbers)

    return mean_value, median_value, mode_value

# Example usage
numbers = [1, 2, 3, 4, 5, 5, 6, 7, 8, 9]
mean_value, median_value, mode_value = calculate_statistics(numbers)

print(f"Mean: {mean_value}")
print(f"Median: {median_value}")
print(f"Mode: {mode_value}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Nazia\Desktop\My Python Program> ^C
PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/AppData/Local/Progra
py"
Mean: 5
Median: 5.0
Mode: 5
PS C:\Users\Nazia\Desktop\My Python Program>
```

Question 19→ Write a function nearly equal to test whether two strings are nearly equal. Two strings a and b are nearly equal when a can be generated by a single mutation on b.

Question_19_a&b.py > ...

```python
def nearly_equal(a, b):
    if len(a) != len(b):
        return False

    mutation_count = 0

    for char1, char2 in zip(a, b):
        if char1 != char2:
            mutation_count += 1
            if mutation_count > 1:
                return False

    return mutation_count == 1

# Example usage:
a = "hello"
b = "hallo"
print(nearly_equal(a, b))  # Output should be True

a = "hello"
b = "hello"
print(nearly_equal(a, b))  # Output should be False
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Nazia\Desktop\My Python Program> ^C
PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/A
True
False
PS C:\Users\Nazia\Desktop\My Python Program>
```

Question 20→ Write a function dups to find all duplicates in the list.

Question_20_Duplicate.py > ...

```python
def find_duplicates(input_list):
    duplicates = set()
    seen = set()

    for item in input_list:
        if item in seen:
            duplicates.add(item)
        else:
            seen.add(item)

    return list(duplicates)

# Example usage:
input_list = [1, 2, 3, 4, 5, 6, 2, 3, 7, 8, 5]
duplicates = find_duplicates(input_list)
print(f"Duplicates: {duplicates}")  # Output should be [2, 3, 5]
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Nazia\Desktop\My Python Program> & C:/Users/Nazia/AppData/Local/Pr
y"
Duplicates: [2, 3, 5]
PS C:\Users\Nazia\Desktop\My Python Program>
```