

**St Stephen College  
University of Delhi**

**Course :- Bsc. (prog) Computer Science**

**Semester :- 3<sup>rd</sup>**

**Year :- 2025-2026**

**Paper Name :- Object Oriented Programming  
Using Python**

**Unique Paper Code :- 2342011104**

**Practical File**

**Roll Number :- 24080582015**

**Name :- Vishal Kumar Jha**

## **Practical 1 :- Write a program to find the roots of a quadratic equation.**

```
main.py
1 import math
2
3 a = float(input("Enter value of a: "))
4 b = float(input("Enter value of b: "))
5 c = float(input("Enter value of c: "))
6
7 # Calculate discriminant
8 d = b*b - 4*a*c
9
10 if d > 0:
11     root1 = (-b + math.sqrt(d)) / (2*a)
12     root2 = (-b - math.sqrt(d)) / (2*a)
13     print("Roots are real and different:")
14     print("Root 1 =", root1)
15     print("Root 2 =", root2)
16
17 elif d == 0:
18     root = -b / (2*a)
19     print("Roots are real and equal:")
20     print("Root =", root)
21
22 else:
23     real = -b / (2*a)
24     imag = math.sqrt(-d) / (2*a)
25     print("Roots are complex:")
26     print(f"Root 1 = {real} + {imag}i")
27     print(f"Root 2 = {real} - {imag}i")
28 
```

Enter value of a: 1  
Enter value of b: 6  
Enter value of c: 2  
Roots are real and different:  
Root 1 = -0.3542486889354093  
Root 2 = -5.645751311064591  
...Program finished with exit code 0  
Press ENTER to exit console.

## **Practical 2 :- Write a program to accept a number 'n' and a. Check if 'n' is prime b.**

**Generate all prime numbers till 'n' c. Generate first 'n' prime numbers This program may be done using functions.**

```
main.py
1 import math
2
3 # Part (a): Function to check if n is prime
4 def isPrime(n):
5     if n <= 1:
6         return False
7     for i in range(2, int(math.sqrt(n)) + 1):
8         if n % i == 0:
9             return False
10    return True
11
12 # Part (b): Function to generate all prime numbers till n
13 def primesTillN(n):
14     print("Prime numbers till {}:".format(n))
15     for i in range(2, n + 1):
16         if isPrime(i):
17             print(i, end=" ")
18     print()
19
20 # Part (c): Function to generate first n prime numbers
21 def firstNPromises(n):
22     print("First {} prime numbers:".format(n))
23     count = 0
24     num = 2
25
26     while count < n:
27         if isPrime(num):
28             print(num, end=" ")
29             count += 1
30         num += 1
31
32     print()
33
34 # Main program
35 n = int(input("Enter a number: "))
36 
```

Enter a number: 23  
23 is a prime number.  
Prime numbers till 23:  
2 3 5 7 11 13 17 19 23  
First 23 prime numbers:  
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83  
...Program finished with exit code 0  
Press ENTER to exit console.

```

54
55
56 # Main program
57 n = int(input("Enter a number: "))
58
59 # Part (a)
60 if isPrime(n):
61     print(n, "is a prime number.")
62 else:
63     print(n, "is not a prime number.")
64
65 # Part (b)
66 primesTillN(n)
67
68 # Part (c)
69 firstNPrimes(n)
70

```

**Practical 3 :- Write a program to create a pyramid of the character '\*' and a reverse pyramid.**

```

main.py
1 # Function to print pyramid of '*'
2 def pyramid(n):
3     for i in range(1, n + 1):
4         print(" " * (n - i) + "*" * (2 * i - 1))
5
6 # Function to print reverse pyramid of '*'
7 def reverse_pyramid(n):
8     for i in range(n, 0, -1):
9         print(" " * (n - i) + "*" * (2 * i - 1))
10
11 # Main program
12 rows = int(input("Enter number of rows: "))
13
14 print("Pyramid:")
15 pyramid(rows)
16
17 print("Reverse Pyramid:")
18 reverse_pyramid(rows)
19

```

Pyramid:  
\*  
\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*  
Reverse Pyramid:  
\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*  
\*\*\*  
\*

...Program finished with exit code 0  
Press ENTER to exit console.

**Practical 4 :- Write a program that accepts a character and performs the following: a. print whether the character is a letter or numeric digit or a special character. b. if the character is a letter, print whether the letter is uppercase or lowercase c. if the character is a numeric digit, prints its name in text (e.g., if input is 9, output is NINE)**

```

main.py
1 # Program to categorize a character
2
3 ch = input("Enter a character: ")
4
5 # Part (a): Check type of character
6 if ch.isalpha():
7     print("It is a letter.")
8
9     # Part (b): Check uppercase/Lowercase
10    if ch.isupper():
11        print("The letter is uppercase.")
12    else:
13        print("The letter is lowercase.")
14
15    elif ch.isdigit():
16        print("It is a numeric digit.")
17
18    # Part (c): Print digit as word
19    names = ["ZERO", "ONE", "TWO", "THREE", "FOUR",
20             "FIVE", "SIX", "SEVEN", "EIGHT", "NINE"]
21    print("In words:", names[int(ch)])
22
23 else:
24     print("It is a special character.")
25
26

```

Enter a character: t  
It is a letter.  
The letter is lowercase.  
...Program finished with exit code 0  
Press ENTER to exit console.

**Practical 5 :- Write a program to perform the following operations on a string: a. Find the frequency of a character in a string. b. Replace a character by another character in a string.**

**c. Remove the first occurrence of a character from a string. d. Remove all occurrences of a character from a string.**

```
main.py
1 # Part (a): Find frequency of a character
2 def frequency(s, ch):
3     return s.count(ch)
4
5 # Part (b): Replace a character by another
6 def replace_char(s, old, new):
7     return s.replace(old, new)
8
9 # Part (c): Remove first occurrence of a character
10 def remove_first(s, ch):
11     return s.replace(ch, "", 1)
12
13 # Part (d): Remove all occurrences of a character
14 def remove_all(s, ch):
15     return s.replace(ch, "")
16
17
18 # Main program
19 s = input("Enter a string: ")
20
21 # Part (a)
22 ch = input("Enter character to find frequency: ")
23 print("Frequency:", frequency(s, ch))
24
25 # Part (b)
26 old = input("Character to replace: ")
27 new = input("Replace with: ")
28 print("Updated string:", replace_char(s, old, new))
29
30 # Part (c)
31 ch = input("Enter character to remove first occurrence: ")
32 print("String after removing first occurrence:", remove_first(s, ch))
33
34 # Part (d)
35 ch = input("Enter character to remove all occurrences: ")
36 print("String after removing all occurrences:", remove_all(s, ch))
```

Enter a string: jhjhdsdjhashjbqhwegd  
Enter character to find frequency: h  
Frequency: 5  
Character to replace: j  
Replace with: p  
Updated string: phphsdphashpbqhwegd  
Enter character to remove first occurrence: s  
String after removing first occurrence: jhjhdsjhashjbqhwegd  
Enter character to remove all occurrences: j  
String after removing all occurrences: hhsdhashbqhwegd  
...Program finished with exit code 0  
Press ENTER to exit console.

**Practical 6:- Write a program to swap the first n characters of two strings.**

```
main.py
1 # Program to swap first n characters of two strings
2
3 s1 = input("Enter first string: ")
4 s2 = input("Enter second string: ")
5 n = int(input("Enter value of n: "))
6
7 # Take first n characters of both strings
8 part1 = s1[:n]
9 part2 = s2[:n]
10
11 # Build new swapped strings
12 new_s1 = part2 + s1[n:]
13 new_s2 = part1 + s2[n:]
14
15 print("After swapping:")
16 print("String 1:", new_s1)
17 print("String 2:", new_s2)
```

Enter first string: Vishal  
Enter second string: Viraj  
Enter value of n: 4  
After swapping:  
String 1: Viraal  
String 2: Vishj  
...Program finished with exit code 0  
Press ENTER to exit console.

**Practical 7 :- Write a function that accepts two strings and returns the indices of all the occurrences of the second string in the first string as a list. If the second string is not present in the first string then it should return -1.**

```
main.py
1 def find_occurrences(s, sub):
2     indices = []
3     start = 0
4
5     while True:
6         pos = s.find(sub, start) # find substring from 'start'
7         if pos == -1:
8             break
9         indices.append(pos)
10        start = pos + 1 # move forward to find next occurrence
11
12    if len(indices) == 0:
13        return -1
14    return indices
15
16
17 # Example usage
18 s = input("Enter main string: ")
19 sub = input("Enter substring: ")
20
21 result = find_occurrences(s, sub)
22 print("Output:", result)
23
```

...Program finished with exit code 0  
Press ENTER to exit console.

**Practical 8 :- Write a program to create a list of the cubes of only the even integers appearing in the input list (may have elements of other types also) using the following: a. 'for' loop b. list comprehension**

```
main.py
1 # Using For Loop
2 lst = [1, 2, "hello", 4, 5.5, 6]
3 result = []
4
5 for item in lst:
6     if type(item) == int and item % 2 == 0:
7         result.append(item ** 3)
8
9 print(result)
10
11 #Using List comprehension
12 lst = [1, 2, "hello", 4, 5.5, 6]
13
14 result = [item**3 for item in lst if type(item) == int and item % 2 == 0]
15
16 print(result)
17
```

[8, 64, 216]  
[8, 64, 216]

...Program finished with exit code 0  
Press ENTER to exit console.

**Practical 9 :- Write a program to read a file and**

- a. **Print the total number of characters, words and lines in the file.**
- b. **Calculate the frequency of each character in the file. Use a variable of dictionary type to maintain the count.**
- c. **Print the words in reverse order.**
- d. **Copy even lines of the file to a file named 'File1' and odd lines to another file named 'File2'.**

```
main.py sample.txt File1.txt File2.txt
1 filename = "sample.txt"
2
3 # ----- Part (a): Counts -----
4 with open(filename, "r") as f:
5     content = f.read()
6
7 # total characters
8 num_chars = len(content)
9
10 # total words
11 words = content.split()
12 num_words = len(words)
13
14 # total lines
15 with open(filename, "r") as f:
16     lines = f.readlines()
17 num_lines = len(lines)
18
19 print("Total Characters:", num_chars)
20 print("Total Words:", num_words)
21 print("Total Lines:", num_lines)
22
23 # ----- Part (b): Character Frequency -----
24 freq = {}
25
26 for ch in content:
27     if ch in freq:
28         freq[ch] += 1
29     else:
30         freq[ch] = 1
31
32 print("\nCharacter Frequencies:")
33 for ch, count in freq.items():
34     print(repr(ch), ":", count)
35
```

Total Characters: 75  
Total Words: 12  
Total Lines: 2  
Character Frequencies:  
'B' : 1  
'e' : 8  
'l' : 5  
'o' : 3  
' ' : 10  
'M' : 1  
'y' : 2  
'S' : 4  
'f' : 2  
'V' : 1  
'i' : 5  
'h' : 3  
'a' : 3  
'j' : 1  
'\n' : 1  
'I' : 1  
'm' : 3  
'C' : 1  
'p' : 1  
'u' : 2  
't' : 4  
'r' : 3  
'S' : 1  
'c' : 2  
'n' : 3  
'd' : 1  
'D' : 1  
'U' : 1  
'v' : 1

```
36 # ----- Part (c): Print words in reverse order -----
37 print("\nWords in reverse order:")
38 rev_words = words[::-1]
39 print(" ".join(rev_words))
40
41 # ----- Part (d): Copy even and odd Lines -----
42 with open("File1.txt", "w") as f_even, open("File2.txt", "w") as f_odd:
43     for i, line in enumerate(lines, start=1):
44         if i % 2 == 0:
45             f_even.write(line)
46         else:
47             f_odd.write(line)
48
49 print("\nEven lines written to File1.txt")
50 print("Odd lines written to File2.txt")
```

Words in reverse order:  
University Delhi from student Science Computer am I Jha Vishal Myself Hello  
Even lines written to File1.txt  
Odd lines written to File2.txt  
...Program finished with exit code 0  
Press ENTER to exit console.

```
main.py sample.txt File1.txt File2.txt
1 Hello Myself Vishal Jha
2 I am Computer Science student from Delhi University
```

```
main.py sample.txt File1.txt File2.txt
1 I am Computer Science student from Delhi University
```

```
main.py sample.txt File1.txt File2.txt
1 Hello Myself Vishal Jha
2
```

**Practical 10 :- Write a function that prints a dictionary where the keys are numbers between 1 and 5 and the values are the cubes of the keys.**

```
main.py sample.txt File1.txt File2.txt Language Python 3 input
1 def print_cubes_dict():
2     cubes = {}
3     for i in range(1, 6):
4         cubes[i] = i ** 3
5     print(cubes)
6
7 print_cubes_dict()
```

...Program finished with exit code 0  
Press ENTER to exit console.

**Practical 11 :- Consider a tuple t1=(1, 2, 5, 7, 9, 2, 4, 6, 8, 10). Write a program to perform following operations:** a) Print half the values of the tuple in one line and the other half in the next line. b) Print another tuple whose values are even numbers in the given tuple. c) Concatenate a tuple t2=(11, 13, 15) with t1. d) Return maximum and minimum value from this tuple.

```
main.py sample.txt File1.txt File2.txt Language Python 3 input
1 t1 = (1, 2, 5, 7, 9, 2, 4, 6, 8, 10)
2
3 # (a) Print half values in two Lines
4 mid = len(t1) // 2
5 print("First half:", t1[:mid])
6 print("Second half:", t1[mid:])
7
8 # (b) Tuple of even numbers
9 even_tuple = tuple(x for x in t1 if x % 2 == 0)
10 print("Even numbers tuple:", even_tuple)
11
12 # (c) Concatenate t2 with t1
13 t2 = (11, 13, 15)
14 t3 = t1 + t2
15 print("Concatenated tuple:", t3)
16
17 # (d) Maximum and minimum values
18 print("Maximum value:", max(t1))
19 print("Minimum value:", min(t1))
20
```

First half: (1, 2, 5, 7, 9)  
Second half: (2, 4, 6, 8, 10)  
Even numbers tuple: (2, 2, 4, 6, 8, 10)  
Concatenated tuple: (1, 2, 5, 7, 9, 2, 4, 6, 8, 10, 11, 13, 15)  
Maximum value: 10  
Minimum value: 1

...Program finished with exit code 0  
Press ENTER to exit console.

**Practical 12 :- Define a class Employee that stores information about employees in the company. The class should contain the following:**

- (i) data members - count (to keep a record of all the objects being created for this class) and for every employee: \$\text{employee number, Name, Dept, Basic, DA}\$ and \$\text{HRA}\$.
- (ii) function members: a. \$\text{\textit{init}}\$ method to initialize and/or update the members. Add statements to ensure that the program is terminated if any of \$\text{Basic, DA}\$ and \$\text{HRA}\$ is set to a negative value. b. \$\text{function salary}\$, that returns salary as the sum of \$\text{Basic, DA}\$ and \$\text{HRA}\$. c. \$\text{\textit{del}}\$ function to decrease the number of objects created for the class. d. \$\text{\textit{str}}\$ function to display the details of an employee along with the salary of an employee in a proper format.

The screenshot shows a Python code editor with the following code:

```
Employee Number: 101
Name: Rahul
Department: IT
Basic: 20000
DA: 5000
HRA: 3000
Salary: 28000
Total Employees: 1

...Program finished with exit code 0
Press ENTER to exit console.
```

```
1 - class Employee:
2 -     # class variable to count objects
3 -     count = 0
4 -
5 -     def __init__(self, emp_no, name, dept, basic, da, hra):
6 -         # terminate program if any value is negative
7 -         if basic < 0 or da < 0 or hra < 0:
8 -             print("Error: Basic, DA, and HRA cannot be negative.")
9 -             exit()
10 -
11     # initialize instance variables
12     self.emp_no = emp_no
13     self.name = name
14     self.dept = dept
15     self.basic = basic
16     self.da = da
17     self.hra = hra
18 -
19     # increase count
20     Employee.count += 1
21
22     # (b) salary function
23     def salary(self):
24         return self.basic + self.da + self.hra
25
26     # (c) destructor
27     def __del__(self):
28         Employee.count -= 1
29
30     # (d) string representation
31     def __str__(self):
32         return f"Employee Number: {self.emp_no}\n"
33         f"Name: {self.name}\n"
34         f"Department: {self.dept}\n"
35         f"Basic: {self.basic}\n"
36         f"DA: {self.da}\n"
37         f"HRA: {self.hra}\n"
38         f"Salary: {self.salary()}\n"
39         f"Total Employees: {Employee.count}"
40
41 # ----- Example Usage -----
42
43 e1 = Employee(101, "Rahul", "IT", 20000, 5000, 3000)
44 print(e1)
```

**Practical 13 :- Write a program to define a class '2DPoint' with coordinates x and y as attributes. Create relevant methods and print the objects. Also define a method distance to calculate the distance between any two point objects.**

```
main.py
1 import math
2
3 class TwoDPoint:
4     def __init__(self, x, y):
5         self.x = x
6         self.y = y
7
8     # method to print object details
9     def __str__(self):
10        return f"Point({self.x}, {self.y})"
11
12     # method to calculate distance from another point
13     def distance(self, other):
14         return math.sqrt((self.x - other.x)**2 + (self.y - other.y)**2)
15
16
17 # ----- Example Usage -----
18
19 p1 = TwoDPoint(3, 4)
20 p2 = TwoDPoint(7, 1)
21
22 print("Point 1:", p1)
23 print("Point 2:", p2)
24
25 print("Distance between points:", p1.distance(p2))
26
27
28
```

Language Python 3    Input

Point 1: Point(3, 4)  
Point 2: Point(7, 1)  
Distance between points: 5.0  
...Program finished with exit code 0  
Press ENTER to exit console.

**Practical 14 :- Inherit the above class (referring to '2DPoint' from Q13) to create a "3DPoint" with additional attribute z. Override the method defined in "2DPoint" class, to calculate distance between two points of the "3Dpoint" class.**

The screenshot shows a Python code editor with a file named main.py. The code defines a base class TwoDPoint and a derived class ThreeDPoint. The ThreeDPoint class inherits from TwoDPoint and adds a new attribute z. It overrides the \_\_init\_\_ and distance methods to include the z coordinate. The code is run in a terminal window, showing the creation of two 3D points (p1 and p2) and the calculation of their distance.

```
main.py
1 import math
2
3 # Base Class (from previous question)
4 class TwoDPoint:
5     def __init__(self, x, y):
6         self.x = x
7         self.y = y
8
9     def __str__(self):
10        return f"Point({self.x}, {self.y})"
11
12     def distance(self, other):
13         return math.sqrt((self.x - other.x)**2 + (self.y - other.y)**2)
14
15
16 # Derived Class
17 class ThreeDPoint(TwoDPoint):
18
19     def __init__(self, x, y, z):
20         super().__init__(x, y) # Initialize x and y from parent class
21         self.z = z
22
23     def __str__(self):
24        return f"Point({self.x}, {self.y}, {self.z})"
25
26     # Overriding distance method
27     def distance(self, other):
28         return math.sqrt(
29             (self.x - other.x)**2 +
30             (self.y - other.y)**2 +
31             (self.z - other.z)**2
32         )
33
34
35 # ----- Example Usage -----
36
37 p1 = ThreeDPoint(1, 2, 3)
38 p2 = ThreeDPoint(4, 6, 8)
39
40 print("Point 1:", p1)
41 print("Point 2:", p2)
42
43 print("Distance between points:", p1.distance(p2))
44
```

**Practical 15 :- Write a program to accept a name from a user. Raise and handle appropriate exception(s) if the text entered by the user contains digits and/or special characters.**

The screenshot shows a Python code editor with a file named main.py. The code defines a custom exception InvalidNameError. It prompts the user for a name and checks if it contains any digits or special characters. If found, it raises the InvalidNameError. Otherwise, it prints a valid message. The code is run in a terminal window, demonstrating the exception handling.

```
main.py
1 class InvalidNameError(Exception):
2     pass
3
4 try:
5     name = input("Enter your name: ")
6
7     # Check for invalid characters
8     for ch in name:
9         if not ch.isalpha() and ch != " ":
10             raise InvalidNameError("Name should not contain digits or special characters.")
11
12     print("Valid name entered:", name)
13
14 except InvalidNameError as e:
15     print("Error:", e)
16
```