

# DELHI UNIVERSITY

*DIYAL SINGH*

*EVENING COLLEGE*



**PROGRAMMING FUNDAMENTALS**

**USING**

# **PYTHON**

**SUBMITTED BY:-**

**JATOTH DINESH**

**B.A. PROGRAM**

**SECTION A**

**ROLL NO.:-23BAA009**

**SUBMITTED TO:-**

**Ms.MRINALI**

# **INDEX**

PRACTICAL LIST	DATE	PAGE NO.	REMARKS																		
<div>1. WAP to calculate total marks, percentage and grade of a student. Marks obtained in each of three subjects are to be input by the user. Assign grades according to the following criteria:<div>Grade A: if Percentage <math>\geq</math> 80 Grade B: if Percentage <math>\geq</math> 60 and Percentage <math>&lt;</math> 80 Grade C: if Percentage <math>\geq</math> 40 and Percentage <math>&lt;</math> 60 Grade D: if Percentage <math>&lt;</math> 40</div></div>																					
<div>2. WAP to print factors of a given number.</div>																					
<div>3. WAP to add N natural numbers and display their sum.</div>																					
<div>4. WAP to print the following conversion table (use looping constructs):<table><tr><th>Height (in Feet)</th><th>Height (in inches)</th></tr><tr><td>5.0 ft</td><td>60 inches</td></tr><tr><td>5.1ft</td><td>61.2 inches</td></tr><tr><td>.</td><td>.</td></tr><tr><td>.</td><td>.</td></tr><tr><td>.</td><td>.</td></tr><tr><td>5.8 ft</td><td>69.6 inches</td></tr><tr><td>5.9 ft</td><td>70.8 inches</td></tr><tr><td>6.0 ft</td><td>72 inches</td></tr></table></div>	Height (in Feet)	Height (in inches)	5.0 ft	60 inches	5.1ft	61.2 inches	.	.	.	.	.	.	5.8 ft	69.6 inches	5.9 ft	70.8 inches	6.0 ft	72 inches			
Height (in Feet)	Height (in inches)																				
5.0 ft	60 inches																				
5.1ft	61.2 inches																				
.	.																				
.	.																				
.	.																				
5.8 ft	69.6 inches																				
5.9 ft	70.8 inches																				
6.0 ft	72 inches																				
<div>5. WAP that takes a positive integer n and the produce n lines of output as shown:<div>* * *</div></div>																					

<p>*** **** (sample output for n = 4)</p>			
6. Write a menu driven program using user defined functions to print the area of rectangle, square, circle and triangle by accepting suitable input from user.			
7. Write a function that calculates factorial of a number n.			
8. WAP to print the series and its sum: (use functions)  $1/1! + 1/2! + 1/3! + \dots + 1/n!$			
9. WAP to perform the following operations on an input string a. Print length of the string b. Find frequency of a character in the string c. Print whether characters are in uppercase or lowercase			
10. WAP to create two lists: one of even numbers and another of odd numbers. The program should demonstrate the various operations and methods on lists.			
11. WAP to create a dictionary where keys are numbers between 1 and 5 and the values are the cubes of the keys.			
12. WAP to create a tuple t1 = (1, 2, 5, 7, 2, 4). The program should perform the following: a. Print tuple in two lines, line 1 containing the first half of tuple and second line having the second half. b. Concatenate tuple t2 = (10, 11) with t1.			

# Practical 1

## SOURCE CODE:-

```
# Function to calculate total marks, percentage, and grade
def calculate_grade(subject1, subject2, subject3):
    total_marks = subject1 + subject2 + subject3
    percentage = (total_marks / 300) * 100
    # Assigning grades based on percentage
    if percentage >= 80:
        grade = 'A'
    elif percentage >= 60:
        grade = 'B'
    elif percentage >= 40:
        grade = 'C'
    else:
        grade = 'D'
    return total_marks, percentage, grade

# Input marks from the user
subject1 = float(input("Enter marks obtained in Subject 1: "))
subject2 = float(input("Enter marks obtained in Subject 2: "))
subject3 = float(input("Enter marks obtained in Subject 3: "))

# Calculate total marks, percentage, and grade
total_marks, percentage, grade = calculate_grade(subject1, subject2, subject3)

# Displaying the results
print("\nTotal Marks: ", total_marks)
print("Percentage: {:.2f}%".format(percentage))
print("Grade: ", grade)
```

## Output:-

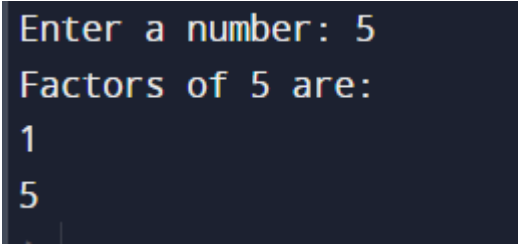
```
Enter marks obtained in Subject 1: 99
Enter marks obtained in Subject 2: 100
Enter marks obtained in Subject 3: 89
Total Marks: 288.0
Percentage: 96.00%
Grade: A
```

## **Practical 2**

### **SOURCE CODE:-**

```
# Function to print factors of a given number
def print_factors(number):
    print("Factors of", number, "are:")
    for i in range(1, number + 1):
        if number % i == 0:
            print(i)
# Input the number from the user
num = int(input("Enter a number: "))
# Call the function to print factors
print_factors(num)
```

### **Output:-**

A screenshot of a terminal window with a dark background. It shows the output of the program for the input number 5. The text is as follows:

```
Enter a number: 5
Factors of 5 are:
1
5
```

## **Practical 3**

### **SOURCE CODE:-**

```
# Function to calculate the sum of first N natural numbers
def sum_of_natural_numbers(n):
    return n * (n + 1) // 2

# Input the value of N from the user
N = int(input("Enter the value of N: "))

# Calculate the sum of first N natural numbers
result = sum_of_natural_numbers(N)

# Display the result
print("Sum of first", N, "natural numbers is:", result)
```

### **Output:-**

```
Enter the value of N: 7
Sum of first 7 natural numbers is: 28
```

## Practical 4

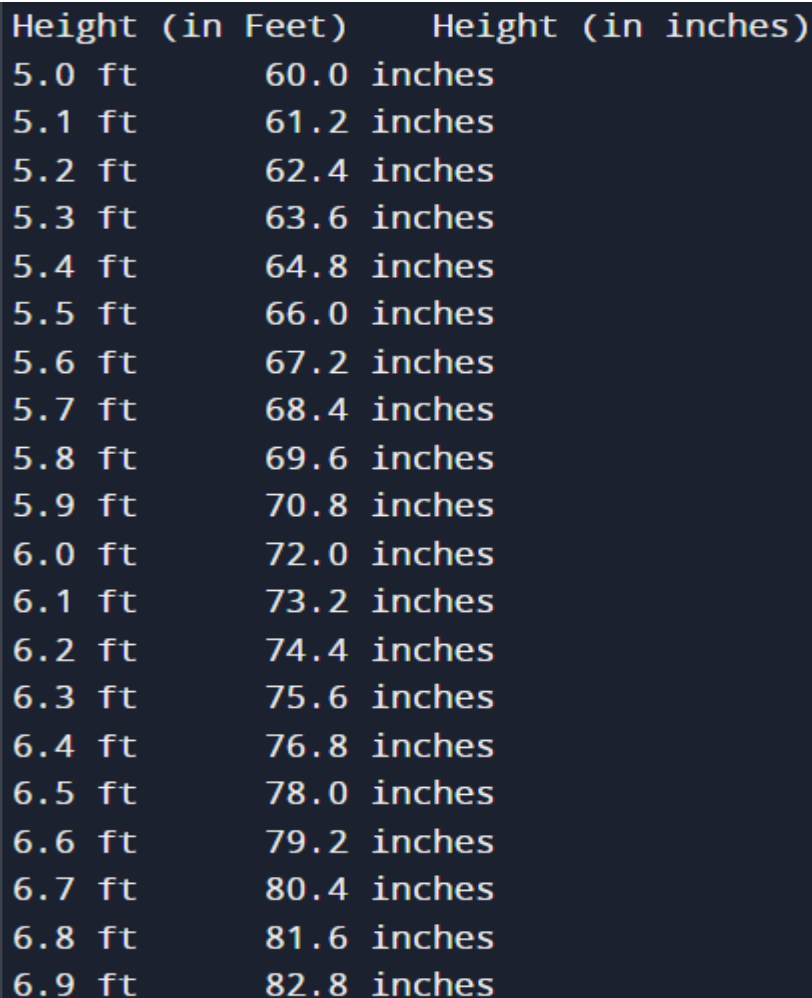
### SOURCE CODE:-

```
# Function to convert height from feet to inches
def convert_to_inches(feet):
    return feet * 12

# Print the table header
print("Height (in Feet)\tHeight (in inches)")

# Loop to print the conversion table
for i in range(5, 7):
    for j in range(0, 10, 1):
        height_in_feet = i + j / 10
        height_in_inches = convert_to_inches(height_in_feet)
        print("{:.1f} ft\t{:.1f} inches".format(height_in_feet, height_in_inches))
```

### Output:-



Height (in Feet)	Height (in inches)
5.0 ft	60.0 inches
5.1 ft	61.2 inches
5.2 ft	62.4 inches
5.3 ft	63.6 inches
5.4 ft	64.8 inches
5.5 ft	66.0 inches
5.6 ft	67.2 inches
5.7 ft	68.4 inches
5.8 ft	69.6 inches
5.9 ft	70.8 inches
6.0 ft	72.0 inches
6.1 ft	73.2 inches
6.2 ft	74.4 inches
6.3 ft	75.6 inches
6.4 ft	76.8 inches
6.5 ft	78.0 inches
6.6 ft	79.2 inches
6.7 ft	80.4 inches
6.8 ft	81.6 inches
6.9 ft	82.8 inches

## Practical 5

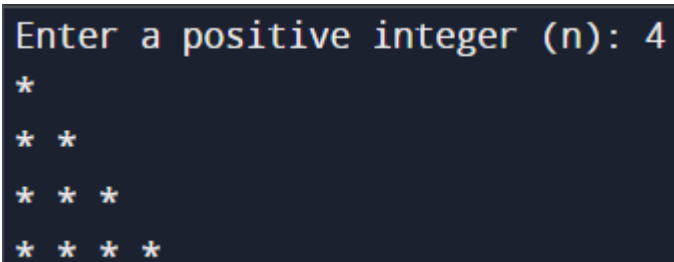
### SOURCE CODE:-

```
# Function to print the pattern
def print_pattern(n):
    for i in range(1, n + 1):
        print('* ' * i)

# Input the value of n from the user
n = int(input("Enter a positive integer (n): "))

# Call the function to print the pattern
print_pattern(n)
```

### Output:-



```
Enter a positive integer (n): 4
*
* *
* * *
* * * *
```



## **Practical 6**

### **SOURCE CODE:-**

```
import math

# Function to calculate the area of a rectangle
def rectangle_area(length, width):
    return length * width

# Function to calculate the area of a square
def square_area(side):
    return side * side

# Function to calculate the area of a circle
def circle_area(radius):
    return math.pi * radius * radius

# Function to calculate the area of a triangle
def triangle_area(base, height):
    return 0.5 * base * height

# Menu function
def menu():
    print("\nMenu:")
    print("1. Rectangle")
    print("2. Square")
    print("3. Circle")
    print("4. Triangle")
    print("5. Exit")

# Input function
def get_user_choice():
    return int(input("Enter your choice (1-5): "))

# Main program
while True:
    menu()
    choice = get_user_choice()
    if choice == 1:
        length = float(input("Enter length of rectangle: "))
        width = float(input("Enter width of rectangle: "))
        print("Area of rectangle:", rectangle_area(length, width))
```

```
elif choice == 2:
    side = float(input("Enter side length of square: "))
    print("Area of square:", square_area(side))
elif choice == 3:
    radius = float(input("Enter radius of circle: "))
    print("Area of circle:", circle_area(radius))
elif choice == 4:
    base = float(input("Enter base of triangle: "))
    height = float(input("Enter height of triangle: "))
    print("Area of triangle:", triangle_area(base, height))
elif choice == 5:
    print("Exiting the program. Goodbye!")
    break
else:
    print("Invalid choice. Please enter a number between 1 and 5.")
```

### **Output:-**

```
Menu:
1. Rectangle
2. Square
3. Circle
4. Triangle
5. Exit
Enter your choice (1-5): 1
Enter length of rectangle: 4
Enter width of rectangle: 5
Area of rectangle: 20.0

Enter your choice (1-5): 2
Enter side length of square: 4
Area of square: 16.0

Enter your choice (1-5): 3
Enter radius of circle: 5
Area of circle: 78.53981633974483

Enter your choice (1-5): 4
Enter base of triangle: 5
Enter height of triangle: 6
Area of triangle: 15.0

Enter your choice (1-5): 5
Exiting the program. Goodbye!
> |
```

## **Practical 7**

### **SOURCE CODE:-**

```
def factorial(n):  
    if n == 0 or n == 1:  
        return 1  
    else:  
        return n * factorial(n - 1)  
  
# Example usage:  
number = int(input("Enter a number: "))  
result = factorial(number)  
print("Factorial of", number, "is:", result)
```

### **Output:-**

```
Enter a number: 7  
Factorial of 7 is: 5040
```

## Practical 8

### SOURCE CODE:-

```
import math

# Function to calculate the factorial of a number
def factorial(num):
    if num == 0 or num == 1:
        return 1
    else:
        return num * factorial(num - 1)

# Function to print the series and its sum
def print_series_and_sum(n):
    series_sum = 0
    print("Series:")
    for i in range(1, n + 1):
        term = 1 / factorial(i)
        series_sum += term
        print("{:.4f}".format(term), end=" ")
    print("\nSum of the series:", "{:.4f}".format(series_sum))

# Input the value of n from the user
n = int(input("Enter the value of n: "))

# Call the function to print the series and its sum
print_series_and_sum(n)
```

### Output:-

```
Enter the value of n: 8
Series:
1.0000 0.5000 0.1667 0.0417 0.0083 0.0014 0.0002 0.0000
Sum of the series: 1.7183
```

## **Practical 9**

### **SOURCE CODE:-**

```
# Function to perform operations on the input string
def string_operations(input_string):
    # a. Print length of the string
    print("a. Length of the string:", len(input_string))

    # b. Find frequency of a character in the string
    char_to_find = input("b. Enter a character to find its frequency: ")
    frequency = input_string.count(char_to_find)
    print(f"Frequency of '{char_to_find}' in the string: {frequency}")

    # c. Print whether characters are in uppercase or lowercase
    upper_count = sum(1 for char in input_string if char.isupper())
    lower_count = sum(1 for char in input_string if char.islower())

    print(f"c. Uppercase characters: {upper_count}")
    print(f"Lowercase characters: {lower_count}")

# Input the string from the user
user_string = input("Enter a string: ")

# Call the function to perform operations on the input string
string_operations(user_string)
```

### **Output:-**

```
Enter a string: dinesh
a. Length of the string: 6
b. Enter a character to find its frequency: s
Frequency of 's' in the string: 1
c. Uppercase characters: 0
   Lowercase characters: 6
```

## **Practical 10**

### **SOURCE CODE:-**

```
# Function to create lists of even and odd numbers
def create_lists():
    even_numbers = []
    odd_numbers = []
    # Populating the lists with even and odd numbers
    for num in range(1, 21):
        if num % 2 == 0:
            even_numbers.append(num)
        else:
            odd_numbers.append(num)
    return even_numbers, odd_numbers

# Function to demonstrate operations and methods on lists
def demonstrate_list_operations(even_list, odd_list):
    # Print the initial lists
    print("Even Numbers List:", even_list)
    print("Odd Numbers List:", odd_list)
    # Concatenate the lists
    combined_list = even_list + odd_list
    print("\nCombined List:", combined_list)
    # Append a number to the even list
    even_list.append(22)
    print("\nEven Numbers List after appending 22:", even_list)
    # Insert a number at a specific position in the odd list
    odd_list.insert(2, 23)
    print("\nOdd Numbers List after inserting 23 at index 2:", odd_list)

    # Remove a number from the combined list
    combined_list.remove(6)
    print("\nCombined List after removing 6:", combined_list)

    # Extend the even list with the odd list
    even_list.extend(odd_list)
```

```

print("\nEven Numbers List after extending with Odd Numbers List:", even_list)

# Sorting the combined list in ascending order
combined_list.sort()

print("\nCombined List after sorting in ascending order:", combined_list)

# Reverse the combined list
combined_list.reverse()

print("\nCombined List after reversing:", combined_list)

# Count occurrences of a number in the combined list
count_of_10 = combined_list.count(10)

print("\nCount of 10 in Combined List:", count_of_10)

# Call the functions to create and demonstrate lists
even_numbers_list, odd_numbers_list = create_lists()
demonstrate_list_operations(even_numbers_list, odd_numbers_list)

```

## **Output:-**

```

Even Numbers List: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
Odd Numbers List: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]

Combined List: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19]

Even Numbers List after appending 22: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22]

Odd Numbers List after inserting 23 at index 2: [1, 3, 23, 5, 7, 9, 11, 13, 15, 17, 19]

Combined List after removing 6: [2, 4, 8, 10, 12, 14, 16, 18, 20, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19]

Even Numbers List after extending with Odd Numbers List: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 1, 3, 23, 5, 7, 9, 11, 13, 15, 17, 19]

Combined List after sorting in ascending order:[1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

Combined List after reversing: [20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 5, 4, 3, 2, 1]

Count of 10 in Combined List: 1
> |

```

## **Practical 11**

### **SOURCE CODE:-**

```
# Function to create a dictionary of cubes
def create_cubes_dictionary():
    cubes_dict = {}
    for num in range(1, 6):
        cubes_dict[num] = num ** 3
    return cubes_dict

# Call the function to create the dictionary
cubes_dictionary = create_cubes_dictionary()

# Print the created dictionary
print("Dictionary with keys as numbers and values as cubes:")
for key, value in cubes_dictionary.items():
    print(f"{key}: {value}")
```

### **Output:-**

```
Dictionary with keys as numbers and values as cubes:
1: 1
2: 8
3: 27
4: 64
5: 125
```



## **Practical 12**

### **SOURCE CODE:-**

```
# Create the tuple t1
t1 = (1, 2, 5, 7, 2, 4)

# a. Print tuple in two lines
half_length = len(t1) // 2
print("a. Tuple in two lines:")
print("Line 1:", t1[:half_length])
print("Line 2:", t1[half_length:])

# b. Concatenate tuple t2 = (10, 11) with t1
t2 = (10, 11)
concatenated_tuple = t1 + t2
print("\nb. Concatenated tuple:", concatenated_tuple)
```

### **Output:-**

```
a. Tuple in two lines:
Line 1: (1, 2, 5)
Line 2: (7, 2, 4)

b. Concatenated tuple: (1, 2, 5, 7, 2, 4, 10, 11)
```