



BHADERWAH CAMPUS  
UNIVERSITY OF JAMMU

# **PYTHON PROGRAMMIN G**

**Practical file**

DEPARTMENT OF COMPUTER SCIENCE  
& IT BHADERWAH CAMPUS



**BCA**  
**Bachelor of**  
**Computer Application**

PYTHON PROGRAMMING

SUBMITTED BY:

DIKSHA RANI

SUBMITTED TO:

Dr. Ashok Sharma

DEPARTMENT OF COMPUTER SCIENCE  
& IT BHADERWAH CAMPUS

Bachelor of Computer  
Applications

**CERTIFICATE**

This is to certify that DIKSHA RANI student of BCA  
7th SEMESTER bearing university ROLL NO.  
713040005 has completed the required number of  
practical of Python Programming under the  
guidance of course CO-ordinate Ms. Jahanvi Kotwal  
from the Department of Computer Science and IT,  
Bhaderwah Campus University of Jammu.

**Signature of Incharge:**

## **ACKNOWLEDGEMENT**

I WOULD LIKE TO EXPRESS MY GRATITUDE TO ALL THOSE WHO GAVE ME THE POSSIBILITY TO COMPLETE THIS PRACTICAL FILE. I WANT TO THANK DEPT.OF COMPUTER SCIENCE & IT FOR PROVIDING ME ALL THE TOOLS AND RESOURCES FOR THE COMPLETION OF MY PRACTICAL FILE.

I HAVE FURTHER MORE TO THANK OUR CONCERNED TEACHER MS. JAHANVI KOTWAL ENCOURAGED ME TO MOVE AHEAD AND COMPLETE MY WORK IN TIME. I WOULD ALSO LIKE TO THANK OUR LAB INCHARGE ,WHOSE SIMULATING SUGGERSTIONS AND ENCOURAGEMENT HELPED ME ALL THE TIME DURING MY WORK.

THANK YOU.

***DJKSHA RANJ***

## INDEX

S.NO	NAME OF THE PYTHON PROGRAM
1.	Wap to find sum of digits of number.
2.	Wap to demonstrate String concatenation.
3.	Wap to find reverse of a number.
4.	Wap to calculate Simple Interest.
5.	Wap to prompt a user to enter a day of the week .If the entered day of the week is between 1 to 7 then display the respective name of the day.
6.	Wap to display inverted half pyramid using “*”.
7.	Wap using while loop to print factorial of a number.
8.	Wap to check whether the entered number is Armstrong number or not.
9.	Wap to demonstrate the use of list slicing.
10.	Wap to demonstrate the use of inbuilt functions.
11.	Wap to print Fibonaccl series upto 8.
12.	Wap to count words/lines in a file.
13.	Wap to create a class Student and print the details of a student.
14.	Wap to show how encapsulation works using private variables.
15.	Wap to implement abstraction using ABC(Abstract Base Case).

1.WAP TO FIND SUM OF DIGITS OF NUMBERS

```
p=int(input("Enter the number:"))
```

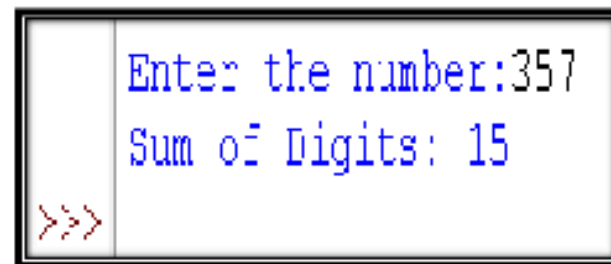
```
sum=0
```

```
for digit in str(p):
```

```
    sum=sum+int(digit)
```

```
print("Sum of Digits:",sum)
```

**OUTPUT:-**

A screenshot of a Python terminal window with a black border. The prompt is '>>>'. The user has entered '357' in response to the prompt 'Enter the number:'. The program has output 'Sum of Digits: 15'.

```
>>> Enter the number:357
Sum of Digits: 15
>>>
```

2.WAP TO DEMONSTRATE STRING CONCATENATION

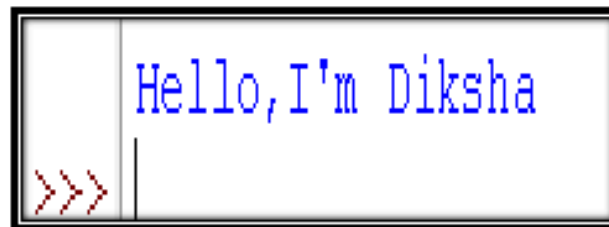
```
var1="Hello,"
```

```
var2="I'm Diksha"
```

```
var3=var1+var2
```

```
print(var3)
```

**OUTPUT:-**

A screenshot of a Python interactive shell window. The window has a black border. On the left side, there is a vertical line separating the prompt from the input/output area. The prompt is represented by three red greater-than signs (>>>). To the right of the prompt, the text 'Hello,I'm Diksha' is displayed in a blue monospace font. The text is aligned to the left of the prompt area.

3.WAP TO FIND REVERSE OF A NUMBER

```
x=9876
```

```
rev=0
```

```
while(x>0):
```

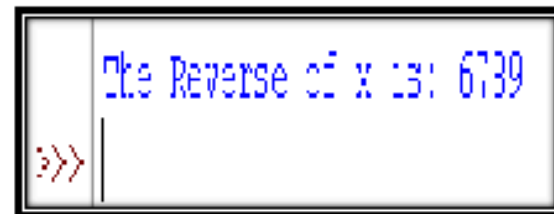
```
    a=x%10
```

```
    rev=rev*10+a
```

```
    x=x//10
```

```
print("The Reverse of x is: "rev)
```

**OUTPUT:-**

A screenshot of a Python interpreter window. The window has a title bar and a main area with a light gray background. On the left side, there is a vertical line. To the left of this line, the text '3>>' is displayed in red. To the right of the line, the text 'The Reverse of x is: 6739' is displayed in blue. The window is bordered by a thick black frame.

```
3>> The Reverse of x is: 6739
```



#### 4.WAP TO CALCULATE SIMPLE INTEREST

P=300

R=1

T=4

Simple\_interest=(P\*R\*T)/100

print("The Simple Interest is:",Simple\_interest)

#### OUTPUT:-

A screenshot of a Python interpreter window. The window has a black border and a white background. On the left side, there is a vertical bar with a red prompt character '>>>'. To the right of the bar, the text 'The Simple Interest is: 12.0' is displayed in blue. The text is centered vertically within the window.

```
>>> The Simple Interest is: 12.0
```

### 5.WAP TO PROMPT A USER TO ENTER A DAY OF THE WEEK

```
day=int(input("Enter a number from 1 to 7: "))
```

```
if day==1:
```

```
    print("is Sunday",day)
```

```
elif day==2:
```

```
    print("is Monday",day)
```

```
elif day==3:
```

```
    print("is Tuesday",day)
```

```
elif day==4:
```

```
    print("is Wednesday",day)
```

```
elif day==5:
```

```
    print("is Thursday",day)
```

```
elif day==6:
```

```
    print("is Friday",day)
```

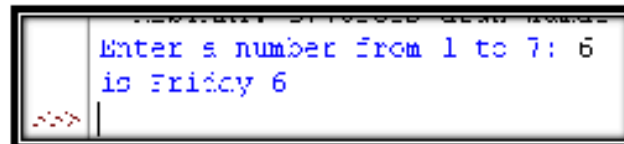
```
elif day==7:
```

```
    print("is Saturday",day)
```

```
else:
```

```
    print("Wrong input!")
```

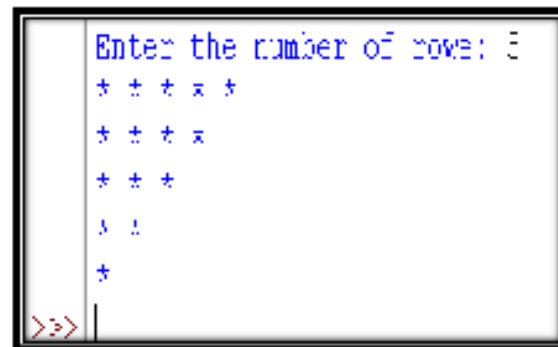
### OUTPUT:-

A screenshot of a Python IDE window. The title bar at the top reads "Python Shell - 27/10/2022 09:00:12". The main text area contains the program's output in blue text: "Enter a number from 1 to 7: 6" followed by "is Friday 6" on the next line. At the bottom left of the text area, there is a red prompt character ">" and a vertical cursor line.

6.WAP TO DISPLAY INVERTED HALF PYRAMID USING "\*"

```
rows = int(input("Enter the number of rows: "))  
for i in range(rows, 0, -1):  
    for j in range(i):  
        print("*", end=" ")  
    print()
```

**OUTPUT:-**

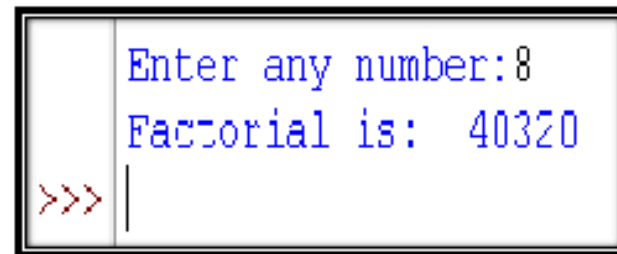


```
Enter the number of rows: 5  
* * * * *  
* * * *  
* * *  
* *  
*  
>>>
```

7.WAP USING WHILE LOOP TO PRINT FACTORIAL OF A NUMBER

```
n=int(input("Enter any number:"))  
f=1  
while n>=1:  
    f*=n  
    n-=1  
print("Factorial is: ",f)
```

**OUTPUT:-**



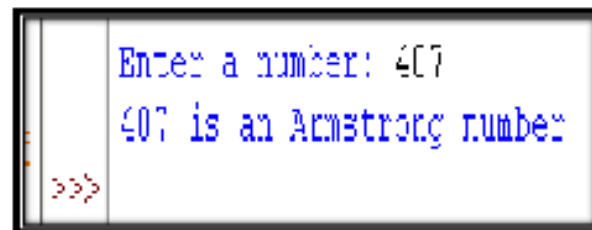
The screenshot shows a Python interactive shell (REPL) with a double border. The prompt is three red greater-than signs (>>>). The user has entered '8' in response to the program's prompt 'Enter any number:'. The program has then printed 'Factorial is: 40320'.

```
>>> Enter any number:8  
Factorial is: 40320  
>>> |
```

**8.WAP TO CHECK WHETHER THE NUMBER ENTERED IS  
ARMSTRONG OR NOT**

```
num = int(input("Enter a number: "))  
sum = 0  
temp = num  
while temp > 0:  
    digit = temp % 10  
    sum += digit ** 3  
    temp //= 10  
  
if num == sum:  
    print(num,"is an Armstrong number")  
else:  
    print(num,"is not an Armstrong number")
```

**OUTPUT:-**



```
Enter a number: 407  
407 is an Armstrong number  
>>>
```

### 9. WAP TO DEMONSTRATE THE USE OF LIST SLICING

```
List = [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
print("Original List:\n", List)
```

```
print("\nSliced Lists: ")
```

```
print(List[3:9:2])
```

```
print(List[:2])
```

```
print(List[:])
```

### OUTPUT:-

```
Original List:
[1, 2, 3, 4, 5, 6, 7, 8, 9]

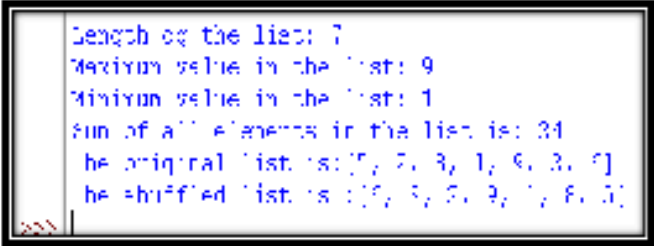
Sliced Lists:
[4, 6, 8]
[1, 3, 5, 7, 9]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
>>>
```

10.WAP TO DEMONSTRATE THE USE OF INBUILT FUNCTIONS

```
num=[5,2,8,1,9,3,6]
length=len(num)
print("Length og the list:",length)
maximum=max(num)
print("Maximun value in the list:",maximum)
minimum=min(num)
print("Minimum value in the list:",minimum)
total=sum(num)
print("Sum of all elements in the list is:",total)
import random
print("The original list is:" + str(num))
random.shuffle(num)
print("The shuffled list is :"+ str(num))
```

**OUTPUT:-**



```
Length og the list: 7
Maximun value in the list: 9
Minimum value in the list: 1
Sum of all elements in the list is: 34
The original list is:[5, 2, 3, 1, 9, 3, 6]
The shuffled list is :[9, 3, 2, 9, 1, 8, 3]
```

11.WAP TO PRINT FIBONACCI SERIES UPTO 8(FIRST\_NUM=0)

```
n = 8
```

```
a = 0
```

```
b = 1
```

```
sum=0
```

```
count=1
```

```
print("Fibonacci Series:",end="")
```

```
while(count<=n):
```

```
    print(sum,end=" ")
```

```
    count+=1
```

```
    a=b
```

```
    b=sum
```

```
    sum=a+b
```

**OUTPUT:-**

A screenshot of a Python terminal window. The output of the program is displayed in blue text: "Fibonacci Series:0 1 1 2 3 5 8 13". Below the output, the prompt ">>>" is visible in red text, followed by a vertical line indicating the cursor position.

```
Fibonacci Series:0 1 1 2 3 5 8 13
>>>
```



12.WAP TO COUNT WORDS/LINES IN A FILE.

```
def count_words_lines(file_path):  
    """  
    Counts the number of lines and words in the given file.  
    :param file_path: Path to the file  
    :return: tuple (line_count, word_count)  
    """  
  
    try:  
        with open(file_path, 'r', encoding='utf-8') as file:  
            line_count = 0  
            word_count = 0  
  
            for line in file:  
                line_count += 1  
  
                words = line.split()  
                word_count += len(words)  
  
            return line_count, word_count  
    except FileNotFoundError:  
        print(f"Error: File '{file_path}' not found.")  
        return None, None
```

```
except PermissionError:
    print(f"Error: Permission denied to read '{file_path}'.")
    return None, None
except Exception as e:
    print(f"An unexpected error occurred: {e}")
    return None, None

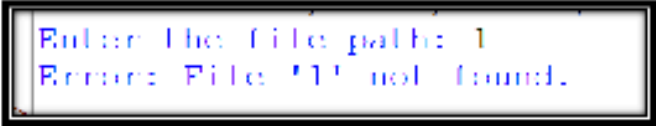
if __name__ == "__main__":

    file_path = input("Enter the file path: ").Strip()

    lines, words = count_words_lines(file_path)

    if lines is not None:
        print(f"Total Lines: {lines}")
        print(f"Total Words: {words}")
```

### **output:-**



```
Enter the file path: 1
Error: File '1' not found.
```

13. WAP TO CREATE A CLASS STUDENT AND PRINT THE  
DETAILS OF A STUDENT.

```
class Student:
    def getStudentDetails(self):
        self.rollno=input("Enter Roll Number : ")
        self.name = input("Enter Name : ")
        self.physics =int(input("Enter Physics Marks : "))
        self.chemistry = int(input("Enter Chemistry Marks : "))
        self.maths = int(input("Enter Math Marks : "))

    def printResult(self):
        self.percentage = (int)((self.physics + self.chemistry +
self.maths) / 300 * 100 );
        print(self.rollno,self.name, self.percentage)

S1=Student()
S1.getStudentDetails()

print("Result : ")
S1.printResult()
```

```
S1.physics += 9
```

```
print("result after adding grace marks...")
```

```
S1.printResult()
```

### **OUTPUT:-**

```
Enter Roll Number : 1  
Enter Name : diksha  
Enter Physics Marks : 10  
Enter Chemistry Marks : 9  
Enter Math Marks : 14  
Result :  
1 diksha 11  
result after adding grace marks...  
1 diksha 14
```

14. WAP TO SHOW HOW ENCAPSULATION WORKS USING PRIVATE VARIABLES.

```
class Employee:
    def __init__(self, name, salary):
        self.name = name
        Self_Salary = None
        self.set_salary(salary)
    def get_salary(self):
        return Self_Salary
    def set_salary(self, salary):
        if not isinstance(salary, (int, float)):
            raise ValueError("Salary must be a number.")
        if salary < 0:
            raise ValueError("Salary cannot be negative.")
        Self.salary = salary
    def display_info(self):
        print(f"Name: {self.name}")
        print(f"Salary: ₹{self_salary}")
try:
    emp = Employee("John Doe", 50000)
    print("Public attribute (name):", emp.name)
```

```
    print("Private attribute (salary) via getter:", emp.  
Get_salary())
```

```
emp. Set_salary(60000)  
    print("Updated salary via getter:", emp.Get_salary())  
    emp. Display_info()  
except Exception as e:  
    print("Error:", e)
```

### **Output:-**

```
Public attribute (name): John Doe  
Private attribute (salary) via getter: 50000  
Updated salary via getter: 60000  
Name: John Doe  
Salary: ₹60000
```

15.WAP TO IMPLEMENT ABSTRACTION USING  
ABC(ABSTRACT BASE CASE).

```
from ABC import ABC, abstractmethod

class Shape(ABC):
    """
    Abstract base class representing a geometric shape.
    All shapes must implement the area() and perimeter()
    methods.
    """
    @abstractmethod
    def area(self):
        """Calculate and return the area of the shape."""
        pass
    @abstractmethod
    def perimeter(self):
        """Calculate and return the perimeter of the shape."""
        pass

class Circle(Shape):
    def __init__(self, radius):
        if radius <= 0:
```

```
        raise ValueError("Radius must be positive.")

    self.Radius = radius

def area(self):

    from math import pi

    return pi * self.radius ** 2

def perimeter(self):

    from math import pi

    return 2 * pi * self.radius

class Rectangle(Shape):

    def __init__(self, length, width):

        if length <= 0 or width <= 0:

            raise ValueError("Length and width must be positive.")

        self.length = length

        self.Width= width

    def area(self):

        return self.length * self.width

    def perimeter(self):

        return 2 * (self.length + self.width)

if __name__ == "__main__":

    try:
```



```
print("Choose shape: 1. Circle 2. Rectangle")
choice = input("Enter choice (1/2): ").strip()

if choice == "1":
    radius = float(input("Enter radius: "))
    shape = Circle(radius)
elif choice == "2":
    length = float(input("Enter length: "))
    width = float(input("Enter width: "))
    shape = Rectangle(length, width)
else:
    raise ValueError("Invalid choice.")

print(f"Area: {shape.Area():.2f}")
print(f"Perimeter: {shape.perimeter():.2f}")

except ValueError as e:
    print(f"Error: {e}")
except Exception as e:
    print(f"Unexpected error: {e}")
```

**OUTPUT:-**

```
Choose shape: 1. Circle 2. Rectangle
Enter choice (1/2): 1
Enter radius: 10
Area: 314.16
Perimeter: 62.83
```