BRAINWARE UNIVERSITY

398, Ramkrishnapur Road, Barasat, North 24 Parganas, Kolkata - 700 125

Laboratory Report / Workshop / Assignment Submission

Session - 2024 - 25

Lab Report No: - 6

Topic Title: - Perform Exponentiation Using a Base and an Exponent.

Name of the Department: - Cyber Science & Technology

Programme Name: - B.Sc (H) ANCS 2024

Semester / Year: - Semester-2, 2024-2025

Course Code: - VAC09009

Course Name: - Python Programming Lab

Name of the Student: - Rupankar Chakraborty

Roll No: -

Registration No: -

Student Code: -BWU/BNC/24/ 157

[illegible]
19.03.25

~~Python~~

In python we use "def" to define a function.

In python while using the functions ~~are~~ return statement is used to end the execution of the function. and it returnes the value of the expression following the return keyword to the caller. A return statement is overall used to invoke a function so that the past statements can be executed.

• Variables in python:

In python there are two types of variables which are commonly used.

1. Global Variable:

In python global variables are those which are not defined inside any perticular function and it has a global scope where as the global variables can be accesseble throughout ~~the~~ the programme and inside every mentioned function.

2. Local Variable:

Local variables in python are those which are initialized inside the function which belongs only to that perticular function. It can't be accessed outside the function.

3. Recursion in function:

In python a recursive function is defined like any other function but it includes ~~to~~ a call to itself. It is often used to break complex problems ~~into~~ into simple ones

~~def of functions~~

```
def factorial (n):
    if n == 1 or n ==0:
        return 1
    else:
        return n*factorial (n-1)
print (factorial (5))
```

explaination: The factorial of the number n denoted as n factorial (n!) is the product of all positive integers less than or equals to n. The recurcive approach involves the function calling itself with the decremented value of n until it reaches the base case (n==1)

i. Base Case:

Base Case is the condition under which the recursion function stops. It is important to prevent infinite loops and to ensure that each reecurcive call reduces the problem in some manner In this factorial example Base Case (n==1)

ii. Recurcive Case:

Recurcive Case is the part of function that includes the call to itself. It is ~~even~~ eventually reches to the base case. In the factorial example the Recurcive Case is return n* factorial (n-1).

Types of Recursion in Python:

In python while using reecurcive function it can be broadly classified into two types.

1. Tail Recursion:

Tail Recursion occurs when the recurcive call function is the last operation to be executed, with no additional ~~work~~ task or calculations following the recurcive call in many programming languages tail recursion can be optimized by the compiler into itterative loops to improve performance and prevent stack overflow.

2. Non Tail Recursion:

Non tail recursion occurs when there are operations or calculations that blindly follows the recurcive call this type prevents the compiler or interpreter from optimizing the recursion into an itteration.

Question:

Write a recurcive python function power(base, exponent) that calculates the exponentiation of a given base and exponent. The function should handle:

i. A base case when the exponent is zero.

ii. Positive exponents using recursion.

iii. Negetive " by converting them to teir reciproker.

The user will enter a base and an exponent, call the function and display the probable output.

```
def power (base, component):
    if comonent  == 0: # base case to handle 0 and 1
        return 1
    elif comopent <0: # negetive case
        return 1 /power (base, -component)
    else: # recurcive case
        return base * power (base, component -1)
base = float (input("Enter the value"))
ex~~com~~ponent = int( input("Enter the exponent"))
 # calculation result
 result = power (base, component)
 print(f"{base} ^ {ex~~com~~ponent} = {result}")
```

Output: Enter the value 3
        Enter the exponent 2
        3.0 ^2 = 9.0

[illegible]
09.03.25