

Papers Dock

PYTHON

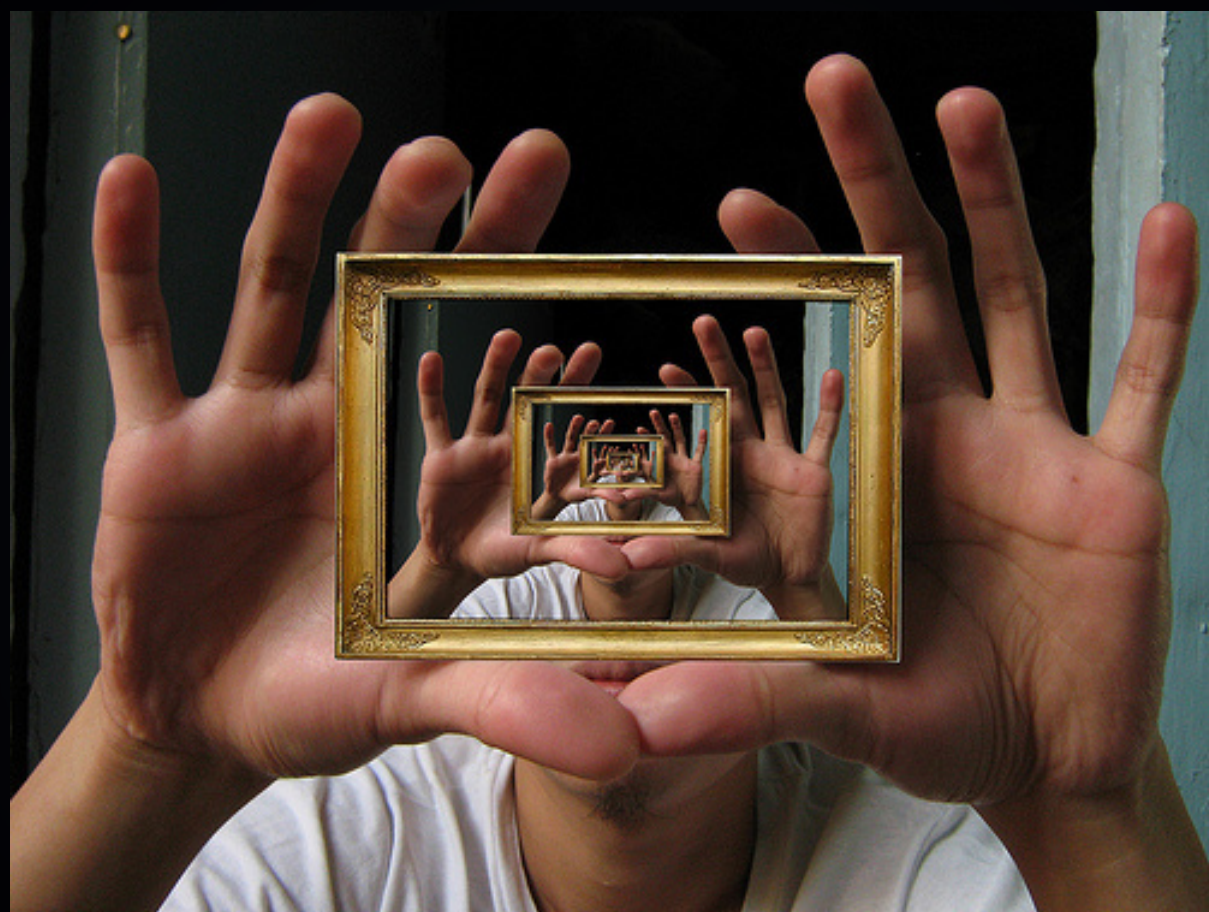
9618

RECURSION

What Is Meant By Recursion ?

When a function / procedure calls itself





```
def recursion(x):  
    y = x * 2  
    print(y)  
    recursion(y)  
  
recursion(4)
```

This is a recursive function in Python that takes an argument x, doubles it, and then calls itself with the result as the new argument. The function will keep calling itself recursively, doubling the argument each time, and printing out the result, until it encounters an error due to reaching Python's maximum recursion depth.

The base case and the general case are two important concepts in recursion that are used to control the flow of a recursive function.

The base case is the condition that determines when the recursion should stop. It is the condition that prevents the function from calling itself again and again, and allows the function to return a value. In other words, the base case is the stopping point for the recursion. It is the case where the problem is simple enough that the function does not need to call itself anymore.

The general case, on the other hand, is the opposite of the base case. It is the condition that determines when the recursion should continue. In the general case, the function is not yet at the stopping point and needs to call itself again with a smaller or simpler version of the problem. This continues until the base case is reached.

```
def recursion(x):  
    y = x * 2  
    print(y)  
    if y < 10000:  
        recursion(y)  
  
recursion(4)
```

the base case is when the value of y exceeds or equals 10000, which means that the recursion stops and the function does not call itself anymore.

```
def recursion(x):  
    y = x * 2  
    print(y)  
    if y < 10000:  
        recursion(y)  
  
recursion(4)
```

The base case is when $y \geq 10000$

The general case is when $y < 10000$

What Is Meant By Recursive Algorithm ?

Any algorithm that have a base case, work towards a base case, have a general case and calls itself.

- (b) The following recursive procedure outputs every even number from the positive parameter value down to and including 2.

The procedure checks if the integer parameter is an even or an odd number. If the number is odd, the procedure converts it to an even number by subtracting 1 from it.

The function MOD(ThisNum : INTEGER, ThisDiv : INTEGER) returns the remainder value when ThisNum is divided by ThisDiv.

Complete the **pseudocode** for the recursive procedure.

```
PROCEDURE Count (BYVALUE ..... : INTEGER)
```

```
    IF ..... (Number, 2) <> 0
```

```
        THEN
```

```
            Number ← Number - 1
```

```
        ENDIF
```

```
        OUTPUT .....
```

```
        IF Number > 0
```

```
            THEN
```

```
                ..... (..... - 1)
```

```
            ENDIF
```

```
ENDPROCEDURE
```

- (b) The following recursive procedure outputs every even number from the positive parameter value down to and including 2.

The procedure checks if the integer parameter is an even or an odd number. If the number is odd, the procedure converts it to an even number by subtracting 1 from it.

The function MOD(ThisNum : INTEGER, ThisDiv : INTEGER) returns the remainder value when ThisNum is divided by ThisDiv.

Complete the **pseudocode** for the recursive procedure.

```
PROCEDURE Count (BYVALUE Number : INTEGER)

    IF MOD (Number, 2) <> 0

        THEN

            Number ← Number - 1

        ENDIF

    OUTPUT Number

    IF Number > 0 62

        THEN

            CALL Count (Number - 1)

        ENDIF

    ENDPROCEDURE
```

