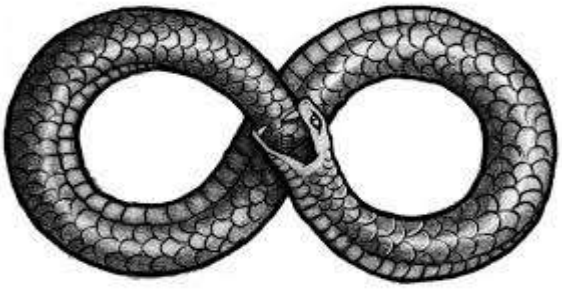
Think About the Structure of Images

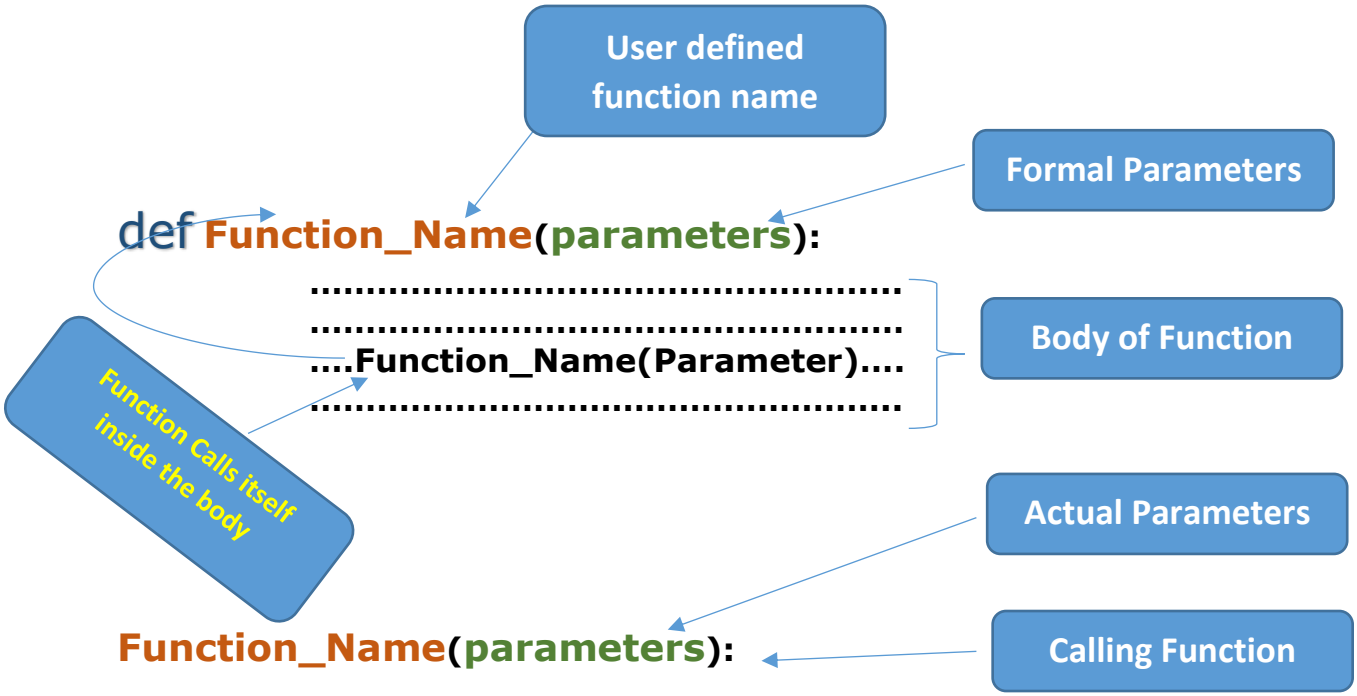


Conclusion:

- **One type of structure in images are repeatedly used.**
- **Repetition is for the limited numbers of time that mean it is not for infinite times.**
- **Every image has its specific start point and end point.**
- **These images with this type of structure are the examples of <u>recursion</u>.**
- **Recursion is similar to Iteration (looping) in Python programming Language.**

Recursion in Python

Recursion is a way of programming, in which a function calls itself directly or indirectly. In other words, a function calls itself inside the body of itself for the finite time.



Direct and Indirect Recursion

```
def abc():
    abc()
```

Function abc() calling itself in its body. So it is direct recursive function.

```
def abc():
    pqr()

def pqr():
    abc()
```

Function abc() calling to another function-pqr() and pqr() calls to abc(). So it is indirect recursive function.

Basic Concept of recursion

Recursion is a technique for solving a large problem by applying the same procedure repeatedly to reduce it to successively smaller problem. Recursive function essentially required following two parts for its implementation

1. **Base case (one or more) in function**

- **Base case is a statement in recursive function, whose result is known or result computed without any recursion statement.**
- **When base case "not defined" or "not reached" in recursive function then infinite recursion will occur and there will be abnormal termination of program. That mean base case must be execute in the function.**

2. **Recursive step in function**

- **Recursive step is a statement in function that calls itself with some parameters to repeat the same procedure.**
- **Recursive step should not call endless otherwise function will caught by infinite recursion.**

Example of recursive function

Program-1
Write function to find sum of n-numbers and use recursion in function.

```
def sum_of_number(n):
    if (n==1):
        return 1
    else:
        return (n+sum_of_number(n-1))

last_num=5
total=sum_of_number(last_num)
print("Sum of numbers from 1 to ",last_num,"=",total)
```

Output:

```
Sum of numbers from 1 to 5 = 15
```

Explanation of program execution:

```
sum_of_number(5)                    # Initial function call
5+sum_of_number(4)                  # First recursion call
5+(4+sum_of_number(3))              # Second recursion call
5+(4+(3+sum_of_number(2)))          # Third recursion call
5+(4+(3+(2+sum_of_number(1))))      # Fourth recursion call
5+(4+(3+(2+1)))
5+(4+(3+3))
5+(4+6)
5+10
15
```

Recursion Vs Iteration (Loop)

- **Recursion and iteration are interchangeable in nature.**
- **When a loop repeats then same memory locations used for variables used in loop.**
- **Loop repeats same code each time when it repeats.**
- **In recursion, the fresh memory locations reserved for each recursive function call.**
- **In recursion, same code will not repeat for its recursive call.**
- **Recursion requires more resource in terms of RAM (memory space) and processor utilization compare to iteration.**
- **Recursion is slower process than Iteration due to extra memory manipulation.**
- **Sometimes recursion makes the code easier to understand than Iteration.**

Program-2

Write function to find factorial of given number by using

(1) Iteration

(2) Recursion

(1) By Iteration	(2) By Recursion

def factorial(num):
    f=1
    while(num>=1):
        f=f*num
        num=num-1
    return f
 	
fact=factorial(5)
print("Factorial=",fact)
 	
Output:
Factorial=120
	def factorial(num):
	    if (num==1):
	        return 1
	    else:
	        return num*factorial(num-1)
	
	fact=factorial(5)
	print("Factorial=",fact)
	
	Output:
	Factorial=120

Assignments

Q.1 Write function to find the sum of list elements by using recursion.

Q.2 Write function to find GCD (Greatest Common divisor) of 2 numbers.

Q.3 Write function to find the square of given numbers.