SNS College of Engineering
Department of Information Technology

Iteration and Recursion

Presented By
M.Malarmathi
AP/IT

Iteration

- Iterative approach is a repetition process until the condition fails
- for ,while loops are used
- Fibonacci series
  - Fibonacci sequence is defined to start at either 0 or 1, and the next number in the sequence is one
  - Each subsequent number in the sequence is simply the sum of the prior two
  - 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,...

  or

  - 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,...

Mathematical definition:

- F(n) = F(n-1) + F(n-2)

- Program using while loop**(finding fibonacci series)**

```
num=int(input("Enter a number"))
i=0
fib0=0
fib1=1
while (i<num):
        if i<=1:
            fibnext=i
        else:
            fibnext=fib0+fib1
            fib0=fib1
            fib1=fibnext
        i=i+1
        print(fibnext)
```

Program using for loop**(finding fibonacci series)**

```
def F_iter(n):
    if (n == 0):
        return 0
    elif (n == 1):
        return 1
    elif (n >1 ):
        fn1 = 0
        fn2 = 1
        print(fn1,fn2)
        for i in range(2, n):
            fn3 = fn1+fn2
            fn1 = fn2
            fn2 = fn3
            print(fn3)
```

Recursion

- The function calls itself until the condition is met
- recursion is like a selection structure, and which makes code smaller and clean
- function partially defined by itself

Recursion(finding fibonacci series)

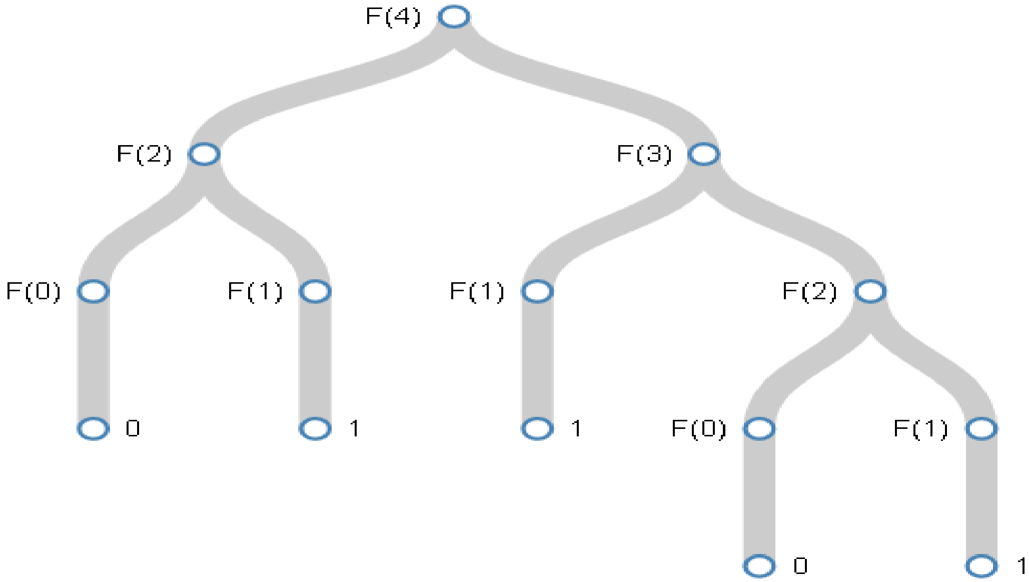
```
def GenerateFibonaci(x):
    if(x == 0):
        return 0
    elif(x == 1):
        return 1
    else:
        return GenerateFibonaci(x-1) + GenerateFibonaci(x-2)
x = int(input("Enter the term till which you want to generate fibonacci sequence: "))
for i in range(x):
    print(GenerateFibonaci(i))
```

when n > 1, the function calls itself

When we call F(4). F(4) will check x= 0 and x = 1 cases and it checks x> case where it calls the function twice with F(3) and F(2).

- F(3) and F(2) then each subsequently call the function again F(3) calls F(2) and F(1), and F(2) calls F(1) and F(0), as shown in the tree structure below

- The F(1) and F(0) cases are the final, terminating cases in the tree and return the value of 1 or 0, respectively.



Difference between Recursive algorithm and Iterative algorithm

Iteration algorithm :

- code may be longer but it is faster than recursive
- consumes less memory compared to recursive approach
- Uses for and while loop

- **Recursive algorithm**
  - Recursive algorithm uses a function that is partially defined by itself
  - Recursive algorithm uses selection structure
  - Infinite recursion occurs if the recursion step does not reduce the problem in a manner that converges on some condition.(base case)
  - Recursion terminates when a base case is recognized
  - Recursion is usually slower then iteration due to overhead of maintaining stack
  - Recursive algorithm uses more memory than iteration
  - Infinite recursion can crash the system
  - Recursion makes code smaller

Thank you