Recursion

Head & Tail Recursion :-

Print Numbers from 1 to 10.

Iterative Solution

```
def count()
    for i in (1....10)
        print i
    end
end
```

Recursive Solution

```
1. def count (n)
2.     if n>0
3.         count(n-1)
4.         print n
5.     end
6. end
```

Recursive call

* There is no base case
* We stop recursion when n becomes 0.

```
1  def count(n):
2      if n > 0:
3          count(n - 1)
4          print(n)
5
6  count(10)
```

* Recursive function needs a parameter which is updated in each recursive call.

Print 10 to 1

Iterative Solution :-

```
def count(n)
    for i in (10....1)
        print i
    end
end
```

Recursive Solution

```
1. def count(n)
2.     if n > 0
3.         print n
4.         count(n-1)
5.     end
6. end
```

Recursive call

* There is no base case

```
1  def count(n):
2      if n > 0:
3          print(n)
4          count(n - 1)
5
6  count(10)
```

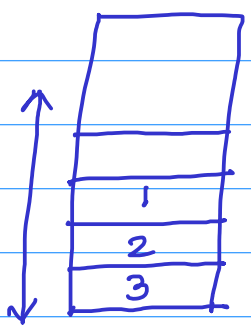
* Comparing the Recursive Solutions

```
1. def count(n)
2.     if n > 0
3.         count(n-1)
4.         print n
5.     end
6. end
```

Head Recursion

* Recursive call happens before some work is done.

* In head Recursion we need a stack frame to maintain State



* In head Recursion Stack will grow with the no. of Recursive calls

```
1. def count(n)
2.     if n > 0
3.         print n
4.         count(n-1)
5.     end
6. end
```

Tail Recursion

* Recursive call is the Last Action

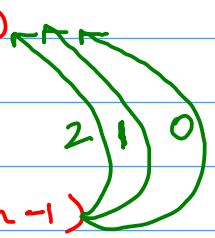
* In tail Recursion we don't need variables to maintain in stack because we have done the work we need to do with variable n.

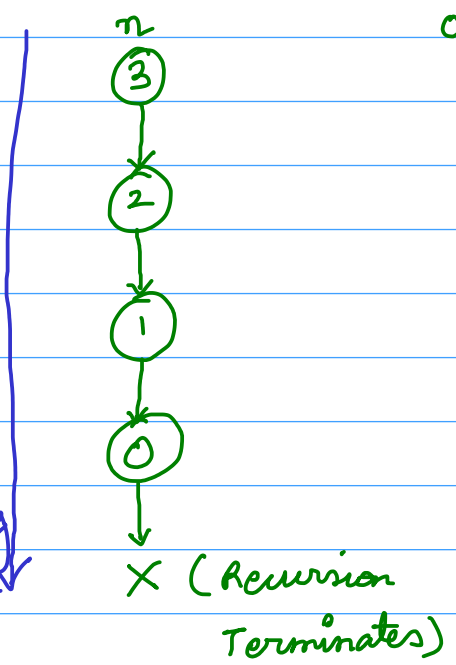
* In tail Recursion Stack will not grow.

* Compilers can optimize code by converting tail Recursion to Iterative Solution.

TAIL RECURSION CALL TRACE

```
1. def count (n)
2.     if n > 0
3.         print n
4.         count (n-1)
5.     end
6. end
```





Head Recursion Call Trace

```
1. def count (n)
2.     if n > 0
3.         count(n-1)
4.         print n
5.     end
6. end
```