1. **def factorial (n):**

Use Python visualizer to help you understanding the execution of the function

1. **if n ==0:**
2. **return 1**
3. **else:**
4. **return (n\*factorial(n-1))**
5. **print (factorial (4))**

Trace the execution of the above code; show how the stack will be used during the execution.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Line  **STACK**   |  | | --- | |  | |  | |  | |  | |  | | 2\*factorial(1) | | 3\*factorial(2) | | 4\*factorial(3) |   STACK | n | n ==0 | Note |
| **06** |  |  | Call function factorial () with parameter = 4 |
| 01 | 4 |  | This is the subroutine |
| 02 |  | FALSE | This is the stopping condition. (Base case) n does not equal 0 so it is false. |
| 05 |  |  | The function calls itself for (n = 3) |
| 01 | 3 |  |  |
| 02 |  | False |  |
| 05 |  |  | The function calls itself for (n=2) |
| 01 | 2 |  |  |
| 02 |  | False |  |
| 05 |  |  | The function calls itself for (n=1) |
| 01 | 1 |  |  |
| 02 |  | False |  |
| 05 |  |  | The function calls itself for (n=0) |
| 01 | 0 |  |  |
| 02 |  | True |  |
| 03 |  |  | Return 1 |

Describe how the stack is used during execution:

* A stack is the output of 4 factorial so it results to 24

2x1=2

3x2=6

4x6=24

Rewrite the above recursive function as iterative algorithm.