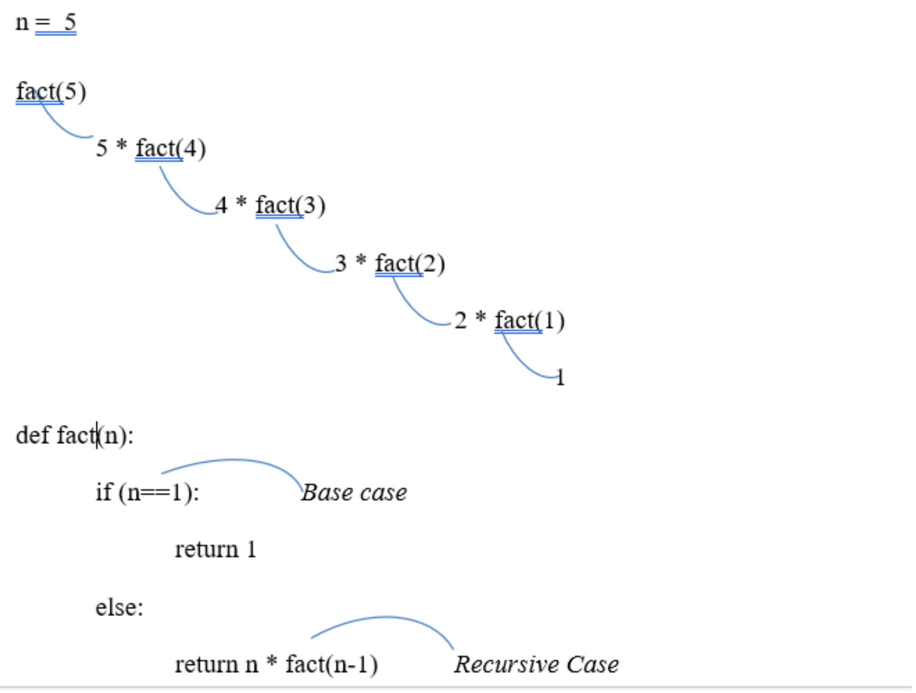
Recursion

- The term **Recursion** can be defined as the process of **defining something** in terms of **itself**.
- In simple words, it is a process in which a **function calls itself directly or indirectly**.
- Consider the following example, in which we can calculate sum of first 5 natural number:



```
n = 5

fact(5)
      5 * fact(4)
            4 * fact(3)
                  3 * fact(2)
                        2 * fact(1)
                              1

def fact(n):
      if (n==1):
            return 1
      else:
            return n * fact(n-1)
```

- **Advantages of using recursion**
  - A complicated function can be split down into smaller sub-problems utilizing recursion.
  - Sequence creation is simpler through recursion than utilizing any nested iteration.
  - Recursive functions render the code look simple and effective.
- **Disadvantages of using recursion**
  - A lot of memory and time is taken through recursive calls which makes it expensive for use.
  - Recursive functions are challenging to debug.
  - The reasoning behind recursion can sometimes be tough to think through.
- **Syntax:**
  - def func(): <--
  - |

- | (recursive call)
- |
- func() ----

- **Example factorial of a number using recursion:**
  - def factorial_number(n):
  - if n==1:
  - return n
  - else:
  - return n * factorial_number(n-1)
  - factorial_number(5)
- **Output:**
  - 120
- **Step to assume solution is recursive:**
  - fact(n)
  - Recursive case n * fact(n-1)
  - Base case n==1.