Recursive/Iterative

Definitions:

- Recursive
  - a function calls itself again and again till the base condition(stopping condition) is satisfied
  - common to functional programming
- Iterative
  - iterative is used to describe a situation in which a sequence of instructions can be executed multiple times
  - each time an iteration

Recursive Rules:

- each recursive call should be on a smaller instance of the same problem, that is, a smaller subproblem
- recursive calls must eventually reach a base case, which is solved without further recursion
  - Base case
  - Recursive Case
- Dynamic Programming and recursion
  - use when ever you compute a recursive input multiple times
  - memoization (caching previosly computed results, use result next time computation is needed)

Example problems:

- drawing fractals
- factorial
- towers of hanoi

Comparison:

- factorial
  - Iterative

```
def factorial(n):
    factorial = 1
    for i in range(2,n+1):
        factorial *= i
    return factorial
```

  - Recursive

```
def factorial(n):
    if (n < 2):
        return 1
    else:
        return n*factorial(n-1)
```

- Generally recursion more elegant, iteration better performance and debugability
- Iterative algorithm
  - more lines of code
  - faster
- Recursive algorithm
  - complex to implement
  - code will be elegant and easy to read
  - tracing is difficult
  - takes more time because of overheads like calling functions and registering stacks repeatedly
  - some complex problems can be solved easily and effectively in recursion