1. Lambda Function

Lambda functions in Python are small, anonymous functions defined using the `lambda` keyword. Unlike standard functions declared with `def`, lambda functions can have any number of arguments but only one expression. The expression is evaluated and returned when the function is called. Lambda functions are useful for short, simple operations that are used temporarily and don't require a formal function definition.

Simple lambda function to add two numbers

```
In [1]: add = lambda x, y: x + y
        print(add(5, 3))
```

```
8
```

Lambda function with filter() to get even numbers from a list

```
In [2]: nums = [1, 2, 3, 4, 5, 6, 7, 8, 9]
        even_nums = list(filter(lambda x: x % 2 == 0, nums))
        print(even_nums)
```

```
[2, 4, 6, 8]
```

Lambda function with sorted() to sort a list of strings by length

```
In [3]: names = ["Alice", "Eve", "Bob", "Chad", "Deborah"]
        sorted_names = sorted(names, key=lambda x: len(x))
        print(sorted_names)
```

```
['Eve', 'Bob', 'Chad', 'Alice', 'Deborah']
```

```
In [4]: is example, we're using a lambda function as the key for the sorted()
```

2. Recursive Function

A recursive function is a function that calls itself in order to break down a problem into smaller instances. It contains a base case to stop the recursion and prevent infinite loops. Recursive methods offer a clean solution for problems like factorials and Fibonacci sequences. However, excessive recursion can lead to a stack overflow error if not handled properly.

Calculate factorial using recursion

```
In [5]: def factorial(n):
            """Returns the factorial of a number using recursion."""
            if n == 1:
                return 1
            else:
                return n * factorial(n-1)
```

```
In [6]: print(factorial(5))
```

```
120
```

Fibonacci Sequence using recursion

```
In [7]: def fibonacci(n):
            """Return the nth value in the Fibonacci sequence."""
            if n <= 1:
                return n
            else:
                return fibonacci(n-1) + fibonacci(n-2)
```

```
In [8]: print(fibonacci(7))
```

```
13
```

```
In [ ]:
```