Python Functions & Modules – Practice Set

This set is based on the topics we've covered so far:

- **Defining Functions in Python**
- **Function Arguments & Return Values**
- **Lambda Functions in Python**
- **Recursion in Python**
- **Modules and Pip – Using External Libraries**
- **Variable Scope and Docstrings**

These exercises will help you get hands-on practice with writing functions, understanding arguments, using recursion, and working with external libraries.

---

1. Defining Functions

1. Write a function `greet()` that prints `"Hello, Python Learner!"` when called.
2. Write a function `square(num)` that returns the square of a given number. Test it with different numbers.

---

2. Function Arguments & Return Values

1. Write a function `full_name(first, last)` that takes first name and last name as parameters and returns a single string in the format `"First Last"`.

2. Write a function `calculate_area(length, width=10)` that returns the area of a rectangle. Test it by calling the function with:
    1. Both `length` and `width`
    2. Only `length` (use default width)

---

3. Lambda Functions

1. Write a lambda function that adds two numbers and test it.
2. Create a list `[1, 2, 3, 4, 5]` and use `map()` with a lambda function to get their squares.

---

4. Recursion in Python

1. Write a recursive function `factorial(n)` that returns the factorial of a number.
2. Write a recursive function `sum_of_digits(n)` that returns the sum of all digits of a given number.

---

5. Modules and Pip – Using External Libraries

1. Import the `math` module and use it to:
    1. Find the square root of 144
    2. Calculate `sin(90°)` (hint: use `math.radians()`)
2. Install and import the `requests` module (if available) and use it to fetch data from `"https://api.github.com"` .

---

6. Variable Scope and Docstrings

1. Write a function `increment()` that has a local variable `counter` initialized to `0` and increments it by `1` each time it is called. Observe whether the value persists across function calls.
2. Write a function `multiply(a, b)` that has a proper **docstring** explaining what it does. Then use `help(multiply)` to display the docstring.

---

7. Bonus Challenges

1. Write a recursive function `fibonacci(n)` that prints the first `n` Fibonacci numbers.
2. Write a function `safe_divide(a, b)` that returns the result of `a / b`, but returns `"Cannot divide by zero"` if `b` is `0`.
3. Create a small module `my_utils.py` with a function `is_even(n)` that returns `True` if `n` is even. Import and use it in another Python file.