Python Decorators

Generator

- Nested Functions
- Closures
- Decorators

Nested Function

- A functional programming pattern
- A function defined in another function
- Variables in the outer function are available and read-only
- Must be made nonlocal for modifying them

```
# outer function
def compute(first, second):
    results = {}
    # inner or nested function
    def add():
        # results, first and second are read-only in inner
        results["sum"] = first+second
    # inner or nested function
    def multiply():
        # results first and second are read-only in inner
        results["product"] = first * second
    # invoking inner functions
    add()
    multiply()
    return results
```

Closure

- **A functional programming pattern**
- **A function returned by another function**
- **Variables in the host function are available and read-only**
- **Must be made nonlocal for modifying them**

```
# host function
def computer(number):

    # inner or nested function
    def increment():
        # number read-only in inner
        nonlocal number
        number += 1
        return number

    #returning function
    return increment
```

Decorator

- **A functional programming pattern**
- **A function that decorates another function without modifying the structure**

```
# function to decorate a target
def loggable(target):
    # additional functionality
    def capability(*args, **kwargs):
        # pre-processing
        print("multplying ", args)
        result = target(*args)
        # post-processing
        print("product: ", result)
        return result
    # ready-to-use decorator
    return capability
```

- **Usage**

```
log_and_multiply = loggable(multiply)
print(log_and_multiply(5, 6))
```

Thanks