Python Functions & Modules — Lesson Notes

Function Basics

```python
def greet(name, greeting="Hello"):
    """Return a personalized greeting."""
    return f"{greeting}, {name}!"

# Keyword arguments
greet(name="Alice", greeting="Hi")

# *args and **kwargs
def log(*args, **kwargs):
    for arg in args:
        print(arg)
    for key, val in kwargs.items():
        print(f"{key}={val}")
```

Decorators

```python
import time

def timer(func):
    def wrapper(*args, **kwargs):
        start = time.time()
        result = func(*args, **kwargs)
        print(f"{func.__name__} took {time.time() - start:.2f}s")
        return result
    return wrapper

@timer
def slow_function():
    time.sleep(1)
```

Modules & Packages

```
my_project/
├── main.py
├── utils/
```

```
│   ├── __init__.py
│   ├── math_helpers.py
│   └── string_helpers.py
```

```python
# main.py
from utils.math_helpers import calculate_area
from utils import string_helpers
```

Generators

```python
def fibonacci(n):
    a, b = 0, 1
    for _ in range(n):
        yield a
        a, b = b, a + b

for num in fibonacci(10):
    print(num)
```

Key Takeaways

1. Use default parameters and keyword arguments for flexible functions.
2. Decorators add behavior to functions without modifying them.
3. Organize code into modules and packages for maintainability.
4. Generators are memory-efficient for large sequences.
