

Class

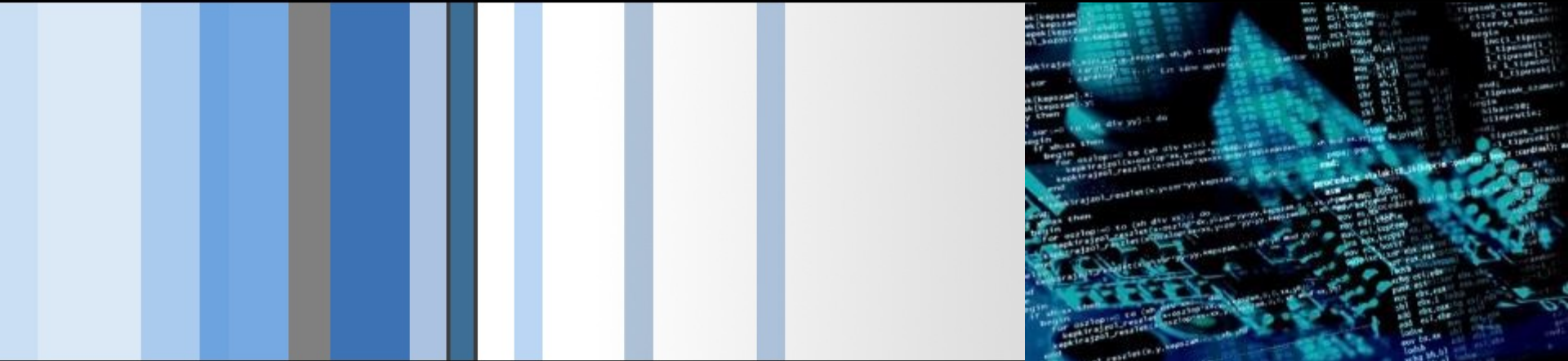
9

Function design and decomposition

CSC 141 · Class 9

Thursday, September 24, 2026

Dr. Si Chen



Today's learning goals

- Tell a void function apart from a fruitful function.
- Compose functions by passing one's return value into another.
- Decompose one long program into several small functions.
- Test each function separately before combining them.

Warm-up: predict before running

```
def double(n):  
    return n * 2  
  
def shout(text):  
    print(text.upper() + "!")  
  
result = double(5)  
shout("ready")
```

- What does result hold?
- What prints, and from which function?

Two kinds of functions

- A void function performs an action, such as printing, and returns nothing.
- A fruitful function computes a value and hands it back with return.
- `greet_visitor` from Class 8 was void.
- `add_tax` and `celsius_to_fahrenheit` were fruitful.

Reading the call site

```
log_status("Ready")  
energy = capped_energy(150)
```

- A bare call on its own line usually means a void function.
- An assignment usually means a fruitful function whose result you plan to use.

A common mix-up

```
def log_status(label):  
    print(label + " status logged.")  
  
result = log_status("Volt Station")  
print(result + 1)
```

- log_status never returns a value.
- result is None.
- None + 1 raises a TypeError.

Composing functions

```
def build_greeting(name):  
    return "Hello, " + name + "  
  
def shout(text):  
    return text.upper()  
  
final = shout(build_greeting("Ash"))  
print(final)
```

- The inner call runs first.
- The inner result, "Hello, Ash!", becomes shout's argument.

Decomposition: one long program

```
subtotal = 42.00
tax = subtotal * 0.06
total = subtotal + tax
print(f"Subtotal: ${subtotal:.2f}")
print(f"Tax: ${tax:.2f}")
print(f"Total: ${total:.2f}")
```

- This works, but every step is locked into one block.
- None of these steps can be reused or tested alone.

The same program, decomposed

```
def tax_amount(subtotal):  
    return subtotal * 0.06  
  
def total_with_tax(subtotal):  
    return subtotal + tax_amount(subtotal)  
  
subtotal = 42.00  
print(f"Subtotal: ${subtotal:.2f}")  
print(f"Tax: ${tax_amount(subtotal):.2f}")  
print(f"Total: ${total_with_tax(subtotal):.2f}")
```

- `total_with_tax` calls `tax_amount` itself.
- Each function can be tested with one line, on its own.

Why decompose?

- Each function has one clear job.
- A mistake in `tax_amount` is easy to find and fix in one place.
- You can test `tax_amount(100)` alone before trusting the whole receipt.
- Longer programs become a set of named, checkable pieces.

Guided build: a grade calculator

- Write two functions.
- `average_score(a, b, c)` returns the mean of three test scores.
- `letter_grade(avg)` returns "A" for 90+, "B" for 80–89, "C" for 70–79, otherwise "F".
- **Combine them:** print the letter grade for three scores you choose.

Grade calculator test cases

scores	average	letter
95, 92, 88	91.67	A
82, 79, 85	82.0	B
70, 65, 60	65.0	F
90, 90, 90	90.0	A

Grade calculator solution

```
def average_score(a, b, c):  
    return (a + b + c) / 3  
  
def letter_grade(avg):  
    if avg >= 90:  
        return "A"  
    elif avg >= 80:  
        return "B"  
    elif avg >= 70:  
        return "C"  
    else:  
        return "F"  
  
avg = average_score(95, 92, 88)  
print(letter_grade(avg))
```

- letter_grade only needs the average, not the three original scores.
- Each function is independently testable.

Practice: a shipping calculator

- Write `weight_fee(pounds)`: \$0.50 per pound.
- Write `rush_fee(is_rush)`: \$8.00 if True, otherwise \$0.00.
- Write a third line that combines both fees into one total and prints it.
- Test 4 pounds not rushed, and 4 pounds rushed.

Explain your decomposition

- **With a partner:**
- Name each function and what it returns.
- Point to the line where you compose two functions together.
- Identify which of your functions is void and which is fruitful, if either is void.
- Run one test case together.

Exit ticket

- What distinguishes a void function from a fruitful function?
- Why did `letter_grade` not need the three original test scores as parameters?
- What happens if you print the result of calling a void function?

Before next class

- Save and rerun your shipping calculator once more without looking at your notes.
- Programming Assignment 2 is due Tuesday, September 29.
- Bring one function that surprised you this week.
- **Next class:** function design, scope, and systematic testing continue.