

Class

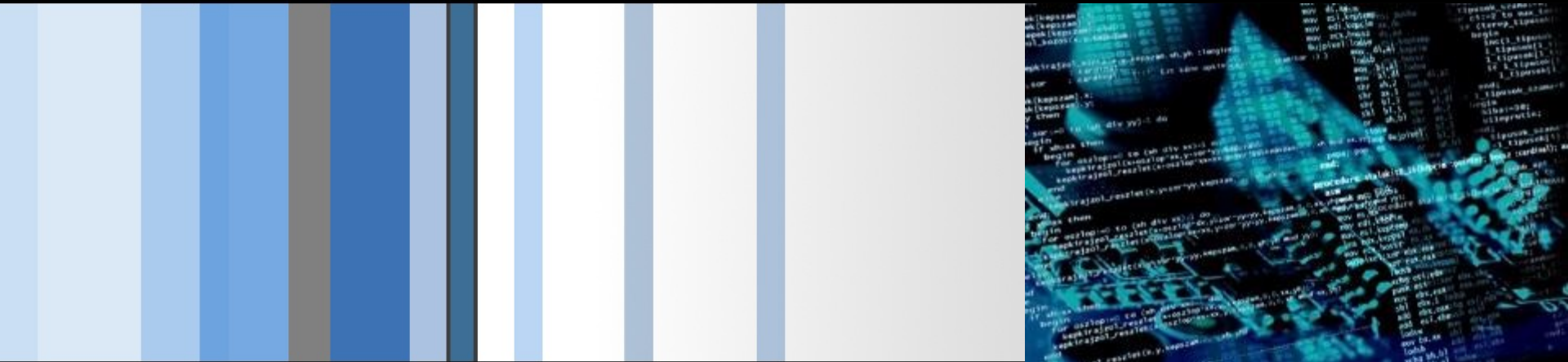
8

Functions and return values

CSC 141 · Class 8

Tuesday, September 22, 2026

Dr. Si Chen



Quiz 2

- 11:00–11:15 AM
- Covers Classes 6–7: nested decisions, testing, and debugging.
- Closed notes. Work independently.
- **Online:** wcu.ninja/csc141/quiz2

Today's learning goals

- Define a function with `def` and call it by name.
- Pass information into a function using parameters.
- Return a value from a function with `return`.
- Use a function's returned value in an expression.

Why write a function?

- **Community center rule:** three different posters each print a similar welcome message.
- Copying the same lines three times risks three different mistakes.
- A function writes the rule once; every poster calls it.

Defining a function

```
def greet_visitor():  
    print("Welcome to the center!")
```

- def starts a function definition.
- The indented block is the function's body.
- Defining a function does not run it yet.

Calling a function

```
def greet_visitor():  
    print("Welcome to the center!")  
  
greet_visitor()  
greet_visitor()
```

- Each call runs the body once.
- Parentheses are required even with nothing inside them.

Trace: two calls

```
greet_visitor()  
# Python jumps into the function body.  
# Output:  
Welcome to the center!  
  
greet_visitor()  
# Python jumps in again, from the top.  
# Output:  
Welcome to the center!
```

- Each call is independent.
- The function does not remember anything between calls yet.

Parameters customize a call

```
def greet_visitor(name):  
    print("Welcome, " + name + "!!")  
  
greet_visitor("Ash")  
greet_visitor("Misty")
```

- **name is a parameter:** a placeholder filled in by each call.
- "Ash" and "Misty" are arguments: the actual values supplied.

Trace: two arguments, one function

```
name = "Ash"      → Welcome, Ash!  
name = "Misty"    → Welcome, Misty!  
name = ""         → Welcome, !
```

return sends a value back

```
def add_tax(price):  
    return price * 1.06  
  
total = add_tax(50)  
print(total)
```

- return hands a value back to the caller.
- The function does not print anything itself.
- total now holds 53.0.

Using a returned value

```
def add_tax(price):  
    return price * 1.06  
  
print("Subtotal:", 50)  
print("With tax:", add_tax(50))  
print("Two orders:", add_tax(50) + add_tax(20))
```

- A returned value behaves like any other value.
- It can be printed, stored, or combined in a larger expression.

A common mistake: no return

```
def add_tax(price):  
    answer = price * 1.06  
    print(answer)  
  
total = add_tax(50)  
print(total + 1)
```

- `add_tax` prints 53.0, but returns nothing.
- `total` is `None`.
- `None + 1` raises a `TypeError`.

Guided build: a temperature converter

- Write `celsius_to_fahrenheit(c)`.
- **Formula:** $F = C \times 9/5 + 32$.
- Return the result; do not print inside the function.
- Call it and print the result yourself.

Converter plan and test cases

celsius	expected fahrenheit
0	32.0
100	212.0
-40	-40.0

Converter solution after your attempt

```
def celsius_to_fahrenheit(c):  
    return c * 9 / 5 + 32  
  
print(celsius_to_fahrenheit(0))  
print(celsius_to_fahrenheit(100))  
print(celsius_to_fahrenheit(-40))
```

- Reveal only after students try.
- Run all three tests before moving on.

Independent practice: a tip calculator

- Write `calculate_tip(bill, percent)`.
- Return $\text{bill} \times \text{percent} / 100$.
- Call it for a \$40 bill at 15%, 18%, and 20%.
- Print each tip rounded to two decimals.

Explain your function

- **With a partner:**
- Name your function's parameters.
- State what it returns.
- Show one call and its result.
- Explain why you used return instead of print inside the function.

Exit ticket

- What is the difference between a parameter and an argument?
- What does a function return if it has no return statement?
- Why can a returned value be used inside another expression?

Before Thursday

- Run your tip calculator again from a saved file, not from memory.
- Bring one function you are unsure about.
- Programming Assignment 2 is due Tuesday, September 29.
- **Next class:** decomposing a program into several functions.