# Higher-Order Functions

# Office Hours: You Should Go!

**You are not alone!**

https://cs61a.org/office-hours/

# Announcements

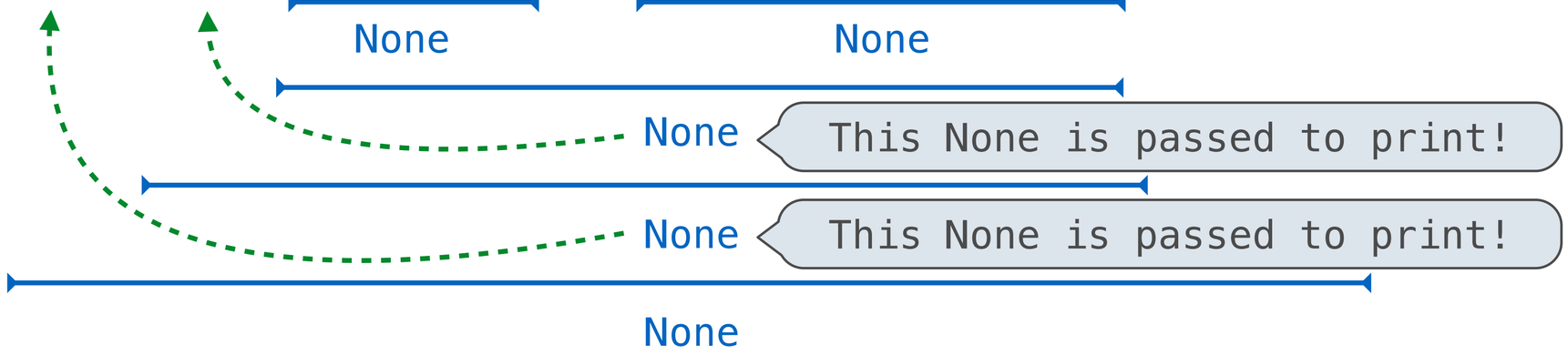
# Print and None Review

# Fall 2022 CS 61A Midterm 1, Question 1c

What does the long expression print?

```
s = "Knock"

print(print(print(s, s) or print("Who's There?")), "Who?")
```




```
Knock Knock
Who's There?
None
None Who?
```

False values in Python: False, 0, '', None *(more to come)*

To evaluate the expression **<left> or <right>**:

1. Evaluate the subexpression **<left>**.
2. If the result is a true value **v**, then the expression evaluates to **v**.
3. Otherwise, the expression evaluates to the value of the subexpression **<right>**.

# Designing Functions

# Describing Functions

A function's *domain* is the set of all inputs it might possibly take as arguments.

A function's *range* is the set of output values it might possibly return.

A pure function's *behavior* is the relationship it creates between input and output.

```
def square(x):
    """Return X * X."""
```

*x is a number*

*square returns a non-negative real number*

*square returns the square of x*

# A Guide to Designing Function

Give each function exactly one job, but make it apply to many related situations

```
>>> round(1.23)
1
```

```
>>> round(1.23, 1)
1.2
```

```
>>> round(1.23, 0)
1
```

```
>>> round(1.23, 5)
1.23
```

Don’t repeat yourself (DRY):  Implement a process just once, but execute it many times

(Demo)

# Higher-Order Functions

1. Functions as Arguments
2. Functions as Return Values

# Functions as Arguments

(Demo)

# Summation Example

```
def cube(k):
    return pow(k, 3)

def summation(n, term):
    """Sum the first n terms of a sequence.

    >>> summation(5, cube)
    225
    """
    total, k = 0, 1
    while k <= n:
        total, k = total + term(k), k + 1
    return total
```

Function of a single argument (*not called "term"*)

A formal parameter that will be bound to a function

The cube function is passed as an argument value

0 + 1 + 8 + 27 + 64 + 125

The function bound to term gets called here

# Program Design

Modularity

Abstraction

Separation of Concerns

# Twenty-One Rules

Two players alternate turns, on which they can add 1, 2, or 3 to the current total

The total starts at 0

The game end whenever the total is 21 or more

The last player to add to the total loses

(Demo)

Break: 5 minutes

# Functions as Return Values

(Demo)

# Locally Defined Functions

Functions defined within other function bodies are bound to names in a local frame

A function that returns a function

```
def make_adder(n):
    """Return a function that takes one argument k and returns k + n.

    >>> add_three = make_adder(3)
    >>> add_three(4)
    7
    """
    def adder(k):
        return k + n
    return adder
```

The name add_three is bound to a function

A def statement within another def statement

Can refer to names in the enclosing function

# Twenty-One Strategies

(Demo)