Conditionals & Functions

CSE 160 Summer 2026

Questions?

Sli.do #cse160

Announcements

- **Homework Assignment 1** due Monday, July 6th at 11:59 PM
- **Programming Practice 1** due Monday, July 6th at 11:59 PM
- **Written Check-In 2** due Monday, July 6th at 11:59 PM
- **No class on Friday, July 3rd!**

Today's Roadmap

1. A Loop Problem
2. Conditionals
3. Conditionals and Loops
4. Intro to Functions

Jupyter Hub

A Loop Problem

Write a loop to print out the **even** numbers between 0 and 10 (inclusive). Your program output should look like this:

0

2

4

6

8

10

Bonus problem

Think about how you would print out all the even numbers in a list of random, unordered numbers.

For example, how would you find all the even numbers in this list:

[9, 16, -100, -54, 48, 10, 32]

Python Building Blocks so Far

- Print Statements
- Expressions
- Variables and Assignment Statements
- Types
  - int, float, string, bool
- Loops
- A loop nested inside of a loop!
- TODAY: conditionals (aka "if statements")

Conditional Statements

- A **conditional statement** allows programmers to control which code block is executed based on whether a condition is true.
- Follows decision-based logic we use in our daily lives
  - **If** it's raining, **then** bring an umbrella
  - **If** the coffee pot is empty, **then** brew some coffee
  - **If** I am hungry, **then** cook dinner

```
If [condition is True], then [do some action]
```

Boolean (True/False)

Code to run

Boolean Expressions evaluate to `True` or `False`

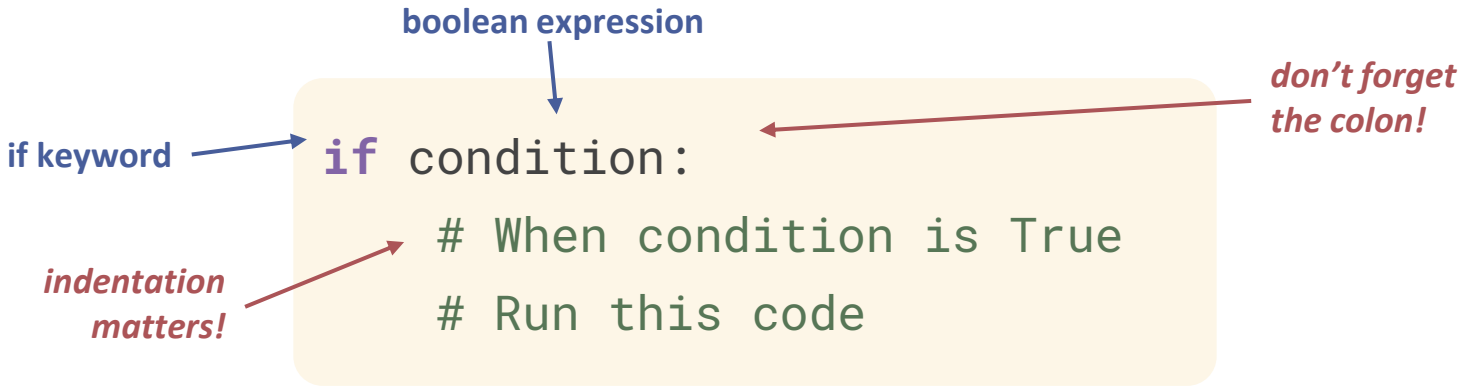
Some example boolean expressions:

```
22 > 4
22 < 4
22 == 4
22 >= 22
(22 > 4) and (5 < 6)
(22 > 4) and (5 > 6)
not (5 <= 6)
not True
```

We can print out the value of an expression: `print((22 > 4) and (5 < 6))`

We can save the value of an expression in a variable: `x = 22 < 4`

If statement “Anatomy”

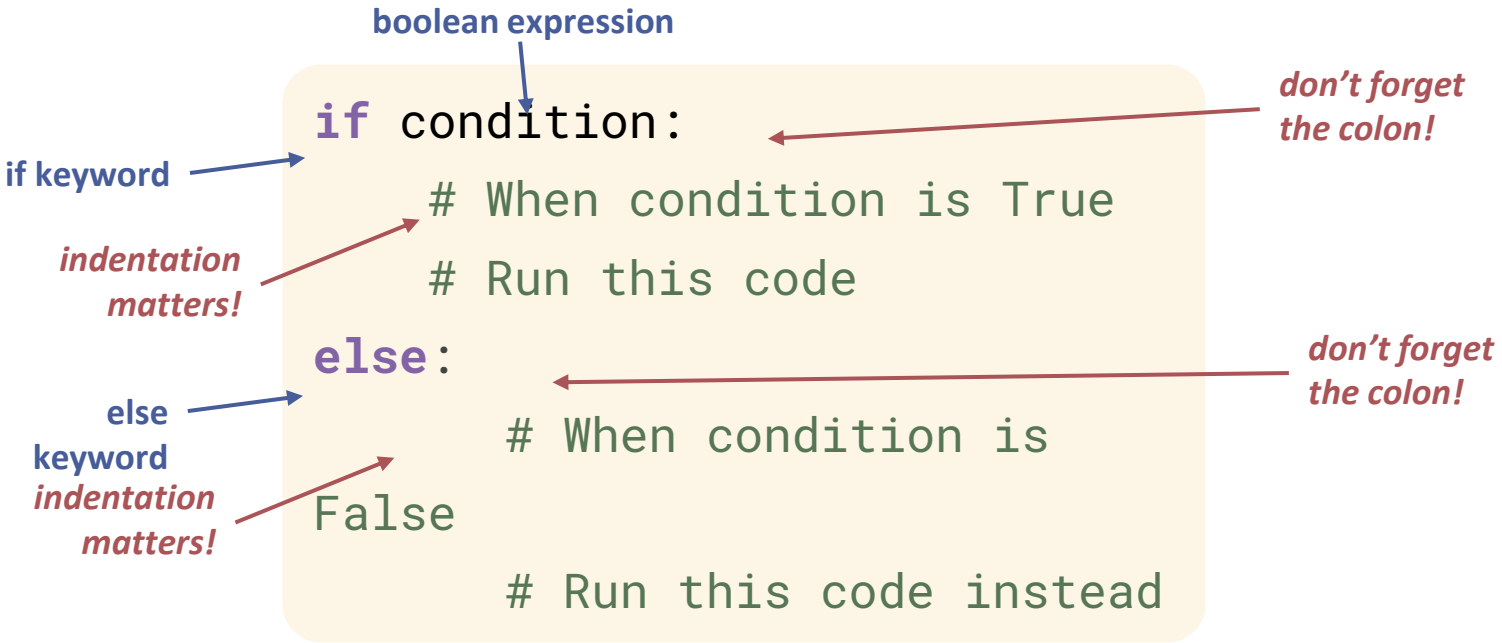


If Statement Example:

Jupyter Hub

```
x = 3
if x == 3:
        print("x is 3!")
```

If-else statement "Anatomy"

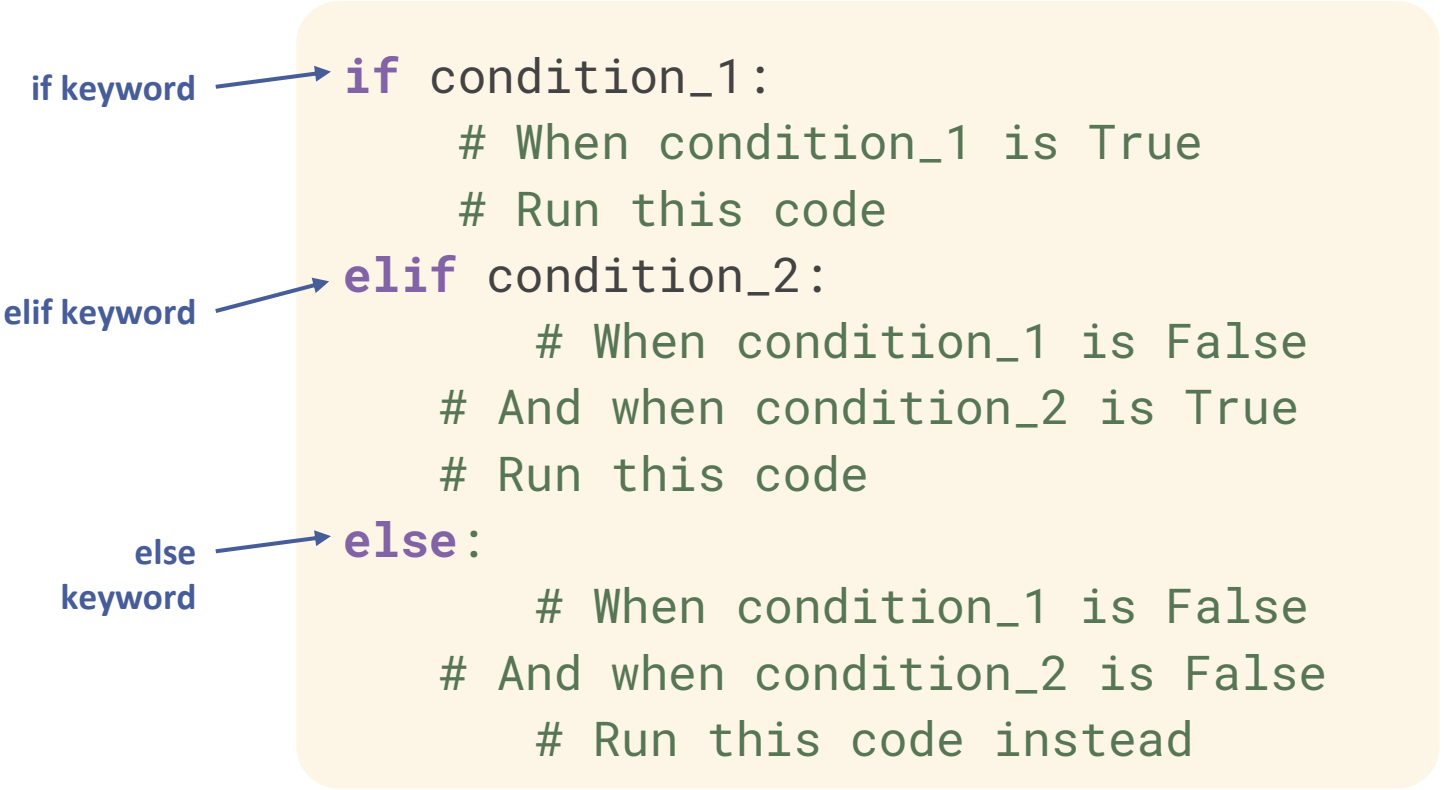


If-else Statement Example:

```
x = 3
if x == 3:
        print("x is 3!")
else:
        print("x is NOT 3")
```

What if we want to choose between more than two possible outcomes?

If-elif-else statement “Anatomy”



If-elif-else Statement Example:

Python Tutor

```
x = 3
if x == 3:
        print("x is 3!")
elif x == 7:
        print("x is 7!")
else:
        print("x is neither 3 nor 7")
```

Think Pair Share

Which statements will be printed?

A Only statement 1

B Only statement 2

C Statement 1 and statement 2

D Only statement 3

E Nothing will be printed

Sli.do

#cse160

```
x = 5
y = 5
if x > 0:
    print("x is greater than 0")
elif y > 0:
    print("y is greater than 0")
else:
    print("x and y are not greater than 0")
```

1

2

3

If-elif-else: check for more cases!

- If-else statements handle two cases
  - The case where something is True (if)
  - The case where something is False (else)
  - Only two possible outcomes
- **If-elif-else statements** can handle more cases!
  - Not every situation is binary (it happens or it doesn't)
  - **If** it's raining, **then** bring my umbrella; **else if** it's snowing, **then** bring my snowboard; **else if** it's sunny, **then** bring my paddleboard; **else** stay indoors.

- **Hypothetically**: you can check as many cases as you'd like
  - Be careful because some cases may overlap!
  - When the first case is True, all other cases are ignored!

If-elif-else: check many, many cases!

```
weather = "raining"
if weather == "raining":
    print("Wear a jacket")
elif weather == "cloudy":
    print("Go for a run")
elif weather == "sunny":
    print("Go paddleboarding")
else:
    print("Stay indoors")
```

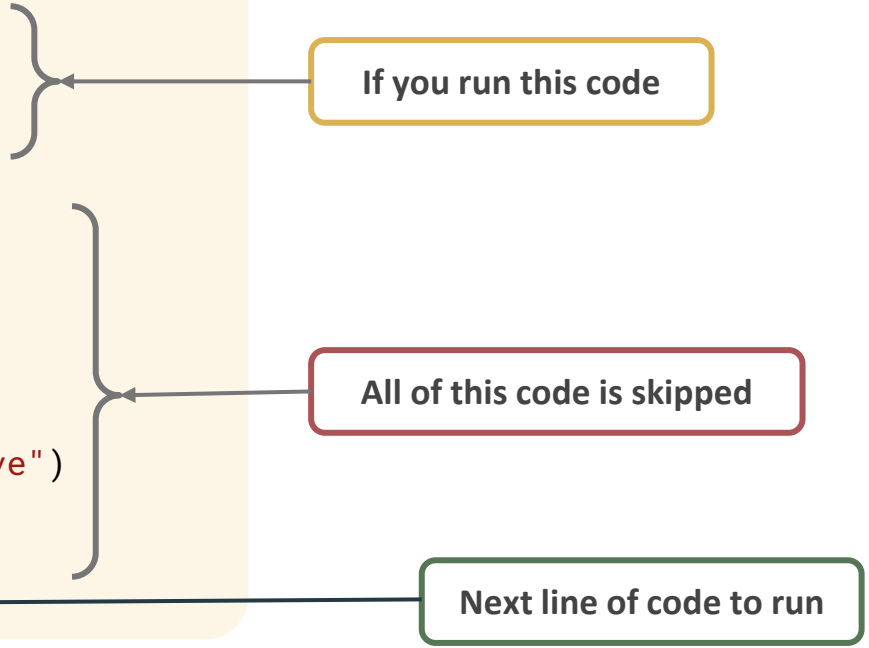
If-elif-else: Order of conditions matters!

```
temp = 30
if temp <= 45:
    print("Cold!")
elif temp <= 80:
    print("Comfortable.")
else:
    print("Hot!")
```

```
temp = 30
if temp <= 80:
    print("Comfortable.")
elif temp <= 45:
    print("Cold!")
else:
    print("Hot!")
```

Is there a better way to write this?

```
val = 0
# calculate absolute value of val
if val < 0:
    print("val is negative")
    print(val)
    result = -val
else:
    if val == 0:
        print("val is zero")
        print(val)
        result = val
    else:
        print("val is positive")
        print(val)
        result = val
print(result)
```



A Loop Problem, Again

Write a loop to print out the **even** numbers in the list: [9, 16, -100, -54, 48, 10, 32]

Output should be:

16

-100

-54

48

10

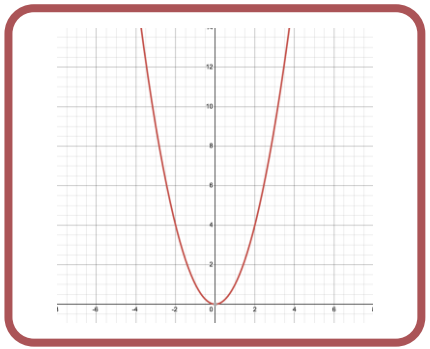
32

Inputs and Outputs

What things in the world take inputs, process them, and then produce outputs?

Title: “Week 1 Done”
Body: “….”

X



f(x) = x²

Functions

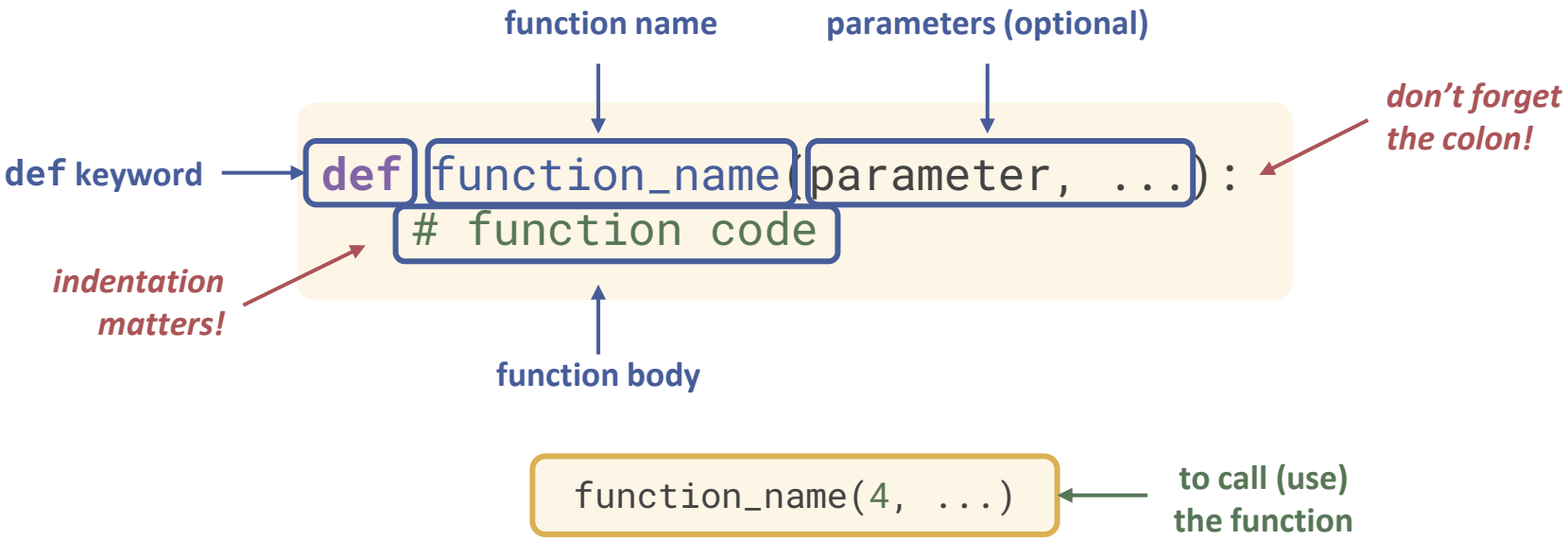
Functions *compartmentalize* code that can be used (or "called") when we want.

Functions

```
def my_func(input):
    # do work
    return output
```

- **Organize** code
- **Reuse** code, *but with* ***different inputs***
- **Abstract** away details

Anatomy of a Function



Return Statements

return is a Python keyword that is used to **output a value from a function**.

```
def function_name(parameter, ...):
    # perform actions
    # produce a value
    return value
```

return statement →

Returning from a Function

return is a Python keyword that is used to **output a value from a function**.

Once a **return** statement is executed, **no more code in the function will be run**.

```
def function_name(parameter, ...):
    # perform actions
    # produce a value
    return value
    print("Hey") # can't run!
```

unreachable code!