#7: Else Statements and Nesting

SAMS SENIOR NON-CS TRACK

Last Time

Understand how **scope** changes where we can access variables

Use **Booleans** to compute whether an expression is True or False

Use **if statements** to make choices about program control flow

Ex 4-1 Feedback

Make sure to read each problem prompt carefully! It will usually ask you to either:

- Set up variables with starting values and write code using them
- Request user input and write code using it
- Write a function and call it using values

An example of a solution in each format is shown to the right.

Ex4-1 was a bit tough for half the class, so let's go over all of the problems now.

```
# Set up a variable
x = 5
print("Result:", x + 2)


# Ask user for input
x = int(input("Enter a number: "))
print("Result:", x + 2)


# Write a function
def addTwo(x):
    return x + 2
print("Result:", addTwo(5))
```

Today's Learning Goals

Use **else** and **elif** statements to make multiple-option decisions

Use **nesting** to combine if statements with other if statements or functions

Conditional Branches

Else conditions for alternatives

Sometimes we want a program to do one of two possible actions based on the conditions. In this case, instead of writing two if statements, we can write a single if statement and give it an **else**. The else will cover the case when the Boolean expression is False.

```
if <boolean_expression>:
    <body_if_true>
else:
    <body_if_false>
```

Else is Attached to If

We can't write an else statement by itself- it has to come immediately after an if statement.

If there are multiple if statements, then the else is associated with the if that came before it.

```
x = 25
if x > 10:
    print("First")
if x < 20:
    print("Second")
else:
    print("Third")
```

Conditional Example

Prediction Exercise: What will the following code print?

```
x = 5
if x > 10:
    print("Up high!")
else:
    print("Down low!")
```

Question: What could we change to get the other statement to print instead?

Question: Can we get the program to print out both statements?

Exercise 1: weekend checker

Go to the schedule page and download the starter file for today's lecture. You'll write exercise code under the comment with the exercise's number.

Exercise 1: write a few lines of code that set a **variable** day equal to "Thursday". The code should print out "It's the weekend!" if the variable holds "Saturday" or "Sunday"; otherwise, it should print "It's a school day.".

Your code should work properly if the value in day is changed.

Multiple Branches

If we want to have more than two options for what the program can do, we can add one or more **elif** statements in between the initial if and final else. The program will only ever enter one branch of the conditional.

```
if <boolean_expression_A>:
    <body_if_A_True>
elif <boolean_expression_B>:
    <body_if_A_False_and_B_True>
else:
    <body_if_both_False>
```

Multi-Branch Example

The following example shows a three-branch conditional that checks whether a number is positive, negative, or zero. Note that we can assign a variable to the result, then print it at the end.

```
x = int(input("Please enter a number: "))
sign = ""
if x > 0:
    sign = "positive"
elif x < 0:
    sign = "negative"
else:
    sign = "zero"
print(sign)
```

Overlapping Branches

If we write a conditional using an if and elif and **both** Boolean expressions are True, we only enter the **first** branch that evaluated to True. The other branches get skipped over.

```
x = 15
if x > 10:
    print("Two digits")
elif x > 0:
   print("One digit")
```

Example: gradeCalculator

Let's write a few lines of code that asks the user to input a grade as a number, then calculates the letter grade that corresponds to that number.

90+ is an A, 80-90 is a B, 70-80 is a C, 60-70 is a D, and below 60 is an R.

Exercise 2: colored circle

Exercise 2: write a few lines of tkinter code that set up two **variables**, x and y, with initial values 100 and 350. The code should draw a circle centered at the x,y coordinate.

The color of the circle depends on where its center point is. If it's in the top-left quadrant of the canvas, the circle should be blue; if it's in the top-right quadrant, it should be red; and if it's in the bottom half of the screen, it should be purple. Your code must use an if-elif-else structure to accomplish this to get full credit.

Note: your code should work if the values assigned to x and y are changed

Nesting Conditionals

Combining Blocks of Code

So far, we've used both functions and conditionals to redirect program flow. We can start to write really interesting programs by **combining these control flow structures together**, so that we can redirect the program in more interesting ways.

We'll combine these structures by **nesting** one structure inside of another, so that one control structure runs inside the body of another.

Nested Conditionals

When we nest conditionals, both conditionals need to be True to reach the innermost statement. This works the same way as combining two Boolean expressions with **and**.

```
x = 50
if x >= 10:
    if x < 100:
        print("Two digits!")
```

Note that the body of the inner if statement has been indented **twice**, to show that it belongs to the inner if.

Nesting and Indentation

When we nest control structures, we'll use **indentation** to show which control structure a line of code belongs to. In general, lines of code at the **same indentation level** belong together.

```
first = True
second = True
if first == True:
    print("#1 body")
    if second == True:
        print("#2 body")
        print("still here")
    print("#2 body")
print("Out of the conditionals")
```

Nesting and Elif/Else Statements

Likewise, when we're pairing up elif or else statements in nested code, we'll pair them with the if statement at the **same indentation level**. This is true even if an inner if statement comes in between them! However, an outer if statement **cannot** come between them.

```
first = True
second = True
if first == True:
    if second == True:
        print("both true!")
else:
    print("first not true")
```

Question: if we want to add an else statement to the inner if, where should it go?

Exercise 3: rock-paper-scissors

Exercise 3: write a few lines of code that will play a basic guessing game with the user.

Ask the user to **input** rock, paper, or scissors. Choose one of the three values to check in the code. (You don't have to generate this randomly- just pick a value yourself).

- If the user guesses **correctly**, ask them to input a second guess, and compare it to a second value (which can be the same as the first, or different).
  - If they get it right again, print out "Wow!"
  - If they get it wrong the second time, print out "Nice try."
- If they get it **wrong** the FIRST time, print out "I beat you!", and don't ask them to play again.

Nested Functions and Conditionals

We can also nest an if statement inside a function! The same indentation rules apply as before.

```
def greetPerson(name):
    if name == "Kelly":
        print("Hello, Prof. Kelly!")
    else:
        print("Hi " + name + "!")

greetPerson("Tak")
print("---")
greetPerson("Kelly")
```

Edge Cases

Using conditionals inside of functions lets us deal with certain **edge cases** in our code. An edge case is an input to the function that doesn't fit the usual pattern of the function and needs to be dealt with specially.

For example, say we want to write a function that draws a line on the canvas between two given points. An edge case would be when those two points are the same. In that case, perhaps we want to draw a circle at that point instead!

```
def drawLine(canvas, x1, y1, x2, y2):
    if x1 == x2 and y1 == y2:
        canvas.create_oval(x1 - 5, y1 - 5, x1 + 5, y1 + 5, fill="black")
    else:
        canvas.create_line(x1, y1, x2, y2)
drawLine(canvas, 100, 120, 300, 50)
drawLine(canvas, 200, 300, 200, 300)
```

Returning Early

When we use conditionals in functions, it lets us use return statements differently then before. Now, instead of needing to return on the very last line, we can return early!

```
def checkAge(yearBorn):
    if yearBorn > 2019:
        return "You haven't been born yet!"
    age = 2019 - yearBorn
    return age
print(checkAge(2001))
print(checkAge(2020))
```

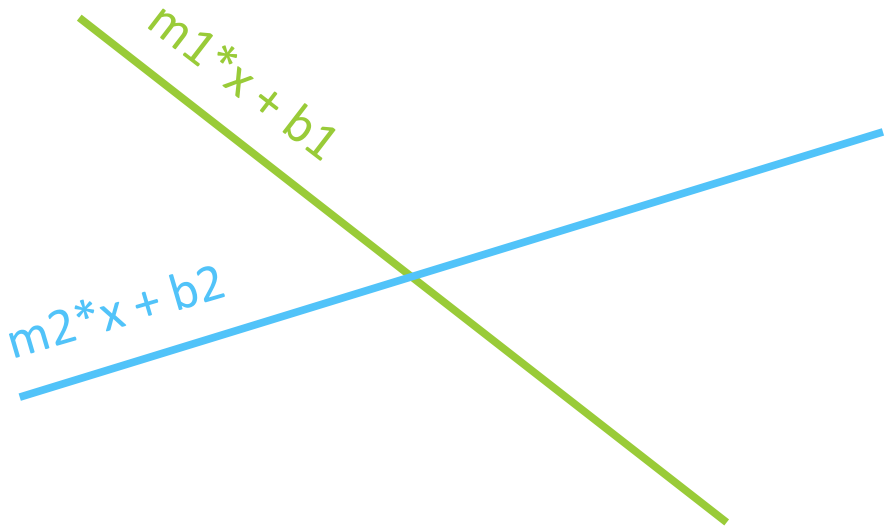
As soon as we reach a return statement, the code will **exit the function**. Therefore, in the second case, only one value (the string) is returned.

Exercise 4: lineIntersection

Exercise 4: write a **function** lineIntersection(m1, b1, m2, b2) which takes two lines, m1*x + b1 and m2*x + b2, and returns the x coordinate where they intersect.

If the two lines are parallel (and don't intersect), return the string "parallel" instead. Recall that two lines are parallel if they have the same slope.

At the top level of the code, call the function and print out its result twice: once with two intersecting lines [3x - 4 and 2x + 7] and once with two parallel lines [5x - 2 and 5x + 9].



Recall that we can find the coordinate x with the formula:

$$x = \frac{b1 - b2}{m2 - m1}$$

Today's Learning Goals

Use **else** and **elif** statements to make multiple-option decisions

Use **nesting** to combine if statements with other if statements or functions