8.1. Intro: What we can do with Turtles and Conditionals

So far, our programs have either been a series of statements which always execute sequentially or operations that are applied to each item in an iterable. Yet programs frequently need to be more subtle with their behavior. For example, a messaging app might only set a message's title bold if it has not been read by the user. Or a video game needs to update the position of all the characters that are not asleep. This is done with something called a **selection** or a **conditional statement**.

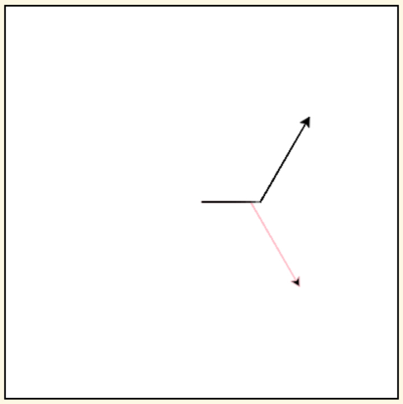
In the context of turtle drawings, using this kind of statement will allow us to check conditions and change the behavior of the program accordingly

Save & Run 5/9/2021, 8:34:49 AM - 2 of 2 Show in CodeLens

```
import turtle
wn = turtle.Screen()

amy = turtle.Turtle()
amy.pencolor("Pink")
amy.forward(50)
if amy.pencolor() == "Pink":
    amy.right(60)
    amy.forward(100)
else:
    amy.left(60)
    amy.forward(100)

kenji = turtle.Turtle()
kenji.forward(60)
```



Activity: 1 -- ActiveCode (ac7_1_1)

In the above code, we first set amy's pen color to be "Pink" and then move her forward. Next we want one of two actions to happen, either amy should move right and then forward, or left and then forward. The direction that we want her to go in depends on her pen color. If her pen color is set to pink - which is determined by writing `amy.pencolor() == "Pink"` which checks to see if the value returned by `amy.pencolor()` is the equivalent to the string "Pink" - then we should have her move right and forward. Else (or otherwise) she should move left and forward. Both things can't happen though. She can't move right, forward *and* left, forward. We then do the same thing for kenji, though in this case, we didn't change kenji's pen color.

It might seem a bit odd to add the conditionals in this example. Wouldn't we already know that we set up amy and kenji's colors, so why would we need a conditional? While it's true that this isn't the *best* place to use a conditional, we can combine conditional statements with for loops to make something pretty cool!

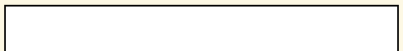
Save & Run 5/9/2021, 8:36:34 AM - 2 of 2 Show in CodeLens

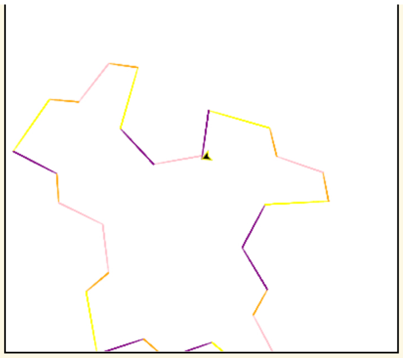
```
import turtle
wn = turtle.Screen()

amy = turtle.Turtle()
amy.pencolor("Pink")
amy.right(170)

colors = ["Purple", "Yellow", "Orange", "Pink", "Orange", "Yellow", "Purple", "Ora


for color in colors:
    if amy.pencolor() == "Purple":
        amy.forward(50)
        amy.right(59)
    elif amy.pencolor() == "Yellow":
```





Activity: 2 -- ActiveCode (ac7_1_2)

The above example combines a for loop with a set of conditional statements. Here, we loop through a list of colors and each iteration checks to see what amy's pen color is. Depending on the pen color, the turtle will move in a certain direction, for a certain distance. Before the for loop iterates, amy's pen color is changed to whatever `color` is in the for loop and it continues. Note how the color doesn't change until the end, so that we can start using whatever color amy is set to initally. This means that the last color in the list `colors` will not be used, though you can see how the icon changes to the appropriate color.

This chapter will further detail how to use conditional statements.

8.1.1. Learning Goals

- To understand boolean expressions and logical operators
- To understand conditional execution
- To be able to write a boolean function
- To know when to use binary, unary, chained and nested conditional statements

8.1.2. Objectives

- To properly evaluate a (compound) boolean expression
- To use parenthesis to properly demonstrate operator precedence
- To use conditional statements to properly branch code

You have attempted 3 of 2 activities on this page

8. Conditionals">

itionals">

8.2. Boolean Values and Boolean Expressions">

✔ Completed. Well Done!

| Back to top