WEEK 3: SELECTION

If-else in Python

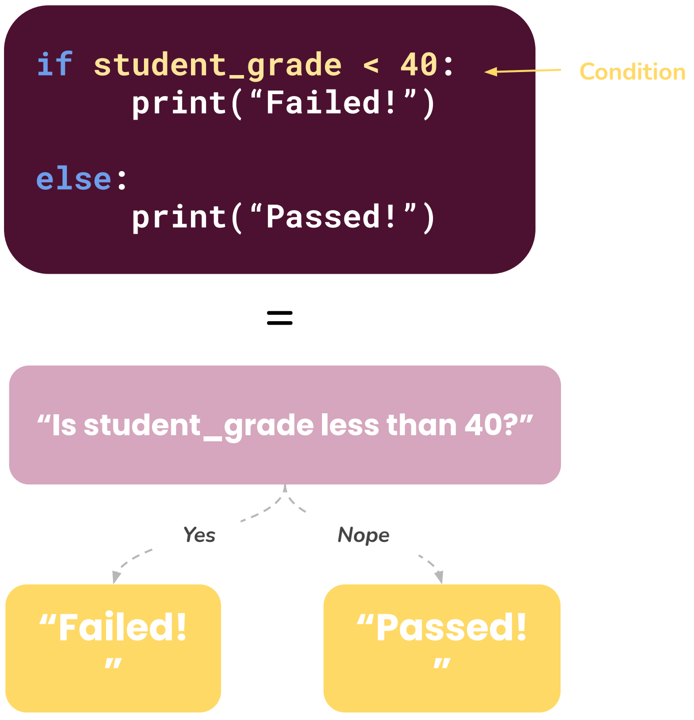
This week we looked at selection, and how we can utilise selection in Python through the *if statement*. We learned that by using an *if*, we run one bit of code or another depending on some kind of condition. We also learned about relational operators and logical operators, but more on that later.

An if statement works by testing some kind of condition, and based on that, running some code. This means we can conditionally do things. For example, we may want to only print out "Failed!" if (and ONLY if) a student's grade is below 40%.

We also learned about the *else* keyword and how it essentially means "otherwise". The *else* keyword allows us to run code if the condition is not satisfied.

```
if student_grade < 40:
     print("Failed!")

else:
     print("Passed!")
```



Experiment with if-else in Python:

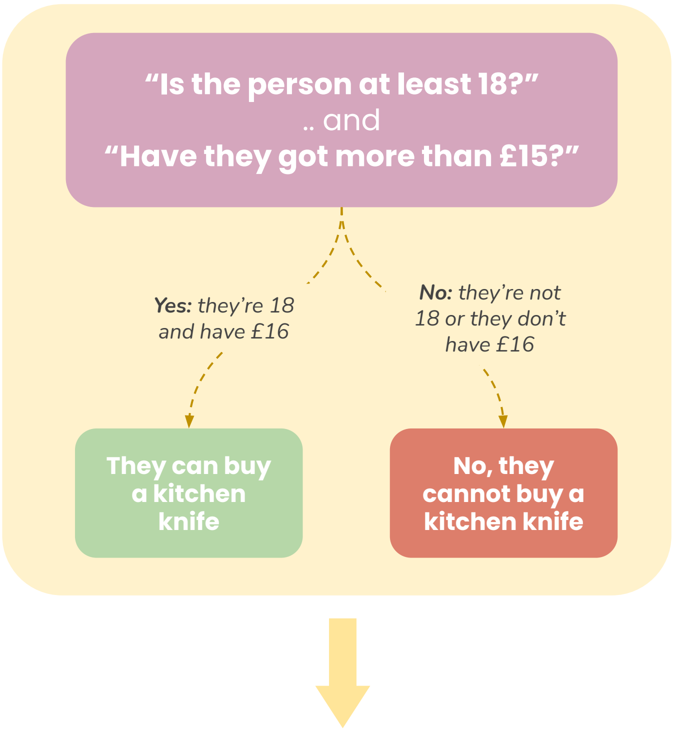
1. Try the example in the top-right of this page. Make a variable called **student_grade**, and test if that value is less than 40. If it is, show "you failed", otherwise show "you passed". Try it out with **student_grade** = 90 and **student_grade** = 10. What do you get?
2. Create a program which will show "boiled!" if a variable **temperature** is more than 99. Otherwise, show "boiling".
3. Make a program which tests if a variable **input_text** is equal to "hello". If it is, print out "hello there!"
4. Expand this to encompass other phrases, like: "how are you?", "goodbye", "how old are you?" and "what's your name?".
5. Expand this even further to let the user type in their text, rather than setting it directly in the code. Look up "python input function" for more information, or rewatch this week's talk (at the end) where we discuss the input() function.

Logical operators

Another thing we looked at was *logical operators*. Before we talk about these, we'll consider relational operators. "Relational" operators are operators which compare two things. For example, if we wrote that **a** < **5**, the relational operator here is "<". This is because it is comparing if **a** is less than 5 -- a relationship.

Logical operators are similar, but fundamentally different. There are three logical operators: **`and`**, **`or`** and **`not`**. Logical operators allow us to build complex conditions where something AND something else has to be true, or if something OR something else is true. Here the **and** & **`or`** operators can be used respectively. For example, using the **`and`** operator, we could test if a variable **`money`** is more than 15 *and* another variable **age** is more than 17. This process would allow us to age-restrict purchases. Conditions with a logical operator are sometimes referred to as "compound conditions".

The **and** / **`or`** operators are also useful for checking if a variable is within a certain range. Finally, the **`not`** operator allows it to test the inverse: if something is NOT true.


"Is the person at least 18?"
.. and
"Have they got more than £15?"
Yes: they're 18 and have £16
No: they're not 18 or they don't have £16
They can buy a kitchen knife
No, they cannot buy a kitchen knife

```
print("Can I see your ID pls");

if money > 15 and age >= 18:
      print("Here's a knife!")

else:
      print("Nope, get out")
```

Experiment with logical operators in Python:

1. Implement the example seen above. Don't forget: you need to make the **money** and **age** variables!
2. Create a basic working discount system. Make a variable called bill and set it to 50. Make two other variables **is_staff** and **has_discount_code** and set both to True initially. Then, write an if statement which will test if either of these is true (using the OR operator), and if so, will give them a discount of 25%. Then, print out the final bill.
    a. What happens if you change one of these variables to False?
    b. What happens if you use the AND operator instead of the OR operator?
    c. What happens if you use the NOT operator before the condition?
    d. Can you expand this to test if a person has enough money?
3. Make a variable called **value** and set it to 5. Without using the > or >= operators, can you test if value is more than 5? *(hint: value > 5 is true if value <= 5 is not true. The < and > operators can be thought of like opposites)*
4. Can you write a program to accept a username and password from the user, and test if both their username is "admin" and password is "1234", showing "Access Granted" if this condition is true?

```
if request == “Limerick”:
        print(“There once was a man from Peru”)
        print(“Who dreamed he was eating a shoe”)
        print(“He awoke with a fright”)
        print(“In the middle of the night”)
        print(“To find his dream had come true”)
```

```
if password_attempts < 5:
        print(“Unable to log in.”)
        password_attempts += 1

else:
        print(“You cannot try again.”)
```

```
# This is similar to the example on the previous
page, but without logical operators

if money > 15:
        print(“You have enough money!”)
        if age > 18:
                print(“You can buy a knife”)
        else:
                print(“.. but you’re not 18”)
else:
        print(“You don’t have enough money”)
```

What’s that space about?

You might have realised that the lines Python runs in an if statement have a space before them. Why is this? Well, we call this process of putting a space before the line *indentation*. We “indent” code using the TAB key on the keyboard.

Why is it needed? The answer is because we need some way of telling Python which bit of code we want to conditionally run. Otherwise Python gets confused and doesn’t know if you want to run a line of code based on the condition. The way of clarifying this is by putting a space before the line!

So far we’ve looked at if statements with just one line of code “inside” them. But actually, we can run anything inside of an if statement. Some examples can be seen to the left.

We can even have an if statement inside another if statement! In that case, we would have indent code inside the first if ONE time, and TWO times for the second if.

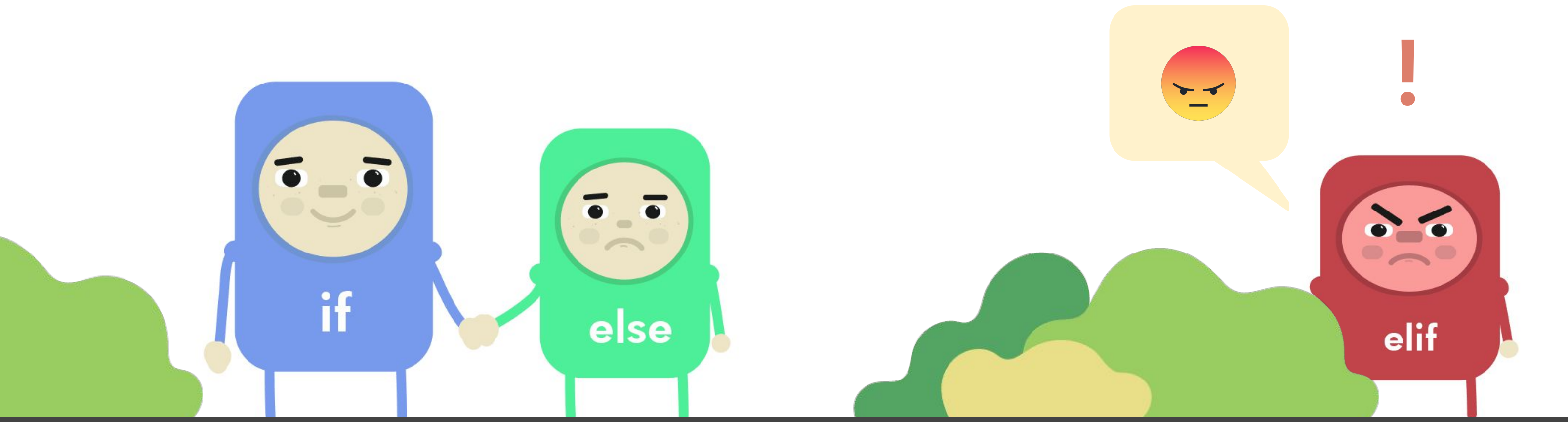
This is related to something referred to as “scope”. We will touch on scope in either the next week, or week after, so don’t worry if it’s confusing!

Experiment with indentation in Python:

1. Try out some of the example above. Test them thoroughly and understand how they work.
2. Write a simple if statement testing some condition of your choice. If this condition is satisfied, print out “yay”. Test it and make sure it runs the code if the condition is satisfied. Then:
    a. Try removing the space before print. What happens? Do you get an error? Why do you think this is?
    b. Add another print statement below the first one, and make sure they’re both indented once. Test to see if it works. What happens if you remove the space before the second print? Does it run? Why?
    c. Test this out and see what happens if the second message is shown. If it does, why is this?
3. Write a “nested” if statement: an if statement within another if statement. Pay close attention to the indentation and ensure you have the correct level of indentation for each line. Can you go one step further? An if statement within an if statement, within an if statement?

Hang on, where's elif?

For those of you with some previous programming knowledge, you might have noticed we left out the **elif** keyword. The **if**, **else** and **elif** keywords are all related, but we didn't have time to discuss what an **elif** is, or what it does. So stay tuned for next week where we'll touch on what an elif is, and how it can be used!



NEXT WEEK:
GETTING LOOPY

THE Ben AND Chris SHOW