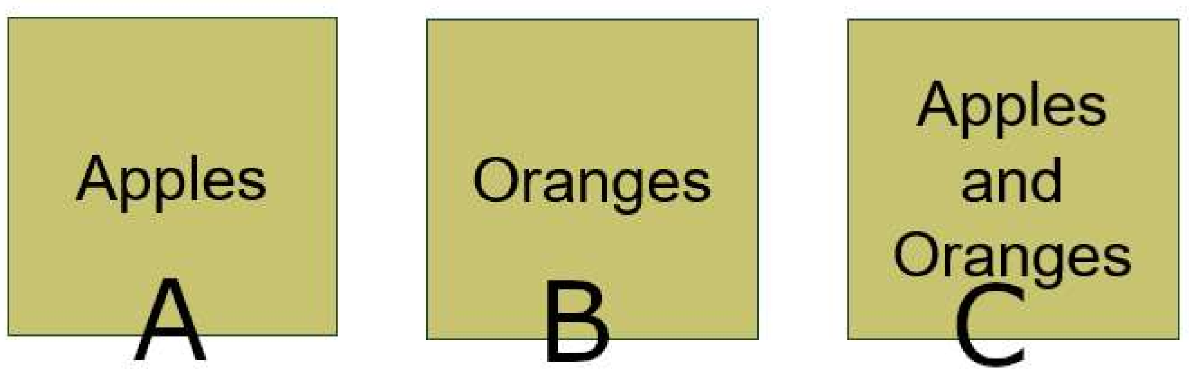
Culture and Coding: Python Unit 1 - Introduction to Python Conditionals

In this section, we will explore conditional operators and the ability to be able to execute code conditionally. First however, let us take a look at our logic with a warm up problem.

Three mislabled boxes problem

- A classic logic problem
- We have three boxes
- They are labled
  - Apples
  - Oranges
  - Apples & Oranges
- However, we know they are *all* mislabled
- We want to correct the labels
- We can only pull *ONE* item from ONE box

- Can we relable the boxes correctly?



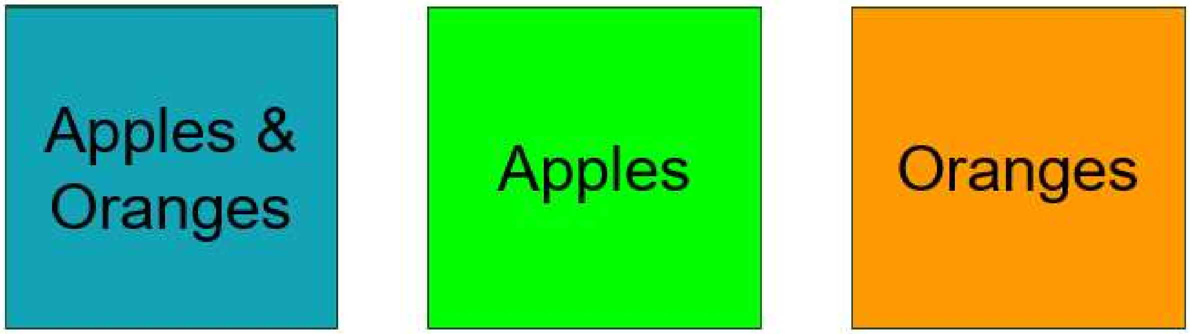
> **DISCUSSION**
>
> - Work together at the table to solve the problem.
> - Document on a piece of paper *how* you solve.
> - A key thing in technical interviews is being able to discuss your logic outloud, so make sure you are talking.

We will give about 5 minutes to discuss. You will be asked which **one box** we will want to start with.

Solution

- Apple box - possible correct answers: Oranges OR Apples and Oranges
- Orange Box - possible correct answers: Apples OR Apples and Oranges

        - If I pick from both boxes, I could end up with either apple or orange - no definitive answer
- Apples and Orange Box - possible correct answers: Apples OR Oranges
    - This seems like a definitive answer - as we know if we pick one, the other is not in there
    - Let's assume we pick an orange from the "Apples and Oranges"
        - That means, that box is Orange
        - The Apples box Apples & Oranges
        - The Oranges box must be Apples
        - We have a definitive answer!



Notice: We didn't just say which box, but we *tried it out*.
Logic problems should always be tested, as should code!

Logic Problems and Coding

Not every programmer likes logic problems (personally, I didn't)

This is because, we are often not trained on how to do them. Logic is really asking ourselves:

- What do you know
- What do we have to work with
- What can we do based on conditions given

And then...

- Coming up with a possible solution
- Testing it out

At that point, it moves from 'logic problem'. To simply: **problem solving** and **coding**

Right now, we can do is limited to a single path, but let's change that!

Conditional Operators

- We already know the basic mathmatical operators
    - +
    - -

    - `*`
    - `/`
    - `=`

However, logic is so important, all programming languages have **conditional** operators.

- Conditional Operators are
    - `<` - is my left **less than** my right
    - `<=` - is my left **less than or equal to** my right
    - `>` - is my left **greater than** my right
    - `>=` - is my left **greater than or equal to** my right
    - `==` - is my left **equal to** my right
    - `!=` - is my left **not equal to** my right

  Based on what they are comparing, they all evaluate to `True` or `False` (notice capital!)

In [ ]:

```
myValue = 5 > 5
print(myValue)

myValue = 5 >= 4 + 1 # conditional operators have lower precidence, so this is 5 >= 5
print(myValue)
```

```
False
True
```

Notice, these are just like standard operations. They give us a value. Which means we can store (like above) or **return** that value.

In [ ]:

```
def is_less_than_zero(val1):
    return val1 < 0

print("TESTING function")
print(is_less_than_zero(-10))
print(is_less_than_zero(11))
```

```
TESTING function
True
False
```

In class activity - Conditionals

Let's work on an in class activity. You can find it linked in canvas.

Your Task:

Given the following code, write a condition statement that produces the following output

```
False
True
True
```

As a group Step 1: Figure out the logic, and write it out (not in code) Step 2: Write it out in code Step 3: Test with other values!

Tip

It can be in two lines or one line, but for reference, here is the code from above.

```
def is_less_than_zero(val1):
    return val1 < 0  ## one line version

def is_less_than_zero(val1):
    myValue = val1 < 0
    return myValue ## two line version
```

In [ ]:

```
def age_check(age):
    return False  # replace false with your conditional statement. This can be *one l


print(age_check(20))  # should print False
print(age_check(21))  # should print True
print(age_check(22))  # should print True
```

```
True
False
True
```

Coditional Code Execution (if/else Statements)

Using conditionals, we can have code only execute given a certain condition. We call this an **if** statement.

- if condition is true:
    - execute the indented block of code
- else
    - execute a different block of code (if any)

Examples:

In [ ]:

```
if 10 > 5:
    print("In first if statement")
    print("As long as it is indented, it is a block of code")

value = int(input("Enter a whole number: "))
if value > 0:
    print(f"{value} is above zero")
else:
    print("it is not above zero")

name = input("Enter your name")
if name == "Dave":
    print(f"I can't do that {name}.")
else:
    print(f"Dave didn't like my answer {name}.")
```

```
In first if statement
As long as it is indented, it is a block of code
42 is above zero
I can't do that Dave.
```

Adding additional options elif

In python (not all languages), if we want to have more options we use `elif` (short for else if).

The rule:

- checks the first if, if False
- check the next elfif
- execute until end
    - else is often included, but technically optional

In [ ]:
```
value = int(input("Enter a whole number: "))

if value > 0:
    print("Value is possitive")
elif value < 0:
    print("Value is negative")
else:
    print("Value is zero")
```

```
Value is possitive
```

Or, let's put in a function, so we can reuse it!

In [ ]:
```
def isPossitive(value):
    ans = "The value is zero" #assuming a value
    if value > 0:
        ans = f"{value} is possitive"
    elif value < 0:
        ans = f"{value} is negative"
    return ans

val = int(input("Enter a possitive number: "))
val2 = int(input("Enter a random number: "))

answer1 = isPossitive(val)
answer2 = isPossitive(val2)

print(f"{answer1} and {answer2}")
```

```
42 is possitive and The value is zero
```

In Class Activity: Part 2

You can access the activity in zyBooks. We will have limited time, but do the best you can.

Your Task

Given the following output, build the if/elif statement in the provided fucntion.

```
OK
NOT OK
OK
OK
NOT OK
Unknown Region, Not OK
```

Notice, like above the 'default condition' is provided at the start. Here is the code from above.

```python
def isPossitive(value):
    ans = "The value is zero" #assuming a value
    if value > 0:
        ans = f"{value} is possitive"
    elif value < 0:
        ans = f"{value} is negative"
    return ans
```

TIP

To accomplish this task, you need to 'nest' if-statements. This is because the block of code can be any block including if-statements.

Here is an example of a nested if-statement.

```python
if name == "Turing":
    if year > 2018:
        answer = "Turing test solved!.. maybe"
    else:
        answer = "Not solved"
elif name ==  "Schneider"
    if year >= 2019:
        answer = "Are you trying Susan's Schneider's AI Consciousness or
Chip Test?
    else:
        answer = "Not sure which test you are talking about"

print(answer)
```

In [ ]:

```python
def age_check_by_region(age, region):
    confirm = "Unknown Region, Not OK"
    # write an if check, that will
    # set the value of confirm
    # based on the values expected below
    # will require a nested if statement, or an additional function
    return confirm


print(age_check_by_region(21, "USA"))  # prints  "OK"
print(age_check_by_region(20, "USA"))  # prints  "NOT OK"
print(age_check_by_region(21, "EURO"))  # prints  "OK"
print(age_check_by_region(18, "EURO"))  # prints  "OK"
print(age_check_by_region(17, "EURO"))  # prints  "NOT OK"
# if anything else is passed into region, it will return "Unknown Region, Not OK"
print(age_check_by_region(32, "YOLO"))
```

Dealing with Logic - DRAW IT OUT!

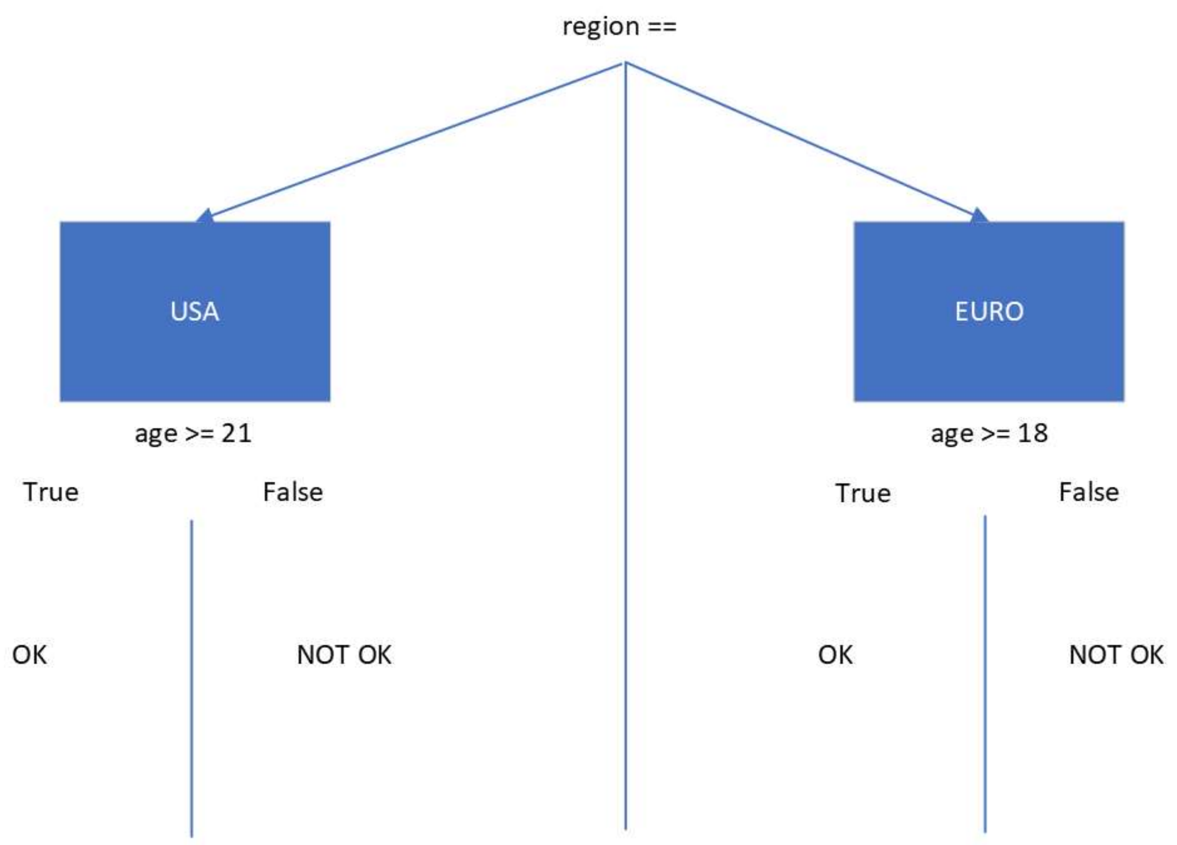
Really, this is the **Super Ninja Trick**

Draw out the "logic tree"

- When you are reading code
- When you are figuring it out, before writing.

Let's look at the example above.

```
* We have two halves right away:
   * region == "USA"
   * region == "EURO"
```



Next Class

> I propose to consider the question, 'Can machines think?'" Alan Turing - Computing and Intelligence, 1950