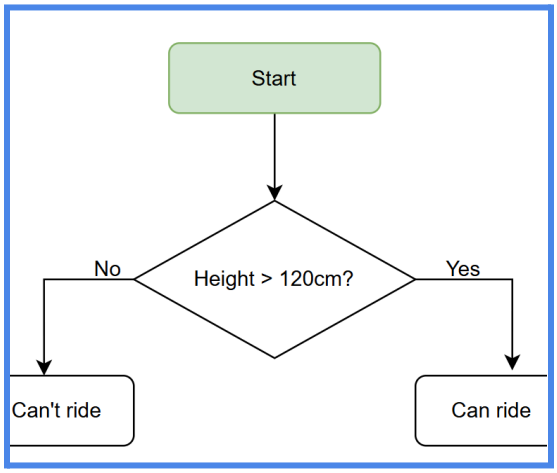
DAY 3:Control Flow and Logical Operators

https://appbrewery.github.io/python-day3-demo/

Control Flow with If / else< and conditional

The **`if`** and **`else`** statements in Python are used for decision-making, allowing your program to execute specific blocks of code based on whether a condition is true or false. Let's make an example: If we work on a roller coaster, we need to sell tickets but we need to check if the person has the appropriate height to enter.

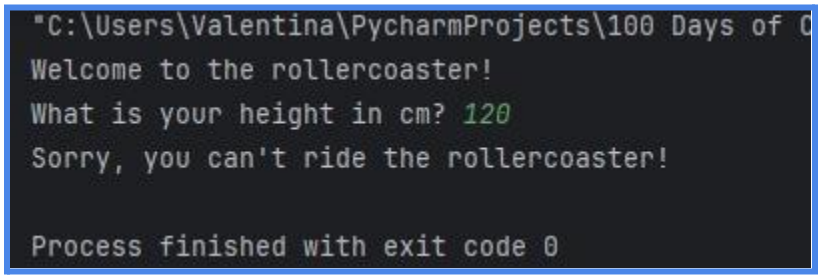


This is basically the structure that were going to use:

```
print("Welcome to the rollercoaster!")
height = int(input("What is your height in cm? "))

if height > 120:
    print("You can ride the rollercoaster!")
else:
    print("Sorry, you can't ride the rollercoaster!")
```

Now, when we use this greater than sign (>) effectively, what we're saying is that is the height greater than 120, which means that it does not include 120. In fact, if I run this

code and if I type in my height as 120, then it actually goes into the else block and prints this.



That's why we need to write >= And now when I run my code and I write 120, it's telling me that I can ride the roller coaster.

Comparison Operators

Operator	Meaning
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Greater than (>) Lesser than(<) we've also seen greater than or equal to (>=) and lesser than or equal to (<=). Now in this table there's also the equal to (==) and not equal to.(!=) For example, if you wanted to check if somebody's height is equal to precisely 120, then you would use two equal signs. It gets a little confusing if you are new to programming because sometimes we're typing one and other times we're typing two.

Single Equal Sign (=) vs Double Equal Signs (==)

The single one is used for asigment
Example: `height = 120` assigns the value `120` to the variable `height`.

The double equal sign It is used for *comparison* to check if two values are equal.
Example: `height == 120` checks if the variable `height` is equal to `120`.

```
if height == 120:
    print("You can ride the roller coaster!")
else:
    print("Sorry, you cannot ride the roller coaster.")
```

Modulo operator

The **modulo operator (%)** gives you the **remainder** when you divide one number by another.

If you have 7 candies and you group them into groups of 3, how many are left over?

- **7 % 3 = 1** (because 7 divided by 3 leaves a remainder of 1).

Examples:

1. **10 % 3 = 1**
   - 3 fits into 10 three times (3 × 3 = 9), leaving **1 left over**.
2. **9 % 2 = 1**
   - 2 fits into 9 four times (2 × 4 = 8), leaving **1 left over**.
3. **8 % 4 = 0**
   - 4 fits into 8 exactly two times (4 × 2 = 8), leaving **0 left over**.

We can use the modulo operator to see if a number is even or odd(pares o impares)

```
print("Lets see if you number is even or odd")
number=int(input("Put here your number"))


if number % 2==0:
    print("Even")
else:
    print("Odd")
```

Nested if and Elif statements

In the last example of the roller coaster we check with the if and else statement if there over 120, but we also need to check the age, If somebody is over 18 years old, then they should be paying the adult price, which is let's say $12, but if they are 18 or under, then they should only be paying $7. But in a nested if statement, once the first condition has passed, we can check for another condition, and then we can have another if else statement inside this if condition.

```
if condition:
    if another condition:
        do this
    else:
        do this:
else:
    do this:
```

Were going to ride our code with the roller coaster example:

```
print("Welcome to the roller coaster!")
height = int(input("What is your height in cm? "))

if height >= 120:
    print("You can enter the roller coaster! 😊")
    age = int(input("What is your age? "))

    if age <= 18:
        print("Please pay $7.00")
    else:
        print("Please pay $10.00")

else:
    print("Sorry, you are not tall enough to ride. 😞")
```

But what happens if we have more than two tickets to offer? That's where the elif enters

```
print("Welcome to the roller coaster!")
height = int(input("What is your height in cm? "))

if height >= 120:
    print("You can enter the roller coaster! 😊")
    age = int(input("What is your age? "))

    if age <= 12:
        print("Please pay $5.00")
    elif age <= 18:
        print("Please pay $7.00")
    else:
        print("Please pay $12.00")

else:
    print("Sorry, you are not tall enough to ride. 😞")
```

We can create ad many elif as we want

Exercise

BMI Calculator with Interpretations

- Add some if/elif/else statements to the BMI calculator so that it interprets the BMI values calculated.
- If the bmi is under 18.5 (not including), print out "underweight"
- If the bmi is between 18.5 (including) and 25 (not including), print out "normal weight"
- If the bmi is 25 (including) or over, print out "overweight"

```
weight = 85
height = 1.85

bmi = weight / (height ** 2)

# 🚨 Do not modify the values above
# Write your code below 👇
if bmi < 1.85:
    print( "underweight")
elif 18.5 <= bmi < 25:
    print("normal weight")
else:
    print("overweight")


```

Multiple Statements in Succession

Coming back to our roller coaster ticketing problem. We want to be able to charge users an extra $3 if they want to purchase a ticket that includes a photo. Even if we've already gotten their age and height and determined their ticket price. This is an extra question. Do you want a photo or not? Yes or no?

```
#if/elif/else
if condition1:
    do A
if condition2:
    do B
else:
    do C
```

```
#MultipleConditions
if condition1:
    do A
if condition2:
    do B
if condition3:
    do C
```

To do this we would write multiple if conditions. If condition1 is true, then do A, but then the code is going to go to the next case and check if condition2 is also true, in which case it will do B, and if the final condition is also true, it's going to do C.
Whereas on the example on the left here, only one of these things, A, B or C will be carried out. Comparing the example on the left where we're using, If / elif / else only one of these things, A, B, or C will be carried out. But on the right-hand side, all three conditions are checked, and if it so happens that all three conditions are true, then A, B, and C will all be executed. So how do we implement this in our code?

1. First I'm going to tell them which type of ticket they're eligible for.
2. Then using an input we're going to ask if the want a picture or not it has to be at the same indentation level as this if block. And then I'm going to save that input inside a variable called wants_photo.
3. So now I'm going to use an if statement to check what their answer was. If it was true then I'm going to add $3, and if it was false I'm just going to skip giving them their ticket price. So if wants_photo is equal to y, well, in this case I'm going to go and add $3 to their bill.
4. I'm going to create a variable called bill and equal to 0 and then at the bottom of the age im going to put bill=5 and depending how old are you were going to increase the prices
5. But once we land in this if statement I'm going to have to add $3 to their bill, no matter which value it is at the moment. So effectively, what I want to do is bill equals the current value of bill plus three.
6. When you want a new value for your variable there is a short way to do it: bill += 3.
7. Finally i'm just going to print to the user their final bill.

```
print("Welcome to the rollercoaster!")
height = int(input("What is your height in cm? "))
bill = 0

if height >= 120:
    print("You can ride the rollercoaster!")
    age = int(input("What is your age? "))
    if age <= 12:
        bill = 5
        print("Child tickets are $5.")
    elif age <= 18:
        bill = 7
        print("Youth tickets are $7.")
    else:
        bill = 12
        print("Adult tickets are $12.")

    wants_photo = input("Do you want a photo? (y/n): ")
    if wants_photo == "y":

        bill += 3

    print(f"Thank you for your patience! Your final bill is ${bill}.")

else:
    print("Sorry, you have to grow taller before you can ride.")
```

Pizza project

The goal is to create a Python Pizza Delivery Program, and the program is going to automatically calculate the bill for the user based on a number of options.

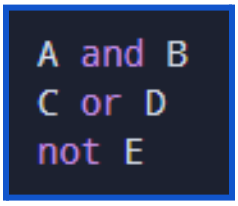
```python
print("Welcome to Python Pizza Deliveries!")
size = input("What size pizza do you want? S, M or L: ")
pepperoni = input("Do you want pepperoni on your pizza? Y or N: ")
extra_cheese = input("Do you want extra cheese? Y or N: ")
bill = 0
#Determinate the base price
if size == "s":
    bill = 15

elif size == "m":
    bill = 20

elif size == "l":
    bill = 25

#The pepperoni price
if pepperoni == "y":
    if size == "s":
        bill += 2
    else:
        bill += 3
#the cheese price
if extra_cheese == "y":
    bill += 1
print(f"Your bill is ${bill}")

```

Notes:

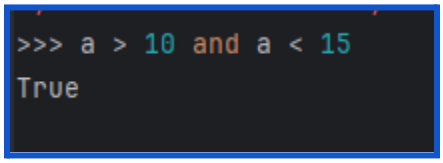
1. Always look the uppercase, very important, if the actions of your code are just available in uppercase can be a problem from the user
2. The if, elif and all the statements that you want to do need to be clear and align.

Logical Operators

There's three of them that are really useful: "and", "or", and "not".

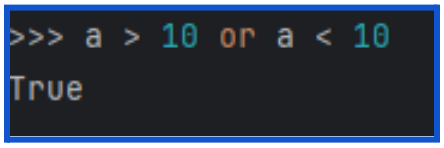


- In the first one when you combine two different conditions using an and operator they both have to be True both A and B for the entire line of code to be True. If just one of them is True, say A is True and B is False, or A is False and B is True, then evaluates to False. Example: We say a = 12



- I hit Enter, then I would get, True, because both a is greater than ten and a is less than 13 are True. That what happens when you use and.

- Now if you only needed one of the conditions to be True, then you could use the "or" operator instead. So if C or D were True, or if they're both True, then it will evaluate to True. It's only when both C and D are False, does this statement actually become False.



A	B	A OR B
true	true	true
true	false	true
false	true	true
false	false	false

- Now the final one is the "not" operator. And all that this does is it basically reverses a condition. So if the condition is False, then it becomes True. If it's True, then it becomes False. It only works when the condition that's being checked is False. So effectively it flips the True to the False and False to True.

```
>>> not a < 0
True
```

- So we know that a is 12, so is a < 0? Well it's not because it's 12. It's definitely not less than zero. But by putting the (not) in front of it, it reverses the condition. So a < 0, this will evaluate to False, but putting the not in front of it it becomes True.
-

A	NOT A
true	false
false	true

In the example of our roller coaster lets add a condition for midlife crisis people (45-55) lets charge $0

```
elif age >= 45 and age <= 55:
    print("Adult tickets in mid life crisis are $0")
```

There's a shorter way to do it, now because we're checking two conditions and we have the "and" logical operator in between, sometimes you want to do it less wordy, you just take away the “and” but from someone that is just learning we can understand it more with it, some im going to leave it like that.

Final Project

"Choose Your Own Adventure Books," where you flip to a different page if you make a particular choice, and then the story kind of evolves as you make your choices.