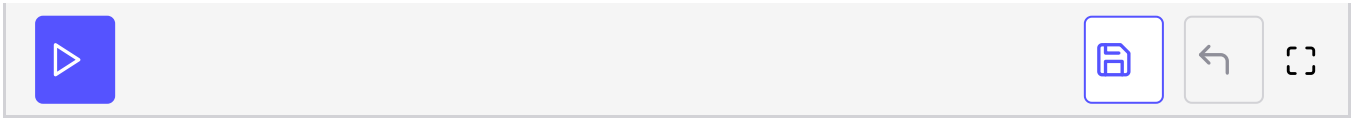
The if statement

Learn how if statements work in Python

Python's if statement is pretty easy to use. Let's spend a few minutes looking at some examples to better acquaint ourselves with this construct.

```
if 2 > 1:
    print("This is a True statement!")
    #This is a True Statement!
```

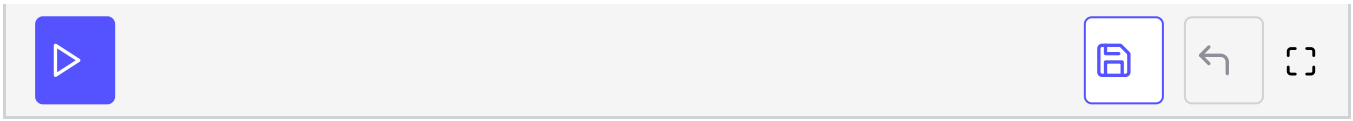


This conditional tests the "truthfulness" of the following statement: 2 > 1. Since this statement evaluates to True, it will cause the last line in the example to print to the screen or **standard out** (stdout).

> **Python Cares About Space**
>
> The Python language cares a lot about space. You will notice that in our conditional statement above, we indented the code inside the **if** statement four spaces. This is very important! If you do not indent your blocks of code properly, the code will not execute properly. It may not even run at all.
>
> Also, do **not** mix tabs and spaces. The recommended number of spaces to indent a block of code is four. You can actually indent your code any number of spaces as long as you are consistent. However, the 4-space rule is one that is recommended by the Python Style Guide and is the rule that is followed by the Python code developers.

Let's look at another example:

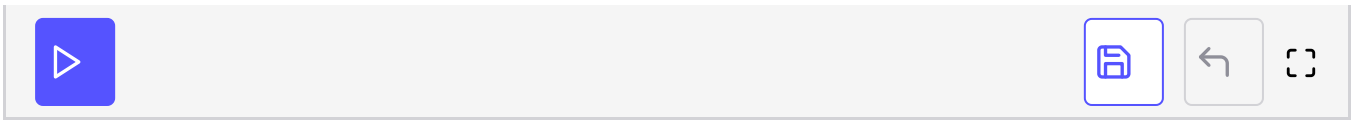
```
var1 = 1
var2 = 3
if var1 > var2:
    print("This is also True")
```



In this one, we compare two variables that translate to the question: Is `1 > 3`? Obviously one is not greater than three, so it doesn't print anything. But what is we wanted it to print something? That's where the **else** statement comes in. Let's modify the conditional to add that piece:

```
var1 = 1
var2 = 3

if var1 > var2:
    print("This is also True")
else:
    print("That was False!")
```



If you run this code, it will print the string that follows the **else** statement. Let's change gears here and get some information from the user to make this more interesting. In Python 2.x, you can get information using a built-in called **raw_input**. If you are using Python 3.x, then you'll find that raw_input no longer exists. It was renamed to just **input**. They function in the same way though. To confuse matters, Python 2.x actually has a built-in called **input** as well; however it tries to execute what is entered as a Python expression whereas raw_input returns a string. Anyway, we'll be using Python 2.x's **raw_input** for this example to get the user's age.

```
# Python 2.x code
value = raw_input("How much is that doggy in the window? ")
value = int(value)

if value < 10:
    print("That's a great deal!")
elif 10 <= value <= 20:
    print("I'd still pay that...")
else:
    print("Wow! That's too much!")
```

```
    print( 'Wow! That's too much!' )
```

Let's break this down a bit. The first line asks the user for an amount. In the next line, it converts the user's input into an integer. So if you happen to type a floating point number like **1.23**, it will get truncated to just **1**. If you happen to type something other than a number, then you'll receive an exception. We'll be looking at how to handle exceptions in a later chapter, so for now, just enter an integer.

In the next few lines, you can see how we check for 3 different cases: less than 10, greater than or equal to 10 but less than or equal to 20 or something else. For each of these cases, a different string is printed out. Try putting this code into IDLE and save it. Trying running this code a few times with different inputs to see how it works.

You can add multiple **elif** statements to your entire conditional. The **else** is optional, but makes a good default.