A Note on Scope and Globals

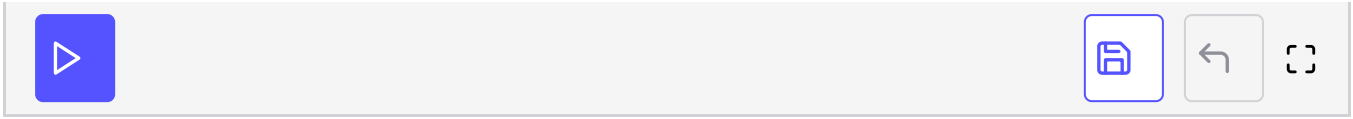
Let's learn about how scope works in python

Python has the concept of **scope** just like most programming languages. Scope will tell us when a variable is available to use and where. If we define the variables inside of a function, those variables can only be used inside that function. Once that function ends, they can no longer be used because they are **out of scope**. Let's take a look at an example:

```
def function_a():
    a = 1
    b = 2
    return a+b

def function_b():
    c = 3
    return a+c

print( function_a() )
print( function_b() )
```



If you run this code, you will receive the following error:

```
NameError: global name 'a' is not defined
```

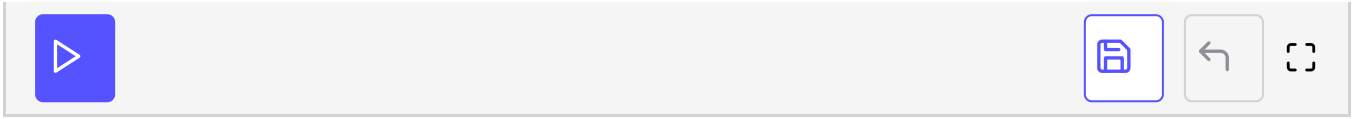
This is caused because the variable **a** is only defined in the first function and is not available in the second. You can get around this by telling Python that **a** is a **global** variable. Let's take a look at how that's done:

```
def function_a():
    global a
    a = 1
    b = 2
    return a+b

def function_b():
```

```
    c = 3
    return a+c


print(function_a())
print(function_b())
```



This code will work because we told Python to make **a** global, which means that that variable is available everywhere in our program. This is usually a bad idea and not recommended. The reason it is not recommended is that it makes it difficult to tell when the variable is defined. Another problem is that when we define a global in one place, we may accidentally redefine its value in another which may cause logic errors later that are difficult to debug.