Local Imports

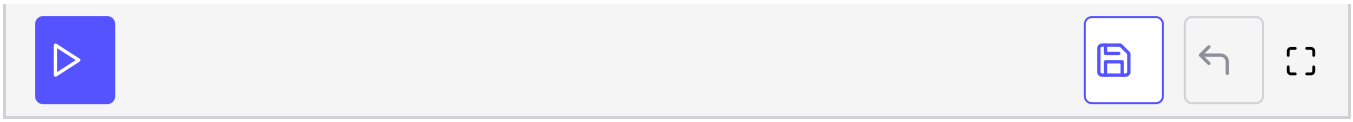
A local import is when you import a module into local scope. When you do your imports at the top of your Python script file, that is importing the module into your global scope, which means that any functions or methods that follow will be able to use it. Let's look at how importing into a local scope works:

```
import math
import sys  # global scope

def square_root(a):
    # This import is into the square_root functions local scope
    import math
    return math.sqrt(a)

def my_pow(base_num, power):
    return math.pow(base_num, power)

if __name__ == '__main__':
    print(square_root(49))
    print(my_pow(2, 3))
```



Here we import the sys module into the global scope, but we don't actually use it. Then in the square_root function we import Python's math module into the function's local scope, which means that the math module can only be used inside of the square_root function. IF we try to use it in the my_pow function, we will receive a NameError. Go ahead and try running the code to see this in action!

One of the benefits of using local scope is that you might be using a module that takes a long time to load. If so, it might make sense to put it into a function that is called rarely rather than your module's global scope. It really depends on what you want to do. Frankly, I've almost never used imports into the local scope, mostly because it can be hard to tell what's going on if the imports are scattered all over the module. Conventionally, all imports should

be at the top of the module after all.