map

The **map** built-in also takes a function and an iterable and return an iterator that applies the function to each item in the iterable. Let's take a look at a simple example:

```
def doubler(x):
    return x * 2

my_list = [1, 2, 3, 4, 5]
for item in map(doubler, my_list):
    print(item)

#2
#4
#6
#8
#10
```

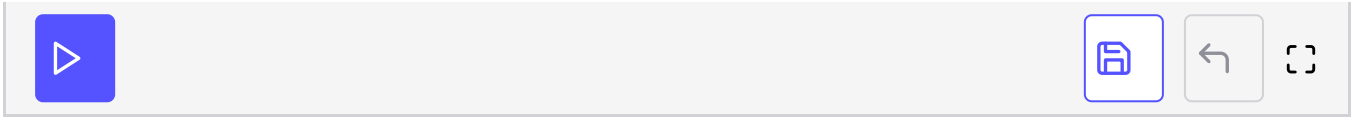
The first thing we define is a function that will double whatever is passed to it. Next we have a list of integers, 1-5. Finally we create a for loop that loops over the iterator that is returned when we call map with our function and list. The code inside the loop will print out the results.

The map and filter functions basically duplicate the features of generator expressions in Python 3. In Python 2, they duplicate the functionality of list comprehensions. We could shorten the code above a bit and make it return a list like so:

```
print (list(map(doubler, my_list)))
#[2, 4, 6, 8, 10]
```

But you can do the same thing with a list comprehension:

```
print ([doubler(x) for x in my_list])
#[2, 4, 6, 8, 10]
```



So it’s really up to you which you want to use.