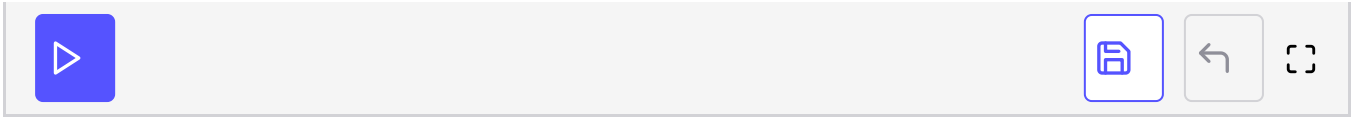
A Simple Function

A function is a block of code that begins with the Python keyword **def** followed by the actual name of the function. A function can accept zero or more arguments, keyword arguments or a mixture of the two. A function always returns something. If you do not specify what a function should return, it will return **None**. Here is a very simple function that just returns a string:

```
def a_function():
    """A pretty useless function"""
    return "1+1"

if __name__ == "__main__":
    value = a_function()
    print(value)
```



All we do in the code above is call the function and print the return value. Let's create another function:

```
def another_function(func):
    """
    A function that accepts another function
    """
    def other_func():
        val = "The result of %s is %s" % (func(),
                                          eval(func())
                                          )

        return val
    return other_func
```

This function accepts one argument and that argument has to be a function or callable. In fact, it really should only be called using the previously defined function. You will note that this function has a nested function inside of it that we are calling **other_func**. It will take the result of the function passed to it,

evaluate it and create a string that tells us about what it did, which it then returns. Let's look at the full version of the code:

```
def another_function(func):
    """
    A function that accepts another function
    """

    def other_func():
        val = "The result of %s is %s" % (func(),
                                          eval(func())
                                          )
        return val
    return other_func

def a_function():
    """A pretty useless function"""
    return "1+1"

if __name__ == "__main__":
    value = a_function()
    print(value)
    decorator = another_function(a_function)
    print(decorator())
```

This is how a decorator works. We create one function and then pass it into a second function. The second function is the **decorator** function. The decorator will modify or enhance the function that was passed to it and return the modification. If you run this code, you should see the following as output to stdout:

```
1+1
The result of 1+1 is 2
```

Let's change the code slightly to turn **another_function** into a decorator:

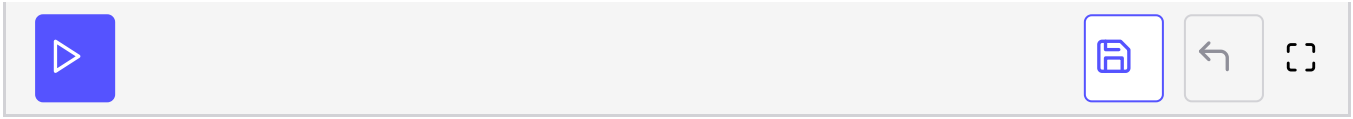
```
def another_function(func):
    """
    A function that accepts another function
    """

    def other_func():
        val = "The result of %s is %s" % (func(),
                                          eval(func())
                                          )
        return val
    return other_func

@another_function
```

```
def a_function():
    """A pretty useless function"""
    return "1+1"

if __name__ == "__main__":
    value = a_function()
    print(value)
```



You will note that in Python, a decorator starts with the <**@*>* symbol followed by the name of the function that we will be using to “decorate” our regular with. To apply the decorator, you just put it on the line before the function definition. Now when we call **a_function**, it will get decorated and we’ll get the following result:

```
The result of 1+1 is 2
```

Let’s create a decorator that actually does something useful.