Decorators 2

Debugging Code

@debug decorator will print the arguments a function is called with as well as its return value every time the function is called:

```
import functools

def debug(func):
    """Print the function signature and return value"""
    @functools.wraps(func)
    def wrapper_debug(*args, **kwargs):
        # reading one dimentin arguments
        args_repr = [repr(a) for a in args]                      # 1
        # reading 2 dimentions arguments --> keys, values
        kwargs_repr = [f"{k}={v!r}" for k, v in kwargs.items()]  # 2
        signature = ", ".join(args_repr + kwargs_repr)           # 3
        print(f"Calling {func.__name__}({signature})")
        value = func(*args, **kwargs)
        print(f"{func.__name__!r} returned {value!r}")           # 4
        return value
    return wrapper_debug
```

Test the decorator

```
@debug
def test(x,y,z):
    return 100
test(1,2,3)
```