Programming Exercise 3

Decorators

In this assignment you will create two decorators. I suggest you follow 2 steps.

Step 1

Recall the **trace** decorator from the Chapter 9 slide set. It counts the number of times a decorated function is executed. Modify **trace** and (rename it as **track**) so that it 1) caches the result of all function calls, so that the decorated function is never called twice with the same arguments, and 2) counts the total number of function calls. Tracking the count of calls is accomplished by using a *function attribute*. To illustrate how this works, consider the recursive function that computes the **n**th Fibonacci number.

```
def fib(n):
    return n if n in (0,1) else fib(n-1) + fib(n-2)
```

This implementation makes many repeated function calls on the same argument. If we decorate it as follows:

```
@track
def fib(n):
    return n if n in (0,1) else fib(n-1) + fib(n-2)
```

then calling **print(fib(10), 'count =' ,fib.count)** yields the following:

```
55, calls = 177
```

Very wasteful! Your modified track decorator should print the following:

```
(1,){} found in cache
(2,){} found in cache
(3,){} found in cache
(4,){} found in cache
(5,){} found in cache
(6,){} found in cache
(7,){} found in cache
(8,){} found in cache
```

The eight trace statements are printed whenever a repeated call is attempted on the arguments shown. Note that both positional and keyword arguments function as the cache dictionary key. This is done with the following statement:

```
key = str(args) + str(kwargs)
```

using the customary parameter names for decorators.

Step 2

Write a **logging decorator** that saves all function calls and their results in the file "log.txt". This requires opening the file for appending:

```
logfile = open('log.txt','a')
```

You will need the wrapper function to open the file for each function call, write the information, and close the file.

When you decorate **fib** with only the **log** decorator, log.txt should contain the output below:

```
@log
def fib(n):
 return n if n in (0,1) else fib(n-1) + fib(n-2)

print(fib(10))   # Prints 55
```

The file *log.txt* contains:

```
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((1,){}) = 1
fib((3,){}) = 2
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((4,){}) = 3
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((1,){}) = 1
fib((3,){}) = 2
fib((5,){}) = 5
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((1,){}) = 1
fib((3,){}) = 2
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((4,){}) = 3
fib((6,){}) = 8
fib((1,){}) = 1
```

```
fib((0,){}) = 0
fib((2,){}) = 1
fib((1,){}) = 1
fib((3,){}) = 2
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((4,){}) = 3
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((1,){}) = 1
fib((3,){}) = 2
fib((5,){}) = 5
fib((7,){}) = 13
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((1,){}) = 1
fib((3,){}) = 2
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((4,){}) = 3
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((1,){}) = 1
fib((3,){}) = 2
fib((5,){}) = 5
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((1,){}) = 1
fib((3,){}) = 2
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((4,){}) = 3
fib((6,){}) = 8
fib((8,){}) = 21
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((1,){}) = 1
fib((3,){}) = 2
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((4,){}) = 3
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
```

```
fib((1,){}) = 1
fib((3,){}) = 2
fib((5,){}) = 5
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((1,){}) = 1
fib((3,){}) = 2
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((4,){}) = 3
fib((6,){}) = 8
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((1,){}) = 1
fib((3,){}) = 2
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((4,){}) = 3
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((1,){}) = 1
fib((3,){}) = 2
fib((5,){}) = 5
fib((7,){}) = 13
fib((9,){}) = 34
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((1,){}) = 1
fib((3,){}) = 2
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((4,){}) = 3
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((1,){}) = 1
fib((3,){}) = 2
fib((5,){}) = 5
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((1,){}) = 1
fib((3,){}) = 2
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
```

```
fib((4,),{}) = 3
fib((6,),{}) = 8
fib((1,),{}) = 1
fib((0,),{}) = 0
fib((2,),{}) = 1
fib((1,),{}) = 1
fib((3,),{}) = 2
fib((1,),{}) = 1
fib((0,),{}) = 0
fib((2,),{}) = 1
fib((4,),{}) = 3
fib((1,),{}) = 1
fib((0,),{}) = 0
fib((2,),{}) = 1
fib((1,),{}) = 1
fib((3,),{}) = 2
fib((5,),{}) = 5
fib((7,),{}) = 13
fib((1,),{}) = 1
fib((0,),{}) = 0
fib((2,),{}) = 1
fib((1,),{}) = 1
fib((3,),{}) = 2
fib((1,),{}) = 1
fib((0,),{}) = 0
fib((2,),{}) = 1
fib((4,),{}) = 3
fib((1,),{}) = 1
fib((0,),{}) = 0
fib((2,),{}) = 1
fib((1,),{}) = 1
fib((3,),{}) = 2
fib((5,),{}) = 5
fib((1,),{}) = 1
fib((0,),{}) = 0
fib((2,),{}) = 1
fib((1,),{}) = 1
fib((3,),{}) = 2
fib((1,),{}) = 1
fib((0,),{}) = 0
fib((2,),{}) = 1
fib((4,),{}) = 3
fib((6,),{}) = 8
fib((8,),{}) = 21
fib((10,),{}) = 55
```

Now decorate **fib** with both **track** and **log:**

```
@track
@log
def fib(n):
   return n if n in (0,1) else fib(n-1) + fib(n-2)
```

```
print(fib(10),'count =',fib.count)
```

The output should then be

```
(1,){} found in cache
(2,){} found in cache
(3,){} found in cache
(4,){} found in cache
(5,){} found in cache
(6,){} found in cache
(7,){} found in cache
(8,){} found in cache
55, calls = 11
```

from the **trace** decorator, and log.txt should contain:

```
fib((1,){}) = 1
fib((0,){}) = 0
fib((2,){}) = 1
fib((3,){}) = 2
fib((4,){}) = 3
fib((5,){}) = 5
fib((6,){}) = 8
fib((7,){}) = 13
fib((8,){}) = 21
fib((9,){}) = 34
fib((10,){}) = 55
```

Submit your *log.txt* and *dec.py* for the latter, decorated case.