Assignment 2

Due February 12, 2017

Submissions are due by 11:59PM on the specified due date. Submissions may be made on the Blackboard course site under the Assignments tab. Late submissions will be accepted up to two days late with a 10% penalty for each day (24 hours).

Make sure your name and FSUID are in a comment at the top of the file.

In this assignment, you may only use the *sys*, *strings*, and *time* modules. The *__future__* and *copy* modules are also available for use.

1 mixed_cipher.py (60 points)

Your task is to create a Python module called mixed_cipher. You may have had some experience with writing programs that implement simple Caesar ciphers. In this assignment, we'll be implementing a small program which takes in a filename as a command-line argument and then performs simple substitution on the file contents with a mixed alphabet. A mixed alphabet can be created by prompting the user for the keyword, removing repeated letters in the keyword, and then writing the remaining letters of the alphabet in the usual order. For example, if my keyword is "computer", then I have the following:

```
Plaintext: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Ciphertext: COMPUTERABDFGHIJKLNQSVWXYZ
```

Your program should output the encrypted text to the screen. Here's an example execution. Let's say I have a file called mixed_test.txt which has the following contents:

```
This is a file. This file has some WORDS in it.
```

A sample run of mixed cipher.py with this test file would produce the following output. Note that capitalization and punctuation are kept as-is in the encrypted text.

```
$ python mixed_cipher.py mixed_test.txt
Please enter a keyword for the mixed cipher: Motherboard
Plaintext: abcdefghijklmnopqrstuvwxyz
Ciphertext: motherbadcfgijklnpqsuvwxyz
Sadq dq m rdge. Sadq rdge amq qkie WKPHQ dj ds.
```

Assignment 2 continues on the next page.

2 log_decorator.py (40 points)

Your goal is to create a decorator that extends a function with some logging statements.

The logging decorator must be called **log** and must be defined in the module log_decorator.py. The decorator may optionally take a single argument that indicates the file to which the statements should be appended. If no argument is provided, the statements are written to stdout. If the filename provided cannot be opened for whatever reason, logging should be directed to stdout.

The logging decorator should extend the function by printing the following items:

1. The name of the function being called. (Hint: Make use of the __name__ attribute available on every function object).
2. A list of the arguments supplied to the function as well as the type of the argument. If no arguments are supplied, you must report this. You might want to make use of __name__ here as well!
3. The output of the function.
4. The execution time of the function (elapsed wall clock time) in seconds. The time should be rounded to 5 decimal places.
5. The return value and the type of the return value. Note that even when multiple values are returned, the return value is a single tuple object. If there is no return value, you must report this.

To demonstrate the behavior of the log decorator, take a look at the following example log.py, which defines (but does not show!) the log decorator:

```python
@log()
def factorial(*num_list):
    results = []
    for number in num_list:
        res = number
        for i in range(number-1,0,-1):
            res = i*res
        results.append(res)
    return results

@log("logger.txt")
def waste_time(a, b, c):
    print("Wasting time.")
    time.sleep(5)
    return a, b, c

@log("logger.txt")
def gcd(a, b):
    print("The GCD of", a, "and", b, "is ", end="")
    while a!=b:
        if a > b:
            a -= b
        else:
            b -= a
    print(abs(a))
    return abs(a)


@log()
def print_hello():
    print("Hello!")

@log(10)
def print_goodbye():
    print("Goodbye!")

if __name__ == "__main__":
    factorial(4, 5)
    waste_time("one", 2, "3")
    gcd(15,9)
    print_hello()
    print_goodbye()
```

The results of executing this module are given below:

```
$ python log.py
*************************************************
Calling function factorial.
Arguments:
  - 4 of type int.
  - 5 of type int.
Output:
Execution time: 0.00002 s.
Return value: [24, 120] of type list.
************************************************

*************************************************
Calling function print_hello.
No arguments.
Output:
Hello!
Execution time: 0.00002 s.
No return value.
************************************************

*************************************************
Calling function print_goodbye.
No arguments.
```

```
Output:
Goodbye!
Execution time: 0.00037 s.
No return value.
************************************************

$ more logger.txt
*************************************************
Calling function waste_time.
Arguments:
  - one of type str.
  - 2 of type int.
  - 3 of type str.
Output:
Wasting time.
Execution time: 5.00539 s.
Return value: ('one', 2, '3') of type tuple.
************************************************

*************************************************
Calling function gcd.
Arguments:
  - 15 of type int.
  - 9 of type int.
Output:
The GCD of 15 and 9 is 3
Execution time: 0.00010 s.
Return value: 3 of type int.
************************************************
```