COP 4521: Spring 2021
Homework - 2

Total Points: 10
Due: Friday 01/29/2021, 11:59pm

1 Objective

The objective for this assignment is to make sure

- You are familiar with native Python and Python syntax.
- You can write small programs that demonstrate Python's capabilities.
- You can use Python to solve small problems.
- You are comfortable with Python's Object Oriented Programming concepts.
- You can do some research on a particular problem and learn about its general background and then use Python to solve the problem. You will be expected to do this at any Software Engineering job, and it is good to practice while you're still in school.

2 log_decorator.py (5 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:

- The name of the function being called. (Hint: Make use of the __name__ attribute available on every function object).
- 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!
- The output of the function.
- The execution time of the function (elapsed wall clock time) in seconds. The time should be rounded to 5 decimal places.
- 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:

```
@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.
**********************************************
```

3 oop.py (5 points)

Define a student class. A student has the following attributes: firstname, lastname, gender which can be male or female, status which can be equal to freshman, sophomore, junior, and senior and gpa.
Then define the following methods for the student class.

- The show_myself method will simply print all the attribute variables when called upon the object. This method has no input arguments.
- The study_time method will increment the gpa of the student according to the following formula: ' gpa = gpa + log(study_time)'. The only input argument of this method is the variable study_time. In addition make sure that the gpa variable never exceeds 4.0 even if the student studies for a very long time.

Create 5 student objects and store the objects in a list called student_list. The five students are: Mike Barnes, Jim Nickerson, Jack Indabox, Jane Miller and Mary Scott. Mike is a freshman, Jim a sophomore, Jack a junior, Jane and Mary are seniors. Their respective GPAs are: 4, 3, 2.5, 3.6 and 2.7. Make sure you assign the gender when you instantiate the objects.
Then call the show_myself method on all of them. I suggest you use a loop for making the objects and a separate loop for showing the objects.

4 Submission

- For this homework, Make sure your python code can be run on Python version ¿= 3.8.
- Please name your file `log_decorator.py and oop.py`
- Please add your name and FSUID as a comment at the top of your programs.
- Please turn in the program through Canvas.
- We are aware that solutions to these problems might exist on the internet. Please turn in your solution to the problem, and not someone else's. We will be running your submission through plagiarism detection software.
- You are also responsible for making sure your submission on Canvas contains the right file and that the file has not been corrupted in any way.