CS 115 Lab 1: Function Practice

In this lab, you will write a few simple functions, with a focus on covering the basics of Python, as well as a few convenient built-in functions. The goal is to get familiar with a variety of data types and arithmetic operations.

In this lab and onwards, **each function needs to include a docstring that describes *what* that function does.**

For each function, take care to include a docstring explaining what it does. Also, fill in your name and the Stevens Honor Code at the top of the file.

Task 1: Sums of Consecutive Numbers

Suppose that you have integers x and y. An occasionally useful quantity is the sum of all numbers between x and y, which is what you will implement in this problem.

Fill out the provided `consecutive_sum` function's body, such that, if given two numbers x and y, it returns the sum of all numbers that are at least x and less than y. In particular, observe that x is included in the sum but y is not.

You may assume that the given numbers will always satisfy $0 \leq x < y$. Furthermore, x and y will both be `int`s, and the output should be one as well.

You might find the following mathematical identity useful:

$$x + (x+1) + (x+2) + \cdots + (y-1) = \frac{y+x-1}{2} \cdot (y-x)$$

CA Check-In

After you complete the above task, talk to the CA in your section.

1. Explain what `consecutive_sum` does in your own words
2. Show the CA the docstring for the function to get their feedback.

Task 2: Comparing Outer Letters

For your next task, you will be working with strings. A palindrome is a string which reads the same forwards and backwards, and this task and the next both rely on this concept.

First, implement the function `same_ends`. This takes in a string consisting of a word (or several), and returns whether or not the first and last letter of the string are the same (in the form of a `bool`).

To complicate matters, however, we add an additional requirement: This check should not depend on case. If a word starts and ends in the same letter but with different capitalization, `same_ends` should return `True`. To accomplish this, there are a number of useful methods for working with strings. You can find a detailed list at https://docs.python.org/3/library/stdtypes.html#string-methods, although we recommend using `lower` or `casefold`, which you invoke on a string `s` by doing `s.lower()`. You can assume that the text is only comprised of letters, numbers, and spaces as long as that is written in the function's docstring.

Task 3: Verifying Palindromes

For the third task, your goal is to implement a function `is_palindromic` that checks whether its argument is a palindrome or not. In other words, given a string `word`, `is_palindromic(word)` should return `True` if and only if `word` is the same as its reversal. **You should not use loops for this question even if you know how to use them**.

HINT: You might want to first figure out how to reverse the word, so that you can compare it to its original value. Furthermore, your solution should also work if `word` is a tuple instead of a string.

Task 4: Movie Ticket Prices

The final task is to implement a function `get_ticket_price` that returns the price of a movie ticket **as an** `int` based on the customer's age. The ranges are inclusive, so the range 3-12 years includes 3 year olds and 12 year olds for example.

- Under 3 years: $0
- 3-12 years: $10
- 13-64 years: $15
- 65 and older: $12

Use `if`, `elif`, and `else` to determine the correct price. Pay extra attention in the docstring to list the following assumptions about input and output:

- The outputted price is in dollars
- Age is assumed to be a positive number

You may assume that $age >= 0$ as long as you write that assumption in the docstring.

Testing

As you are developing your code, it is important to actively test it, to ensure that it works. To help with this, we have provided you with a testing file.

To use this testing file, we recommend the following steps:

1. Save this testing file in the same directory/folder where you have your solution to the lab.
2. Run the testing file the same way you would run any other Python program.

This file will then run through several test cases, checking to see if your code matches the expected behaviour. After it finishes, it should output a summary. If it outputs OK (like below), that means that your code passes all the test cases.

If you find that any tests are failing, please read it to fix your code. Ask a CA for help if you're stuck!

```
............
----------------------------------------------------------------------
Ran 12 tests in 0.000s

OK
```

Otherwise, if it indicates that some test cases failed, you will get an output like below. In this case, you should correct your code before you submit it on Canvas.

```
........F.EF
[ERRORS OMITTED FOR SPACE]
----------------------------------------------------------------------
Ran 12 tests in 0.001s

FAILED (failures=2, errors=1)
```

Submission Instructions

For this lab/homework, you must submit the following files, with exactly the following names:

- `lab1.py`
- `lab1_report.txt`

Remove any code outside imports and function definitions. For example, any function calls to test your code should be removed from `lab1.py` before submission.

Remember to put your name and the Stevens Honor Pledge on every file you submit! Failure to put the pledge can result in marks being deducted, with repeat penalties potentially getting stricter between assignments.

Submissions must be handled through Gradescope (accessible through the homework page on Canvas). Grades will be released through Gradescope once the homework is graded.

Hints and Suggestions

- A good docstring includes the expected input of the function, expected output of the function, and the relationship between the input and output.
- You may choose to add types to your docstring but it is not required

Notes and Reminders

- **Remember to include docstrings!** If your submission does not have docstrings, or has docstrings that are not related to your code, you will not get credit for them.
- We have provided a test file for you to test your solutions. By running this test file, you should be able to verify that your files are named properly, and that your code works correctly.
- Test frequently! Try out many different cases to verify that your code runs. Our provided test file may not be comprehensive, and so you should ensure to test your code on many other cases as well.
- Code that doesn't compile (throws a `SyntaxError` on being run) will receive a 0 for the assignment! Make sure you submit code that can be run, even if it does not pass all the test cases.
- Make sure to include your name and pledge.
- Make sure to remove any code outside function definitions!

Rubric

Category	**Description**	**Points**
Name	For stating your name on every file for submission.	5
Pledge	For writing the full Stevens Honor Pledge on every file for submission.	5
Check-In	For performing the CA Check-In during the lab.	20
`consecutive_sum`	For correctly implementing `consecutive_sum`.	20
`same_ends`	For correctly implementing `same_ends`.	10
`is_palindromic`	For correctly implementing `is_palindromic`.	15
`get_ticket_price`	For correctly implementing `get_ticket_price`.	20
Report Questions	For filling out the questions on the report.	5
Total		100