PA03: Introduction to Python Functions

University of Virginia Department of Computer Science
CS 1112 - Intro to Programming
Fall 2024

Due by: 11:00pm on Wednesday, October 2, 2024

Contents

Objective: This assignment introduces students to writing functions in Python, including defining functions using the `def` keyword, declaring a function's parameters, returning values from functions, and calling functions within other functions. By the end of this assignment, students should be able to understand how to declare a function, how to use the function parameter(s), how to return a value from that function, and how to call that function, including from within other functions.

TL;DR

1. Implement **seven (7) functions** (remember to name your file **my_functions.py**):
   - `is_even(given_value)`
   - `add_string_contents(given_str)`
   - `subtract_string_contents(given_str)`
   - `operate_on_string(given_str)`
   - `right_triangle_area(base, height)`
   - `trapezoid_area(first_base, second_base, height)`
   - `fahrenheit_to_kelvin(fahrenheit_temp)`
2. Use **Python multi-line docstrings** to comment each function thoroughly. Include additional **in-line comments** within your code (doesn't have to be every line). Remember, you **WILL** be graded on both these aspects for this assignment. *Failure to include both multi-line docstring comments and in-line comments will result in some loss of points.*
3. If you create your own **variables** make sure they are given a **proper name** that makes sense in the context
4. **Test your function solutions** in a file **other than the one you submit.** See next section for more details.
5. **Submit** your file **my_functions.py** to Gradescope by the **deadline** mentioned above.

You must match our file name, my_functions.py, ***exactly*** .
If you do not match our filename exactly, our autograder will be unlikely to find it, and it will therefore be unable to generate useful feedback to aid you on your submission.

1 Task

You're starting a new class off right when you notice, *gee, it seems like I'm being asked to do a lot of busywork.* It's almost like you could automate your homework problems instead of doing everything by hand! At this moment, you remember that you can use *functions* to do **just that**!

For this assignment, you'll need to implement seven functions in a file called `my_functions.py` (named exactly). You must use **multi-line Python docstrings** to comment each function thoroughly. You must also use **in-line comments** to document the code inside of your functions (doesn't have to be every line.) Please pay attention to the function names, number and type of parameters (for example, the `right_triangle_area()` function is defined with two (2) parameters: a "base" and a "height"), and what each function should return:

1.1 is_even(given_value) (10 points)

This function should take any number as an argument called `given_value` which may be positive or negative. If `given_value` is even, return `0`. If `given_value` is odd, return `1`. Your function must return the **integer 0 or 1** only (not strings). (*Hint: how do I check if a number is even or odd?*).

This function should work for any (real) integer. You may assume that our tests will not try to call your function with a string or other nonsensical input, nor will our tests try to call your function with a `float` argument.

```
is_even(given_value)
```

1.2 add_string_contents(given_str) (10 points)

Given a string `given_str` that consists of numeric characters, add up each number character in the string and return a numeric total as the result.
Example usage:

```
add_string_contents("123456")  # should return the number 21
# (21 = 1 + 2 + 3 + 4 + 5 + 6)
add_string_contents("723")  # should return the number 12 = 7 + 2 + 3
add_string_contents("32091")  # should return the number 15
# (15 = 3 + 2 + 0 + 9 + 1)
```

You may assume that our tests will pass in strings that **are not empty** and that consist of only numbers; you do not need to try and check for letters in the strings which can't be converted to numbers (that is, we will not test your code with a string like "123x7"). You may also assume that we will not pass in a minus sign ("-") as part of the string, so you do not need to check for this either.

1.3 subtract_string_contents(given_str) (10 points)

Given a string `given_str` that consists of numeric characters, take the first character and subtract every following character in the string from it, returning a numeric difference as the result.
Example usage:

```
subtract_string_contents("123456")  # should return a number -19
# (-19 = 1 - 2 - 3 - 4 - 5 - 6)
subtract_string_contents("723")  # should return 2 = 7 - 2 - 3
subtract_string_contents("32091")  # should return -9
# (-9 = 3 - 2 - 0 - 9 - 1)
```

The same guidance and assumptions given for `add_string_contents` apply here: you do not need to check for letters in the string, nor do you need to check for minus signs ("-"). Strings we use to test your function **will not be empty** and, for this function, will be at least three (3) characters long.

```
subtract_string_contents(given_str)
```

1.4 operate_on_string(given_str) (20 points)

Given a string `given_str`:

- If the **length** of the string is *even* (**NOT** if the number is even), then *add up* the contents of `given_str` and return that numeric result. **You must use the `is_even` function you previously wrote to check this.** (That is, call `is_even` inside this function.)
- If the **length** of the string is *odd* (**NOT** if the number is odd), then take the first character of `given_str` and *subtract* all following characters from that first character, returning the numeric result.

(Hint: you have functions to do *all three of these tasks* already – see the three previous functions (1.1, 1.2, and 1.3)! It should be a matter of *choosing* what to do based on input).

1.5 right_triangle_area(base, height) (10 points)

Given numeric arguments for `base` and `height`, return the numeric value for the area of a right triangle using the following formula:

$$A = \frac{1}{2}bh, b = \text{base}, h = \text{height} \tag{1}$$

You do not need to check whether the arguments are numbers or not; we will only test with numeric arguments.

1.6 trapezoid_area(first_base, second_base, height) (10 points)

Given numeric arguments for `first_base`, `second_base`, and `height`, return the numeric value for the area of a trapezoid using the following formula:

$$A = \frac{(b_1 + b_2)h}{2}, b_1 = \text{first base}, b_2 = \text{second base}, h = \text{height} \tag{2}$$

Like with `right_triangle_area`, you may assume that we will test only with numeric arguments, so you do not need to check your input.

1.7 fahrenheit_to_kelvin(fahrenheit_temp) (10 points)

Given a temperature argument in degrees Fahrenheit, do the following (pseudocode):

1. Convert from Fahrenheit to Celsius using the formula $C = \frac{5}{9}(F - 32)$
2. Convert from Celsius to Kelvin using the formula $K = C + 273$
3. Return the Kelvin temperature as a numeric value.

You may assume that we will only give you physically sensible arguments (you don't need to worry about your function returning a result below absolute zero, which wouldn't make sense). Be careful with the order of operations here, especially in the Fahrenheit-to-Celsius conversion! *Use the pseudocode above to write your solution in only* ***one*** *function.*

1.8 Hints and Tips

As we have mentioned in class, **strings** in Python can be processed and operated on character-by-character (in technical terms, they are "iterable"). You can do this in one of two ways:

- Use a for loop with the upper bound being `len(string)`:

```
for i in range(0, len(string)):
    # access individual characters with string[i] (where i is
    # the index into the string) then, operate accordingly
```

- Use a special loop called a "for-in" loop. The `in` keyword in Python is used in expressions of the form `x in y` to deal with individual elements `x` of a collection `y` one at a time.

```
for character in string:
    # "character" is one individual character in "string"
    # access and operate on a character by interacting with "character"
```

Please make sure your submission conforms to the following guidelines:

- Match our function names and the number of parameters they expect *exactly*.
- **Do not print anything** within your functions. Instead, **return** values *from* your function.
  - We will be able to get your output without `print` statements; in fact, including `print` statements may throw off the autograder and cause your submission to be graded incorrectly.
- **Do not "hard-code"** your solutions. They should be *general* solutions to the problem we present; rather than trying to fit your code to specific test cases, we are pushing you to develop general logic you can use in your code.

Our autograder will be very particular about these details. In addition to checking what your function returns against the expected value that we would get, if your function does not follow our format (or does unexpected things like print), our autograder may call it incorrectly, which is likely to produce errors. We do check for these and do our best to let you know the specific problem, though.

Keep the following in mind as you write your code, as you will be graded on these aspects:

- Don't forget your header at the top of your `.py` file (see section 2.1)
- Use appropriate variables names throughout your program
- Include in-line comments throughout your code to explain what you are doing (see example of in-line comment below:)

```
food = "apple"  # This is a string representing an apple
```

- Document all of your functions with **multi-line docstrings**. Note that the following is what we are referring to:

```
def func(first_param, second_param):
    '''
    First line - Description of the function - what does it do?
    :param first_param: what is it's data type,
        what does this parameter represent in the function?
    :param second_param: what is it's data type,
        what does this parameter represent in the function?
    :return: what is it's data type,
        what does the return value represent in the function?
    '''
    # BODY OF CODE HERE
```

(Remember, multi-line docstring comments are **not** a replacement for regular in-line comments. There is a place and need for both!)

- Cite your resources clearly or state you did not use any

Helpful guidance: do not wait until the last minute to start this assignment! Don't hesitate to reach out to your professor or TAs if you have any questions. We are here to help!

1.9 Testing Your Solution

Test your function solutions by doing something similar to the following in a file other than the one you submit (i.e., not in the `my_functions.py` file.):

```
import my_functions
# Test is_even() function with various numbers (both even and odd)
print("Testing is_even() with an even and an odd number:")
number1 = 22
number2 = 437
print(my_functions.is_even(number1))
print(my_functions.is_even(number2))

# Test add_string_contents() function with whatever you like, e.g. "123456"
print("Testing add_string_contents() with numeric strings
numeric_string = "123456"
print(my_functions.add_string_contents(numeric_string))
# Or you can hard-code the input "250" instead of passing in a variable:
print(my_functions.add_string_contents("250"))
...
# Write more tests, etc. for the other functions. Test with several
# different input values to check the robustness of your solutions
```

2 Submission and Collaboration Policy

2.1 Your Submission–Comment Header

Your `.py` file should contain the following information (header) at the **top** of your file:

```
# NAME: e.g. I. Lv. Sneks
# Email: e.g. ils3py@virginia.edu
# PA Number and Name: e.g. PA## - Name of the Assignment
# Resources Used (if applicable):
```

For **Resources**, include URLs to any online resources where you studied or reviewed code that is specific to these problems, including any physical textbooks you referenced. Include any other resources you used here. You **do not need to include** the names of TAs or Professors that you consulted with. You also do not need to reference course slides or materials provided for you (e.g. on Canvas). In fact, *please feel free* to refer to lecture slides and demos we have done in class!

If you **did not** reference any online/outside resources, please **keep this line in your header** *(do not delete it!)* but simply state that you "*Did not use any online, outside, or physical resources for this assignment.*"

2.2 Collaboration Policy for this Homework

Absolutely no collaboration with any fellow students (who are not current CS 1112 course TAs) is allowed. Your work must represent individual effort. You must write your own code: not *just* type it, but also compose it yourself as your own original work.

You must cite any and every source you consult other than those explicitly provided by the course itself. If you work with, obtain, or receive help from another source (websites, textbooks, tutors, online videos, etc.), nothing should be copied directly or retyped into the submitted solution. References must be documented in the header as previously stated.

2.2.1 Generative AI

We recognize that generative AI tools like Chat GPT-3.5 can be useful in situations like those where you may need that simple "spark" to get moving on an assignment. We kindly request that if you use generative AI, you use it for *guidance*, **not** to *complete the assignment for you*. As noted in the syllabus, use of generative AI must meet the following guidelines:

- **Transparency:** you must acknowledge that you used generative AI tools and indicate specifically what you used the tool(s) for.
- **Originality:** you are ultimately responsible for your own work, including understanding what you submit. Simply copying output from a generative AI tool without understanding it is highly discouraged.
- **Learning:** generative AI should be employed as a *learning tool*. We reserve the right to ask you questions about what you submit. If it appears you do not understand what you submitted, we may deduct points on your submission accordingly.
- **Academic Integrity:** you should use generative AI in good faith in a way that respects academic integrity, taking care to avoid plagiarism, improper attribution, and unauthorized use as stated above.

- **Guidance:** if in doubt, ask a professor or a TA for more questions on allowed use! We won't mind the question, and would much rather you discuss tricky cases with us before you submit than afterwards. We're here to help!

2.3 Gradescope

Submit your completed file, which **must be named `my_functions.py` exactly**, to Gradescope by **11:00pm on the date given at the top of this assignment (first page)**. If your file has a format we do not expect (e.g. `.txt`, `.docx`, or `.pdf`), our autograder will **not** be able to properly test your code. Recovering the contents of the file will be difficult since the file format will scramble the file contents from standard `.py` format, which is basically just plaintext.

Important reminder: Do not submit your separate testing file to Gradescope, just submit your main Python file with the functions that you wrote.

Helpful guidance: Do not wait until the last minute to start this assignment! Remember you can **submit multiple times** to Gradescope. Look at the feedback you receive and then go back and fix your code, then you can resubmit. There is NO penalty for submitting multiple times to Gradescope. If you face any difficulties or have questions, post on Piazza. Also, don't hesitate to reach out to your professor or teaching assistants (TAs) for guidance. We are here to help!