Goals

By the end of this assignment, you will have
  • practice with testing and debugging

Note

Where examples for testing are needed, be sure to include (1) normal input (2) examples that test all the paths through a function (3) boundary input, e.g., 0, 1, empty strings, etc. Functions do NOT need to check argument data types (argument types are assumed to be as given in the type contract for each function.

Include at least four (additional) test cases for ttriangle. Include at least six (additional) test cases for find_min_and_max and my_average.

(0) The following function is broken. The docstring describes what it is supposed to do.

def ttriangle(n):
    '''(int) -> None

    Prints a right triangle with n lines, where the first line prints 1 'T'
    and the last line prints n 'T's. If n is <= 0, do not print anything.
    None value is returned.

    For example, >>> ttriangle(6)
    T
    TT
    TTT
    TTTT
    TTTTT
    TTTTTT
    '''
    ct = 1
    while ct < n:
        print("T" * ct)

    return #None

a) Generate a series of examples to test ttriangle.

b) Fix the bug(s) in the current version of ttriangle. Comment any code you change.
The following function is broken. The docstring describes what it is supposed to do.

def find_min_and_max(values):
    '''(string) -> None

    Find the maximum and minimum values in a non-empty string of integers and print them.
    None value is returned.

    >>> find_min_and_max('45312')
    The minimum value is 1.
    The maximum value is 5.
    '''
    mmin = 0
    mmax = 0

    for value in values:
        if value > mmax:
            mmax = value
        if value < mmin:
            mmin = value

    print('The minimum value is', mmin)
    print('The maximum value is', mmax)

    return #None

a) Generate a series of examples to test find_min_and_max. Put them in the form of the example in the docstring.

b) Fix the bug(s) in the current version of find_min_and_max. Comment any code you change.

(2) Based on text ch. 15-6

Suppose you have a data set of customer satisfaction survey results, with product ratings from 1 (poor quality) to 9 (high quality). Zeros indicate no answer. Here is a function that computes and returns the average product rating from the data set, a function parameter of type string. Zeros do not count toward the average. For no usable data, return 0.
def my_average(dataset):
    """(string) -> float

    returns average of values in input string values,
    but zeros do not count at all

    >>> my_average('23')
    2.5
    >>> my_average('203')
    2.5
    ""
    count = 0
    total = 0
    for value in dataset:
        if value != '0':
            total += int(value)
            count += 1
    
    avg = total / count
    return avg

a) Generate a series of examples to test my_average. Put them in the form of the example in the docstring.

b) Fix the bug(s) in the current version of my_average. Comment any code you change.

(3) As Time Goes By with bugs

Oh, no – someone started playing around with working code without making a copy of the working version first. Now function minutesToYears is not returning the correct result.

Find and fix the bugs in minutesToYears and/or its auxiliary functions. The revised functions should work for all of the examples given in the docstrings.

Comment any changes you make in the code.
def minutesToHours(minutes):
    '''(number) -> float
    convert input minutes to hours; return hours
    >>> minutesToHours(60)
    1.0
    >>> minutesToHours(90)
    1.5
    >>> minutesToHours(0)
    0.0
    '''
    hours = minutes / 60
    hours = round(hours, 2)
    print(hours)

def hoursToDays(hours):
    '''(number) -> float
    convert input hours to days; return days
    >>> hoursToDays(24)
    1.0
    >>> hoursToDays(100)
    4.17
    >>> hoursToDays(0)
    0.0
    '''
    days = hours / 24
    return days

def daysToYears(days):
    '''(number) -> float
    convert input days to years; return years
    >>> daysToYears(365)
    1.0
    >>> daysToYears(100)
    0.27
    >>> daysToYears(0)
    0.0
    '''
    days = 365
    years = days / 365
    years = round(years, 2)
    return years
def minutesToYears(m):
    '''(int) -> float

    input number m minutes is converted to equivalent number of years. return years.
    call auxiliary functions to do each step

    >>> minutesToYears(525600)
    1.0
    >>> minutesToYears(5256000)
    10.0
    >>> minutesToYears(394200)
    0.75
    >>> minutesToYears(0)
    0.0
    '''

    minutesToHours(m)
    hoursToDays(h)
    daysToYears(d)

    return y
Grading Rubric

This project will be marked out of 35 points: 2 points for fixing the bug(s) in triangle and 4 points for generating good test cases; 4 points for fixing the bug(s) in find_min_and_max and 6 points for generating good test cases; 6 points for fixing the bug(s) in my_average and 6 points for generating good test cases; 7 points for fixing the bug(s) in minutesToYears and its auxiliary functions.

Getting Started

Write your program using an editor window, so you can save your file.

Examples are for testing your code. You do not need to turn in the results of code tests.

Programming style is important! Remember
• Include a docstring in every function
• Use whitespace between operators and operands
• Use descriptive variable names
• Add appropriate comments.

Finishing & submitting your work

When you have completed all of the problems, add additional comments to your code to make sure the functions for each problem are clear. Check the list of functions above to make sure that your Python file includes all of them. Do a final Save command to save the code in the editor window as a file with the name cis122project7.py.

To submit your project, login to Blackboard. From the menu on the left hand side of the screen, choose "Projects". In the projects folder, choose "Submit Bugs Bugs Bugs". In Section 2 of the page that is displayed, scroll down to "Attach file" and choose "Browse My Computer". Locate cis122project7.py (the file you just created), and double click on it. The file name will appear on the "Attached Files" list.

At the bottom of Section 2, you will see a "Comments" window. This is where you credit all of the sources of any help you may have received on this assignment, including your partner if you are working in a programming pair. This is also the place to include any feedback you may have about the assignment and/or any remaining questions you may have.

Scroll down to Section 3 and hit the "Submit" button. You may re-submit your project up until the project deadline. Only the final submission will be graded.