CS 65
Fall 2025

Lab 7: Value Returning Functions

Due: Saturday, November 8th
Total points: 4

In this lab, you are going to practice defining your own functions, including those with parameters and return values.

A way of testing software, in which you test only the function that is being developed, is called a *unit test*.

Similar to Lab 6, I will be specifying the name and parameter list of four specific functions you must implement. The automated tests will test if you have implemented each function correctly. You can add your own code to test each function (and you are encouraged to do so), but all that matters is the function implementation.

First, create a new file in Thonny and save it. Name the file Lab7.py. Furthermore, the functions must also be named exactly `letter_grade` and `letter_grade_extra_credit` and `convert_to_liters` and `bmi`

Part 1:
Create a function named **`letter_grade`**. The function should accept one parameter, named score. should return the corresponding string based on the score input.

Score	**Return value**
> 100 or < 0	"input error"
>= 90	"A"
>= 80	"B"
>= 70	"C"
>= 60	"D"
< 60	"F"

Hint: your function should contain:

```
def letter_grade(score):
    """
    Convert a score to a letter grade

    Parameter:
        score: an integer or floating point value

    Returns:
        a string, the corresponding grade for the score
    """
```

Write some test code to run your function. For example:

```
print(letter_grade(90))
print(letter_grade(88))
print(letter_grade(100.1))
print(letter_grade(59))
```

Should result in

```
A
B
input error
F
```

Part 2:

Create a function named **`letter_grade_extra_credit`**. The function should accept two parameters, named `score` and `extra_credit`. The function should first check to see that the `extra_credit` parameter is not a negative number. If `extra_credit` is negative, then the function should return "`invalid`". Otherwise, the code should add the `score + extra_credit` and return the appropriate letter grade as defined in Part 1.

Hint: Your code should contain

```
def letter_grade_extra_credit(score, extra_credit):
```

Example: the following code

```
print(letter_grade_extra_credit(90,-1))
print(letter_grade_extra_credit(88,2))
print(letter_grade_extra_credit(75.5,6))
print(letter_grade_extra_credit(40.25,20))
print(letter_grade_extra_credit(99,1.01))
```

Should result in

```
invalid
A
B
D
input error
```

Part 3:
Create a function named **`convert_to_liters`**. The function should accept one parameter, named **`gallons`**. The function should return the input parameter, an amount in gallons, to liters. The ratio needed to convert gallons to liters is as follows:

Hint: Your code should contain

```
def convert_to_liters(gallons):
```

Example:
A call to the function in this form:

```
print(convert_to_liters(20))
```

Should return:

```
75.7082
```

Part 4:
Create a function named **`bmi`**. (The body mass index (BMI) is a convenient rule of thumb that screens for weight categories that may lead to health problems, but it does not diagnose the body fatness or health of an individual. [source].)

The function should accept two parameters, named: **`weight`**, and **`height`** – listed in this exact order. You can assume the input will be in pounds and inches. The formula for calculating the bmi (according to the CDC) is

$$703 \times \text{weight (lbs)} / [\text{height (in)}]^2$$

The following BMI values are used to broadly classify a person into four categories: underweight, normal weight, overweight, and obese:

BMI Categories	**BMI Value**
Underweight	< 18.5
Normal Weight	18.5 – 24.999

Overweight	25.0 – 29.999
Obese	>= 30.0

Hint: your code should contain:

Your **`bmi`** function should return the appropriate string based on the bmi formula.

Example:
A call to the function in this form:

```
print(bmi(175,66))
```

Should return:

```
Overweight
```

For full credit, each of your functions should have complete docstrings (multi-line comments at the top of each function that specifies the name of the function, a description of the parameters, and a description of the return value). Your code must also have your name in comments.

One last reminder: the file you submit in Xuexitong must be named exactly Lab7.py (note the capitalization) and it must contain functions named exactly `letter_grade` and `letter_grade_extra_credit` and `convert_to_liters` and `bmi`