due: 01.30.2022 Sunday 11:59pm

Before attempting this homework, please read the *primer* (`primer_3.0.pdf`). You will go over this primer in Lab01, but should read it in advance. It is especially important to download Anaconda before Lab01 (it is a large download that will require some time).

HW1 syntax tested

• the numeric types int and float

• the arithmetic operators (+ - * / // **)

• the math module (math.sqrt, math.pi)

• variable assignment (optional: you may avoid it by directly returning the results, rather than first storing them)

• function calls and branching (call with arguments, parameters, return) (although we have built all of this function structure for you, and you just need to use it)

In this first homework, you will use Python as a calculator, similar to the way you would use your HP calculator in a physics class. This is already incredibly useful. Even though Python in this homework acts like a glorified calculator, it is an elegant calculator, with less tedium and more control.

HW1 syntax constraints

Certain Python syntax will be banned in each homework.

Almost anything we have covered in class so far is allowed.

The following syntax is banned for HW1: *strings, sequences like lists, recursion.*

If you don't know what some (all?) of these words mean, that is to be expected; we will cover the meaning of these terms in later lectures. Associated keywords are also banned.

HW1 style constraints

• each line should contain no more than 80 characters

• do not hardcode the test cases into your solutions (this is counterproductive anyway, since we will be testing your code using different test cases)

Grading

Grading: a problem is correct if all our tests pass, otherwise it will be considered as False.

Each Questions Worth 20 points

*** Do not forget to include *"independent completion form"* ***

Instructions

Each step indicates the relevant section of the primer_3.0, where you can find help on this issue.

To work on HW1:

• install Anaconda (if you haven't done it in lab01 yet)

• build a HW1 directory under your home directory (cs103sp22)

• download hw1.ipynb from Canvas

• move this file to your HW1 directory

• start Jupyter Notebook and from dashboard navigate to hw1 folder

• open hw1.ipynb in Jupyter Notebook and start working on the questions

• in the notebook, edit the first markdown cell, add your full name and blazerid and Run it

HW1, CS103 Spring 2022

name:

blazerid:

• in the notebook, edit the myName function, replacing James Bond with your name (please leave everything else the same, including the quotes)

• in the notebook, solve the practice problem (see below) in hw1.ipynb

• starting with the first problem, solve the rest of the HW1 problems by writing code in the file hw1.ipynb using the Jupyter Notebook, do not forget to run the cell each time you make a change.

• once you are confident that your code works, submit on Canvas. **Do not forget to include your Independent Completion Form (ICF).**

Practice problem

f(x) Write the function `f(x)` that takes a float value `x`, and returns the value of the linear polynomial `5x−3`. This Python function is similar to the mathematical function f(x) = 5x−3.

Sample Input:	Expected Output:
x = 5	22
X= 0	−3
X = 2.5	9.5

code provided:

```
def f(x):
    return #add your code here
```

correct answer:

```
def f(x):
    return 5*x - 3
```

HW1 problems

areaCircle(r)

Write the function `areaCircle(r)` that takes a positive float value, representing the radius of a circle, and returns the area of this circle.

Sample Input:	Expected Output (~):
r = 1	3.14159265
r = 0.1	0.0314159265
r = 12.34	478.387906280

specialOperation(n1,n2)

Write the function `specialOperation(n1,n2)` that takes two positive integers n1 and n2. You will return their product if the product is less than 100. If their product is bigger than 100, you will return their sum

Sample Input:	Expected Output (~):
n1 = 4, n2 = 5	20
n1 = 12, n2 = 7	84
n1 = 15, n2 = 10	25

phoneBill (m,tx) Write the function `phoneBill(m,tx)` that takes two int m and tx as input and returns a float. The function takes total number of minutes and text messages that user spends in a particular month and returns the bill amount. These is the policy of the carrier company:

- The plan includes 50 minutes of airtime and 50 text messages for $15.00 a month.
- Each additional minute of airtime costs $0.25, while additional text messages cost $0.15 each.
- All cell phone bills include an additional charge of $0.44 to support 911 call centers.
- The entire bill (including 911 charge) is subject to 5 percent sales tax.
- The function will take the total number of minutes and text messages and compute/return the total amount to be paid.

Sample Input:	Expected Output:
m = 70, tx = 120	32.4869
m = 50, tx = 50	16.212
m = 127, tx = 30	36.4244

grader (avg_exams, avg_hw, attendance)

Write the function "**grader**" that takes two floats; **avg_exams** and **avg_hw**, and one integer **attendance** as input and returns a **bool** value. The function will check any student's grade and attendance to decide if the student pass the course or fail. If the student pass, the function returns true, otherwise it returns false. These are the criteria to pass the course:

- The `attendance` must be greater than 17 to pass.
- The `avg_exams` and the `avg_hw` must be greater than 70.
- Either `avg_exams` or `avg_hw` must be greater than 80 (or both of them).

Example Function Calls

`grader(76,100,24)` → returns True

`grader(100,90,16)` → returns False

`grader(72,78,22)` → returns False

radToDegree(rad)

Write the function "`radToDegree`" that takes a radian value and converts it to degree. You can check this link to learn about the radian/degree conversion.

Sample Input:	Expected Output (~):
rad = 100	5729.58
rad = 27	1546.99
rad = 1	57.29

Submission:

Make sure that all your code is correct and producing the correct output. Then, upload your hw1.ipynb file into Canvas. **Do not forget to sign and upload your independent completion form as a pdf.**

Pep talk Despite the long discussion of each problem; the actual code is quite short. This is a common characteristic of good code: lots of thought leading to terse but clear code. In particular, each function in HW1 may be implemented using one line of code in the body of the function for some questions.