ITI1120 Winter 2018 - ASSIGNMENT 1

Read the instructions below carefully. The instructions must be followed. This assignment is worth **5%** of your grade. The assignment is due on `Jan 29th, 8:00am`. No late assignments will be accepted.

This is an individual assignment. Please review the Plagiarism and Academic Integrity policy presented in the first class, i.e. read in detail slides 16-19 of the lecture slides of the first class. You can find the lecture slides on Brightspace. While at it, also review Course Policies on slide 15.

The goal of this assignment is to learn and practice (via programming) the concepts that we have learned so far: numbers, algebraic expressions, boolean expressions, strings, operations on strings, type conversion, variables, use of Python's builtin functions including input and output functions, designing your own functions, documenting your own functions via docstrings, and testing your functions. Before you can start this assignment, you need to know how to use a (IDLE's) text editor to edit python modules (i.e. files) as well as how to use python shell interactively to test your code. If you have no idea how to do these things watching video of the 3rd lecture, for example, will help. Submit your assignment by the deadline via Brightspace (as instructed and practiced in the first lab.) You can make multiple submissions, but only the last submission before the deadline will be graded. What needs to be submitted is explained next.

The assignment has 12 programming questions (in Section 1 below). Each question asks you to design one function. Put all these functions (for all the questions below) in ONE file only, called `a1_xxxxxx.py` (where `xxxxxx` is replaced with your student number). Within this file, `a1_xxxxxx.py`, separate your answers (i.e. code) to each question with a comment that looks like this:

```
########################################################################
# Question X
########################################################################
```

To have an idea on what this file `a1_xxxxxx.py` should look like, I included with this assignment a solution to an nonexistent assignment. You can view this mockup solution by opening the included file called `a1_mockup_assignment_solution.py`

In addition to `a1_xxxxxx.py` you must submit a separate file called `a1_xxxxxx.txt`. What should be in that file is explained below. Thus for assignment 1 you have to **SUBMIT TWO FILES**:

- `a1_xxxxxx.py` and
- `a1_xxxxxx.txt`

as instructed in lab 1. Submit your assignment by the deadline via Brightspace (as instructed in the first lab.)

Your program must run without syntax errors. In particular, when grading your assignment, Tas will first open your file `a1_xxxxxx.py` with IDLE and press Run Module. If pressing Run Module causes any syntax error, **the grade for the whole assignment will be zero**.

Furthermore, for each of the functions below, I have provided one or two tests to test your functions with. For example, you should test question 1 by making function call `lbs2kg(5)` in the Python shell. To obtain a partial mark your function may not necessarily give the correct answer on these tests. If your function gives any kind of python error when run on the tests provided below, that question will be marked with zero points.

After reading each of these questions once, go to the Section 2: “Testing your code" below and see what the output of your function should give. In that section, you can find a couple offunction calls and the required results for each question. Studying these example function calls will help you a lot to understand what each individual question requires, or rather what its function should do.

To determine your grade, your functions will be tested both with examples provided in Section 2: "Testing your code" and with some other examples. Thus you too should test your functions with more example than what I provided in Section 2.

Each function has to be documented with *docstrings*. In particular, each function has to have docstrings that specify:
- type contract
- description about what the function does (while mentioning parameter names)
- preconditions, if any

The purpose of this assignment is to practice concepts that you have seen in the first 2.5 weeks of class. Thus this assignment does not require use of loops, if and other branching statements, lists ... etc, except for question 12 (and Question 10 if you want to be creative).

Thus you must solve the questions below without loops, `if` and other branching statements, lists etc ... unless explicitly stated otherwise in the question. Questions that break this rule will receive zero since they suggest that the required understanding of the material covered in first 2.5 weeks is not attained.

Global variables are not allowed. If you do not know what that means, for now, interpret this to mean that inside of your file `a1_xxxxxx.py` variables can only be created (ie. Assigned value) inside of the bodies of your functions.

To avoid confusion, unless otherwise specified in the questions here is what you can use in this assignment:
- comparison operators: `<,<=, ==, !=, >, >=`
- Boolean operators: `and, or, not`
- arithmetic operators: `+, -, *, /, **, %, //`
- the following Python built in functions: `print, input, round, len, int, float, str`
- string operators: `+, *`
- any function from the `math` module (recall `import math, dir(math)`, and then you can call `help` on any function in `math` module. eg. `help(math.sqrt)`)
- anything from `Turtle` module
- keywords: `def, return`
- `if` statements *(only in question 12 and possibly question 10)*

Section 1: Assignment 1 questions

1. (2 points) Write a function `lbs2kg(w)` that returns the given weight, `w`, expressed in pounds as weight in kilograms.
2. (2 points) Write a function `id_formater(fn, ln, appelation, city, year)` that returns a string of the form `"appelation. ln,fn (city, year)"`.
3. (2 points) Write a function `id_format_display()` that prompts the user for the first name, last name, appellation, place of birth, and year of birth. The function should then print the ID of the user using the same format as specified in question 2. Your `id_formater_display()` function must use a function call to your `id_formater` function from question 2.
4. (2 points) Write a function `l2loz(w)` that takes a non-negative number `w` as input and returns a pair of numbers `(l,o)` such that `w = l + o/16` where `l` is an integer and `o` is a non-negative number smaller than 16.
5. (2 points) The *median* of a group of numbers is the number from that group that has the property that at least half of the elements in the group are smaller or equal to it and at least half of the elements in the group are bigger or equal to it. In a sorted list of numbers, the median can be found in the middle of the list (well, the middle is only well defined if the list has odd number of elements). For example, the median of this group of numbers: 10,11,13,15,16,23,26 is 15. Median of this group of numbers: 7,1,3 is 3. The median of this group of numbers: 2,2,5 is 2. Write a function `median3(num1,num2,num3)` that prints a message for each of the given three numbers, `num1`, `num2` ,`num3`, stating if the number is a median. See the test runs in Section 2 to understand in what format should the function `median3` print its results.
6. (2 points) Write a function `forms_triangle(a, b, c)` that returns `True` if three segments of length `a`, `b` and `c` can form a triangle, otherwise it returns `False`. You may assume that your function will be tested with `a`, `b` and `c` that are non-negative integers . You are not allowed to use `if` (or other branching) statements. Here are the details about when 3 numbers (i.e. 3 side lengths) form a triangle. If you are given three sticks, you may or may not be able to make a triangle with them. For example, if one of the sticks is 10cm long and the other two are 1cm long, you will not be able to form a triangle. For any three non-negative numbers, there is a simple test that determines if it is possible to form a triangle with them. It states: If any of the three numbers is greater than the sum of the other two, then you cannot form a triangle. Otherwise, you can. Use this test for your function. In mathematics, this fact is known as the "Triangle Inequality".
7. (2 points) Write a function `below_parabola(a,b,p,q)` that returns `True` if the point `(p,q)` in the plane (i.e., the point with `x`-coordinate `p` and `y`-coordinate `q`) is below or on the graph of the parabola `y = ax^2 + b`, otherwise it returns `False`. You may assume that your function will be tested only with positive numbers for `a` (and arbitrary numbers for `b`). (Note that `x^2` stands for `x` squared.)
8. (2 points) Write a function `efun(x)` that takes as input a positive number `x` and solves the following equation for `y` and returns `y`. The equation is $e^{17y}=x+2$ where `e` refers to the mathematical constant.
9. (2 points) Write a function `ascii_name_plaque(name)` that takes as input a string representing a person's name and draws (using `print` function) a name plaque as shown in the examples given in Section 2 below. Recall that <u>you may not use loops nor `if`/ branching statements</u>. (Question 10 and 12 are the only exceptions to that rule.)
10. (2 points) Write a function `draw_court()` that draws lines of basketball court using `Turtle` graphics. See this image for example. If you want to be really precise you can easily find NBA court specificiations. If you are imaginative, instead of function `draw_court()` write a function `my_fun_drawing()` that draws something else fun, but it should not be simpler than `draw_court()`.
11. (4 points) Suppose that a grading scheme for a course with 3 assignments, one midterm and one final is: each assignment is worth 5%, the midterm is worth 35% and the final is worth 50% of the grade. Write a function

`projected_grade(a1,A1,a2,A2,m,M)` that predicts (i.e. computes) the final grade percentage as follows. The student obtained `a1` points out of maximum `A1` for assignment 1; `a2` points out of maximum `A2` for assignment 2 and `m` points out of maximum `M` for the midterm. To compute the final course grade, assign to assignment 3 the average percentage of assignment 1 and assignment 2 and assign to the final exam the same the percentage as obtained on the midterm exam. For example, if the student got 12/20 in assignment 1 and 24/24 in assignment 2, the grade in assignment 3 should be 80% since that is the average of the two percentages. If the student obtained 9/12 in the midterm, the grade of the final exam should also be 75%. Using the given grading scheme and the predicted grades for assignment 3 and the final exam, function `projected_grade` should return the predicted final percentage for the course. You can assume that the function will be tested with reasonable values `a1`,`A1`,`a2`,`A2`,`m`, and `M`.

12. (4 points) Write a function called `projected_grade_v2()` that prompts the user to enter 6 numbers (the meaning of these six numbers is the same as `a1`,`A1`,`a2`,`A2`,`m`,`M` in the previous question). Compute the predicted percentage of the final course grade with the slight modification to the grading scheme where if the average on the midterm and the final is less than 50% the final course grade percentage is whatever is smaller of the two grading schemes. Print a meaningful message to a user about their predicted final grade and the function should return the predicted final grade. You may (or rather should) use `if` statements in this question.

Section 2: Testing your code

We would like to see evidence that you have tested your functions in python shell, like we did in class. Do this by opening your assignment solution, i.e. opening `a1_xxxxxx.py`, with IDLE and then test each of your functions individually. Then copy and paste the python shell output into a text file called `a1_xxxxxx.txt`. The contents of `a1_xxxxxx.txt` must have something like this in it:

```
>>> # testing Question 1
>>>
>>> lbs2kg(1)
0.453592
>>>
>>> lbs2kg(130.2)
59.05767839999999
>>>
>>> lbs2kg(5)
2.26796
>>>
>>>
>>> # testing Question 2
>>>
>>> id_formater("Albert","Einstein", "Dr", "Bern", 1879)
'Dr. Einstein, Albert (Bern, 1879)'
>>>
>>>
>>> # testing Question 3
>>>
>>> id_formater_display()
What is your first name? Harry
What is your last name? Potter
What is your appellation? Wizard
Where were you born? Godric's Hollow
What is your year of birth? 1980

Wizard. Potter, Harry (Godric's Hollow, 1980)
>>>
>>>
>>> # testing Question 4
>>>
>>> l2loz(7.5)
(7, 8.0)
>>>
>>> l2loz(9.25)
(9, 4.0)
>>>
>>>
>>> # testing Question 5
>>>
>>> median3(18, 1, 8)
18 is a median. That is  False
1 is a median. That is  False
8 is a median. That is  True
>>>
>>> median3(1, 8, 1)
1 is a median. That is  True
```

```
8 is a median. That is  False
1 is a median. That is  True
>>>
>>>
>>> # testing Question 6
>>>
>>> forms_triangle(20,20,30)
True
>>> forms_triangle(2,30,1)
False
>>>
>>>
>>> # testing Question 7
>>>
>>> below_parabola(1, 0, 0, 0)
True
>>> below_parabola(1, 0, 1, 1)
True
>>> below_parabola(1, 0, 1, 2)
False
>>> below_parabola(2.5, 0, 1, 2)
True
>>>
>>>
>>> # testing Question 8
>>>
>>> efun(7)
0.12924850454918937
>> efun(21)
0.18444083623112645
>>> efun(123456789)
1.0959648107275306
>>> efun(0.2)
0.046379844727310014
>>>
>>>
>>> # testing Question 9
>>>
>>> ascii_name_plaque("abba")
**************
*            *
*            *
*  __abba__  *
*            *
*            *
**************
>>>
>>> ascii_name_plaque("Captain Kara 'Starbuck' Thrace")
****************************************
*                                      *
*                                      *
*  __Captain Kara 'Starbuck' Thrace__  *
*                                      *
*                                      *
****************************************
>>>
>>> ascii_name_plaque("Seven of Nine")
**********************
*                    *
*                    *
*  __Seven of Nine__  *
*                    *
*                    *
**********************
>>>
>>>
>>> # testing Question 10
>>>
>>> draw_court()
>>>
>>>
>>> # testing Question 11
>>>
>>> projected_grade(10, 10, 15, 15, 30,30)
100.0
>>> projected_grade(12, 20, 24, 24, 9,12)
```

```
75.75
>>> projected_grade(18, 20, 16, 19, 60,100)
64.06578947368422
>>> projected_grade(19, 20, 16, 16, 49,100)
56.275
>>>
>>>
>>> # testing Question 12
>>>
>>> projected_grade_v2()
How many points did you get in Assignment 1? 10
What was the maximum possible number of points for Assignment 1? 10
How many points did you get in Assignment 2? 9
What was the maximum possible number of points for Assignment 2? 10
How many points did you get on the midterm? 49
What was the maximum possible number of points for the midterm? 100
Your predicted final grade is 49.0 %
>>> projected_grade_v2()
How many points did you get in Assignment 1? 10
What was the maximum possible number of points for Assignment 1? 10
How many points did you get in Assignment 2? 9
What was the maximum possible number of points for Assignment 2? 10
How many points did you get on the midterm? 0
What was the maximum possible number of points for the midterm? 100
Your predicted final grade is 0.0 %
>>> projected_grade_v2()
How many points did you get in Assignment 1? 10
What was the maximum possible number of points for Assignment 1? 10
How many points did you get in Assignment 2? 17
What was the maximum possible number of points for Assignment 2? 17
How many points did you get on the midterm? 100
What was the maximum possible number of points for the midterm? 100
Your predicted final grade is 100.0 %
```