ASSIGNMENT 1

Read the instructions below carefully. The instructions must be followed. This assignment is worth 4% of your grade. The assignment is due on Monday 1st of October 8AM. 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 pages 15-18 of course outline (ITI1120-course-outline-syllabus-Fall2018.pdf). You can find that file on Brightspace under Course Info. While at it, also review Course Policies on pages 13 and 14.

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 14 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 2 by making function call `mh2kh(5)` in the Python shell. To obtain a partial mark your function may not necessarily give the correct answer on these tests. But 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 of

function 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 (as will be explained in Lecture 5 on Sept 19). 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. Question 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.

In addition to docstrings, Lecture 5, on Sept 19, will cover compound boolean expressions (i.e boolean expressions that use and, or and not boolean operators). They will be needed for some questions Finally, we will also cover if statements, in Lecture 5. They will only be needed in question 12.

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) Two numbers a and b are called pythagorean pair if both a and b are integers and there exists an integer c such that $a^2 + b^2 = c^2$. Write a function pythagorean_pair(a,b) that takes two integers a and b as input and **returns** True if a and b are pythagorean pair and False otherwise.
2. (2 points) Write a function mh2kh(s) that given the speed, s, expressed in miles/hour **returns** the same speed expressed in kilometres/hour.

3. (2 points) Write a function in_out(xs,ys,side) that takes three numbers as input, where side is non-negative. Here xs and ys represent the x and y coordinates of the bottom left corner of a square; and side represents the length of the side of the square. (Notice that xs, ys, and side completely define a square and its position in the plane). Your function should first prompt the user to enter two numbers that represent the x and y coordinates of some query point. Your function should **print** True if the given query point is inside of the given square, otherwise it should print False. A point on the boundary of a square is considered to be inside the square.
4. (2 points) Write a function safe(n) that takes a non-negative integer n as input where n has at most 2 digits. The function determines if n is a safe number. A number is *not safe* if it contains a 9 as a digit, or if it can be divided by 9. The function should test if n is safe and return True if n is safe and False otherwise.
5. (2 points) Write a function quote_maker(quote, name, year) that returns a sentence, i.e. a string of the following form: In year, a person called name said: “quote” See the next Section 2 below for some examples of how your function must behave.
6. (2 points) Write a function quote_displayer() that prompts the user for a quote, name, and year. The function should then print a sentence using the same format as specified in the previous question. (To do that, your solution must make a call to quote_maker function from the previous question to obtain a string that you then print).
7. (2 points) In this question you will write a function that determines the result of a rock, paper, scissors game given choices of player 1 and player 2. In particular, write a function rps_winner() that prompts the user for choice of player 1 and then choice of player 2, and then it displays the result for player 1 as indicated in the examples given in Section 2. You may assume that the user will only enter words: rock, paper or scissors in lower case. Recall that paper beats rock, that rock beats scissors and that scissors beat paper. If both players make the same choice, we have a draw.
8. (2 points) Write a function fun(x) that takes as input a positive number x and solves the following equation for y and returns y. The equation is $10^{4y}=x+3$.
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 you may not use loops nor if/branching statements. (Question 10 and 12 are the only exceptions to that rule.)
10. (2 points) Write a function draw_car() that draws, with Turtle graphics, the car from the image that you can find at the end of the assignment - except that the car in your drawing has to have some lively colour (i.e not be white like the one in the drawing). If you are imaginative, you may choose to draw a more complex drawing of a car or instead of function draw_car() write a function my_fun_drawing() that draws something else fun. Whatever you choose it should not be simpler than the car on the given image. In this questions you can use loops and/or if statements if you want to draw something more complex.
11. (2 points) Write a function alogical(n) , that takes as input a number, n, where n is bigger or equal to 1, and returns the minimum number of times that n needs to be divided by 2 in order to get a number equal or smaller than 1. For example 5.4/2=2.7. Since 2.7 is bigger than 1, dividing 5.4 once by 2 is not enough, so we continue. 2.7/2=1.35 thus dividing 5.4 twice by 2 is not enough since 1.35 is bigger than 1. So we continue. 1.35/2=0.675. Since 0.675 is less than 1, the answer is 3. In particular, these calculations determine that 5.4 needs to be divided by 2 three times minimum in order to get a number that is less than or equal to 1. See Section 2 for more examples. Recall that you may not use loops nor if/branching statements. (Question 10 and 12 are the only exceptions to that rule.)
12. (5 points) Write a fancy time formatter, time_format(h,m), that takes a time of day, expressed in hours, h (integer 0 to 23) and minutes, m (integer 0 to 59). The function first rounds the minutes to the nearest 5 minutes, and then returns the time as

descriptive string. In particular, after rounding the minutes, the string has to be such that it uses "o'clock" and "half past" for 0 and 30 minutes, respectively. For less than 30 minutes, it has to use "past" format and for more than 30 minute it has to use "to" format. See the examples in Section 2 for clarification and examples on how your function must behave. Your code for rounding minutes to the nearest 5 minutes should not take more than ONE line of code. You may use if statements in this question.

13. (2 points) Write a function cad_cashier(price,payment) that takes two real non-negative numbers with two decimal places as input, where payment>=price and where the second decimal in payment is 0 or 5. They represent a price and payment in Canadian dollars. The function should return a real number with 2 decimal places representing the change the customer should get in Canadian dollars. Recall that in Canada, while the prices are expressed in pennies, the change is based on rounding to the closest 5 cents. See the examples in Section 2 for clarification and examples on how your function must behave.
14. (5 points) Suppose that a cashier in Canada owes a customer some change and that the cashier only has coins ie. toonies, loonies, quarters, dimes, and nickels. Write a function that determines the minimum number of coins that the cashier can return. In particular, write a function min_CAD_coins(price,payment) that returns five numbers (t,l,q,d,n) that represent the smallest number of coins (toonies, loonies, quarters, dimes, and nickels) that add up to the amount owed to the customer (here price and payment are defined as in the previous question). You program must first call cad_cashier function, from question 13, to determine the amount of change that needs to be returned. Then before doing anything else, you may want to convert this amount entirely to cents (that should be of type int). Once you have the total number of cents here are some hints on how to find the minimum number of coins.

Hints for your solution (algorithm) for question 14:
To find the minimum number of coins the, so called, *greedy strategy* (i.e. *greedy algorithm*) works for this problem. The greedy strategy tries the maximum possible number of the biggest-valued coins first, and then the 2nd biggest and so on. For example, if price is $14.22 and payment $20, the customer is owed $5.80 (after rounding to closest 5 cents), thus 580 cents, the greedy strategy will first figure the maximum number of toonies it can give to the customer. In this case, it would be 2 toonies. It cannot be 3 toonies as that equals $6 and the cashier would return too much money. Once the cashier returns 2 toonies, he/she still needs to return 180 cents. The next biggest coin after toonie is a loonie. So the greedy strategy would try loonies next. Only 1 loonie can fit in 180 cents, so the cashier should next return 1 loonie. Then there is 80 cents left. The next biggest coin to consider is a quarter … and so on … ending with nickels. (For this example the function should return (2,1,3,0,1)). Thus for this question, you are asked to implement this strategy to find the optimal solution. See section 2 for more examples.

Side note: in the Canada (and most other) coin systems, the greedy algorithm of picking the largest denomination of coin which is not greater than the remaining amount to be made will always produce the optimal result (i.e. give the smallest number of coins). This is not automatically the case, though: if the coin denominations were 1, 3 and 4 cents then to make 6 cents, the greedy algorithm would choose three coins: one 4-cent coin and two 1-cent coins whereas the optimal solution is two 3-cent coins.

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
>>>
>>> pythagorean_pair(2,2)
False
>>> pythagorean_pair(6,2)
False
>>> pythagorean_pair(6,8)
True
>>> pythagorean_pair(300,-400)
True
>>>
>>> # testing Question 2
>>>
>>>
>>> mh2kh(5)
8.0467
>>>
>>> mh2kh(110.4)
177.67113600000002
>>>
>>>
>>> # testing Question 3
>>>
>>> in_out(0,0,2.5)
Enter a number for the x coordinate of a query point: 0
Enter a number for the y coordinate of a query point: 1.2
True
>>> in_out(2.5,1,1)
Enter a number for the x coordinate of a query point: -1
Enter a number for the y coordinate of a query point: 1.5
False
>>> in_out(-2.5,1,2.1)
Enter a number for the x coordinate of a query point: -1
Enter a number for the y coordinate of a query point: 1.5
True
>>>
>>>
>>> # testing Question 4
>>>
>>> safe(93)
False
>>> safe(82)
True
>>> safe(29)
False
>>> safe(36)
False
>>> safe(9)
False
>>> safe(7)
True
>>>
>>>
>>> # testing Question 5
>>>
>>> quote_maker("Everything should be made as simple as possible but not simpler.", "Albert Einstein", 1933)
'In 1933, a person called Albert Einstein said: "Everything should be made as simple as possible but not simpler."'
>>>
```

```
>>> quote_maker("I would never die for my beliefs because I might be wrong.", "Bertrand Russell", 1951)
'In 1951, a person called Bertrand Russell said: "I would never die for my beliefs because I might be wrong."
>>>
>>>
>>> # testing Question 6
>>>
>>> quote_displayer()
Give me a quote: The best lack all conviction while the worst are full of passionate intensity.
Who said that? Bertrand Russell
What year did she/he say that? 1960
In 1960, a person called Bertrand Russell said: "The best lack all conviction while the worst are full of passionate intensity."
>>>
>>>
>>> # testing Question 7
>>>
>>> rps_winner()
What choice did player 1 make?
Type one of the following options: rock, paper, scissors: rock
What choice did player 2 make?
Type one of the following options: rock, paper, scissors: paper
Player 1 wins. That is False
It is a tie. That is not True
>>> rps_winner()
What choice did player 1 make?
Type one of the following options: rock, paper, scissors: paper
What choice did player 2 make?
Type one of the following options: rock, paper, scissors: rock
Player 1 wins. That is True
It is a tie. That is not True
>>> rps_winner()
What choice did player 1 make?
Type one of the following options: rock, paper, scissors: scissors
What choice did player 2 make?
Type one of the following options: rock, paper, scissors: paper
Player 1 wins. That is True
It is a tie. That is not True
>>> rps_winner()
What choice did player 1 make?
Type one of the following options: rock, paper, scissors: paper
What choice did player 2 make?
Type one of the following options: rock, paper, scissors: paper
Player 1 wins. That is False
It is a tie. That is not False
>>>
>>>
>>> # testing Question 8
>>>
>>> fun(7)
0.25
>> fun(20)
0.3404319590043982
>>> fun(999999997)
2.25
>>> fun(0.1)
0.12284042345856817
>>>
>>>
>>> # testing Question 9
>>>
>>> ascii_name_plaque("vida")
**************
*            *
*  __vida__  *
*            *
**************
>>> ascii_name_plaque("Captain Kara 'Starbuck' Thrace")
****************************************
*                                      *
*  __Captain Kara 'Starbuck' Thrace__  *
*                                      *
****************************************
>>> ascii_name_plaque("Seven of Nine")
```

```
***********************
*                     *
*  __Seven of Nine__  *
*                     *
***********************
>>>
>>>
>>> # testing Question 10
>>>
>>> draw_car()
>>>
>>>
>>> # testing Question 11
>>>
>>> alogical(5.4)
3
>>> alogical(4)
2
>>> alogical(1000)
10
>>> alogical(4200231)
23
>>>
>>>
>>> # testing Question 12
>>>
>>> time_format(8,0)
“8 o'clock"
>>> time_format(8,2)
"8 o'clock"
>>> time_format(8,59)
"9 o'clock"
>>> time_format(8,8)
"10 minutes past 8 o'clock"
>>> time_format(8,32)
"half past 8 o'clock"
>>> time_format(8,48)
"10 minutes to 9 o'clock"
>>> time_format(17,42)
"20 minutes to 18 o'clock"
>>> time_format(23,13)
"15 minutes past 23 o'clock"
>>> time_format(23,42)
"20 minutes to 0 o'clock"
>>> time_format(0,29)
"half past 0 o'clock"
>>> time_format(11,59)
"12 o'clock"
>>> time_format(23,58)
"0 o'clock"
>>> time_format(0,1)
"0 o'clock"
>>> time_format(11,1)
"11 o'clock"
>>>
>>>
>>> # testing Question 13
>>>
>>> cad_cashier(10.58,11)
0.4
>>> cad_cashier(98.87,100)
1.15
>>> cad_cashier(10.58,15)
4.4
>>> cad_cashier(10.55,15)
4.45
>>> cad_cashier(10.54,15)
4.45
>>> cad_cashier(10.52,15)
4.5
>>> cad_cashier(10.50,15)
4.5
```

```
>>>
>>>
>>> # testing Question 14
>>>
>>> min_CAD_coins(10.58,11)
(0, 0, 1, 1, 1)
>>> min_CAD_coins(98.87,100)
(0, 1, 0, 1, 1)
>>> min_CAD_coins(10.58,15)
(2, 0, 1, 1, 1)
>>> min_CAD_coins(10.55,15)
(2, 0, 1, 2, 0)
>>> min_CAD_coins(10.54,15)
(2, 0, 1, 2, 0)
>>> min_CAD_coins(10.52,15)
(2, 0, 2, 0, 0)
>>> min_CAD_coins(10.50,15)
(2, 0, 2, 0, 0)
>>> min_CAD_coins(3, 20)
(8, 1, 0, 0, 0)
```