ASSIGNMENT 1

Read the instructions below carefully. The instructions must be followed. This assignment is worth 2.5% of your grade. The assignment is due on Monday, 25th of September at 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 16-20 of course outline (i.e. slides of Lecture 1). You can find that file on Brightspace under Lecture 1. While at it, also review Course Policy on missing assignments on page 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.

IMPORTANT NOTE for this and future assignments:
Your grade will partially be determined by automatic (unit) tests that will test your functions. All the specified requirements below are mandatory (including function names, and the behaviour implied by examples in Section 2). Any requirement that is specified and not met may/will result in deduction of points.

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 a 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 f_to_k(90)

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 examples than what I provided in Section 2.

Each function must be documented with docstrings (as will be explained in the upcoming lectures). 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 Question 10 if you want to be creative). Thus you must solve the questions below without loops, if and other branching statements, lists etc. Questions that break this rule will receive zero since they suggest that the required understanding of the material covered in the 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.

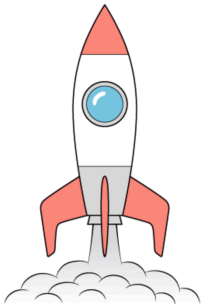
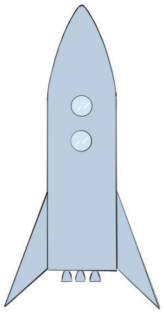
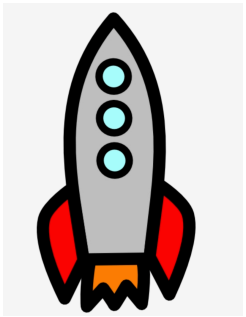
To avoid confusion, unless otherwise specified in the questions here is what you can use in this assignment:
- comparison operators: <,<=, ==, !=, >, >=
- Boolean operators: and, or, not
- arithmetic operators: +, -, *, /, **, %, //
- the following Python built in functions: print, input, round, len, 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 the math module. eg. help(math.sqrt))
- anything from Turtle module
- keywords: def, return

Section 1: Assignment 1 questions

1. (2 points) Write a function called f_to_k(t) that **returns** the temperature, t, given in degrees Fahrenheit as a temperature expressed in degrees Kelvin.
2. (2 points) Write a personalized poem maker in the form of a function called poem_generator(). This function prompts the user for the name and city of birth and then prints a poem that includes that name and city of birth. You can make up your own poem or adapt one from here for example https://literarydevices.net/limerick/.

3. (2 points) Write a function called impl2loz(w) that takes a non-negative number w as input and **returns** a pair of numbers (l,o) such that w = l + o/16 and l is an integer and o is a non-negative number smaller than 16. Note that the solution l and o are unique.
4. (2 points) Write a function pale(n) that takes a positive integer n as input where n has four digits. The function determines if n is a pale number. A number is *not pale* if and only if it has at least two consecutive digits each equal to 3, or if its last digit is divisible by 4. The function should test if n is pale and **returns** True if n is pale and False otherwise. (In this function, you are not allowed to use strings.)
5. (2 points) Write a function bibformat(author,title,city,publisher,year) that **returns** a string of the form: author (year). title. city: publisher. See the next Section 2 below for some examples of how your function must behave.
6. (2 points) Write a function bibformat_display() that prompts the user for a book title, name of the author, year of publication, publisher and the headquarter city of the publisher. The function should then print the information about the book in the same format as specified in the previous question. (To do that, your solution must make a call to bibformat function from the previous question to obtain a string that you then print).
7. (2 points) Write a function compound(x, y, z) that takes as input three integers x, y and z and returns True if x is the only even number among the 3 integers and if at least one pair of the three integers adds up to a number greater than 100. Otherwise the function returns False. (Recall that you are not allowed to use any type of if (i.e. branching) statements here.)
8. (2 points) Write a function funct(p) that takes as input a positive number p (greater or equal to 11) and solves the following equation for r and prints the sentence "The solution is r", where r is replaced by the number that solves the equation for the given p. The equation is $5^{r^2} - p + 10 = 0$
9. (2 points) Write a function gol(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 is the only exceptions to that rule.)

10. (2 points) Write a function draw_rocket() that draws, with Turtle graphics, a rocket or a spaceship as in the image below. Your drawings should not be much simpler than what is in the images below but it can be more complex. 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 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.
12. (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). Your program must first call the cad_cashier function, from question 11, 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 12:

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 the 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
>>>
>>>
>>> f_to_k(90)
305.3722222222222
>>>
>>> f_to_k(-457.87)
1.0
>>>
>>> # testing Question 2
>>>
>>>
>>> poem_generator()
Enter your name: Vida
Enter your city of birth: Mostar

Vida had funny funny hair
With tons and tons to spare
Vida's clippings made a wig
It was very big
And caused the townsfolk of Mostar to stare
>>>
>>>
>>> # testing Question 3
>>>
>>> impl2loz(7.5)
(7, 8.0)
>>>
>>> impl2loz(9.25)
(9, 4.0)
>>>
>>>
>>> # testing Question 4
>>>
>>> pale(1128)
False
>>> pale(3443)
True
>>> pale(3351)
False
>>> pale(3333)
False
>>> pale(4331)
False
>>> pale(3423)
True
>>> pale(4533)
False
>>>
>>>
>>> # testing Question 5
>>>
>>> bibformat("George R. R. Martin", "A Game of Thrones", "New York City", "Bantam Spectra", 1996)
'George R. R. Martin (1996). A Game of Thrones. New York City: Bantam Spectra.'
>>>
>>>
>>> # testing Question 6
>>>
>>> >>> bibformat_display()
Enter the title of a book: Guns, Germs, and Steel: The Fates of Human Societies
Enter the name of the author? Jared Diamond
What year was the book published? 1997
Enter the name of the publisher: W. W. Norton
In what city are the headquarters of the publisher? New York City
Jared Diamond (1997). Guns, Germs, and Steel: The Fates of Human Societies. New York City: W. W. Norton.
>>>
>>>
```

```
>>> # testing Question 7
>>>
>>> compound(80,40,31)
False
>>> compound(80,41,31)
True
>>> compound(8,1,31)
False
>>> compound(80,1,31)
True
>>> compound(80,1001,-44)
False
>>> compound(-2,1001,-43)
True
>>>
>>>
>>> # testing Question 8
>>>
>>> funct(11)
The solution is  0.0
>>> funct(12)
The solution is  0.6562595203678139
>>> funct(10300)
The solution is  2.395927506817742
>>> funct(1000000000)
The solution is  3.588326771135425
>>>
>>>
>>> # testing Question 9
>>>
>>> gol(5.4)
3
>>> gol(4)
2
>>> gol(1000)
10
>>> gol(4200231)
23
>>>
>>>
>>> # testing Question 10
>>>
>>> draw_rocket()
>>>
>>> # testing Question 11
>>>
>>> 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 12
>>>
>>> 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)
```