LBYCPA1

Programming Logic and Design Laboratory

Laboratory Module 2

Basic Syntax, Variables, Data Types, Numbers, Casting

By

John Carlo Theo S. Dela Cruz || LBYCPA1 || EQ1

INTRODUCTION

This module is mostly concerned with programming grammar, variable naming, and declaration. Additionally, the session discusses and teaches about data types, numbers, and casting. Together with module 1, this module will serve as a basis for programming. As with Module 1, the materials and tools for this module will be Jupyter notebooks and Pycharm.edu.

As mentioned in the form, the objective of a lab report is to provide an opportunity for learning rather than to contribute to field knowledge. With that stated, the act of coding, such as flowcharts and pseudocodes. Python enables ease of use when it comes to writing; users, such as myself, can easily grasp the code and identify syntax or logical issues. With the help of Jupyter notebook, you may review your code while you're still working; the platform will also identify any issues in your code once it's run. It is a simple necessity to study the fundamentals; regardless of your field of endeavor, returning to the fundamentals will influence us, students in some way. The module covers variable declaration and naming, as well as statements, values, and expressions. Additionally, the module would teach the many Python data types, including strings, numbers, tuples, and many others. In Python, type conversion, or casting, would also be included in the second module.

(a) Objectives

1. To understand the different data types in Python
2. To understand and be familiarize with various arithmetic operators.
3. To solve advanced mathematical problems using Python
4. To properly run the various arithmetic operators
5. To manipulate a certain function.
6. To execute and run Karel the Bot

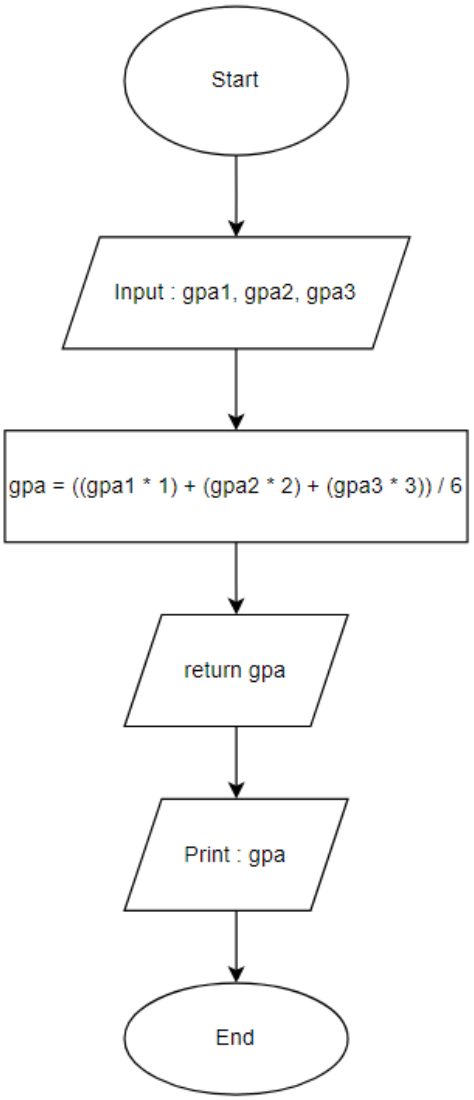
(b) Materials and Tools

1. Module 2
2. Karel the Bot
3. Jupyter Notebook
4. Pycharm.edu
5. Diagrams.net
6. W3schools
7. StackOverflow

PROCEDURES (*Individual*) / EXPERIMENTAL PLAN

Each exercise requires understanding the problem, identify the expected input and outputs, and formulate the process of a program. Proper planning is a must in coding, I used flowcharts, algorithms, and pseudocodes before I can execute our codes:

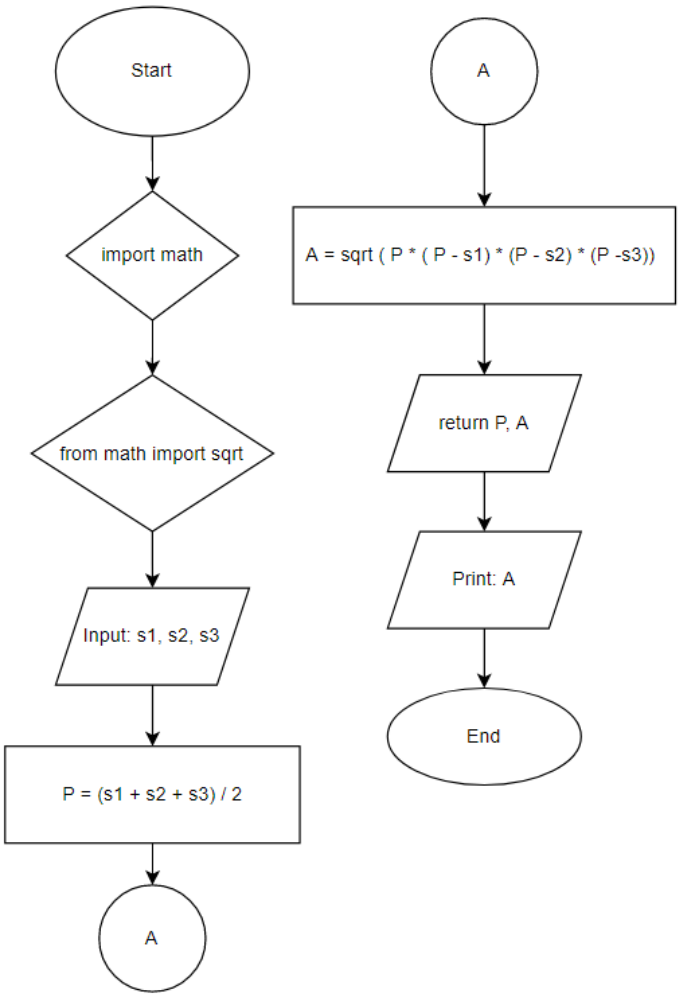
1. Exercise 1: GPA Calculator
    a. This program is a GPA Raw Calculator, and it requires the user to enter their grades of the subject in COEDISC, LBYCPA1 and CALENG1; then the program formulates and will show the GPA. In formulating the GPA, I simply followed the guide on the notes once again and declared a variable gpa.



$$Gpa = [Grade] \times [Unit]$$

$$= \frac{\Sigma Gpa}{\Sigma unit}$$

2. Exercise 2: Triangle Area calculator
    a. This program is basically a calculator, and it requires the users to input 3 sides to find the area of the triangle. I followed the guide from my notebook, and I imported the built-in function math and imported a specific function called square root. In this problem, it was specified that we will use the function of a square root, so I searched a formula and I found Heron's Formula that fits with the criteria of this activity.



Just use this two step process:

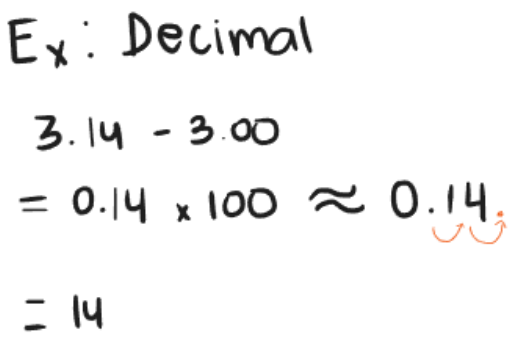
Step 1: Calculate **"s"** (half of the triangles perimeter):

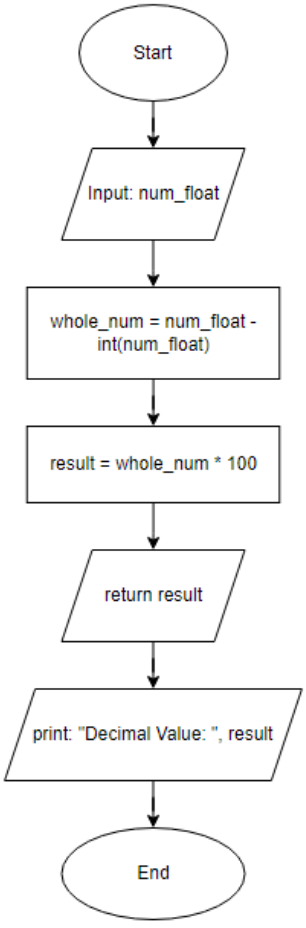
$$s = \frac{a+b+c}{2}$$

Step 2: Then calculate the **Area**:

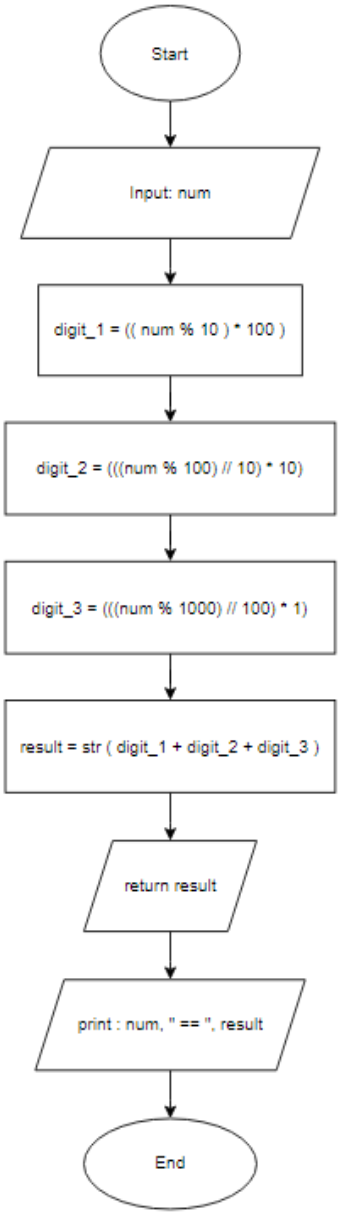
$$A = \sqrt{s\,(s-a)\,(s-b)\,(s-c)}$$

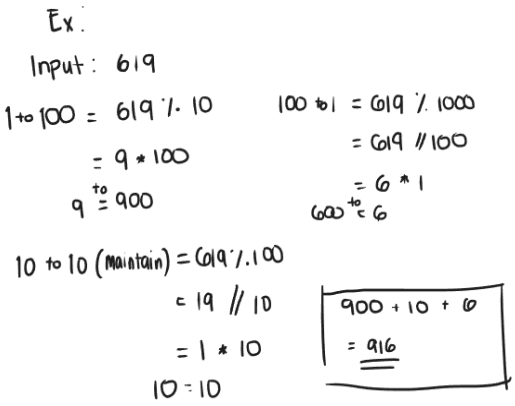
3. Exercise 3: Decimal points calculator
    a. The program requires the users to input a number with 2 decimal points and the program gets the value of the 2 decimal points and transform it into a whole number. The thought process in this activity is to subtract the input with 2 decimal points, minus the whole number of that specific input (3.14 – 3 = 0.14). Once we get the result, I multiplied the result with 100 to transform it to a whole number.





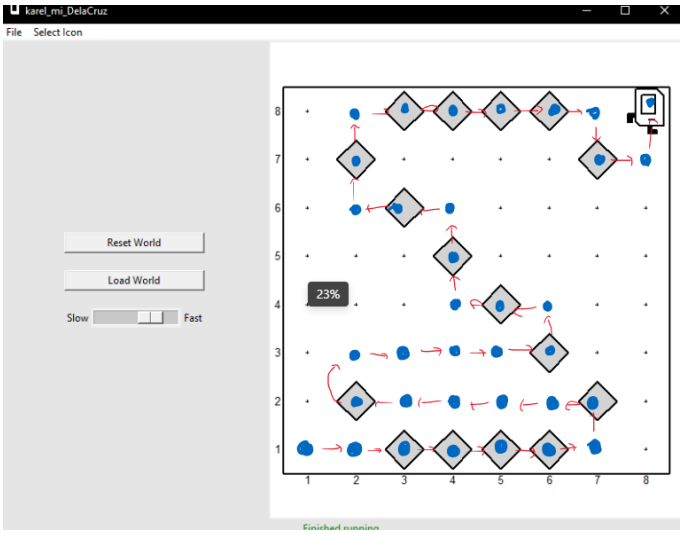
4. Exercise 4: 3-digit Reverse
    a. This program requires an input of a 3-digit number and the program reverses the position of the number. By using modulo and division integer, my initial plan here is to get every place value and manipulate them (Ones Value into Hundredths), by getting the remainder and by using the division integer 10 or 100 (to simplify and it can end with 0) then multiply it with a specific place value. Since we have the value of each place value, we just need to add them all to combine them all together.





5. Exercise 5: Karel the Bot

a. I downloaded the Karel the Bot python package and simply followed the guide on how the bot can be manipulated with specific syntax to create and draw my middle initial.



Writing Middle initial (S.) using beepers: PSUEDOCODE:

```
function move_place_beeper_east:
    move()
    move()
    put_beeper()
    move()
    put_beeper()
    move()
    put_beeper()
    move()
    put_beeper()
    move()

function move_place_beeper_north:
    turn_left()
    move()
    put_beeper()

function move_place_beeper_west:
    turn_left()
    move()
    move()
    move()
    move()
    move()
    put_beeper()

function move_north:
    turn_left()
    turn_left()
    turn_left()
    move()

function move_place_beeper_east:
    turn_left()
    turn_left()
    turn_left()
    move()
    move()
    move()
    move()
    put_beeper()
```

```
function move_north:
    turn_left()
    move()

function move_place_beeper_west:
    turn_left()
    move()
    put_beeper()
    move()

function move_place_beeper_north:
    turn_left()
    turn_left()
    turn_left()
    move()
    put_beeper()
    move()

function move_place_beeper_west:
    turn_left()
    move()
    put_beeper()
    move()

fucntion move_place_beeper_north:
    turn_left()
    turn_left()
    turn_left()
    move()
    put_beeper()
    move()

function move_place_beeper_east:
    turn_left()
    turn_left()
    turn_left()
    move()
    put_beeper()
    move()
    put_beeper()
    move()
```

```
    put_beeper()
    move()
    put_beeper()
    move()

function move_place_beeper_south:
    turn_left()
    turn_left()
    turn_left()
    move()
    put_beeper()

function move_east_north:
    turn_left()
    move()
    turn_left()
    move()
    turn_left()
    turn_left()
    turn_left()
```

The Introduction together with individual Procedures and plan comprises the Experimental Plan and Conducting Experiment/ Activity criteria in the Final Laboratory Report Rubric:

CRITERIA	EXEMPLARY (90-100)	SATISFACTORY (80-89)	DEVELOPING (70-79)	BEGINNING (below 70)	WEIGHT
Experimental Plan (Flowchart/ Algorithm) ***(SO-PI: B1)***	Experimental plan has supporting details and diagram/algorithm that is stated and well explained	Experimental plan has supporting details and diagram/algorithm that is stated but not explained	Experimental plan is vague or brief. It has supporting details and doesn't have diagram/algorithm	No experimental plan presented	30%
Conducting Experiment/ Activity ***(SO-PI: B1)*** ***(SO-PI: K1)***	Objective and Materials used (modern engineering tools, software, and instruments/equipment) are identified. Steps are easy to follow for conducting and experiment/activity.	Objective is not stated. Materials used (modern engineering tools, software, and instruments/equipment) are identified. Steps are easy to follow for conducting and experiment/activity	Does not provide enough information to conduct an experiment/activity	No Objective, Materials used (modern engineering tools, software, and instruments/equipment), and steps for conducting experiment/activity provided.	20%

RESULTS AND DISCUSSION/COMPUTATIONS (*Include the program output screenshots, and discussions per problem solution*)

1. Familiarization Exercise 1 Result: GPA Calculator

Explanation:

In this Activity, we were assigned to develop a program that will calculate the GPA on 3 specific courses. The input of the program is expressed as a float since, we will be inputting the grades each course, and it is in a form of a decimal number. I categorized each subject in the variable (gpa1, gpa2, gpa3) separately. Before I write the code of the formula for computing the GPA, I checked first our course flowchart to know and identify the number of units of COEDISC, LBYCPA1, and CALENG1. The code is executed, and its formula would be the SUBJECT multiply by how many UNITS, apply this to the three subjects, get the sum and divide it with the total number of UNITS to calculate the raw GPA. Line 1 defined the function cal_GPA3 and the value would return on "return gpa" and it also indicates the end of the function. The final section of the code prints the output of the program which is the expected output or the calculated final

GPA grade.

```
def calc_GPA3(gpa1, gpa2, gpa3):
    # MAKE SURE THAT THE LINES BELOW ARE INDENTED
    # Assign the computed GPA to a variable named gpa


    gpa = ((gpa1 * 1) + (gpa2 * 2) + (gpa3 * 3)) / 6 #Formula of the GPA Calcula


    return gpa # make sure that the result is a float

# IMPLEMENT THE CODE BELOW THAT WILL HAVE THE SAME FORMAT AS THE SAMPLE DIALOG A
# Ask for the three GPAs and store them to gpa1, gpa2, and gpa3 variables
gpa1 = float(input("What is your COEDISC GPA? "))
gpa2 = float(input("What is your LBYCPA1 GPA? "))
gpa3 = float(input("What is your CALENG1 GPA? "))

# Calculate the GPA
gpa = calc_GPA3(gpa1, gpa2, gpa3)

print("Your GPA is: ", gpa)
# Replace this line with your code implementation
```

```
What is your COEDISC GPA? 2.5
What is your LBYCPA1 GPA? 3.0
What is your CALENG1 GPA? 3.5
Your GPA is:  3.1666666666666665
```

2. Familiarization Exercise 2 Result: Triangle Area Calculator

Explanation:

In this Problem, we are tasked to develop a Triangle Area Calculator; in which the user must enter the triangle's sides, and in this exercise, we were tasked to use the function of a square root to find the area of the triangle. I searched a formula on google and its name is Heron's Formula in which, we get the sum of the sides of the triangle divided by 2 to get the Perimeter of the triangle, and to find the Area we will use the function of the square root to find the Area:

Just use this two step process:

Step 1: Calculate **"s"** (half of the triangles perimeter):

$$s = \frac{a+b+c}{2}$$

Step 2: Then calculate the **Area**:

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

Importing the built-in function of python called math so we can use the operator of the square root; write the formula inside the function and as the value returns from the function, all we need to do is to print out the expected output or the final answer of the area and declaring the final answer as a float so the result would be exact.

```
# Indicate the name of the formula used just below this comment (put inside the 
FORMULA_NAME = "triangle_area"

import math
from math import sqrt # Hint: You have to use the square root function

def triangle_area(s1, s2, s3,):

    # --- Heron's Formula

    P = (s1 + s2 + s3)/2
    A = sqrt (P * (P - s1) * (P - s2) * (P - s3))

    return P
    return A # this must be of float type as well

s1 = float(input("Side 1: "))
s2 = float(input("Side 2: "))
s3 = float(input("Side 3: "))

A = triangle_area(s1, s2, s3,)

print("Triangle Area is: ", A)
```

```
Side 1: 3
Side 2: 4
Side 3: 5
Triangle Area is:  6.0
```

3. Familiarization Exercise 3 Result: Decimal Point Calculator

Explanation:

This is a tricky problem since we will only use arithmetic operators and type conversion functions. The program is expected for the user to input a number with 2 decimal places and the output would be the 2 decimal places that the user has inputted. The thought process of this problem is first, I subtracted the user's input minus the user's input in Integer form (Whole Number), the result would just be the 2 decimal places that the user has inputted and to make it a whole number; get the value of the result and multiply to transform the decimal into a whole number. Finally print the expected output and set the function into an integer so the output would only be a whole number.

```
def tenths_hundredths(num_float):
    # Note that the variable num_float is a float
    # The code must not use string manipulation functions,
    # only arithmetic operations and type conversion are allowed
    # The answer should be stored to result variable as an integer

    whole_num = num_float - int(num_float) # Subtract num_float to a its integer
    result = whole_num * 100 # Divide to make the decimal into a whole number

    return result # make sure that the result is an integer

num_float = float(input("Enter a number with a decimal: "))

result = int(tenths_hundredths(num_float)) # ---- Set into an Integer

print("Decimal Value:", result)
```

```
Enter a number with a decimal: 3.14
Decimal Value: 14
```

```
Enter a number with a decimal: 3.14
Decimal Value: 14
```

4. Familiarization Exercise 4 Result: 3-Digit Reverser

Explanation:

The user is required to input any 3-digit number and the program will reverse the placements of each digit. The program should perform only with arithmetic functions to reverse the digits and the result must be a string type. I divided my code into 3 parts so I can identify and take out the value of each place value.

- digit_1 = num % 10 * 100
    - The input would be divided by 100 and get the remainder of the **Ones' value** then multiply it by 100 to transform into a **Hundredths' place value**
- digit_2 = num % 100 // 10 * 10
    - The input would be divided by 100 and get the remainder of the **Tens' value**; use division integer and divide the result by 10, and multiply it by 10 to transform into a **Tens' place value**
    - You get the remainder of the tens value then // 10 to get the tens value and multiply by 10
- digit_3 = num % 1000 // 100 * 1
    - The input would be divided by 100 and get the remainder of the **Hundredths' value**; use division integer and divide the result by 10, and multiply it by 10 to transform into a **Ones' place value**
- result = str (digit_1 + digit_2 + digit_3)
    - In this line of code, it is basically getting the sum of all digits and making sure that the data would be saved.

Finally, as the function returns the value; The program would print out the expected output / reversed digits.

```python
def reverse3(num):
    # Note that the variable num is an integer
    # The code must not use string manipulation functions,
    # only arithmetic operations and type conversion are allowed
    # Hint: You need to use modulo and integer division operations

    # Modulo - remainder - %
    # Integer division - divides to a whole number only - //

    digit_1 = ((num % 10) * 100)
    # You get the remainder of the ones value then multiply by 100

    digit_2 = (((num % 100) // 10) * 10)
    # You get the remainder of the tens value then // 10 to get the tens value and multiply by 10

    digit_3 = (((num % 1000) // 100) * 1)
    # You get the remainder of the hundreds value then // 100 to get the hundreds value and multiply by 1


    result = str( digit_1 + digit_2 + digit_3 )
    # Add all digits
    # Make sure to save the reversed digits to the result variable and
    # convert it to a string

    return result # the result variable must be of string type

num = int(input("Enter a Number: "))
result = (reverse3(num))
print(num, " == ", result)

# print(type(reverse3(753))) # must display 'str'
```

```
Enter a Number: 123
123  ==  321
```

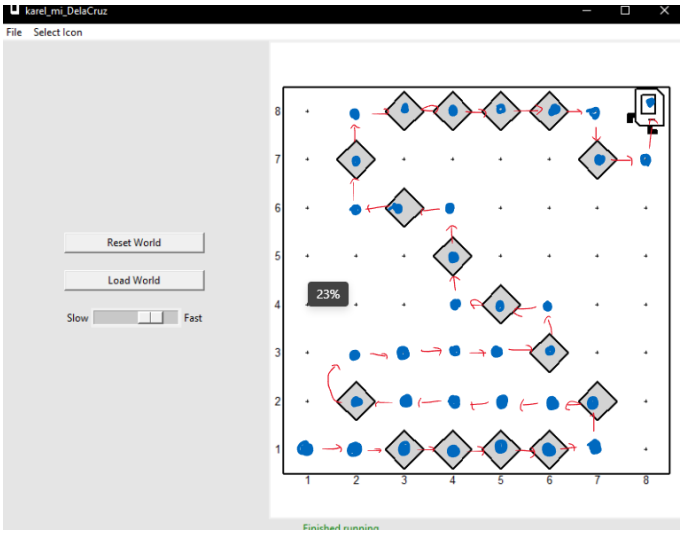
```
Enter a Number: 123
123  ==  321
<class 'str'>
```

5. Familiarization Exercise 5 Result: Karel the Bot

Explanation:

In this activity, I first had to install the Karel Python package through the Anaconda and used Pycharm.edu to create and input the code for Karel the Robot. We used the Karel functions to accomplish the activity to spell out my middle initial of S in the Karel 2D environment. The functions are the following:

- move()
  - This function instructs the Karel Bot to move one tile on what direction is the bot facing.
- put_beeper()
  - This function instructs the Karel Bot to move one tile on what direction is the bot facing.
- turn_left()
  - This function instructs the Karel Bot to rotate their direction to 90 to the left / counterclockwise

- 

The Results and Discussions constitutes the data criteria in the Lab Report Evaluation Rubric:

Codes/Data/ Program	Data is well utilized in the program. Program codes are easy to read. Program output has no error. Questions are answered completely and correctly	Data is somewhat utilized in the program. Program code are easy to read. Program output has an output but logically incorrect. Some questions are answered completely and correctly	Data is not utilized in the program. It has a missing significant code/syntax in the program.	No program presented	30%

CONCLUSION:

The second module focuses on the fundamentals of Python. The best programming language is foreign to my understanding, and I've enjoyed learning it and experimenting on it. I wish to have a better understanding and develop an appreciation for programming. Like every module's objective, we should assure ourselves that we understand and execute the programs smoothly.

I gained a lot of understanding about Python, and I am getting used to it. The module aided me tremendously throughout the coding process, from accessing my notes to pointing up errors in my code, and even creating a cell in the Jupyter Notebook to experiment what those functions can do and how can I manipulate it. I've

learned to be patient because there were a lot of errors as I process and execute my codes. It's a trial-and-error basis for me, and I hope in the future I can easily find and analyze my code right away. Overcoming the error would need me to be more vigilant and precise with my coding and spelling. I feel that it is something that I can work on and improve upon. Apart from that, I loved this problem set and can honestly claim that I learnt a lot. This module helped me think outside the box and trying to think simple solutions to solve these problems, just like the problem in which we will get the value of the decimal value, it trains us to be critical thinkers and to analyze the problems better. There are a lot to improve upon. Apart from that, I honestly claim that I learnt a lot.

My advice to individuals attempting this module for the first time is that you must truly listen to the lectures and comprehend the module. Third-party resources such as w3 schools and stackOverflow can assist you if you are having difficulty understanding the modules and the syntax. Another piece of advice is that when it comes to coding, you must be conversant with variable naming and most especially the indentions, Python is very sensitive with indentions and syntax.

The rest of the rubric criteria are as follows:

Grammar, logical presentation, and format ***(SO-PI: G1)***	The report was grammatically correct, logically presented and used the required format.	The report had minimal grammatical errors and somewhat presented logically. The required format was used.	The report had a lot of grammatical errors and not logically presented; the required format was barely used.	The report had a lot of grammatical errors, was not logically presented and the required format was not used.	20%

REFERENCES (*Enumerate references in APA format*)

- w3school (n.d.). Retrieved from. https://www.w3schools.com/python/python_intro.asp
- stackOverflow (n.d.). Retrieved from. https://stackoverflow.com/
- Math is Fun. (n.d.). Heron's Formula. Retrieved from. https://www.mathsisfun.com/geometry/herons-formula.html

APPENDIX (*Attach all the source codes here per problem category*)

1. Exercise 1: GPA Calculator

```
def calc_GPA3(gpa1, gpa2, gpa3):

    gpa = ((gpa1 * 1) + (gpa2 * 2) + (gpa3 * 3)) / 6

    return gpa

gpa1 = float(input("What is your COEDISC GPA? "))
gpa2 = float(input("What is your LBYCPA1 GPA? "))
gpa3 = float(input("What is your CALENG1 GPA? "))

gpa = calc_GPA3(gpa1, gpa2, gpa3)

print("Your GPA is: ", gpa)
```

2. Exercise 2: Triangle Area calculator

```
FORMULA_NAME = "triangle_area"
import math
from math import sqrt

def triangle_area(s1, s2, s3,):

    # --- Heron's Formula
    P = (s1 + s2 + s3)/2
    A = sqrt (P * (P - s1) * (P - s2) * (P - s3))

    return P
    return A

s1 = float(input("Side 1: "))
s2 = float(input("Side 2: "))
s3 = float(input("Side 3: "))

A = triangle_area(s1, s2, s3,)

print("Triangle Area is: ", A)
```

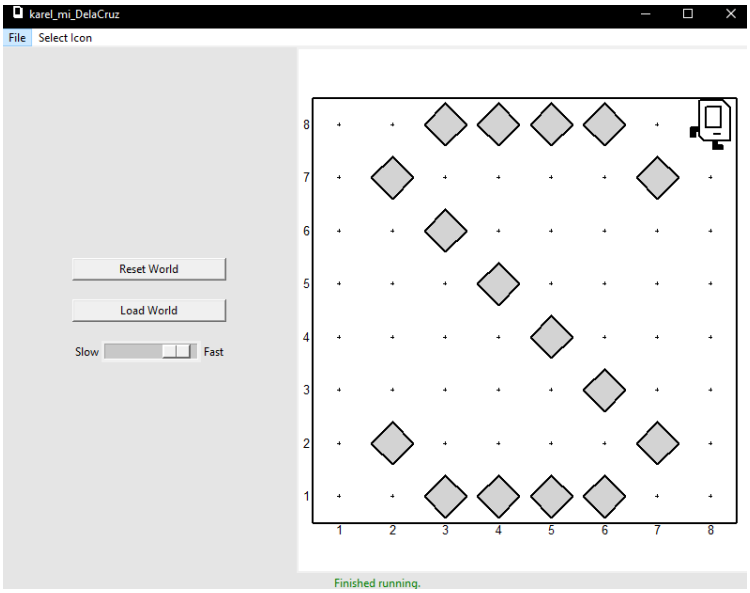
3. Exercise 3: Decimal points calculator

```
def tenths_hundredths(num_float):

    whole_num = num_float - int(num_float)
    result = whole_num * 100

    return result

num_float = float(input("Enter a number with a decimal: "))

result = int(tenths_hundredths(num_float))

print("Decimal Value:", result)
```

4. Exercise 4: 3-digit reverser

```
def reverse3(num):

    digit_1 = ((num % 10) * 100)

    digit_2 = (((num % 100) // 10) * 10)

    digit_3 = (((num % 1000) // 100) * 1)

    result = str( digit_1 + digit_2 + digit_3 )

    return result

num = int(input("Enter a Number: "))
result = (reverse3(num))
print(num, " == ", result)
```

5. Exercise 5: Karel




```
from stanfordkarel import *

def main():
    """Karel code goes here!"""
    # move() - move
    # turn_left() - rotate 90 degrees
    # put_beeper()
    move()
    move()
    put_beeper()
    move()
    put_beeper()
    move()
    put_beeper()
    move()
    put_beeper()
    move()
    turn_left()
    move()
    put_beeper()
    turn_left()
    move()
    move()
    move()
    move()
    move()
    put_beeper()
    turn_left()
    turn_left()
    turn_left()
    move()
    turn_left()
    turn_left()
    turn_left()
    move()
    move()
```

```
    move()
    move()
    put_beeper()
    turn_left()
    move()
    turn_left()
    move()
    put_beeper()
    move()
    turn_left()
    turn_left()
    turn_left()
    move()
    put_beeper()
    move()
    turn_left()
    move()
    put_beeper()
    move()
    turn_left()
    turn_left()
    turn_left()
    move()
    put_beeper()
    move()
    turn_left()
    turn_left()
    turn_left()
    move()
    put_beeper()
    move()
    put_beeper()
    move()
    put_beeper()
    move()
    put_beeper()
    move()
    turn_left()
    turn_left()
    turn_left()
    move()
    put_beeper()
    turn_left()
    move()
    turn_left()
    move()
    turn_left()
    turn_left()
    turn_left()

if __name__ == "__main__":
    run_karel_program()
```