Lab 9 – CSE 101 (Spring 2019)

1. Objectives

The primary objectives of this lab assignment are:

- CSV file processing using Python, and
- Object oriented programming using Python.

2. CSV file processing with Python

Visit the following link that explains how to use csv library in Python for reading csv files. The graduate TAs will walk you through the article along with examples to clarify concepts of parsing csv files using Python.

https://realpython.com/python-csv/#writing-csv-file-from-a-dictionary-with-csv

The following sections will be covered:

- Reading CSV files with csv
- Reading CSV files into a dictionary with csv
- Writing CSV files with csv
- Writing CSV file from a dictionary with csv

3. Object Oriented Programming

Download student.py and use_student.py into your lab9 folder and do the following in the given order.

1. Add the following method to Student class in student.py

```
def __init__(self, name, id, major, gpa):
    self.name = name
    self.id = id
    self.major = major
    self.gpa = gpa
```

   Note that there is no underscore in front of an instance variable. This is a convention that some people use, but it is not that common.

2. Add the following two lines to main in use_student.py and run it.

```
s1 = student.Student('Amy', 1, 'CS', 3.21)
print('s1:', s1)
```

   Make sure you understand what the `__init__` method is doing in this context. That is, the constructor `Student(...)` call invokes the `__init__` method. Also note what the output looks like: or something like that. It means that `s1`is an object that is found in memory location `0x1021442b0>` (a hexadecimal, i.e., base 16 number). In other words, that is the string representation of the object `s1`. Hm... Wouldn't you like to see a more meaningful string representation of the object than that? Well, the next step will change that.

3. Add the following method to Student class in student.py and run the main in use_student.py.

   ```
   def __repr__(self):
       return '(' + self.name + ', ' + self.major + ')'
   ```

   Well, do you see a better string representation of `s1` printed now? You can see that `print` function requires a string form to display. When a string representation of an object such as `s1` is needed, the Python system calls the special method `__repr__` automatically and use the return value of the method.

4. Add the following two lines to `main` in `use_student.py` and run it.

   ```
   s2 = student.Student('Ken', 2, 'TSM', 3.42)
   print('s2:', s2)
   ```

5. Add the following method to `Student` class in `student.py`.

   ```
   def __eq__(self, other):
       return self.id == other.id
   ```

6. Add the following two lines to `main` in `use_student.py` and run it.

   ```
   print('s1 == s1:', s1 == s1)
   print('s1 == s2:', s1 == s2)
   ```

   The == operator automatically triggers a call to the `__eq__` method.

7. Add the following method to `Student` class in `student.py`.

   ```
   def __lt__(self, other):
       return self.gpa < other.gpa
   ```

8. Add the following two lines to `main` in `use_student.py` and run it.

   ```
   print('s1 < s1:', s1 < s1)
   print('s1 < s2:', s1 < s2)
   ```

   The < operator automatically triggers a call to the `__lt__` method.

9. Similarly add one for the 'greater than' ('>') operator.

10. The methods that we have added to `Student` so far are called special methods. There are more special methods of this kind, but I will leave it up to your exploration. Instead, now we will add some regular kind of methods. Add the following method to `Student` class in `student.py`.

```
    def change_major(self, new_major):
        self.major = new_major
```

11. And, add the following two lines to `main` in `use_student.py` and run it.

```
  s1.change_major('TSM')
  print('s1:', s1)
```

    and verify that `s1`'s major is now changed to `TSM`.

12. Let's add one more regular method. Add the following method to `Student` class in `student.py`.

```
    def change_gpa(self, new_gpa):
        self.gpa = new_gpa
```

13. And, add the following two lines to `main` in `use_student.py` and run it.

```
  s1.change_gpa(s1.gpa + 0.3)
  print('s1.gpa:', s1.gpa)
```

    and verify that s1's GPA is now changed to a new value. Note how an instance variable is accessed in the main using a dot notation.

14. We can even create a list of `Student` objects and do something with it. Add the following in the `main` in `use_student.py` and run it.

```
  tsmers = [s1, s2]
  sum = 0.0
  for s in tsmers:
      sum = sum + s.gpa
  print('Average GPA = ' + str(sum/len(tsmers)))
```

15. This gives you a quick tutorial on how to create a class, how to create some objects using the class, and use them in a user code, for example in the main of `use_student.py`. Now, that you are familiar with this process, let's use it to solve some real problems.

4. Problem: Polar Coordinates of a Point

Create a file named polarcoord.py and follow the instructions below.

Define a class named Point that will represent a point on a graph. To create a point pass the x and y co-ordinates to the constructor:

```
>>> p1 = Point(1,1)
>>> p2 = Point(4,5)
```

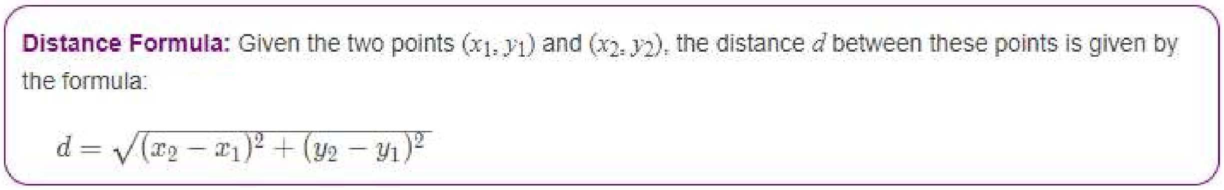
Include the following methods in your class (the examples refer to the two points p1 and p2 shown above):

• The __repr__ method should display the point in standard mathematical notation, e.g.

```
>>> p1
(1,1)
```

• A method named `dist` should compute the distance between two points, e.g.

```
>>> p1.dist(p2)
5.0
```



Distance Formula: Given the two points (x_1, y_1) and (x_2, y_2), the distance d between these points is given by the formula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Read more: https://www.purplemath.com/modules/distform.htm

• A method named polar should return a pair of values corresponding to the polar coordinates of the point:

```
>>> p1.polar()
(1.4142135623730951, 0.7853981633974483)
```

The polar coordinates of a point (x, y) are a pair of numbers (r, q) where r = sqrt(x^2 + y^2) and and q = $\tan^{-1}$ y/x (Python's math library has a function named atan that computes $\tan^{-1}$). The formula for polar coordinates is valid only if the x-coordinate of a point is greater than 0. The polar method should return None if x is negative or 0.

Read more: http://tutorial.math.lamar.edu/Classes/CalcII/PolarCoordinates.aspx

5. Submission

Submit completed `polarcoord.py` program on blackboard.