COMP 348: Principles of Programming Languages Assignment 2

Summer 2023, sections AA and AB

June 6, 2023

Multi-paradigm Programming with Python

Date posted: Tuesday, June 6th, 2023.

Date due: Sunday, June 18th, 2023, by 23:59.

Weight: $8\frac{1}{3}\%$ of the overall grade.

Overview

In this assignment you implement an interactive python application that reads and processes some databases of shapes in text format. Supported shapes are: *Shape* (the generic shape object), *Circle*, *Ellipse*, and *Rhombus*.
Upon loading a database file, the program constructs every shape by calling its class constructor and keeps the object in memory. Repetitions are allowed, and therefore the resulting collection is a multi-set.

The assignment does not enforce any particular file structure. You may use your own module structure. However, using good practices is encouraged. The only restriction is that your main module must be put in the file named “shapedb.py”. All required files must be accessed within the same directory that the “shapedb.py” module is located.

Operations

The following functions / operations of the program are as follows, which may be selected by the user via a Command Line Interface (CLI) menu system:

- LOAD «file»: loads a database of shapes; the database is a multi-set; the «file» represents a file in the filesystem.
- TOSET: converts the current multi-set in memory to a set (removes duplicates).
- SAVE «file»: saves the current in-memory database to a file.
- PRINT: prints the current in-memory database to the standard output.
- SUMMARY: prints the summary of the in-memory database to the standard output.
- DETAILS: prints the detailed information of the in-memory database objects to the standard output.
- QUIT: terminate the program.

Note that in the *TOSET* operation, the repetitions are removed, in which case, the equality of the elements is determined by the shape name as well as all the parameters (i.e. two circles with different radiuses would be considered as two different objects).

[Database] File Format

The database in this assignment is a file that contains shape information. Each shape is presented in a separate row. Every line in the text file consist of shape name and parameters to construct the shape. A generic *Shape* does not have any parameter; A *Circle* receives a radius parameter; An *Ellipse* receives two parameters (the semi-major and semi-minor axes, in no specific order); A *Rhombus* receives two diagonals. An example is given below:

```
shape
rhombus 10 20
circle 2
ellipse 2 4
shape
ellipse -1 4
rhombus 20 10
ellipse 2 4
```

In the above example, the second ellipse record is erroneous and therefore the program displays an error message and continues loading the next rhombus. The erroneous objects may not be added to the database.

```
Error: Invalid Ellipse on line 6: ellipse -1 4
```

The error messages are not limited to the processing errors. Any program error (e.g. invalid file/path, etc.) must be reported as well. Before processing a file, the program displays a message indicating that it is processing the file.

```
Processing <<path-to-file>>...
```

After the processing is finished, the program displays a message indicating how many rows were processed.

```
Processed 8 row(s), 7 shape(s) added, 1 error(s).
```

In the above example, while the two rhombuses may look equal, they are considered different objects. The first and the last ellipse, however, are equal. The two generic shapes are equal, as well.

[Database] Objects

The database objects are implement using the following class hierarchy, by which the attributes and methods of each class are specified:

1. Shape

   (a) the class constructors receives no parameter.

   (b) id: a read-only integer id, automatically assigned to all shapes upon creating; first object: 1, second object 2, and so on.

   (c) method print(): prints the id and the name of the shape, the perimeter, and the area. The name of the shape is the class name (i.e. Shape for this class). This method must dynamically read the class name at runtime (in child classes this method correctly prints the class name, without being explicitly implemented[1]).

[1]See how to obtain runtime class name in Python.

(d) method perimeter(): default method to be implemented by child classes; returning nil by default.

(e) method area(): default method to be implemented by child classes; returning nil by default.

2. Circle

   (a) the class constructor receives the radius as the only parameter.

   (b) method perimeter(): overridden, returns the perimeter of the circle.

   (c) method area(): overridden, returns the area of the circle.

3. Ellipse

   (a) the class constructor receives the semi-major and semi-minor axes: a and b, in an arbitrary order. The greater and the lesser values of the two parameters are to be assigned to the semi-major and the semi-minor axes, respectively.

   (b) method perimeter(): not to be implemented.

   (c) method area(): overridden, returns the area of the ellipse: $A = \pi ab$.

   (d) method eccentricity(): additional method that returns the linear eccentricity of the ellipse: $c = \sqrt{a^2 - b^2}$. In case of error, the method returns nil.

4. Rhombus

   (a) the class constructor receives the two diagonals.

   (b) method perimeter(): overridden, returns the perimeter of the rhombus.

   (c) method area(): overridden, returns the area of the rhombus.

   (d) method side(): returns the side of the rhombus: $a = \sqrt{p^2 + q^2}/2$.

   (e) method inradius(): calculating the radius of a circle inscribed in the rhombus: $r = p \cdot q/(2\sqrt{p^2 + q^2})$, where p, q denote the diagonals. In case of error, the method returns nil.

Program Output

For all operations except for *PRINT*, *SUMMARY*, and *DETAILS*, the program must make sure your program appropriately reports the result to the user. It is advised that your write the program is verbose mode. For example if a multi-set is converted to a set, you may display a message that indicates so. The messages for processing database files were discussed above, as well.

The *PRINT* operation calls `print()` method on each object and displays the result on the screen. A sample output is given below:

```
1: Shape, perimeter: undefined, area: undefined
2: Rhombus, perimeter: 44.72136, area: 100 in-radius: 4.47214
3: Circle, perimeter: 12.56637, area: 12.56637
4: Ellipse, perimeter: undefined, area: 25.13274, linear eccentricity: 3.46410
5: Shape, perimeter: undefined, area: undefined
6: Rhombus, perimeter: 10, area: 0, side: 10, in-radius: 0
7: Ellipse, perimeter: undefined, area: 25.13274, linear eccentricity: 3.46410
```

The *SUMMARY* operations shows a stat summary of the objects in the database. The statistics contain the total number of shapes per item, ordered alphabetically, followed by the total number of "shapes". You may use dict to implement the above structure in memory. An example is given below.

```
Circle(s): 1
Ellipse(s): 2
Rhombus(es): 2
Shape(s): 7
```

Note that rhombuses, circles, and ellipses are counted as shapes as well.

And finally, the *DETAILS* operation lists all objects in a format that may be saved in the database:

```
shape
shape
circle 2
ellipse 2 4
ellipse 2 4
rhombus 10 20
rhombus 20 10
```

If prior to the *DETAILS* operation, had the user issued a *TOSET* operation, the following result would have been displayed (or perhaps saved to a file upon calling *SAVE* operation.

```
shape
circle 2
ellipse 2 4
rhombus 10 20
rhombus 20 10
```

Naming Convention

While using proper phyton naming convention is expected in class names, attributes, and methods, the shape names in the database are case insensitive. The *DETAILS* and *SAVE* operations print names in small letters (see above examples). The *PRINT* and *SUMMARY* operations use the class-names in your Python code (i.e. `Circle` as opposed to `circle` in the database).

Submission

Your submission will have multiple source files. At the very least it will have a file called `shapedb.py`. Feel free to add additional python files to more easily manage your code. Do not include any database files, since the marker must use their own test set to ensure that all students are evaluated in the same way. Once you are ready to submit, place the .py files into a zip file. The name of the zip file will consist of "a2" + last name + first name + student ID + ".zip", using the underscore character "_" as the separator. For example, if your name is John Smith and your ID is "123456", then your zip file would be combined into a file called "`a2_Smith_John_123456.zip`". The final zip file will be submitted through the course web site on Moodle. You simply upload the file using the link on the assignment web page.

Please note that it is your responsibility to check that the proper files have been uploaded. No additional or updated files will be accepted after the deadlines. You can not say that you accidentally submitted an "early version" to Moodle. You are graded only on what you upload.

Note that assignments must be submitted on time in order to receive full value. Appropriate late penalties (< 1 hour $= 10\%$, < 24 hours $= 25\%$) will be applied, as appropriate.

References

1. File I/O:
   `https://www.programiz.com/python-programming/file-operation`
2. Class name in Python:
   `https://favtutor.com/blogs/class-name-python`