COMP 348: Principles of Programming Languages
Assignment 3 on Python and PROLOG

Fall 2021, sections U and DD

November 2, 2021

Contents

1 General Information

Date posted: Wednesday November 3rd, 2021.

Date due: Tuesday, December 07th, 2021, by 23:59[1].

Weight: 5% of the overall grade.

2 Introduction

This assignment targets 1) Multi-paradigm Programming using Python, 2) Logic Programming using PROLOG.

3 Ground Rules

You are allowed to work on a team of 3 students at most (including yourself). Each team should designate a leader who will submit the assignment electronically. See Submission Notes for the details.
ONLY one copy of the assignment is to be submitted by the team leader. Upon submission, you must book an appointment with the marker team and demo the assignment. All members of the team must be present during the demo to receive the credit. Failure to do so may result in zero credit.
This is an assessment exercise. You may not seek any assistance from others while expecting to receive credit. **You must work strictly within your team).** Failure to do so will result in penalties or no credit.

4 Your Assignment

Your assignment is given in two parts, as follows. 1) Multi-paradigm Programming using Python, 2) Logic Programming using PROLOG.

[1]see Submission Notes

4.1 Multi-paradigm Programming using Python

4.1.1 Lists and Generators

Q 1. Define a function that receives a number n and prints the sequence of Lucas numbers that are less than n.

`https://brilliant.org/wiki/lucas-numbers/`

The Lucas sequence has the same recursive relationship as the Fibonacci sequence, where each term is the sum of the two previous terms, except that the first two numbers in the sequence are: 2, and 1. The first few elements of the sequence are: $2, 1, 3, 4, 7, 11, 18, ...$

Q 2. Define a function that receives a number n and returns the first n numbers of a Lucas sequence in a list.

(a) Implement using a regular function using loops and parallel assignments

(b) Implement using a generator function

4.1.2 Sets, Multisets, and Dicts

Q 3. Write a function that receives a sequence and create a "set" using `list` collection. Do not use `set`.

Q 4. Python does not have a built-in support for multisets. Design a class that implements the multi-set functionaries:

(a) Adding an element to the set.

$\{1, 2, 3\} + 1 = \{1, 1, 2, 3\}$

(b) Removing all occurrences of an element from a set.

$\{1, 1, 2, 3\} \backslash 1 = \{2, 3\}$

(c) Determining the multiplicity (repetition) of an element in the set.

$m(\{1, 1, 1, 2, 2, 3\}, 1) = 3$

(d) Union of the set with another set: max multiplicity per element, a new set is returned

$\{1, 2\} \cup \{2, 2, 3\} = \{1, 2, 2, 3\}$

(e) Intersection of the set with another set: min multiplicity per element

$\{1, 1, 2, 2, 3\} \cap \{2, 2, 2, 3, 4\} = \{2, 2, 3\}$

(f) Difference Update: removing elements of another set from the current set and updating the current set.

$\{1, 1, 1, 2, 2, 3\} - \{1, 2, 2, 2\} = \{1, 1, 3\}$

4.1.3 Files and Text Processing

This section consists of three parts that are related, but may be implemented independently.

Q 5. Classes in Python

Define the following class hierarchy in Python

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 explicitly being implemented).
   (d) method perimeter(): default method to be implemented by child classes; returning `None` by default.
   (e) method area(): default method to be implemented by child classes; returning `None` 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 `None`.

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 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 `None`.

Q 6. File / Text Processing

Write a Python program that reads text file that contains the shape information (see previous question). Every line in the text file consist of shape name and parameters to construct the shape. The program creates the shapes and calls the print method for each newly created shape and displays the result on the screen. In case the shape is an ellipse, the linear eccentricity is displayed in the following line. In case of a rhombus, the in-radius of the rhombus is displayed in the next line. In case of errors (i.e. having negative values for axes, radii, etc.), the object is created, however an error message is displayed on the console. As such, no further detail is to be displayed (i.e. no eccentricity is shown for an invalid ellipse). An example given below:
Input:

```
shape
rhombus 10 20
circle 2
ellipse 2 4
shape
ellipse -1 4
rhombus 5 0
```

Output:

```
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
Error: Invalid Ellipse
6: Ellipse, perimeter: undefined, area: undefined
7: Rhombus, perimeter: 10, area: 0
  in-radius: 0
```

Q 7. Arrays and Dicts

Extend the above program to display the statistics after processing the file. The statistics contain the total number of shapes per item, ordered alphabetically, followed by the total number of "shapes". An example is given below.
Output:

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

Use dict to implement the above structure in memory.
Note that rhombuses, circles, and ellipses are counted as shapes as well.

4.2 Logical Programming with PROLOG

Fact Representation, Queries, Unification, and Resolution

Q 8. Provide a knowledge-base of clauses specifying you and your team's courses in PROLOG.

- In your database include your student information (name and id) as well as all courses that each member of the team is taking this semester. The courses include course name and course number.
- Write a query to return the list of courses taken by each member.
- Write a query to return the team size.
- Write a query to return the unique courses taken by the whole team.
- Use `sort/2` to sort the result of the previous query.
- Unify the expression `[A,B|C]` with the above result. Provide the values for A, B, and C.

Short Programs

Q 9. Using `append/3`, write a Prolog query that receives a list, a start index, and a length, and returns the sublist of the list. Use zero-based indexing.

Examples are given in the following:

```
?- sublist([1, 2, 3, 4], 1, 2, O)
O = [2, 3].
?- sublist([1, 2, 3, 4], 0, 0, O)
O = []
?- sublist([1, 2, 3, 4], 0, 10, O)
false
```

Q 10. Write a prolog procedure `every-other/2` that receives a source list as its first argument and produces a list in the second argument whose elements are the elements of odd indexes in the source list.

Examples are given in the following:

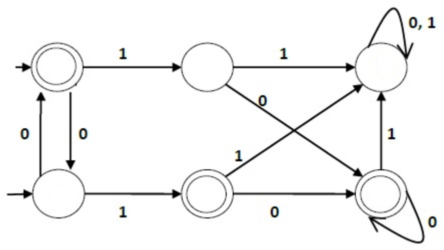
```
?- every-other([], L)
L = [].
?- every-other([1], L)
L = [1].
?- every-other([1, 2], L)
L = [1].
?- every-other([1, 2, 3], L)
L = [1, 3].
?- every-other([1, 2, 3, 4], L)
L = [1, 3].
```

Make sure your procedure is efficiently terminated.

Q 11. Write a Prolog query with arity 2 to return the first n numbers of a Lucas sequence in a list.

PROLOG Applications

Q 12. Given the Finite State Machine in the image below,



1. Represent the FSM in Prolog.
2. Write a Prolog query to determine whether the four sequences of "`0`", "`1`", "`0 1`", and "`1 0`" are accepted by the machine.
3. Run the queries and verify the answers.

5 What to Submit

The whole assignment is submitted by the due date under the corresponding assignment box. Your instructor will provide you with more details. It has to be completed by ALL members of the team in one submission file.

Submission Notes

Clearly include the names and student IDs of all members of the team in the submission. Indicate the team leader.

IMPORTANT: You are allowed to work on a team of 3 students at most (including yourself). Any teams of 4 or more students will result in 0 marks for all team members. If your work on a team, ONLY one copy of the assignment is to be submitted. You must make sure that you upload the assignment to the correct assignment box on Moodle. No email submissions are accepted. Assignments uploaded to the wrong system, wrong folder, or submitted via email will be discarded and no resubmission will be allowed. Make sure you can access Moodle prior to the submission deadline. The deadline will not be extended.

Naming convention for uploaded file: Create one zip file, containing all needed files for your assignment using the following naming convention. The zip file should be called a#_studids, where # is the number of the assignment, and studids is the list of student ids of all team members, separated by (_). For example, for the first assignment, student `12345678` would submit a zip file named `a1_12345678.zip`. If you work on a team of two and your IDs are `12345678` and `34567890`, you would submit a zip file named `a1_12345678_34567890.zip`. Submit your assignment electronically on Moodle based on the instruction given by your instructor as indicated above: `https://moodle.concordia.ca`
Please see course outline for submission rules and format, as well as for the required demo of the assignment. A working copy of the code and a sample output should be submitted for the tasks that require them. A text file with answers to the different tasks should be provided. Put it all in a file layout as explained below, archive it with any archiving and compressing utility, such as WinZip, WinRAR, tar, gzip, bzip2, or others. You must keep

a record of your submission confirmation. This is your proof of submission, which you may need should a submission problem arises.

6 Grading Scheme

Q1	8 marks
Q2	7 marks
Q3	5 marks
Q4	10 marks
Q5	15 marks
Q6	10 marks
Q7	5 marks
Q8	10 marks
Q9	5 marks
Q10	8 marks
Q11	10 marks
Q12	7 marks

Total: 100 pts.

References

1. Lucas Sequence: `https://brilliant.org/wiki/lucas-numbers/`
2. Multiset: `https://en.wikipedia.org/wiki/Multiset`