Department of Mathematics and Computer Science
University of Southern Denmark, Odense

November 3, 2009
Marco Chiarandini

DM533 - Introduction to Artificial Intelligence

Assignment 0, Fall 2009

The submission deadline for this assignment is 11.59 of Thursday, November 12.

We will use Python during the course. The amount of programming required is not large, and you are not expected to be an expert in python.

To get you familiarized with this programming language and the submission procedure, you are asked to submit some code for this assignment. The evaluation on this assignment is the following: if you make an honest effort at most of the questions you will pass, otherwise fail. As with any programming assignment, include with each answer a transcript showing that the code works on a few different inputs. The name for the assignment is `ass0` and your solution files should be called as indicated below. Your file should run on Python at IMADA machines and should include explanations and output traces (as needed) in portions commented with semicolons. Please try to comment your code and use meaningful variable names.

For submission instructions, please see the Appendix. It contains information on how to submit this and all further assignments. Failure to submit properly could result in a non-passed assignment.

Exercises

1. Exercise 3.8. A biref textual reply can be submitted in `doc` with name `ex8.txt` (you may skip the plot of point a) in the reponse).

2. Read the short Python tutorial available from the course web page and submit solutions for your `buyLotsOfFruits` function and your `shopSmart` function in a file called `buyLotsOfFruits` and
`shopSmart.py`, respectively.

3. Write a python function `count-trees(n)` which returns the number of distinct strictly binary trees with n leaves. A strictly binary tree is one in which every node other than a leaf has exactly two children. A leaf has no children. There is 1 strictly binary tree with 1 leaf (just a single node tree), 1 strictly binary tree with 2 leaves, 2 distinct strictly binary trees with 3 leaves, and 5 distinct strictly binary trees with exactly 4 leaves. `count-trees(n)` can be implemented fairly easily using recursion. Test your program for $n = 1$ through 10. [Hint: Suppose a tree with n leaves has a leaves in its left subtree and b leaves in its right subtree, where $a + b = n$; how many such trees are there?] Optional: measure the runtime as a function of n. File name: `counttrees.py`.

4. A standard mathematical set function is powerset, which computes the set of all subsets of a set. For example,

```
>>> powerset("a, b")
[None, (a), (b), (a b)]
```

Implement this as a recursive function. It may require other subsidiary functions.

File name: `powerset.py`

5. Data types

   a. Write class for points and line segments in two dimensions.
   b. Write a function `distance(p1,p2)` that returns the distance between two points.
   c. Write a function `midpoint(l)` that returns the midpoint of a line segment.
   d. Write a function `intersectp(l1,l2)` that decides if two line segments intersect. [Hint: the location of an arbitrary point on AB can be written as `vA + (1-v)B`; a point on CD is `wC +(1-w)D`; the lines cross where these are equal. This gives two equations (equating both x and y parts) for `v` and `w`. Solve these to find the intersection point, if any. Then you need to check that the intersection is actually on both segments - this can be determined by looking at the values of `v` and `w`.]
   e. A polygon can be defined by a list of points, where each point is assumed to be connected to the next. Define methods for calculating the area of a polygon. Begin with subtypes such as rectangle, square, triangle, regular polygon. Formulas for these can be found on the web. Optionally, you can provide a method that works for any polygon. You can assume that the list of points supplied to build any particular shape does in fact describe such a shape.

   File name: `geometry.py`.

6. Download the AIMA (textbook) code in http://code.google.com/p/aima-python/. Make sure it works by compiling the utilities and agents subsystems. If you want, you can try to write a better vacuum world agent. See the overview and the instructions on how to use the code and become familiar with it. Read Exercises 2.7-2.12 that take you through the process of developing an environment class.

7. Using the methods at the previous item, try doing Exercise 3.19 from [B1].

   File name: `vacuum.py`.

Appendix A Submission Instructions

An archive containing the electronic version of documents and source code of the program must be handed in through the Blackboard system within the deadline. The procedure is the following:

- choose the course DM533 in Blackboard,
- choose ”Assignment Hand in” in the menu on the left,
- fill the form and conclude with submit,
- print the receipt (there will be a receipt also per email).

The electronic archive to hand in must be organized as follows. It expands in a main directory named with the assignment name (eg, `ass0`) and 10 digits of your CPR number (e.g., `ass0-0309073090`, no separation between first 6 and last 4 digits). The directory has the following content:

```
ass0-CPRN/README
ass0-CPRN/doc/
ass0-CPRN/src/
```

The directory `doc` contains written replies to the exercises posed where there is a requirement for this. Documents in this directory must have the name mentioned in the exercise and be in .pdf or plain text format (extension .txt) *Do not put your name in the author field of these documents, your CPR number in the directory name will identify your work*. The file README provides instructions or specifications for the programs. The directory `src` contains the python sources. Names of these files depend on the assignment.

Programs must work on IMADA's computers under Linux environment and with the compilers and other applications present on IMADA's computers. Students are free to develop their program at home, but it is their own responsibility to transfer the program to IMADA's system and make the necessary adjustments such that it works at IMADA.

Reports and codes handed in after the deadline will generally not be accepted. System failures, illness, etc. will not automatically give extra time.