6CCS3AIN/7CCSMAIN (Artificial Intelligence)

Practical 01: Getting started with Pacman

(Version 1.2)

1 Overview

This practical is the first step towards the first coursework. In it you will get setup to run the Pacman game that will be the basis of the corsework for the module, and write your first code to control Pacman.

Since the work on Pacman will include working with Python, these instructions also have some pointers to Python resources, which may be helpful, and details of how to install Python on thelab machiens should you need to do that.

2 Start with Python

Python has been installed on the Informatics computing environment. As a result, you should be able to run it directly. Try this:

1. Boot the lab machine into CentOS Linux (this is default)
2. Log in.
3. Open a Terminal window.
4. Type:

   `python`

   on the command line.

You should get something like this:

```
Python 2.7.12 |Anaconda 4.2.0 (64-bit)| (default, Jul  2 2016, 17:42:40)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
>>>
```

You won't get exactly this — this is what I get on my laptop, and the version of Python installed in the lab[1] is 2.7.5, but you should be able to see the Python environment starting up and reporting for duty. If you do, you are ready for Section 3.

If you get an error, sorry, but there is a problem which could be to do with your account, or with the particular setup on the machine you are using. Please talk to one of the TAs.

[1] In fact there are two versions of Python installed in the lab, Python 2.7 and Python 3. Python 2.7.5 is the version that runs for me when I use the command `python`.

3 Start with Pacman

The Pacman code that we will be using for the 6CCS3AIN/7CCSMAIN labs and coursework was developed at UC Berkeley for their AI course. The folk who developed this code then kindly made it available to everyone. The homepage for the Berkeley AI Pacman projects is here:

`http://ai.berkeley.edu/`

Note that we will **not** be doing any of their projects directly. Note also that the code only supports Python 2.7, so that is what we will use[2].

You should:

1. Download:

   pacman.zip

   from KEATS.
2. Save that file to your account at KCL (or to your own computer).
3. Unzip the archive.

   This will create a folder `pacman`
4. From the command line (you will need to use the command line in order to use the various options), switch to the folder `pacman`.
5. Now type:

   `python pacman.py`

   This will open up a window that looks like that in Figure 1
6. The default mode is for keyboard control, so you should be able to play this game out using the arrow keys.

Playing Pacman is not the object here — don't worry if there is an issue with controlling Pacman using the keys — but you will need to run this code to do the coursework. So, if the code causes an error, get help.

When you are tired of running Pacman, move on to the next section.

4 Code to control Pacman

Now we work towards controlling Pacman by writing code. The file `sampleAgents.py` contains several simple pieces of code for controlling Pacman. You can see one of these run by executing:

```
python pacman.py --pacman RandomAgent
```

[2] If you insist on using Python 3, you are on your own, and you may find that the code you submit for the coursework does not run, and so you lose marks.

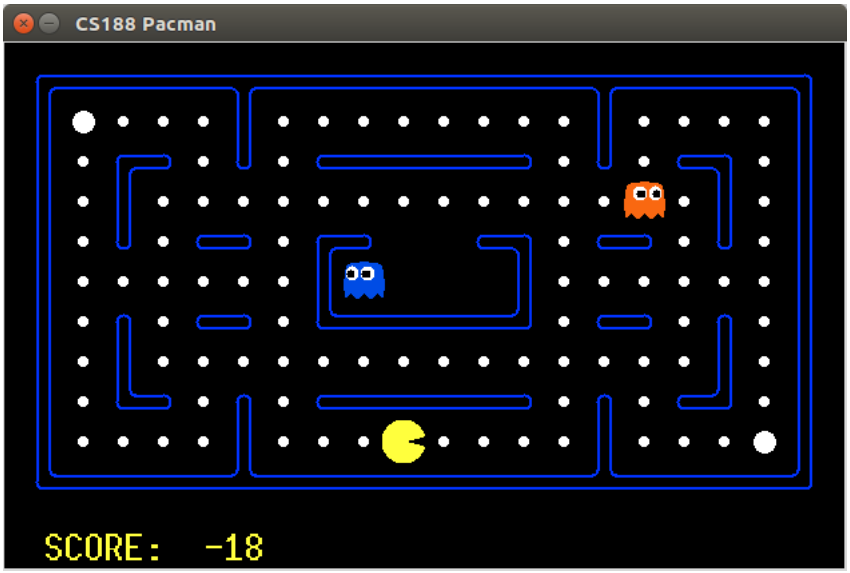


Figure 1: Pacman

This is not a good player (it is just picking from the available actions at random), but it shows you a couple of things.

First, you execute an agent that you write by using the `--pacman` option, followed by the name of a Python class. The Pacman code looks for this class in files called:

```
<something>Agents.py
```

and, when it finds the class, will compile the relevant class. If the class isn't in an appropriately named file, you will get the error:

```
Traceback (most recent call last):
  File "pacman.py", line 679, in <module>
    args = readCommand( sys.argv[1:] ) # Get game components based on input
  File "pacman.py", line 541, in readCommand
    pacmanType = loadAgent(options.pacman, noKeyboard)
  File "pacman.py", line 608, in loadAgent
    raise Exception('The agent ' + pacman + ' is not specified in any *Agents.py.')
```

Now open your favorite editor and look at `sampleAgents.py`. If you look at `RandomAgent` you will see that all it does is to define a function `getAction()` This function is the only thing that is required to control Pacman[3]. The function is called by the game every time that it needs to know what Pacman does — at every "tick" of the game clock — and what it needs to return is an action. That means returning expressions that the rest of the code can interpret to tell Pacman what to do.

In the basic Pacman game, `getAction()` is passed commands like:

```
Directions.STOP
```

[3]Of course, controlling Pacman to do something well may require a number of functions in addition to `getAction()`.

which tells Pacman to not move, or:

```
Directions.WEST
```

which tells Pacman to move towards the left side of its grid (North is up the grid).

However, for your coursework, you have to pass this direction to the function `api.makeMove()` first, just as the classes in `sampleAgents.py` do. For now (check out the file `api.py`) this function just passes the same value right back. However, in later weeks we will be using different versions of `api.py` which change the value that they are passed. This will, naturally, make controlling Pacman harder.

`sampleAgents.py` contains two more agents, `RandomishAgent` and `SensingAgent`. Try running them both.

`RandomsishAgent` picks a random action and then keeps doing that as long as it can.

`SensingAgent` doesn't move, but it does grab all the data that Pacman can access about its world and display it. Note that `SensingAgent` makes use of the functions in `api.py` to find out about the world. You will have to do that as well for these practicals and courseworks. As with movement, the `api` functions will change over the course of the semester to make Pacman's life harder.

5 Now you control Pacman

Now it is time for you to write code to control Pacman. You should write a `GoWestAgent`[4], which always tries to have Pacman go west on the grid when it is possible. What happens when `WEST` is not a legal move is up to you.

Getting the `GoWestAgent` to work is the minimum you should plan to get done this week. You will likely need more than the one hour of scheduled lab time to do this.

6 More

If you want to try other things (all of which will help you with preparation for the coursework):

1. Write code for `HungryAgent` which uses information about the location of the food to try to move towards the nearest food.
2. Write code for `SurvivalAgent` which uses the location of Pacman and the ghosts (and any other information that may be helpful) to stay alive as long as possible.

You may find that running Pacman in a smaller world helpful, in which case try:

```
python pacman.py --pacman RandomAgent --layout testMaze
```

or

```
python pacman.py --pacman RandomAgent --layout smallClassic
```

[4]This agent wants you to know that while it is a big fan of the fabulous 1970s Village People song "Go West" (`https://www.youtube.com/watch?v=1wc-AQJ2MYo`), it is no fan at all of the 1980s band Go West.

7 Python resources

1. The folk behind the Berkeley AI course have provided some Python Basics which will get you going in Python:

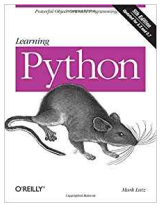
   `http://ai.berkeley.edu/tutorial.html#PythonBasics`

2. You can also access the Data Science Mini-Course on KEATS:

   `https://keats.kcl.ac.uk/course/view.php?id=50927`

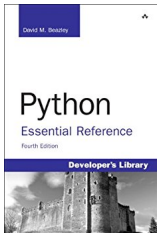
   which has a range of introductory material and exercises.

3. If you like to learn from a book, the O'Reilly volume *Learning Python*:

   is one I recommend.

4. If you like to have a book as a reference guide, then *Python: Essential Reference*:

   is one I find helpful.

There are also many, many Python resources online.

8 Version list

- Version 1.0, October 1st 2017.
- Version 1.1, October 7th 2017.
  - Fixed typos
  - Removed the section on how to install Python to try not to aggrevate the NFS problem in the labs.
- Version 1.2, October 8th 2017.
  - Fixed more typos