NUI Galway CT5132/CT5148
Programming and Tools for Artificial Intelligence
2020/2021 Syllabus & Course Outline

James McDermott

Module code and title

NUI Galway CT5132 Programming and Tools for Artificial Intelligence
NUI Galway CT5148 Programming and Tools for Artificial Intelligence - Online

Module co-ordinator

Dr. James McDermott, School of Computer Science

- `<james.mcdermott@nuigalway.ie>`, phone 091 49 3549.
- `http://jmmcd.net`.
- Office 441, IT Building.
- Office hours: 4.30-5.30pm Thursday (online office hours for CT5148). By appointment (for CT5132).

Module Overview

This module is about programming and computational tools required for artificial intelligence. It uses the Python language (in particular Python 3) as the main vehicle, but also focusses on conceptual material rather than just the language itself. It moves fast through introductory Python workings. It covers several important Python libraries in detail. It discusses approaches to building re-usable, high quality code but not software engineering per se. It also visits some extra topics such as version control, high-performance computing, and an introduction to the R language for statistics.

The module is core for the NUI Galway MSc in Artificial Intelligence (MScAI) Part-time (online) and Full-time (classroom). The syllabus and assessment will be the same for both.

This module will be divided into the following main topics:

1. Weeks 1-2: Introductory Python.
2. Weeks 3-4: Python data libraries: Numpy, Pandas, Matplotlib, Seaborn, and friends.
3. Week 5: High-performance computing.
4. Weeks 6-7: Scikit-Learn for machine learning.
5. Weeks 8-10: Further libraries and models of programming for AI.
6. Weeks 11-12: Introductory R: some side-by-side comparisons between R and Python; R for statistics and the Tidyverse.

Prerequisites

At least one semester of computer programming in any language at undergraduate level, or comparable industry experience, is a prerequisite.

Applicants to the MScAI usually have a 2.1 degree in computer science or computer engineering, including programming in multiple languages. Some will have studied Python or R or both. However it is expected that very few students on the MScAI will have studied all of the topics we will cover in this module. Thus this module will teach Python "from scratch", but in practice will move at a fast pace through introductory material to take account of students' background.

We will assume students are able to use the command line (e.g. Terminal on Mac, Powershell on Windows) and the filesystem, including: cd, mkdir/makedir, ls/dir.

Module learning outcomes

Students who complete this module will be able to:

- Read and write simple Python programs, e.g. for data munging, with a high degree of comfort.
- Use numerical Python libraries for manipulation, input/output, and visualisation of numerical data using Numpy array types and the Numpy ecosystem.
- Use R for simple statistics and data exploration.
- Use essential tools for AI, such as regular expressions, graphs, memoisation, machine learning, combinatorial programming;
- Plan/design a program using any of the above facilities; test it; document it; execute it locally or in the cloud as appropriate.

Assessment strategy

- Continuous assessment: 50%.
  - Assignment 1: Numerical integration: 10%.
  - Assignment 2: Solving problems from the Abstract Reasoning Corpus: 25%.
  - Assignment 3: Manipulating and Visualising Data in R: 15%.
- End-of-semester exam: 50%.

Link between assessment strategies and learning outcomes

The continuous assessments are take-home and will measure the student's ability to interact with coding environments, generate functional code, and create associated documentation. The final exam is closed-book, and will assess the student's code planning and code analysis abilities, as well as understanding of key concepts.

Textbook

Two books are recommended, both available for free:

- Jake Vanderplas, *A Whirlwind Tour of Python*, `https://jakevdp.github.io/WhirlwindTourOfPython/`, fast-paced, suitable for students who have some experience programming in another language. Free pdf (CC0 license) is linked from that site.
- Allen B. Downey, *Think Python 2nd edition*, `https://greenteapress.com/wp/think-python-2e/`. Very good for introductory and some deeper programming concepts. Also his *Think Complexity*. An artist of CS pedagogy.

The following books and resources are for background, for more detail, and for future reference:

- Wes McKinney, *Python for Data Analysis*. Focus on Numpy and Pandas libraries, by the author of Pandas.
- Al Sweigart, *Automate the Boring Stuff with Python*, `https://automatetheboringstuff.com/`. All about automating things that take hours when done by hand, such as renaming files, munging data, and gathering data.
- Toby Segaran, *Programming Collective Intelligence: Building Smart Web 2.0 Applications*, 2007. `https://www.amazon.co.uk/Programming-Collective-Intelligence-Building-Applications/dp/0596529325`. Ignore the subtitle: this book is about giving an intuitive understanding of several important machine learning algorithms based on "collective intelligence" in very accessible Python: recommender systems, genetic algorithms, clustering, etc. For our purposes, it is good programming practice.
- Hal Abelson, Jerry Sussman and Julie Sussman, *Structure and Interpretation of Computer Programs*, 1984. `https://mitpress.mit.edu/sites/default/files/sicp/index.html`. It gives a deeper understanding of what programming is, how program parts can be made to fit together, and how a program is evaluated. The original version uses Scheme as the teaching language, but the lessons apply to Python. Some of the code has been translated to Python on the web also and there is a pdf by "Wizardforcel" which we'll call SICPPy.
- Stuart Russel and Peter Norvig, *Artificial Intelligence: A Modern Approach* (AIMA) is very large and is the most-used textbook for "good old-fashioned AI", i.e. topics like planning, search, symbolic AI, but not machine learning/deep networks. It may be useful as background material. There is a Python version of the code: `https://github.com/aimacode/aima-python`.
- Peter Norvig, *Udacity CS212*, one of the earliest MOOCs covering data structures and algorithms in Python, and AI concepts such as constraint programming. Covers some of the AIMA material but focussing on the design of programs rather than deep AI.
  `https://eu.udacity.com/course/design-of-computer-programs--cs212`

Student communication and feedback

Communication to students will be via Blackboard (including Announcements and Discussion Board) and email.

Students are encouraged to raise questions at all times – especially before/during/after class (CT5132), or during the weekly lab/Q&A/office hours (CT5148).

Students should use email or the Blackboard Discussion Board (see Figure 1) for asynchronous questions. In particular, students should use the Discussion Board for questions about content, regulations, assessment or other items which may be of general interest. Email, phone or a meeting in-person should be used for queries which contain any personal information.

Email correspondence should comply with standard professional email etiquette. In particular, students should send email only from their NUI Galway email accounts, and should state their full name and ID in every email. Emails concerning group projects should cc all group members.

Plagiarism

Students are reminded of the University's policies on plagiarism. Plagiarism includes both deliberate and accidental re-use of material from other students in the programme, or other sources, without appropriate citation/referencing/acknowledgement. Plagiarism and any attempted subversion of the assessment process can lead to severe penalties up to and including expulsion from the University. Information and guidelines are available from the NUI Galway website:

- `http://www.nuigalway.ie/plagiarism/`

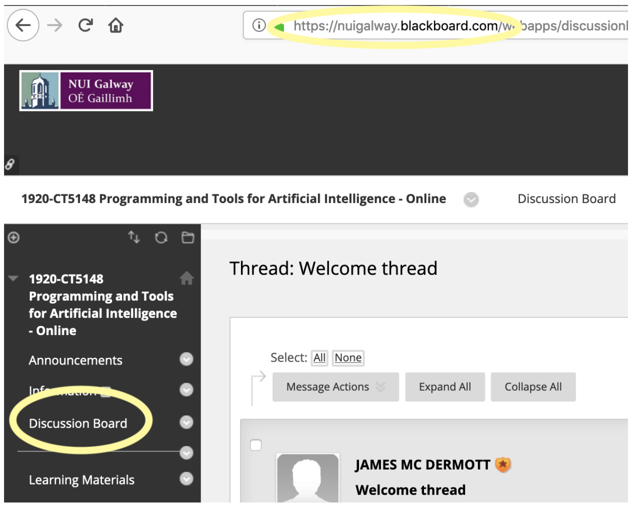

Figure 1: Accessing the Blackboard Discussion Board

Delivery and session times

The CT5132 (classroom) module is delivered as a flipped classroom. That means you consume the video materials and readings in advance of the weekly on-campus sessions. These sessions are called "labs", but they are for exercises, tutorial-style material, and discussion. There are two sessions: Thursday 2-3pm and Thursday 3-4pm. You can choose which session you prefer: sign up in Bb – Groups. The location for both is Arts and Science Building Room 105 (downstairs in Chemistry). Please bring your laptop, and install Anaconda in advance.

The CT5148 (online) module is delivered asynchronously. You consume the materials in your own time. We will hold an optional weekly live lab/Q&A/office hours, and for this I propose Thursdays, 4.30-5.30pm. (We can change later on if needed.) This will be in Bb Collaborate (Bb Virtual Classroom).

Week by week

This is a provisional week-by-week outline of the course, subject to change during the semester.

Week 0: Preparation
- Homework: read this (Syllabus) document.
- Homework: install Anaconda: `https://www.anaconda.com/products/individual` contains downloads for Windows, Mac, Linux. Choose Python 3.8 and 64-bit.

Week 1: Introduction to Python (for experienced programmers)
- Introduction: running a first Python program in a terminal, Jupyter Notebook, Spyder, and ipython; basics.
- import and writing our own modules
- The substitution model and the environment model of evaluation;
- Compound data structures;
- Example: driving in a grid world (also doctests);
- Python functions, including pass-by-value and copying a list;
- Homework: read Vanderplas up to page 44.

Week 2: More Python
- Example: histograms (also duck typing, defaultdict, file i/o);
- Unpacking;

- Comprehensions and generators;
- Functional programming;
- Example: itertools and factorial design of experiments;
- Errors, Exceptions and tracebacks;
- Strings, formatting;
- Homework: read Vanderplas up to page 76.

Week 3: Floating-point and Numpy

- Floating-point: `repr`, machine epsilon, underflow, overflow, NaN, inf;
- Numpy;
- Example: heart rate;
- Example: numerical integration;
- Numpy 2D arrays and fancy indexing;
- Example: fractals;
- Numpy input/output.
- Homework: read Vanderplas up to page 89.

Week 4: Scipy, Matplotlib, Seaborn, Pandas

- Plotting with Matplotlib and Seaborn;
- Pandas;

Week 5: Efficiency and HPC

- Computational complexity;
- High-performance computing and ICHEC;

Week 6: Introduction to Scikit-Learn

- Introduction to Scikit-Learn;
- Supervised learning;
- Train-test splits;
- Analysing prediction errors;
- Clustering;
- Representation learning.

Week 7: Scikit-Learn: Model Selection

- Hyperparameter tuning and cross-validation;
- Feature selection;
- Pipelines;
- Object-oriented programming;
- Mimicking the Scikit-Learn API.

Week 8: Graphs and Applications

- Finite State Machines;.
- Graphs and graph theory;
- NetworkX;
- Version Control with Git and GitHub.

Week 9: Parsing and Generating Text

- Regular expressions;
- eval and exec;
- Grammars, code generation, and generative art.

Week 10: More Advanced Tools

- Memoization;
- n-grams and the Aaronson Oracle;
- FFT with Scipy;
- Introspection.
- Homework: install RStudio.

Week 11: Introduction to R

- Introduction to R;

- Dataframes (tibbles) and the tidyverse;
- Tidy data;
- dplyr.

Week 12: Data transformation in R

- dplyr joins;
- Plotting with ggplot;
- Statistics in R.