Welcome to COMP 302
Programming Languages and Paradigms

Prakash Panangaden
School of Computer Science, McGill University

Winter 2020

1 Introduction

Since the mid 1950s programming in a high-level language has been dominated by "imperative" languages, of which the most early example is Fortran. In such languages, the basic linguistic unit is an *instruction* for modifying the state of a machine, hence the name "imperative" language. Around the start of the present century object-oriented programming became popular as the basic teaching language (a big mistake, in my opinion) and as a result Python, Java or C++ are the common first languages that a student learns today. This builds on the imperative paradigm by offering higher-level abstractions. Unfortunately, these are complicated and many students learn the Object Oriented Paradigm in a half-baked way.

From the earliest days of high-level programming languages, however, the language Lisp has offered an alternative view of programming as a *prescription* for evaluating expressions. Still more recently, new programming languages have emerged that embody new "programming paradigms". A truly new language offers a new viewpoint on computing.

In this course we will study some new (and not so new) programming paradigms; functional programming, imperative programming and object-oriented programming. The main ideas I wish to expose you to are

1. Procedural abstraction, functions as "first-class" citizens.
2. Data abstraction.
3. The notion of state.
4. Object-oriented programming.
5. Type systems, polymorphism and genericity.
6. Streams and lazy evaluation.
7. Language processing techniques.

There is no suitable text-book that meets the needs of this class so you will have to rely on class notes and material that I will provide. In the old days I taught this course using Scheme and I used "Structure and Interpretation of Computer Programs" by Abelson and Sussman. This is a

fantastic book. It is no mere text churned out by hacks who have nothing better to do. However, Scheme is an untyped language and it has become clear that understanding a *typed* language is crucial.

The programming languages that we will use are:

1. O'Caml and
2. Java.

All the course assignments will be in O'Caml. We will be using the LearnOCaml environment. This allows you to use OCaml through a website that we maintain. You do not have to install OCaml on your machine though, of course, you are free to do so. However, we will not give you assistance with installing OCaml on your own machine. Our version of the LearnOCaml environment will have the exercises loaded on for you to do. You can use the environment to develop your solutions and you can submit the assignment through the system *which will automatically grade it.* All grading is done by this automated system so the TAs are there to support your learning not to mark your assignments.

Some of the ideas that we discuss are illustrated by other languages, for example, **Fsharp**, **Haskell** or **Sml**. You will be exposed to concepts embodied in these languages but we will not require you to program in them. The point of this class is not to master the syntax of several different languages but, rather, to *learn concepts of programming languages.* In the past, I used Sml which I liked but which many students hated because it was harder to get documentation (or copy solved assignments from the web). Accordingly, I then switched to F# which the students liked much better but it was a big problem to get every student to install F# and a suitable IDE on their machines. This term I am using OCaml for the second time; my colleague Brigitte Pientka has used it for a few years now. The OCaml language is very close to F# and to SML at least for the basic core. Conceptually the functional programming core is *exactly* the same as that of Sml.

It is important to emphasize what this course is not. **It is not a course on a dozen different languages**. There is a myth that knowing more languages is better. The fallacy in this argument is that most people who know several languages actually know only one *type of language* but in several different forms.

More than any other course I have taught, people come to this class with their own expectations about what they will learn. Let me say at the outset that the purpose is not to cram you with the syntax of the latest popular language. Many people in the past have made remarks like "none of my friends have ever heard of Sml, why are we learning it?" One answer to this question is that I am trying to teach you programming language *ideas.* I have been told that students do not find SML resources on the web. I think you will find enough OCaml resources on the web. But let me rant for a moment: I am *sick and tired* of hearing students *whine* about not finding things on the web. People have to learn to think for themselves. If you can't find the answer to the homework on the web I am happy. I want you to do it yourself. End of rant.

Underlying this subject is a very interesting mathematical theory, "the mathematical semantics of programming languages." I will not talk about that subject in this class.

2 Grading

There will be 8 assignments, all of which are OCaml programming exercises. The homework accounts for 24% of the total course marks; 3% for each assignment. We will not give extensions unless you have a valid medical reason which includes mental health issues. Being busy with other classes is not a valid reason nor is having to go to another city for a hackathon or a rock concert. There will be 3 quizzes, each one worth 2% for a total of 6%. These quizzes are to be done online using myCourses. You will have 60 minutes to complete the quiz once you start, but you can start and finish the quiz any time on the day that it is due.

The midterm examination accounts for 10% and the final examination will be 60% of the total. The midterm date will be announced later. The final examination will be in the final examination period. We will permit a one-page cheat sheet for the midterm and 3 pages for the final.

3 Staying in Touch

There will be three class web pages. Please ensure that you have access to them. One of them is an open web page that we maintain. The URL is:

`http://www.cs.mcgill.ca/~prakash/Courses/302/comp302.html`

Homeworks, announcements, notes and other information will be posted here. The second web page will be set up through the course management system (myCourses) at McGill. This will be used for posting your grades and for the quizzes. The third is a Piazza web site which you can use to communicate with each other, with me and with the TAs. This term we will be using Piazza for class discussion. The system is highly catered to getting you help fast and efficiently from classmates, the TA, and myself. Rather than emailing questions to the teaching staff, I encourage you to post your questions on Piazza. If you have any problems or feedback for the developers, email team@piazza.com.

Find our class Piazza page at: `https://piazza.com/mcgill.ca/winter2020/comp302/home` but note that I will be using the web page first mentioned for announcements.

There a closed Facebook group which will be looked at occasionally by me but I will not answer questions there. I have seen students confidently answer questions incorrectly on other groups which is why I keep an eye on it. By all means set up another closed group to criticize the course anonymously if you like. Please note that statements made on Facebook have no official seal of approval. Please do not come to me and say that someone on the Facebook group told you that something was the right way to solve a question.

We will hold office hours. My office hours are Tuesday and Thursday from 11:30 to 1:00, right after class in Room 105 North Wing McConnell. That is just after class so the first few minutes will be after class in the classroom and then I will walk over to my office for longer questions.

Please **do** come and see me during office hours. Do not feel that you are infringing on their time – office hours represent time has been reserved specifically to see **you**. If you would like to discuss a program, please bring a copy of the code or a laptop with the code on it. We cannot figure out what

is wrong with a program just from a description of its behaviour. You should walk into my office during office hours even if there is someone there talking to me. There are times when students come and "hang around" in my office throughout the period (which I welcome) so if you wait for them to leave you will never get a chance to talk to me.

There are some TAs for the class. The teaching assistants will hold office hours to be announced later on the web site.

4 Academic Integrity

McGill University values academic integrity and treats it seriously. Please see the website

`http://www.mcgill.ca/integrity`

for more details. In the present class we expect that the work that you turn in is your own. You can discuss ideas and approaches with your friends but you cannot blindly copy their solutions or solutions that you found on the web. This includes solutions that you might have found on CourseHero or Docuume. If we find that someone has handed in a correct solution that they cannot satisfactorily explain to me, we will treat it as a case of cheating and prosecute accordingly.