感想：

一学期間、おつかれさまでした。プログラミングという活動に興味がでたら何よりです。ではまたどこかで！ 瀧川

最終回および課題提出時の質問の Q＆A：

Q.もし今後プログラミングに関する不明点があった場合にメール等で質問しても宜しいでしょうか。

A.どうしても解決できなかったらぜひ聞いてください！(別の授業の課題とかでなければ)

Q. 私はプログラマーになりたくてこの授業をとりました。よく youtube やテレビで年収 1000 万を超えるプログラマーが出てきます。この人たちは何が優れているからこんなに評価されている人ですか？

A. 授業を通して、少しは分かったと思うのですが、プログラミングというのは結局「ソフトウェアというモノを設計して実際に作る作業」であって、プログラミング言語が使えることがすごいのではなく、プログラミング言語で「何を作ったか」「何を作れるか」が決定的に大事です。(プログラミング言語が使える人は世の中にいっぱいいるので) 英語が読み書きできることがすごいのではなく、英語で読み書きできる「内容」がすごい人が、小説家・随筆家・コピーライター・官僚・経営者などとして活躍するのと同じですね。英語ができる自体がすごいのではなく(それはある種最低条件？)、英語で何を語るかが価値を決めるのです。最近は github などで自分の作ったものを自由に公開できるので、おっと思うもの(新しいゲーム、プログラミング言語、サービス、など)を作ってそれが実際に世の中のニーズにあうものなら自ずと評価は高いです。もう一つ、プログラミングで一番難しいのは上流の工程、つまり「何を作るか」「どうやって作るか」を決める作業です。ここに携わる人が高給で、ただ決まった仕様のプログラムを実際にコーディングする人が高い給料をもらっているわけではありません。ビルを建てるとすると、設計した建築家の仕事のほうが、実際にコンクリートを打設して壁や床を作る人より高度な能力が求められるので高いお金が発生するのと同じですね。「設計」ができる人は、ふつう「プログラミング言語が自由自在に扱える」だけではなく、それ以外の知識(数学や物理学・生物学、工学、ビジネスや産業、インターネットや通信)が重視されるとも言えます。後者の知識とセンスが「何を作るか」「どう作るか」の良さを決めるので。

Google の技術者のほうが、ホームページ制作会社の技術者よりも、ふつう給料が高い理由は、それだけ Google が作っているソフトウェアのほうがお金をうむ(従って責任と高い知識・スキルを伴う)ということを意味しています。また給料や環境が良い会社には世界中から優れた人たちが応募するのでその意味でも選別されるということもあります。

瀧川は 2015 年まで「ソフトウェア工学」という授業を担当していました。ソフトウェア製造のプロセスについては参考になるかもしれないので、リンク貼っておきます。資料アクセス: ID「se2015」パスワード「goroumaru」
https://art.ist.hokudai.ac.jp/~takigawa/course/se2015/

Q. 先生くらいプログラミングができるようになるとオープンソースコードでなくても、ソフトを使えばこんな感じのコードだろうなと予想がつくものですか？

A. 私だけではなく世の中でソフトウェア作成に携わっているそれなりの経験を積んだ技術者はだいたいそうだと思います。まあ時間さえあれば基本的には作れるものなので。ただ大枠がわかるのと実際にそれを作るのにはだいたいとてもギャップがあるのもソフトウェア業界のあるあるです。あと、最近の人工知能系のアプリケーションは普通の意味での知識ではちょっと仕組みが分からないと思います。(私はたまたま専門なので分かりますけど)

Q. エンジニアとプログラマーの違いって何ですか？

A. 通常ソフトウェアエンジニアは直接プログラムを作る(コーディングする)だけが仕事ではありません。何を作ってほしいのか客先に出向いて相談したり、ニーズを発掘してこういうものを作れば良いのだと考えたり、ハードウェア(サーバーやネットワーク機器やストレージ機器の調達・納品・設定・運用・管理など)や関連するサービス(ネットショップ/e-コマ

ースの場合、運送や決済など)との連携をとったり、ソフトウェアが絡むプロジェクト全体について仕事をします。プログラマーは狭義には「プログラムする人」なので、レベルは色々ありますが、とにかくプログラミングする人すべてです。ただ確定した仕様書(設計書)を元に、それを実際にプログラミング言語でコードにする作業は、最近は人件費の低い東南アジアやインドあたりにアウトソースすることも多いので、実際のところ、代替が聞く人員であり、人件費の高い日本では与えられた挙動をするプログラムのコーディングだけで生計を立ててるひとはそんなに多くはないのかも。なお、一部のテック系の会社を覗き、開発会社でもコンサルティング会社(業務を改善するためのシステムの導入・運用・管理も行う)でも、給料が上がって出世するにしたがって「設計」「企画」など上流の仕事で責任を持つようになるので、現場でコーディングする機会はへっていきます(多くの場合、コードすら書かない)。

Q. 先日、人型のアンドロイドが感情を持ち、自立して行動するようになるというストーリーのゲームをやったのですが、将来的に擬似的な感情を作り出すことはできるのでしょうか？またそういった研究はなされているのでしょうか？

A. 研究はもちろんたくさんされています。工学的なものと認知科学的なものとがあります。前者でいえば、Pepper くんや Siri や Alexa などの対話を伴うシステムではどう会話すれば人間は自然に感じるかという意味で色々な試行錯誤を今も続けています。後者の問題も、紀元前の哲学からある由緒正しい問題ですが、「ロボットが感情を持っているか」の前に、「自分以外の人間が感情を持っているか」すら、実は確認が難しいことです。どうすればそれが判定できるでしょうか？前者、後者といいましたがこの２つの区別さえ実はそれほど簡単ではありません。ただ、おそらくデータさえ取れるのであれば、人間が見て「感情があるように」見えるシステムを作ることは十分可能だと思います。ただこれは感情を作り出したことにはならない気もしますよね(「哲学的ゾンビ」問題とか呼ばれています。「中国語の部屋」問題という人工知能の有名な古典的思考問題も同じようなものです)。人間は実際騙されやすいので。人工知能が「思考するか」「心をもつか」「感情が芽生えるか」などはすべて同じ根深い難しい問題に起因していて、これについて最近何かが急速に分かったわけではありません。逆に言えば人類の永遠のテーマですね。下記の２つのスライドを参考にしてみてください。

何から始めればいいかわからない人の 人工知能(AI)と機械学習の入門
https://www.slideshare.net/itakigawa/ai-107946616

データ社会を生きる技術 ～人工知能の Hope と Hype～
https://www.slideshare.net/itakigawa/hopehype

Q. 先生が好きなプログラミング言語はなんですか。

A. lisp(および scheme や emacs lisp などその方言)という言語がとても好きでした。Fortran について２番目に古いプログラミング言語ですが、数学的で AI 研究の当時の主流言語になり、80-90 年代は common lisp として一般に使われました。情報理工の人は Emacs というエディタをそのうち実験とかで使うことになると思いますが、設定から拡張まで lisp でプログラムすることで実現できる非常に優れたエディタです。瀧川は 20 年くらい emacs を使っていました。訳あって数年前から vim というエディタを使うほうが多くなりましたけど。Emacs か vim かはプログラマが愛してやまないエディタの二大巨頭だと思います。一般的には vim はおすすめではありませんが、私は次の本を読んで vim が好きになりました。
「実践 Vim 思考のスピードで編集しよう!」https://www.amazon.co.jp/dp/4048916599
一般に、lisp だけではなく(mathematica や R などでも実は重要な)「関数型言語」が数学的でとても好きです。最近の言語では、Haskell や Scala や F#や Clojure や Clean かな？(lisp 自体はいまはまったく主流とは言えないと思います)

Q. 実際に Google 社で採用される人はどのような能力、学歴をもった人なのでしょうか？

A. 本社なのか日本の Google でも良いのかにもよりますけど、Google は非常に大きい会社です。開発や企画あるいは研究に携わっている人はごく一部で大勢は、事務職・営業職などから機器製造や運用保守など様々でなんとも言えません。学歴はあまり重要ではない気がします。最近は youtuber じゃないですが、自分のプログラムを github などで自由に公開で

きるし、有用なプログラムは広く世界中で使われるようになります(python が良い例ですね)。良いものつくってんなーとかなったら(学歴問わず)就職できるかもしれないと思います。
たとえば、私の研究室でも最近でも Google に就職した学生はいます。彼の例でいくと、まず博士課程までいっていて、その分野で一定の専門知識を持っていました。またプロコン(プログラミング・コンテスト)が好きで、同好会みたいなもので活動したり、各種の大会に参加したあと、参加資格が切れてもお世話や裏方や解説などをしていました。また、各種会社のインターンなどにも言っていました。Indeed は選抜がありますが費用を全持ちでアメリカでのインターンをやっていてそれにも言っていましたね。こういう、博士号が取れるほどの専門知識やプログラミングでのスキルや活動はおそらく有効に働いたように思います。ただ Google も普通の会社ですので、良いところ悪いところあります。研究室の先輩には Google に就職して活動して辞めて他の会社へうつった人もいますし、まあそれ以外にも色々会社はありますね。なお、プロコンサークルはうちの研究室でいつも勉強会しています。学部も色々なので興味あればぜひ。

Q. 瀧川先生は Python などの言語をはじめて学んだとき、ネット上で無料で学べるサイトなど利用しましたか？
A. 当時はそもそもそういう教材はなく(インターネットが使えるようになりはじめたくらいなので！)、本と公式ドキュメントしかなかったので利用してません。最近、なにか別の言語を読む必要が出た時は結構使います。公式のチュートリアル(Tutorials)とかが良いんですよね。私は情報理工出身なので授業でもオーソドックスに習いましたし(テキストの文法を一個づつやるような)、当時は選択肢がなかったので本を買って勉強していましたね。

Q. 最近、情報の授業で、html を用いてページを作りました。１枚のページを作るのに多くの文字列を入力しましたが、なぜ、表示される画面を word などで書いて表示されるという方法がとられないのかわかりません。
A. これは最初歩の授業のレベルでは単にまわりくどいだけにしか感じないかもねえ。なお、word は html で内容を保存できるので、word で書いて html で保存しても全く OK です。なぜ HTML という書き方があるかといえば、あるやり方で文字列を書くとレンダリング(見栄えのレイアウトや文字の大きさ、画像の表示など)はその規則にそってやってくれるというのは実は便利で、端的に言えばプログラム自体が「HTML」で出力すれば、色々なことをブラウザに表示できるからです。この静的なフォーマットを HTML(もうすこし細かい見栄えの調節を CSS)でやるのは、Google や Facebook や twitter や Scratch など Web ブラウザ上で動くすべてのサービスにとって必要な基盤なのです。なお、動的なことをやるのが通常 JavaScript と呼ばれるプログラミング言語で、HTML、CSS、JavaScript はブラウザさえあれば練習できます。

Q. Scratch で作成したものを Python で読み込むことはできますか。
A. できるんじゃないかな？Scratch のプログラムは結局ただの文字列(テキスト)なので。Scratch の挙動を再現できるか？という質問だとしてもがんばれば勿論できると思うけど、それには Scratch のエンジンを python で書き直さなければいけません。Scratch のソースコードは公開されている(過去の Q & A を参照のこと)し、まあ何でも可能です。