第二回 研究プログラミング WINGS-ABC チュートリアル

登壇者：片岡麻輝、関森祐樹、堀口修平

2021年10月23日（土）17：00-19：00

登壇者紹介

片岡麻輝　総合文化研究科広域科学専攻、修士一年。WINGS-ABC三期生。大泉研究室にて、計算神経科学および理論神経科学の研究を行っている。現在はシミュレーションベースでの研究テーマに取り組んでいる。株式会社ACESで機械学習アルゴリズムの開発にも従事。

関森祐樹　海洋技術環境学専攻、修士一年。WINGS-ABC二期生。生産技術研究所2部、巻研究室で、複数の海中ロボットシステムの研究をしている。複数の海中ロボットシステムを可能とする、基礎的なシステム設計や情報処理アルゴリズムを検証するために、ソフトウエアを作成している。株式会社FullDepthで水中ロボットの制御研究開発にも従事。

堀口修平　情報理工学系研究科数理情報学専攻、博士一年。WINGS-ABC一期生。生産技術研究所の小林徹也研究室で、免疫系の理論研究をしている。数理モデルのシミュレーションのほか、Webアプリ・ロボット・生物画像処理や機械学習アルゴリズムの開発でプログラミングを行う。

研究プログラミングチュートリアル　概要

研究現場のプログラミングで起こりうる問題をスキット（寸劇）形式で紹介し、それぞれについて選択問題と解説を行う形式で進行します。扱うテーマは以下の通りです。最後に、皆さんと研究プログラミングについて一緒に考える、ディスカッションの時間を設けてます。

第一回（情報収集と共有）

- コードの共有
- 環境構築
- パラメータ管理
- バージョン管理

第二回（信頼性と再現性の担保）

- コードの読みやすさ
- Linterと型チェック
- テスト
- 再現性

用語集

コード:コンピュータに解釈や実行させること目的として命令やデータ

プログラム:コンピュータへの命令や処理が記載されたもの

スクリプト:ソースコードで即座に実行できるもの

ソフトウエア:コンピュータの物理的な部分(ハード)の総称

アプリケーションソフトウエア(アプリケーション):オペレーティングシステム(OS)上で動作するソフトウエア

リポジトリ：アプリケーション開発で、システムを構成するデータやプログラムの情報を収納したデータベース

ライブラリ：ある特定の昨日を持ったプログラムを他のプログラムから引用しできるように部品化し、それらを集めてファイルに収納したもの

モジュール：ある機能を実現する、ひとまとまりのプログラム機能や要素

クラス：オブジェクトを生成するための設計図や雛形

シンタックス：プログラミング言語の構文や文法

フレームワーク：アプリケーションなどの実装に必要な機能や定型コードをライブラリとして事前に用意したもの

プラットフォーム：アプリケーションでは、ミドルウエア、ライブラリ、言語処理系（ランタイム）等のこと

バージョン：同名プログラムの新旧を区別する、番号や符号

参考文献

Boswell, D., Foucher, T., & Kado, M. (2012). リーダブルコード より良いコードを書くためのシンプルで実践的なテクニック. (角 征典, Trans.). オライリージャパン.

Cormen, T. H., & Leiserson, C. E. (2009). Introduction to algorithms (3rd ed.). The MIT Press.

Lubanovic, B., Suzuki, H., & Nagao, T. (2015). 入門 Python 3. オライリージャパン.

Microsoft. (2016, April 14). Developing inside a container using Visual studio code remote development. RSS. Retrieved October 19, 2021, from https://code.visualstudio.com/docs/remote/containers.

Pressman, R. S. (2010). Software engineering: A practitioner's approach (7th ed.). McGraw-Hill Higher Education.

Software Freedom Conservancy. (n.d.). git. Git. Retrieved October 19, 2021, from https://git-scm.com/.

呉 珍詰. (2020). 初心者のためのコンテナ入門教室. Think IT（シンクイット）. Retrieved October 19, 2021, from https://thinkit.co.jp/series/9490.

湊川 あい, & DQNEO. (2017). わかばちゃんと学ぶ Git使い方入門. シーアンドアール研究所.

van Rossum, G., Warsaw, B., & Coghlan, N. (2014). Python コードのスタイルガイド. pep8-ja. Retrieved October 19, 2021, from https://pep8-ja.readthedocs.io/ja/latest/.

SHIFT. コンポーネント（単体、ユニット）テストとは？手法などを例で紹介 . *ソフトウェアテストの SHIFT*. Retrieved October 20, 2021, from https://service.shiftinc.jp/column/3636/.

Dutta, S., Legunsen, O., Huang, Z., & Misailovic, S. (2018). Testing probabilistic programming systems. Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. https://doi.org/10.1145/3236024.3236057

河野 晋策 . あなたの生産性を向上させるJupyter Notebook Tips. リクルート メンバーズブログ. Retrieved October 20, 2021, from https://blog.recruit.co.jp/rtc/2018/10/16/jupyter_notebook_tips/.

第二回
研究用ソフトの信頼性と再現性の担保

登場人物

片岡：主人公のM1大学院生

堀口：先輩

関森：教授

（この物語はフィクションです。実際の人物・団体とは一切関係ありません。）

第一回のあらすじ

神経科学分野の研究室に入室したM1の片岡は、神経回路モデルにダイナミクスと情報処理の関係を研究するために、シミュレーションプログラムを作成している。先輩の堀口にプログラムを共有してもらい、シミュレーションを使うための環境構築をした。研究を進める準備が整ったところで、パラメータを管理しながら、モデルをシミュレーションした。片岡は、シミュレーションプログラムを、他のメンバーと共有できるよう、バージョン管理を始めた。

スキット5:「コードの読みやすさ」

あらすじ

M1片岡が入学からしばらく経ち、堀口先輩の神経モデルもわかるようになってきた。片岡はこれまでの知見に基づいて新しいモデルを考案した。ここからは自分で考案したモデルをコードに落とし込んでいく作業が必要になるが、できるだけトラブル等の少ないコーディングを行っていきたい、と片岡は思っている。

問5:「コードの読みやすさ」

以下のうちプログラムを書く上でどれが一番大事でしょうか

1. 高速に動作するプログラムを早くつくること
2. コードを短くすること
3. 誰が見たとしても最短時間で理解できるように書くこと

スキット6:「Linterと型チェック」

あらすじ

コードを書いていく中、片岡は、プログラム実装を補助する、様々なツールが存在することを知った。どうやらプログラミングの研究の多い先輩の中にはVS Codeというアプリを使う人が多いみたいなので、自分も使ってみることにする。

問6:「Linterと型チェック」

Linter 使ってシンタックスエラー(文法的な誤り)を解消したから、プログラムは問題なく実行するだろう。何が問題でしょう？

1. インデントや改行のずれは解消されない
2. データ型(float, int, str, ndarrayなど)が間違っていても見逃される
3. 変数が定義されてなくても見逃される

スキット7:「テスト」

あらすじ

片岡は、Linterではシンタックスなどしかデバッグできないことや、データ型が正しく実装されている必要があることを学んだ。加えて、これらをデバッグしても、意図した通りのアルゴリズムでプログラムが動くことは保証できないことにも気づく。

問7:「テスト」

テストを用いて可能な限り想定通りの動作をするプログラムを短い時間で作ることを目的にした場合、クラス、関数などへの分割は次のどの方針を意識するのが適切でしょうか？（どれが一番理想に近いでしょうか？）

1. 一部の繰り返し現れる処理だけを切り出してモジュールにする
2. 意味的な区切りのある部分はすべて異なるモジュールとして切り出す
3. 処理が煩雑になってきた場合に限り、見やすいようにモジュール分割する

スキット8:「再現性」

あらすじ

片岡が研究室に配属されてから9ヶ月が経った。かなり研究も進み、来年度の学会投稿に向けて、研究室内で進捗を共有することになった。
他の研究者がシミュレーションや計算の結果をハンズオン的に触れられる状態を作ると良い、と片岡は教授にアドバイスを貰った。将来的には、インターネット上で公開し、論文や学会の発表でも広く検証してもらえるようにしたい、と教授は考えている。片岡は、Jupyter Notebookを介して計算方法などを共有したいが、どのように書き進めていくのが良いかわからない。

問8:「再現性」

実験やデータ解析を多くの人に再現してもらえるように、Jupyter notebookで処理の結果を公開することにしました。より再現性の高い共有のためには、次のどの方針でnotebookを書くのが適切でしょうか？

1. 処理をすべてひとつのnotebookファイルにまとめて書く
2. 計算自体は他のスクリプト等で完結させておいて、notebook内では出力データの描画や表示のみを行う
3. 複雑で時間のかかる計算は別のスクリプトに分けて、比較的時間のかからない計算処理と結果の表示だけをnotebookに残す

ディスカッション（10分）

今まで作成してきたプログラムを振り返り、チュートリアルで紹介された知識/テクニックと照らし合わせて、改善できそうなポイントを考えてみましょう。

第二回のまとめ

1. コードの読みやすさ
   - 誰が見ても理解できるコードを心がける
   - 変数名の付け方、コメントの書き方、コーディング規約などは、一貫性を持たせる
2. Linterと型
   - Linterを活用してシンタックスエラーを減らす
   - データ型が正しく設定されているか確認する
3. テスト
   - 単体テストをつくるだけでも、デバッグの労力を大幅に減らすことができる
   - プログラムは適度に分解して、テストがしやすいように作成することを心がける
4. 再現性
   - 計算処理のすべてをnotebookにまとめてはならない
   - 時間のかかる処理はスクリプト、単純な処理は notebookを使う