【一般教育演習】プログラミングで問題を解く：集計から人工知能まで・第十回 2018 年 6 月 21 日

メッセージ：自分で新規の Jupyter Notebook を作成し、Python コードを一緒にかけるノートだと思って、自分なりの「まとめ」を作ってみて、最初はそれを見ながら作業すると定着がいいかなと思います。あとでその Jupyter Notebook ファイルをレポートとしてメール提出してもらおうと思いますので、学びの記録と思って各自作成してみてください。！

1. Web サイト
   スライド、資料、事務連絡などはすべてココでシェアする予定↓
   http://art.ist.hokudai.ac.jp/~takigawa/prog/
   「毎回の作業」「Jupyter 使い方」を忘れた人はここで確認

2. 瀧川のメールアドレス 提出や質問
   takigawa@ist.hokudai.ac.jp

3. 教室
   5/10（木）～8/2（木）E209 教室

4. 今日の内容
   - フォローアップ Q & A
   - 簡単な Jupyter notebook を一から自作して瀧川(上記 2)にメール送付
   - 今日の演習で使うデータのダウンロード
   - 外部ライブラリ pandas および matplotlib を使ったデータ分析と可視化の体験

5. 参考情報
   - Python 公式チュートリアル
     https://docs.python.jp/3/tutorial/
   - Markdown
     https://guides.github.com/features/mastering-markdown/

先週の質問への回答：

Q. 新規の Jupyter Notebook の作成の方法がわかりません。
A. もっとも多い質問だったので、今回、補足資料で「やってみる」課題を作ってみました。Notebook を新規作成して、そこに何を書くかは他の授業を受ける時のノートと同じで自由です。受験をくぐり抜けた人たちなので、そもそもノートの取り方のコツが分からないってこともない気がしますが、まあ端的に行って「あとで自分が見た時、思い出しやすいように書く」のが大切です。例を入れたり、自分なりに別の例を作ってみたり、Web で調べたことを書いておいたり、という感じかな？

なお、「Jupyter notebook tutorial」とかで調べると、そもそも色々な notebook を見つけることができると思いますし、そういうのをダウンロードして試してみるのも役に立つと思います！

Q. 新規に Notebook を作る点はわかりましたが、見出しや code 文、今日の復習問題のような実行されない文で分けたり、文字の大きさを変えたりする方法がわかりません。
A. これは「jupyter の使い方」資料にもありますが、practice05 に解説を入れています。セルのタイプの種類と Markdown 記法の問題ですね。例えば見出しは先頭を「#」「##」で始めるだけです。

Q. print の’ ‘はスペースで変換すると似たようなものが出てきますが、使えるのは一つだけですか？
A. 半角英数(ascii 文字)の「’」だけです。全角のものは、日本語なので文字コードが違います。よく分からなければ「文字化け」についての回の資料を再チェックしてみてください。(なお、フォント(文字のみばえ)はまた別の問題です)

Q. ファイル出力について、なぜ、f.readline()を続けて実行していくと 1 行づつの読み出しになるのか(中身のないカッコの意味も不明)
ファイルというのは、実体としてはなんらかの 0 と 1 の並びとして、ハードディスクなりメモリなりの一部分を占める情報です。例えば、「こんにちは<改行>こんばんは」という中身のテキストファイルだとすると、「こ」「ん」「に」「ち」「は」「<改行>」「こ」「ん」「ば」「ん」「は」という文字コードを 2 進数で表記したものが並んでいます。文字コードが何番かは、文字化けのところで紹介したように encoding によりますがとにかく体系ごとに何らかの番号づけがされているのでそれを二進数で表して 0 と 1 にしたものとして表現されています。
ファイルの読み出しというのは、それを先頭から読んでいく操作になります。で、readline()というのは「<改行>」文字が出てくるまで文字を読んでいくという関数なんですね。なので、f でファイルオープンして、f.readline()したら、まず最初から「こ」「ん」「に」「ち」「は」「<改行>」まで読んで、その結果を文字列「こんにちは<改行>」として返します。この時点で「この改行まで読んだよ」という情報は f の情報の一つとして内部的に記録しておきます。なので、再度、f.readline()が呼び出された場合、その保存した今まで読んだ場所から、また「改行」(もしくはファイルの終端)が出るまで文字を読み出します。操作としては readline()がやっていることはこれだけです。
実際は、改行までではなくて一文字づつ読み込みたかったり、あるいはそもそも文字コードではないもの(音楽ファイルとか画像ファイルとか)を読み込むとかの場合は、また違う単位での読み出しが必要になり、演習では扱いませんが、python のファイル入出力ではいずれも扱うことができます。

Q. open(‘zen.txt’, ‘r’)の r や w について、r→読み出し、w→ファイルとして保存、という認識で正しいか
A. r は’read’、w は’write’です。ファイルというのは、ハードディスクになんらかの 0 と 1 の列で保存されたデータ列です。それを読みだすときは、read で「r」、書き出すときは「w」ですね。ファイルの保存というのは基本的に(読み出した内容を)書き出すことです。

Q. practice04 の小クイズで、なぜ print(a,b)が(20,10)ではなく、(a,b)になるのかわかりません。
a=20、b=10 は定義してるんじゃないのですか。
Practice04 のこの例を実際に jupyter notebook に書いて、実行すると下記のようになると思います。

```
In [1]: a = 20
        b = 10
        b = a
        print(a, b)
```

```
20 20
```

FizzBuzz ができないという人にも共通する最初によくあるハードルだと思うのですが、二つのことがポイントです。

- **「=」は「イコールではなく代入」**
- **プログラムは順番に実行される**

と考えると、理解できるかなと思います。これは日本語で書くと以下のような操作になります。

1) 変数 a に 20 を代入しておく
2) 変数 b に 10 を代入しておく
3) 変数 b に変数 a に代入してある値を代入しておく
4) 変数 a と変数 b に今代入してある値を出力(print)

これが「順番に」実行されます。とすると、a, b は 20, 20 になるのが分かるでしょうか？？ 3)の時点で b の内容は 1)で入れておいた 20 で上書きされているわけですね。以下の例で再度考えてみましょう！
if 文も上から順番に実行されるため、まず a % 2==0 が満たされてしまったら、そのあとの a == 78 も満たすとしてもそれは実行されません。「条件分岐」というように、この実行の結果は、here か、there か、other か、どれか 1 つで排他的な処理になります。

```
In [2]: a = 20
        b = 10
        b = a
        b = a + b
        a = 2 * (b - 1)
        print(a, b)
```

```
78 40
```

```
In [3]: if a % 2 == 0:
            print('here')
        elif a == 78:
            print('there')
        else:
            print('other')
```

```
here
```

Q. ファイルの読み込みは URL などがわかって入れば、どのサイトからも文が読み取れるのでしょうか。
A. 読み取れます。ただし、そのファイルがどのような形式なのか(文字が並んだものなのか、画像などを含む複合文書なのか、音楽ファイルや写真ファイルなのか)によって読み込み方が変わります。この演習では文字データの場合のみしか扱いませんが、より一般には 0 と 1 のどんな並びでも読み書きします。例えばインターネットのデジタル通信もネットワークを介したファイル読み書きみたいなものです。

Q. github にビジネスという項目があったのですが、お金がもらえるんですか？
A. お金をもらえるのではなく取られます(笑)。Github(や bitbucket)はソースが全世界に公開されてもよければ、無料で誰でも使える「git」と呼ばれるもっともよく使われるソースコード管理ソフトウェア(バージョン管理)のネット上の保存先です(リポジトリと呼ばれます)。それに SNS 機能がついたものですね。ただし、会社のソフトウェアとかで全世界にソースコードが見えたりしては困る場合もあります。そういうところ場合は有料プランにすると全公開されない設定が使えるようになる感じですね。Github は長らくオープンソース業界の代名詞で、就活するときも、ここに自作のプログラムをいろいろ載せておくことが履歴書がわりもなってきましたが、つい最近、Microsoft 社によって買収されました。ニュースでも流れてので知ってる人もいるかもですね。これからどういう運営になるかは MS 社次第ですね。

Q. プログラミング言語は基本的に英語に近いように思えますが、ũ や ü などアルファベット表記以外の文字が扱われるようなものも存在するのでしょうか。
A. プログラムの表現にある文字は原理上は何でも良いので利便性の問題のみです。「文字化けについて」の回を見てもらえると思い出せると思いますが、要は文字というのは機械の中では単に「数字(ある体系で何番の文字か)」です。その意味では、何でもいいのですが、まあ特殊文字というのは入力しづらいし(欧では専用キーボードがあるが日本語キーボードでは例えばウムラウト ä は入力が面倒ですよね)、他の国の人が使うと文字化けを起こしうるし、フォントが対応してなければそもそも見えないしで、不便なので、実用言語ではあまり使いません。が、変数名に使うのは OK な場合もあります。例えば Python では下記は問題ありません(入力が面倒臭いのでまあ使う人はいないと思いますけど)。なお、日本語で書くプログラミング言語も存在はします。
https://ja.wikipedia.org/wiki/日本語プログラミング言語

```
aü = 34
θ = 3
φ = 34
Å = 3
print(aü, θ, φ, Å)
```

```
34 3 34 3
```

```
年齢 = 23
性別 = '男'
if 年齢 > 20 and 性別 == '男':
    print('入場許可')
else:
    print('入場不可')
```

```
入場許可
```

Q. 既に例が書いてある所は見て理解するのみで良いのでしょうか。あとメールアドレスはどれですか。
A. 時間がない場合は理解するのみでも仕方ないですが、基本的に自分で読み書きして見ないと身につきません。車の運転を教科書読むだけでできないのと同じで、知識であると同時にスキルでもあるので、実技や練習は避けて通れません(教養としてちょっとどんなもんか知っときたいとかいう場合を除く)。なので例を少し変えて実行してみたり、自分でちょっとバリエーションを考えてみたり、という作業が大切です。
メールアドレスは最初のページの 2 です。

Q. 演習で習わなかったものを学ぶのはいつどれくらいの時間をかけてやれば良いですか。
A. 授業の範囲ということでは、特にやる必要はありません。習ってない知識が必要になる課題を出したりはしませんし、定期試験もやらないので、特にこれを一式覚えてなければいけないということも全くありません(レポート/Notebook の提出はしてもらいますが)。趣旨は「プログラミングとは何か」を体験し、自分自身の言葉で捉えて、必要があれば自分自身でさらに独習できる素地を作る、というのみです。
なので、例えば、Python のプログラムを使えるレベルで身に付けたいなら、この演習のあと、自由に自習に進んでください。中高生でもできる人はいるし、何のハードルもありませんし、どこまでやらなければならないというリミットも何もありません。教育課程があってここまでやらなければならないという範囲も特にありません。どれくらい時間をかけるかも完全に個人の自由です。やっただけ、スキルが身につく、という点につきます。

Q. 長いプログラミングだと何行くらいになるのですか？今のプログラムの長さでも慣れないので気になっています。
A. 授業ではあまり長いプログラムを作ることはしません。基本的に各回簡潔で知識を増やして行きます。今くらい？ただ、プログラマになりたいとかバイトしたいとかであれば、本当に実用的なプログラムはもちろん数万行から数百万行というのはザラです。しかし、

- 普通、それを全て一人で作ることはあまりなく、チームで作る
- 非常に大規模なプログラムを作るには、別の知識が必要 (コードを実際に書く人よりチームで仕事を率いるマネジャーの方が高度職ですし、設計においては上流であればあるほど高給です。設計指針さえ、きちんとできれば実際にそれをプログラムに落とし込むコーディング自体は労働力が安価な東南アジアなどにアウトソースする会社も多いです)

という感じですね。なお、実用的なプログラムは github などで見れるので、そういうのを見てみると、ちゃんと動いているプログラムがどれくらいのコード量なのかを知ることができるかもしれません。