SICP in Python Japanese Edition Documentation

リリース *2015.04.29*

Paul Hilfinger, Shingo Onobori

2015 年 04 月 29 日

目次

Contents:

第 1 章

第一章：関数を用いた抽象概念の構築

1.1 導入

コンピューターサイエンスは極めて幅の広い学問分野である。分散システム、人工知能、ロボット工学、グラフィックス、サイバーセキュリティ、科学技術計算、コンピューターアーキテクチャ、およびそれらから派生する分野は、新たな技術や発見によって毎年拡大している。コンピューターサイエンスのこの急速な発展により、人間の生活においてコンピューターサイエンスが関わらない分野はほとんど無くなってしまった。広告、通信、科学、芸術、娯楽、そして政治に至るまで全てが、コンピューターサイエンスの領域として再発明されてしまった。

コンピューターサイエンスのこの驚くべき生産性は、華麗で強力ないくつかのアイデアを基礎としているからこそ為し得たのである。全ての計算は、情報を表現することと、情報を処理する論理と、論理の複雑さを織りなす抽象概念を設計することから始まっている。これらの基礎を習得するために我々は、コンピューターが如何にプログラムを解釈し計算プロセスを実行するか、正確に理解しなければならない。

カリフォルニア大学バークレー校では長い間、これらの基礎的なアイデアを教えるために、古典的教科書である「計算機プログラムの構造と解釈 (SICP)(ハロルド・エイブルソン、ジェラルド・ジェイ・サスマン、ジュリー・サスマン 共著)」を用いてきた。ありがたいことに、原著者たちが上記の本に再利用可能なライセンスを認めている。そのため、この講義メモを作成するにあたって大いに参考にさせて頂いた。

我々の知的な旅の同行にあたり、乗客は予備知識を必要としないし、また必要のないものだと我々は考えている。

> 我々は計算プロセスという概念を学習する予定である。計算プロセスとは計算機における抽象概念である。計算プロセスが発達していくと、データと呼ばれる他の抽象概念を操作することがことが分かるだろう。計算プロセスの発達は、プログラムと呼ばれる規則のパターンによって方向づけられている。人間は計算プロセスに指示を与えるためにプログラムを書く。実際、我々は呪文を唱えることによって、コンピューターの魂を呼び出しているのだ。
>
> 計算プロセスを降臨させるために我々が使うプログラムは、まるで魔術師の呪文のようだ。呪文は記号表現を用いて注意深く構築されており、我々のプロセスが果たして欲しい処理を記述するために秘伝の深遠なプ

ログラミング言語を使っている。

計算プロセスは、正確にコンピューターが動作するならば、正確かつ厳密にプログラムを実行する。従って、見習いの魔術師のように、未熟なプログラマーは研鑽に励み、計算プロセスが呼び出される手順を理解し、対処しなければならない。

–エイブルソン&サスマン, SICP(1993)

1.2 Python でのプログラミング

– 言語とはあなたが慣れ親しんだように学ぶものではない – アリカ オークレント

計算機にも分かるよう手続きを定義するために、我々はプログラミング言語を必要とする。いや、プログラミング言語とはむしろ、様々な人間と様々なコンピューターが手続きをしっかり理解するために必要とするものかもしれない。このコースでは、我々はプログラミング言語 Python を学習していく。

Python は今や幅広く使われている言語であり、様々な仕事から求人が出されているほどだ。Web プログラマ、ゲームエンジニア、科学者、研究者、さらには新しいプログラミング言語の開発デザイナまで。あなたが Python を学び始めたその時から、あなたはこれら何百万人もの強力な開発者コミュニティの一員なのだ。開発者コミュニティは極めて重要な組織だ。メンバーはお互いに、問題の解決を図ったり、コードとノウハウの共有をしたり、協力してソフトウェアや道具を作りあげていく。参加したメンバーは、その貢献に対して、しばしば名声や世界的な評価を得ていく。もしかするとあなたもいつの日か、エリート Pythonista(訳註:Python を使う人のことをパイソニスタと呼ぶ) として名を挙げているかもしれない。

Python そのものは、巨大なボランティアコミュニティの成果物であり、彼らは自身たちの多種多様な成果に誇りを持っている。Python という言語は、1980 年代後半に Guido van Rossum(訳註:グイド・ヴァン・ロッサム、GvR とも書く) によって創案され、実装された。彼の書いた「Python 3 Tutorial」の最初の章では、なぜ Python が数多ある言語の中から今日の人気を勝ち取ったのかが記してある。

Python は教育用言語として優れている。なぜなら、その歴史を振り返っても、Python の開発者は Python コードが人間にとって理解しやすいことを重視しているからだ。その上、コードの美しさ、簡潔さ、可読性の基礎として Zen of Python(Python の禅) の精神を用いて強化している。Python はこのコースに特にふさわしい。というのも、Python の持つ幅広い機能は、我々がこれから探ろうとしている多種多様なプログラミングスタイルをサポートしているからだ。Python でのプログラミングに決まった一つのやり方というものは無いが、開発者コミュニティの中で共有されている約束事というものがある。その約束事が、プログラムを読んだり、理解したり、既存のプログラムを拡張するのを助けてくれる。そうして、Python が持つ素晴らしい柔軟性と理解のしやすさが組み合わさることで、学生が様々なプログラミングパラダイムを探り、そこで得た新たな知識を多数の進行中のプロジェクトに応用することを助けてくれる。

この一連の講義メモは、抽象化されたデザインと厳格な計算モデルを応用した技術を紹介すると共に、Python の機能を解説することで、SICP の精神を守り伝えていくことを狙いとしている。加えて、この講義メモは、いくつ

かの進んだ言語機能と図解した例題を含む、Python プログラミングへの実践的導入書となるだろう。このコースを進めていけば、きっとあなたは自然と Python を学ぶことが出来る。

しかしながら、Python は多くの機能を備えたリッチな言語であるため、我々がコンピューターサイエンスの基礎的な概念を学ぶにつれて、その機能をじっくりと段階的に説明していくことにする。Python の機能を一気に網羅的に学習したいという意欲的な学生には、Mark Pilgrim の書いた Dive Into Python 3 をオススメする。これはオンラインで無料で読むことが出来る。この本の目的はこのコースの目的とは異なっているが、Python を使う上で非常に有用な実用的な情報を含んでいる。注意すべきこととして、この講義メモと違って、Dive Into Python 3 は読者が何かしらのプログラミングの経験があることを想定している。

Python プログラミングを始める最良の方法は直接インタプリタを触ってみることだ。このセクションでは Python 3 のインストール方法と、インタプリタを用いた対話環境の初期設定と、プログラミングの始め方を説明している。

1.3 Python 3 のインストール

全ての優れたソフトウェアと同様に、Python もまた、いくつかのバージョンがある。このコースでは最新の安定板である Python 3(執筆時は Python 3.2) を用いることにする。多くのコンピューターでは古くなったバージョンの Python が既にインストールされているが、このコースでは必要ない。あなたはどんなコンピューターを用いてもよいが、Python 3 がインストールされている前提で話を進める。心配は要らない、Python は無料である。

Dive Into Python 3 では主要なプラットフォームでのインストール手順が記されている。これらの手順は Python 3.1 で書かれているが、あなたがインストールするときは Python 3.2 の方が良いだろう (このコースにおいては大した違いではないけれど)。カリフォルニア大学バークレー校電気電子工学及びコンピューターサイエンス学部 (EECS) に備わっている教育用マシンには既に Python 3.2 がインストールされている。

1.4 対話的セッション

Python の対話的セッションにおいては、あなたは何かしらのコードをプロンプト、>>>、の後に入力することになる。Python インタプリタは入力された事柄を読み取って評価し、様々なコマンドを計算する。

対話的セッションを開始するにはいくつかの方法がある。その方法はそれぞれ特徴があり異なっている。とりあえず全ての方法を試してみて、好みの方法を見つけて欲しい。以下の方法は全て同じインタプリタを使っている。

・Python 3 を走らせるために最も単純で広く使われている方法は、ターミナルのプロンプトで python3 と入力することである (Mac/Unix/Linux)。Windows の場合は Python 3 アプリケーションを開くことである。

・上の方法よりユーザーフレンドリな方法は、Idle 3(idle3) と呼ばれるアプリケーションを起動することである。Idle は書いたコードを色付けし (シンタックスハイライトと呼ぶ)、使い方のヒントを提示し、ソースコードのエラー部分に印を付けてくれる。Idle はいつも Python に同梱されている。つまり、あなたは既にそれを使える状態にある。

・Emacs エディタ内で対話的セッションをバッファとして走らせることが出来る。学習に多少のコストはかかるものの、Emacs はどんなプログラミング言語にとっても強力で万能なエディタである。使い始めるにはこの講座の Emacs チュートリアル (訳註:後述リンク) を読むと良い。

// 訳者註：UCB EECS のコース CS61A(http://cs61a.org) に、
// Emacs, SublimeText, Vim のチュートリアルがありました。
// お好みのものを使って下さい。
//
// Emacs
// http://cs61a.org/lab/emacs/
//
// Sublime Text
// http://cs61a.org/lab/sublime/
//
// Vim
// http://cs61a.org/lab/vim/

多くの場合、Python プロンプトは >>> で表される。対話的セッションの開始に成功すると、それが表示される。この講義メモではプロンプトを用いて例を記述する。以下がその例だ。

```
>>> 2 + 2
4
```

使い方：各セッションでは、どんな入力があったかというヒストリを記録している。ヒストリにアクセスするためには、<Control>-P(前へ戻る) もしくは<Control>-N(次へ進む) を押す (訳註:Ctrl キーを押しながら P や N のキーを押す)。<Control>-D でセッションを抜けることが出来る。その際、このヒストリも消えてしまう。

1.5 最初の例題

And, as imagination bodies forth
The forms of things to unknown, and the poet's pen
Turns them to shapes, and gives to airy nothing
A local habitation and a name.

—ウイリアム・シェイクスピア、真夏の夜の夢

Python の導入として、いくつかの Python の言語機能を用いた例題から始めようと思う。次のセクションにおいては、我々はゼロから始めて、言語を一つ一つ積み上げていく。このセクションは実装予定の強力な言語機能の予習だと考えてほしい。

Python は様々な分野でのプログラミングで共通して使われる組み込み関数をサポートしている。構文解析、グラフィック表示、インターネット経由でのコミュニケーションなどだ。Import 文を用いて

```
>>> from urllib.request import urlopen
```

と組み込み関数を読み込めば、インターネット上のデータにアクセスできるようになる。上の例では、urlopen と呼ばれる関数が利用できるようになる。urlopen 関数は、インターネット上のモノの場所を示す、統一リソースロケータ (URL) 内のコンテンツにアクセス

第 2 章

Indices and tables

- genindex
- modindex
- search