

ステップ30

Python [基礎編]

ワークブック



はじめに

本書は Python の基礎を習得することが目的の書籍です。

また、すでに他の Python 入門書籍・サイト・ビデオなどで学習していて、それら学習内容の定着具合を確かめたい方にも活用いただける教材です。自身の習熟度合い・理解度を客観的に見ることは難しく、プログラムを書いたり読んだり、レビューをもらったり、それらを繰り返して自分に足りない部分を学んでいくのが一般的ですが、本書は演習問題が付属したワークブック形式となっており、習熟度・理解度の確認が容易になっています。

最近では機械学習が注目されたこともあり、Python 入門書籍で簡単な画像認識などを扱うことも多いのですが、本書では Python の基礎的な内容を中心に学びます。どのような分野に進んでも本書で学んだ内容が土台となるはずです。

■ 読者対象

プログラミングをはじめて学ぶ工業高校生を対象読者に想定しています。工業高校生に限らず、はじめてプログラミングを学ぼうとする大学生や社会人の皆さんにも役に立つと思います。

■ 本書の構成

本書では次の構成をとります。

- 要点として、記憶してほしい語彙や概念を示します。
- ワーク [基礎] として、簡単な問題を出題します。
- ワーク [応用] として、やや高度な問題を出題します。

ワークの中には、答えが複数出てくる問題もあります。学校などの多くの方が同時に学習する環境ならば、ぜひ皆さんに様々な解答を出していただき、他人の発想を学んだり、その解答が適切かを議論していただければと思います。

Step 01	Python とは.....	7
	1.1 Pythonの歴史.....7 1.2 Pythonの特徴.....7	
Step 02	四則演算	9
	2.1 Pythonでの四則演算 + α9 2.2 ZeroDivisionError.....9	
	2.3 演算の優先順位.....10	
Step 03	変数	12
	3.1 変数の使い方.....12 3.2 変数名に使える名前.....12	
	3.3 変数の応用①.....13 3.4 変数の応用②.....14	
Step 04	組み込み型 文字列①.....	16
	4.1 組み込み型とは.....16 4.2 文字列を扱う.....16	
	4.3 組み込み関数.....17	
Step 05	組み込み型 文字列②.....	20
	5.1 文字列のメソッド.....20 5.2 インデクシングとスライシング.....21	
Step 06	組み込み型 リスト①.....	24
	6.1 リストとは.....24 6.2 リストのメソッドと、組み込み関数.....25	
Step 07	組み込み型 リスト②.....	28
	7.1 ミュータブル.....28 7.2 ミュータブルな型の注意点.....29	
Step 08	組み込み型 タプル.....	31
	8.1 タプルとは.....31 8.2 リストとの違い.....31	
	8.3 アンパック.....32	
Step 09	組み込み型 辞書	35
	9.1 辞書とは.....35 9.2 辞書のメソッド.....36	
Step 10	組み込み型 集合	39
	10.1 集合とは.....39 10.2 集合のメソッド.....40	
Step 11	組み込み型 まとめ.....	42
	11.1 bool型.....42 11.2 NoneType型.....42	
	11.3 組み込み型まとめ.....42	

Step 12	条件分岐 if文①	44
	12.1 if文の基本.....44 12.2 インデント.....45	
Step 13	条件分岐 if文②	48
	13.1 比較演算.....48 13.2 ブール演算.....49	
	13.3 暗黙のTrue、False.....50	
Step 14	繰り返し for文①	53
	14.1 for文の基本.....53 14.2 breakとelse.....55	
Step 15	繰り返し for文②	58
	15.1 指定回数の繰り返し.....58 15.2 rangeオブジェクト.....59	
	15.3 enumerateとzip.....60	
Step 16	繰り返し while文	64
	16.1 while文とは.....64	
Step 17	関数①	67
	17.1 関数とは.....67 17.2 引数.....68	
	17.3 戻り値.....70	
Step 18	関数②	72
	18.1 デフォルト引数.....72 18.2 可変長位置引数.....73	
	18.3 可変長キーワード引数.....74 18.4 キーワード専用引数.....75	
Step 19	関数③	77
	19.1 スコープ.....77 19.2 ミュータブルな型の注意点 再び.....79	
Step 20	クラス①	82
	20.1 クラスとは.....82 20.2 self.....83	
Step 21	クラス②	87
	21.1 継承とは.....87 21.2 オーバーライド.....88	
	21.3 superで親のメソッドを呼ぶ.....89	
Step 22	クラス③	92
	22.1 クラスの属性.....92 22.2 インスタンス属性との区別.....93	
Step 23	モジュール	96
	23.1 モジュールとは.....96 23.2 モジュールの直接実行.....97	
Step 24	パッケージ	100
	24.1 パッケージとは.....100 24.2 __init__.py.....101	
Step 25	入出力	104
	25.1 ファイルの書き込み.....104 25.2 ファイルの読み込み.....105	

	25.3 他のモード……105	25.4 エンコーディング……105	
Step 26	例外		109
	26.1 例外を捕まえる……109	26.2 finally と else……110	
Step 27	ライブラリ		113
	27.1 標準ライブラリとは……113	27.2 サードパーティ製ライブラリ……114	
Step 28	迷路アプリケーション①		116
	28.1 迷路アプリケーションの概要……116	28.2 プロトタイプを作る……117	
Step 29	迷路アプリケーション②		122
	29.1 移動処理を実装する……122	29.2 座標のチェック処理……123	
Step 30	迷路アプリケーション③		128
	30.1 クラスを使う……128		
付録 A	Python のインストール		133
	A.1 Windows……133	A.2 Mac……134	
付録 B	対話モードで実行する		135
	B.1 対話モードに入る……135	B.2 他のバージョンの Python を使う……135	
	B.3 対話モードを試す……135		
付録 C	Python スクリプトの実行		136
	C.1 Python スクリプトの実行方法……136	C.2 エディタ・IDE の紹介……136	
	C.3 IDLE の開き方と使い方……137		
	索引		141

※ワークの解答は、以下の Web ページからダウンロードできます。

<http://----->

Python とは

学習のはじめに、Python というプログラミング言語について説明します。Python とは何か？ Python の何がいいのか？ そのような質問をされても答えられるようにしましょう。

要点

1.1 Python の歴史

1989 年の 12 月、オランダ人のガイド・ヴァンロッサム (Guido van Rossum) はクリスマス休暇の暇つぶしとしてプログラミング言語の開発を始めました。これが Python です。その後 1991 年 2 月に alt.sources ニュースグループ上でバージョン 0.9.0 が一般公開され、今ではバージョン 3 も後半です。

4 大 P 言語 (Perl, Python, PHP, Ruby) の中では 2 番目に古く、意外に感じるかもしれませんが Java や C# よりも年上となります。1957 年の FORTRAN (フォートラン) から始まるプログラミング言語の歴史から見ると 90 年前後というのは最近に感じますが、最近よく使われているメジャーな言語の中ではそれなりに古く、歴史のある言語といえます。

余談ですが、「Python」という名前の由来はイギリスのコメディ番組「空飛ぶモンティ・パイソン」から取っています。

1.2 Python の特徴

Python の特徴は多くありますが、代表的なものは可読性・生産性・汎用性の 3 つです。

■ 可読性

可読性とはプログラムの読みやすさのことです。Python は「実行可能な疑似コード」と表現されるほど、自然言語に近い言語です。他言語では何かと出てくる丸括弧などの記号も少なくなるように設計されています。コードが読みやすいとそれだけ学びやすく、覚えやすく、上達が早くなりますし、他人との共同作業もはかどります。

■ 生産性

ある時間内や行数で、どれだけのプログラムや処理が作れるかを生産性といいます。Python はスクリプト言語と呼ばれるグループの仲間です。スクリプト言語の多くはプログラムの実行が簡単で、少ないコードでたくさんのことができます。C や Java といった言語に比べるとコードの量が半分以下になることも珍しくありません。

■ 汎用性

Python は幅広い用途に使える言語なため、汎用的な言語といわれます。Web アプリケーショ

ンや科学分野をはじめ、多くの分野で実際に利用されています。汎用目的に作られた言語は数多くありますが、ある分野での開発が現実的ではない言語もあります。例えば、Web が主戦場の PHP でスマホアプリの開発は（少なくとも今は）現実的ではないでしょう。Python にも不得意なこと、または他言語の方が得意な分野はもちろんあるのですが、総合的にはとても汎用的な言語です。

プログラミング言語に最も必要なものは、コミュニティです。どんなに素晴らしい言語であっても、利用者が少なければ今後の発展は見込めず、情報の取得に苦労するでしょう。Python はどうでしょうか？ ご安心ください、とてもホットな言語です。ある言語がどれだけ人気があるかの指標として、以下の 3 サイトはよく参考にされます。どのサイトでも Python は高い位置に存在することが分かります。

<http://pypl.github.io/PYPL.html>

<https://www.tiobe.com/tiobe-index/>

<http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2017>

ワーク

基礎

次のうち、Python の説明として正しいものを 1 つ選びましょう。分からない語句は、Google 検索を活用しても構いません。

1. Python は生産性が低く、読みづらい言語だ。
2. Python はオブジェクト指向をサポートしていない。
3. Python は可読性、生産性、汎用性に優れた静的言語だ。
4. Python は可読性、生産性、汎用性に優れたスクリプト言語だ。

四則演算

Pythonに付属している対話モードはちょっとした電卓にも使えます。対話モードで四則演算 + α を試してみましょう。安物の電卓より、Pythonでの数値計算のほうが便利です。Pythonのインストールは付録Aを、対話モードの使い方は付録Bを見ましょう。

要点

2.1 Pythonでの四則演算 + α

足し算、引き算、掛け算、割り算のことを四則演算といいます（すでに知ってますよね?）。Pythonでは四則演算はもちろん、他にも様々な計算ができます。実行結果2.1.1で確認しましょう。

実行結果 2.1.1 対話モードでの実行例

```
>>> 1 + 2
3
>>> 3 - 2
1
>>> 5 * 2
10
>>> 5 ** 2
25
>>> 4 / 2
2.0
>>> 4 // 2
2
>>> 10 % 3
1
```

掛け算が \times ではなく $*$ （アスタリスク）に、割り算が \div ではなく $/$ （スラッシュ）になります。 $/$ による割り算は小数点が表示されますが、小数点部分を切り捨てたいならば $//$ とします。 $**$ で指数の計算ができ、 $\%$ （パーセント）記号で剰余（割った余り）を求めることもできます。

2.2 ZeroDivisionError

数学では、0で割ることは許されない行為です。Pythonでは0で割ったときに実行結果2.2.1のように表示されます。

実行結果 2.2.1

```
>>> 1 / 0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
```

これは例外と呼ばれるもので、Python で実行中に検出されたエラーのことです。この例外を扱う方法は後に説明しますが、今はエラーがこのように表示されることを覚えてください。

2.3 演算の優先順位

演算には優先順位があり、それは数学での優先順位と同じです。

実行結果 2.3.1

```
>>> 10 + 20 - 10 / 5 * 100 ** 0
28.0
```

実行結果 2.3.1 では、まず 100^{**0} の指数が計算され、掛け算・割り算の後に足し算・引き算が計算されます。通常の数学における優先順位と同じです。

実行結果 2.4.1

```
>>> 10 + (20 - 10) / 5 * 100 ** 0
12.0
```

実行結果 2.4.1 を見てください。丸括弧をつけてみました。丸括弧内の演算が優先されるのも同様です。

ワーク

基礎①

次のうち、ZeroDivisionError となるものを 1 つ選びましょう。

1.

```
>>> 5 - 0
```
2.

```
>>> 5 * 0
```
3.

```
>>> 5 / 0
```
4.

```
>>> 6 ** 0
```

基礎②

「+」「-」「*」「/」記号を1つずつ使い、括弧の中を埋めましょう。

```
>>> 10 (    ) 5
15
>>> -10 (    ) 10
-20
>>> 10 (    ) 3 (    ) 5
6.0
```

応用

「%」「//」「**」記号を1つずつ使い、括弧の中を埋めましょう。

```
>>> 10 (    ) 3
3
>>> 10 (    ) 3
1
>>> 5 (    ) 2
25
```

変数

変数を使うことで、計算結果や各種データを保存することができます。Pythonに限らずプログラミングにおいては重要な概念となるので、しっかり覚えましょう。

要点

3.1 変数の使い方

実行結果 3.1.1 は、price という変数に整数の 100 を保存し、その変数を使うサンプルコードです。

実行結果 3.1.1

```
>>> price = 100
>>> price
100
>>> price + 10
110
```

変数は、計算結果やデータを保存するのに使われます。price = 100 のように左辺に名前を、右辺に値や式を書きます。これを「変数 price に 100 を代入する」と表現します。代入が終われば、次からは price という名前で自動的に 100 の値が使われます。2 行目の price は 100 と表示され、4 行目の「price + 10」は「100 + 10」となり 110 と表示されます。= は数学では等しいという意味ですが、Python では代入を意味する記号ですので注意してください。また、他の言語でよくある変数名だけを宣言する機能はありません。

3.2 変数名に使える名前

変数名は自由につけることができますが、いくつかのルールがあります。

- 変数名の先頭には _ (アンダースコア)、もしくはアルファベットしか使えない
- 変数名の 2 番目以降には、数字、アルファベット、アンダースコアが使える
- 予約語は使えない

予約語とは、Python プログラムの構文として使われているキーワードです。予約語を確認するには、実行結果 3.2.1 のように入力してください。