実習で身につく！

新しい Python の教科書

～ Python の基本スキルから機械学習の初歩まで～

Python 3.9

Beginner

AI

Machine learning application

Web scraping

SCC

実習で身につく！

新しい Python の 教科書

～ Python の基本スキルから機械学習の初歩まで～

境 祐司 著

著者略歴

境 祐司

Instructional Designer, Adobe Community Evangelist
インストラクショナルデザイナーとして教育設計、IDマネジメントなどを中心に活動。2016年より、クリエイティブ活動におけるAI（人工知能）活用のプロジェクトに参画、AIシステムやロボティクス関連の実証実験に携わる。
2017年より、Adobe Community Evangelistとして、主にAdobeSensei（AdobeのAI技術の総称）やXD関連のイベント登壇、講演などをエンタープライズ対象に開始。
2020年3月以降、コロナ禍におけるオンラインを前提としたビジネススタイルに完全移行し、クリエイティブ作業のリモートワークやプロジェクト管理システム、商談システムなどの検証を行いながら実践導入、パートナー企業へのアドバイスを行っている。

Yuji Sakai

Instructional Designer, Adobe Community Evangelist
Yuji Sakai is an Instructional Designer who performs instructional design and educational management for companies and schools.
Since 2016, he participated in a project on the use of artificial intelligence (AI) in creative activities and was involved in demonstration experiments related to AI systems and robotics. Since then, he has implemented the neural network-based "Style Transfer" into his creative work in practice.
Since 2017, he has been speaking at Creative Cloud product events as an Adobe Community Evangelist (Japan).

Twitter ■ @commonstyle
Web ■ design-zero.tv

著書

『新しいJavaScriptの教科書』（SCC）『Adobe Museランディングページ制作ガイド ～コード知識ゼロで作るWeb広告』（技術評論社）など。

サンプルプログラム／サポート情報のご案内

下記の本書専用サポートページでは、サンプルプログラムのソースコード、正誤情報や、本書に記載された項目等に関する補足情報などを、必要に応じて掲載します。

本書専用サポートページ
https://www.scc-kk.co.jp/scc-books/b-415.html

なお、サポートページの内容は必要に応じて随時更新されますので、ご注意下さい。

はじめに

Pythonは「書きやすく」「読みやすい」言語として知られており、プログラミングを初めて学ぶ人にとっても敷居が低い「学びやすい」言語になっています。

AI（人工知能）ブームと共に脚光を浴びたプログラミング言語ですが、歴史を見るとJavaよりも古く、バージョン1.0のリリースは1994年1月（開発が始まったのは1989年）ですから、すでに27年近く経っています。

PythonにはAI関連の便利なライブラリが充実していたことから、2012年以降、知名度・認知度を一気に高めました。

人気が高いだけに学習リソースは豊富で、基礎を学習した後は独学でレベルアップしていくことができます。世界中に活発なコミュニティがあり、最新情報や高度なテクニックを学べることも、Pythonを習得する大きな利点だと言えるでしょう。

本書は、プログラミングの初心者を対象としており、Pythonを学ぶための準備からプログラムの書き方、基本文法（変数やデータ型、演算子、リスト）、制御構文、関数までの基礎学習、そして簡単なアプリの作成や機械学習を体験する演習などを含み、全12章で構成されています。

プログラミングの初心者が学習をやめてしまう理由はさまざまですが「頭で覚える」ことに偏ってしまうと、学習が進むにつれて辛くなってくることは間違いありません。

本書は、最初に仕様を理解したら、実際にプログラムを書いて理解を深めていく「実践型」の学習方法を取り入れていますので、体系的に習得していくことが可能です。「まずは動かしてみよう！」という意識が重要だと考えています。

本書をきっかけにプログラミングの楽しさ、奥の深さを感じてもらえたら幸いです。

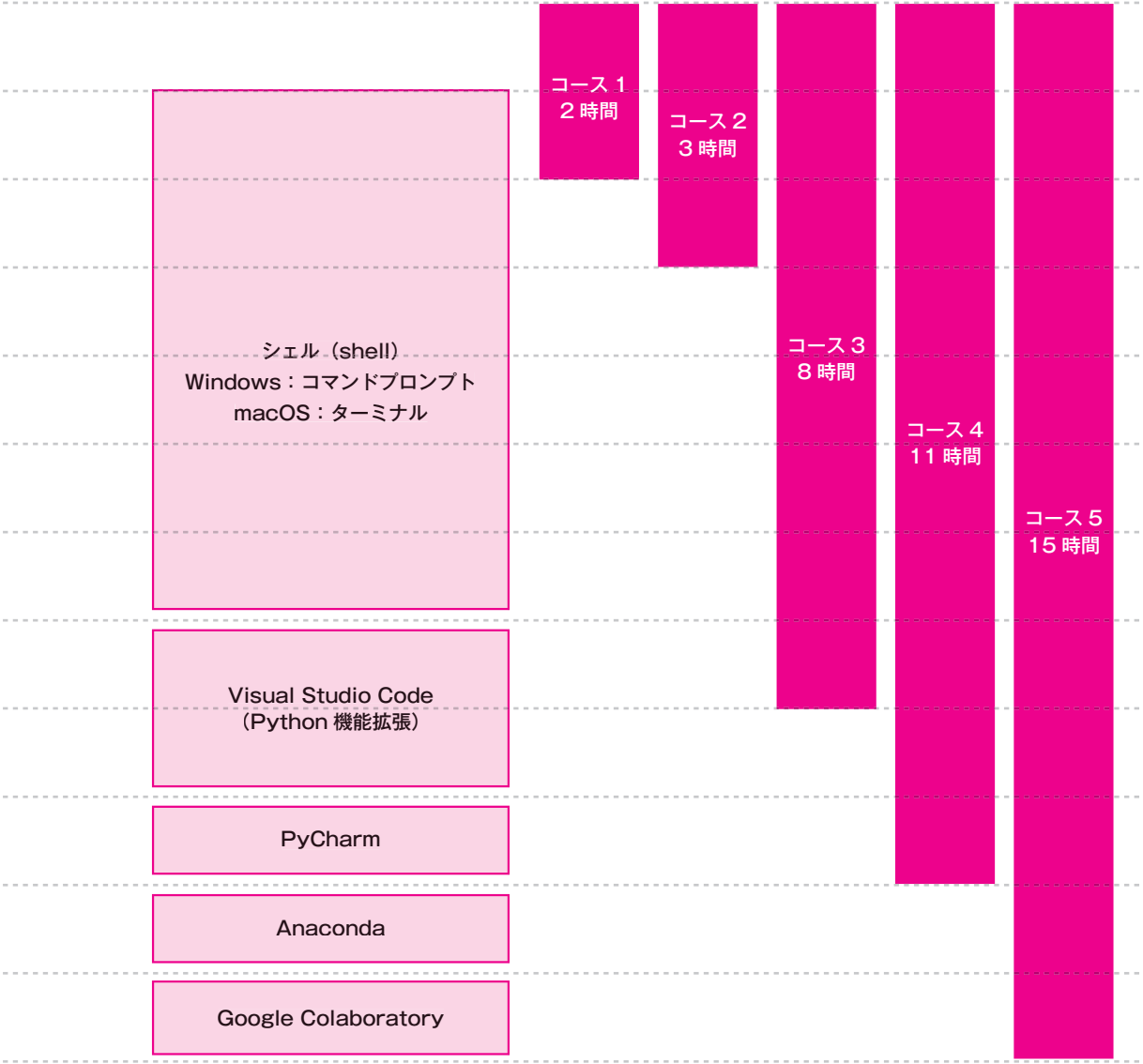
2020年11月

境 祐司

本書の読み方

本書は全 12 章を通して、Python に関するさまざまな知識を解説しています。15 時限の授業を想定して構成していますが、ご自身の学習しやすいペースで読み進めてください。

コンピューターとデータ活用についての理解、プログラミングによる問題の発見と解決について	Chapter 1 Python の概要
	Chapter 2 学ぶための準備と基本文法
Python の基本文法 OS に付属しているシェルを使用して基本文法を学ぶ	Chapter 3 変数とデータ型
	Chapter 4 演算子
	Chapter 5 リスト
	Chapter 6 制御構文
	Chapter 7 関数
コードエディタを使用した演習 初歩的なデスクトップアプリのプロトタイプを制作	Chapter 8 基本演習：アプリのプロトタイプ制作
コードエディタを使用した概念の理解のための学習	Chapter 9 オブジェクト指向プログラミング
Python 専用の統合開発環境を使用した演習	Chapter 10 ライブラリの活用
Python のディストリビューションを使用した演習	Chapter 11 総合演習：Web スクレイピング
クラウドの Python 実行環境を使用した演習	Chapter 12 機械学習


コース 1
2 時間
コース 2
3 時間
コース 3
8 時間
コース 4
11 時間
コース 5
15 時間
シェル（shell）
Windows：コマンドプロンプト
macOS：ターミナル
Visual Studio Code
（Python 機能拡張）
PyCharm
Anaconda
Google Colaboratory

目次

Chapter 12 機械学習 301

Chapter

1

Pythonの概要

Python は「書きやすく、読みやすい」言語として知られています。また、Chromebook のような安価なパソコンとネットに接続できる環境があれば、すぐに Python の学習を始めることができます。Chapter 1 では、プログラミング言語の歴史から Python の概要までを学習していきます。また、簡単なコードを書いて初歩のプログラミングを体験します。

本章で学ぶこと

▶ 達成目標：

1. 「プログラミング」や「プログラム」の言葉の意味について説明できる
2. プログラミング言語の歴史を大まかに理解することができる
3. Python がなぜ「書きやすい言語」と言われているのか理解できる
4. 「スクリプト言語」や「スクリプティング」の言葉の違いについて説明できる
5. なぜ AI の分野で利用されているのか説明できる

▶ 学習ポイント：

- 「プログラミング」や「プログラム」の言葉の意味について解説します
- 機械語から高水準言語までのプログラミング言語の歴史について学べます
- Python が「書きやすく、読みやすい」言語として認知されている理由を解説します
- 「スクリプト言語」や「スクリプティング」、「スクリプトファイル」などの用語について解説します
- Python が AI の分野で利用されている理由について解説します

▶ 学習の流れ：

- Step-01 コンピューターとプログラミング言語
- Step-02 Python の歴史と特徴
- Step-03 なぜ AI の分野で利用されているのか？

Step 01

コンピューターとプログラミング言語

プログラミング言語の歴史を学び、Python を使った簡単なプログラミングを体験します。

機械語から高水準言語まで

プログラミング言語とは「コンピューターと人間の両方」がプログラムの内容を理解できる言語のことです。

「プログラミング」とは人間がコンピューターに対して命令を書くことで、プログラムを実行するとコンピューターが命令に従って処理します。人間が書く命令文のことを「ソースコード」と呼び、書き方や単語などを定義した「文法」に従います。ソースコードのまとまりが「プログラム」です。

- **プログラミングとは人間がコンピューターを動かすために命令を書くこと**
- **プログラムとは人間が書く命令文（ソースコード）をまとめたもの**
- **プログラミング言語には文法がある**

コンピューターが登場した1940年代は「機械語」を使ってプログラムを書いていました。ソースコードは「8B150216」のような数字とアルファベットの羅列で、最終的に「0か1かの2進数」に変換されてコンピューターのCPUで実行されます。人間より機械に都合のよい言語だったわけです。

私たちは0〜9までの10進数で「数」を扱っていますが、コンピューターは0と1の2進数で数を処理しています。前者を「10進数」、後者を「2進数」と呼びます。

10進数の「31」は2進数にすると「11111」になり桁数が増えます。例えば「11111」を「1」と「1111」で表し、10進数にすると「1」と「15」になりますが、これを16進数にすると「1F」となり桁数がかなり減ります。

16進法は10〜15をアルファベットのA〜Fで表しますので、2進数「10000000」を「80」で表すことができます。桁数の多い2進数は16進数に変換することで扱いやすくなります。

10進数	2進数	16進数
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
32	100000	20
64	1000000	40

現在普及しているプログラミング言語では、英語を書くように作業を進められますが、機械語は「機械」が理解しやすい言語であり、人間がプログラムを書くのは至難の業でした。

状況が変わったのは「アセンブリ言語」の登場からです。人間が理解可能な記号に置き換えることができるようになりました。アセンブリ言語を使えば「MOV AX,DX」のように記述することが可能になったのです。

1950年代の後半からは、FortranやCOBOLなどの「高水準言語（高級言語）」と呼ばれる、より人間の言語に近い表現が使えるようになり、プログラミングも身近な技術として世界中に普及していきました。

その後、if文やwhile文、for文などで制御構造を自由に記述できるPascalやC言語、ALGOLなどの構造化言語が登場します。JavaやC++、C#などは、この時代のC言語の性質を継承している言語です。

※if文やwhile文などはChapter 6で学習しますので、ここでは「より人間が理解しやすい」プログラミング言語になったということを覚えておいてください。

- **機械語　→例：8B150216**
- **低水準言語　→例：MOV AX,DX**
- **高水準言語（高級言語）　→例：print ('おはよう！')**

コンピューターを動かすプログラミング言語は、機械語から「低水準言語」のアセンブリ言語、そして「高水準言語（高級言語）」へと進化し、人間が理解しやすい言語に変わりました。

特に、PythonやJavaScript、PHPなどは、シンプルな文法で初学者にとって敷居の低い「学びやすい言語」として知られています。

Pythonは書きやすく、読みやすい言語

PythonAnyWhereでプログラミングを体験

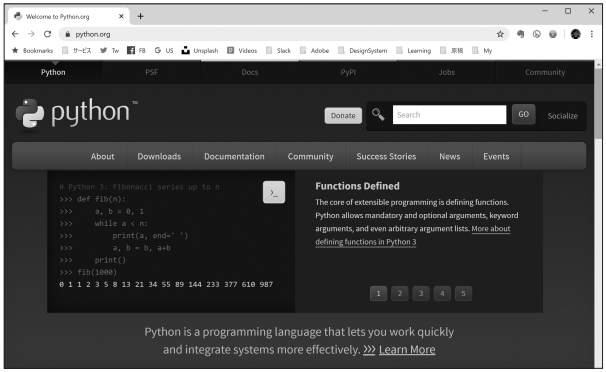
Pythonがどのくらい「書きやすい」のか、実際に体験してみましょう。Pythonの公式サイトにはクラウドサービス「PythonAnyWhere」が組み込まれており、誰でも利用できるようになっています。

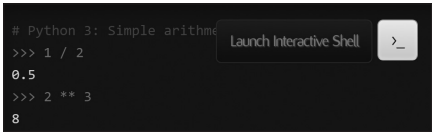
Pythonを自分のパソコンにインストールする必要はありません。Webブラウザーがあれば、すぐにコードを書くことができます。

※PythonAnyWhereは、Python2.x／3.xをサポートしているオンラインのIDE（統合開発環境）およびWebホスティングサービスです。iPadやChromebookでも利用できるため、学校などでも利用されています。

それでは、Pythonの公式サイトをWebブラウザーで開いてください。トップページに「Launch Interactive Shell」の黄色のボタンがありますのでクリックします。

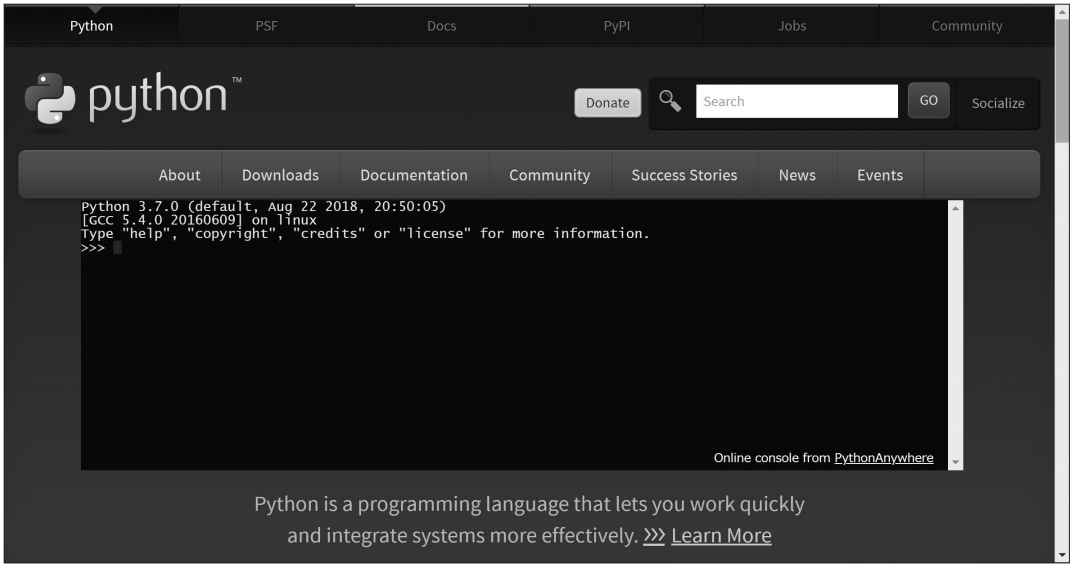
- **Welcome to Python.org（Python公式サイト）**
  https://www.python.org/





Python が利用できるシェルが起動します。少し時間がかかります。

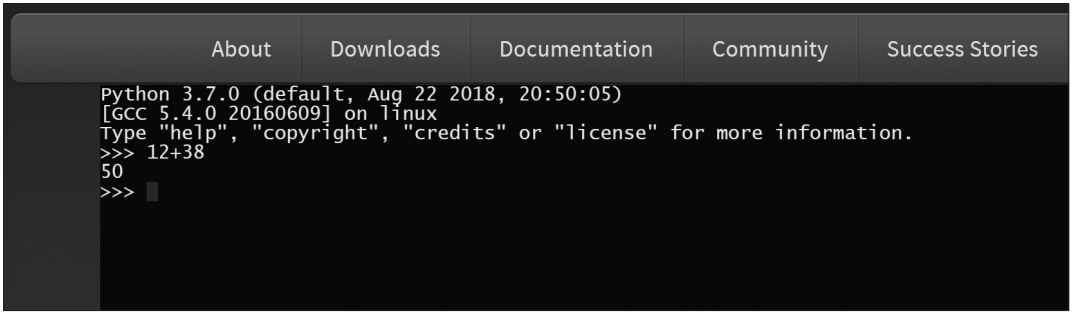
※シェルとは、コマンド（命令）を入力してコンピューターを操作するためのアプリケーションソフトです。Chapter 2 で学習します。



最後の行に「>>>」と表示されたら準備完了です。

「12 + 38」と半角数字と記号を入力して、Enter キーを押してください。数字と記号の間に半角スペースを入れておくと見やすくなります。「50」と表示されたでしょうか？

簡単な計算ですが、これで Python が正しく動いていることが分かりました。



次は3行のコードを書いてみましょう。

1行目です。半角英数字で入力してください。入力後はEnterキーを押します。

```
x = 8
```

2行目を入力してEnterキーを押してください。

```
y = x * 6
```

最後です。「print」の後に「()」を追加して、中に「y」を入れてください。

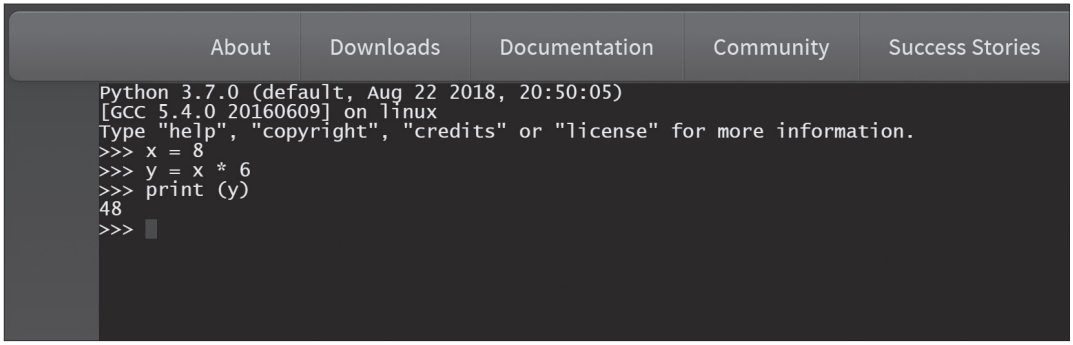
※()は半角の記号です。

```
print (y)
```

入力後、Enterキーを押すと計算が実行され「48」と表示されます。

1行目でxに8が代入され、2行目では「8 × 6」が実行され（掛け算は「*（アスタリスク）」を使います）、計算の結果がyに代入されます。そして、最後の行でyに代入された計算結果を表示するという流れになります。

入力したコードの内容についてはこれから学習していきますので理解する必要はありません。ここでは「Pythonのプログラムを動かす」という体験を目的とします。



このように、Chromebookなどの安価なパソコンやインターネットに接続できる環境があれば、すぐにPythonの学習を始めることができます。Pythonのコードもシンプルで「書きやすい」「読みやすい」と評価されている理由が理解できたと思います。

Chapter 2からは自分のパソコンにPythonをインストールして、学習を進めていきますが、環境構築不要のクラウドサービスなど、さまざまな利用形態が提供されていることを覚えておいてください。

Step 02

Pythonの歴史と特徴

Python の歴史を学び、スクリプト言語やスクリプティングなどの用語について理解します。

他人のプログラムを容易に理解できる言語

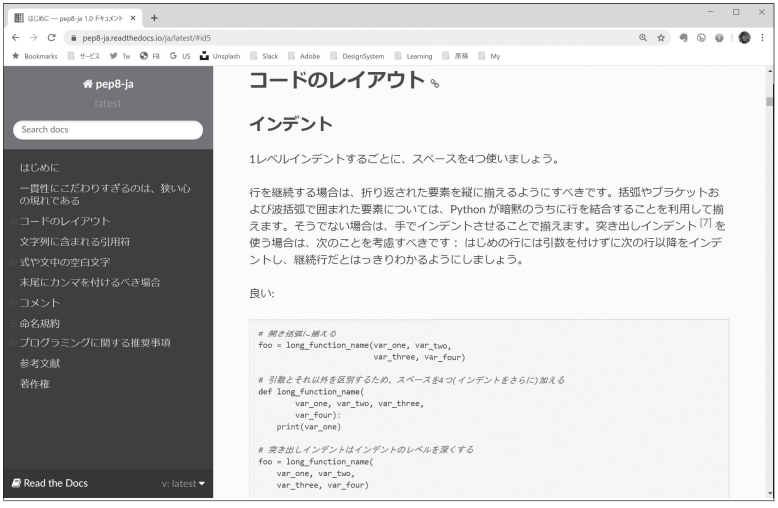
Pythonはプログラミング言語の中でも、初学者にとって「書きやすく」「読みやすい」言語であることはStep-01の体験で理解できたと思います。

Pythonはプログラムを簡潔に書けるだけでなく、インデント（字下げ）で処理の区切りを表すため、誰が書いても同じような体裁の記述になります。つまり、他の人が書いたプログラムを容易に理解できるということです。

Pythonのコードを書く際のガイドラインである「PEP8」には、コードのレイアウトについて記されており、冒頭には「1レベルインデントするごとに、スペースを4つ使いましょう。」と書かれています。

▶ **参考：**

- **Python コードのスタイルガイド　コードのレイアウト：インデント**
  https://pep8-ja.readthedocs.io/ja/latest/#id5



また、Pythonは「Battery Included」と言われています。まるで「電池が付属している製品」のように、インストール後すぐに本格的なプログラミングができるという意味です。

ライブラリと呼ばれる「小さなプログラム（汎用的な機能）」が豊富で、必要に応じて簡単に使うことができますので、少ないコードでも高度なプログラムを書くことができます。

大規模なプロジェクトでも採用されている

文法がシンプルで「書きやすく読みやすい」プログラミング言語でありながら、大規模なWebサービスなどでも利用されています。

例えば、月間ユーザー数が10億人を超えているInstagram（インスタグラム）や画像共有サービスのPinterest（ピンタレスト）、オンラインストレージサービスのDropbox（ドロップボックス）などの開発にもPythonが使われました。

Pythonは「オブジェクト指向プログラミング（OOP:Object Oriented Programming）」言語です。Pythonを習得するにはオブジェクト指向について理解しなければいけませんが、ここでは詳しく解説しません。言葉だけ覚えておいてください。

オブジェクト指向は、インターネットが商用化された1994年以降、Javaのブームと共に普及していきました。Simula67（1967年）やSmalltalk（1980年）など長い歴史がありますが、一般に普及し始めたのは90年代半ばです。

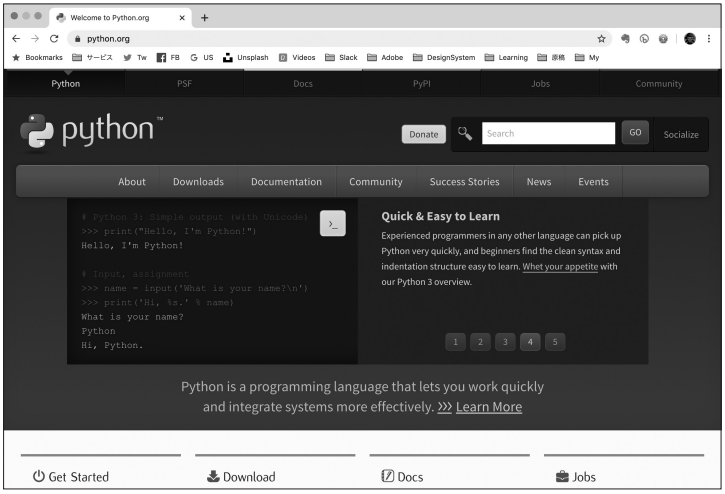
オブジェクト指向プログラミングに分類される代表的な言語には、C++やJava、Ruby、C#、Objective-C、Visual Basic.NET、Kotlinなどがあります。

機械学習の世界的なブームが追い風

現在、Pythonはプログラミング言語のランキングで上位をキープしている大変人気のある言語として知られていますが、歴史はJavaよりも古く、開発が始まったのは1989年です。バージョン1.0が公開されたのは1994年1月で、インターネットが商用化された年なのです。すでに27年近く経っています。

節目は2012年です。ディープラーニングと呼ばれるAI（Artificial Intelligence：人工知能）技術の研究報告が大きな注目を集めましたが、この人工知能の領域でよく利用されているプログラミング言語がPythonだったのです。

Pythonの人気は、ディープラーニング／機械学習の世界的なブームが追い風となって急速に拡大していったと考えてよいでしょう。



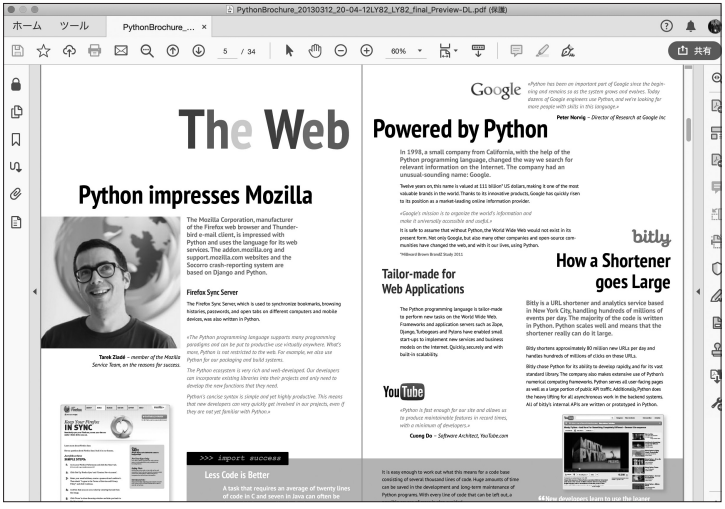
▶ 参考：

- **Welcome to Python.org（Python公式サイト）**
  https://www.python.org/

Pythonのパブリシティ

2001年3月には、Pythonソフトウェア財団（Python Software Foundation：PSF）が設立され、資金調達や知的財産権の管理、Pythonカンファレンスの運営などを行っています。

財団の公式サイトからPythonのパンフレット（PDF）をダウンロードできます。



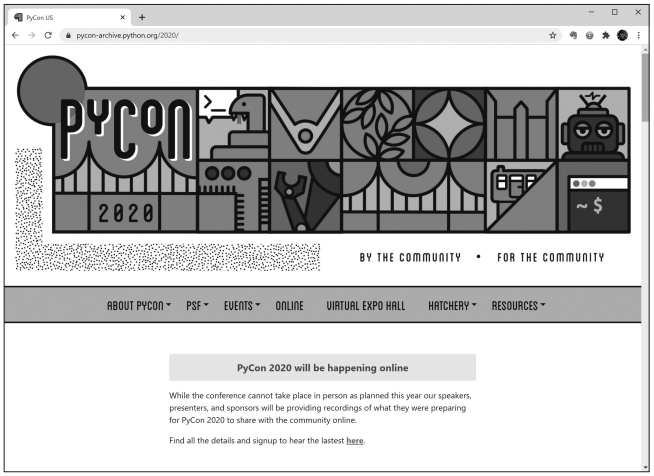
▶ **参考：**

- **Learn more about the PSF Python Brochure Project**
  https://brochure.getpython.info/learn-more

2020年のPyConカンファレンスはオンラインで開催されました。

PyConの公式サイトは、PythonのWebアプリケーションフレームワークDjangoと（Djangoを使って開発された）コンテンツ管理システムのSymposionで構築されています。

- **PyCon US**
  https://pycon-archive.python.org/2020/



国内にも一般社団法人PyCon JPがあり、日本で開催されるPyConカンファレンスを運営しています。2020年は8月28日、29日にオンラインで開催しました。カンファレンスのセッションはビデオ収録されており、YouTubeチャンネルで公開されています。

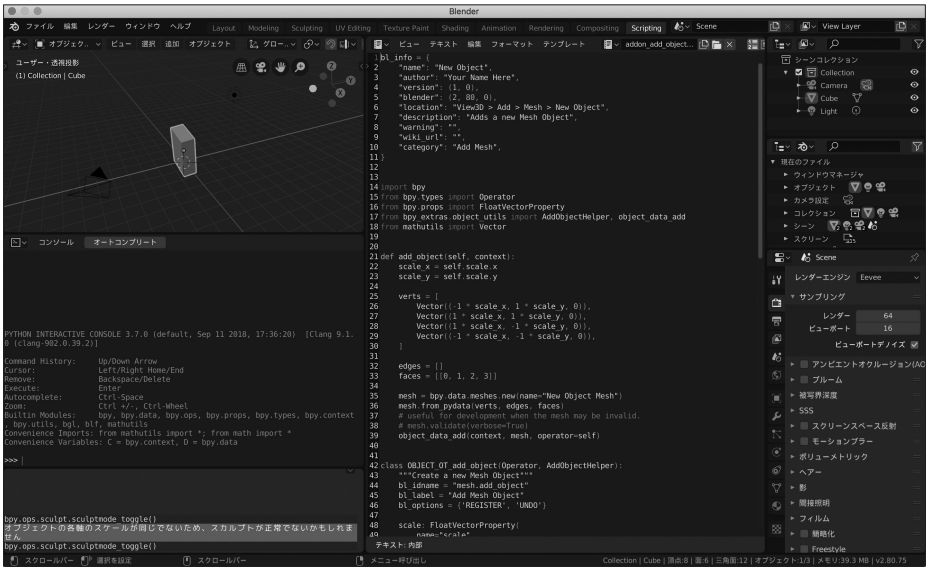
▶ **参考：**

- **一般社団法人PyCon JP**
  http://www.pycon.jp/

スクリプト言語としてのPython

Pythonは、3DCG制作ツールの「Blender（ブレンダー）」や3Dモデリングツール「Rhinoceros 3D（ライノセラス・スリーディー）」を操作することができ、この場合は「スクリプト言語」と呼ばれ、ある特定の作業の「スクリプト」を書くための専用言語という扱いになります。

スクリプトを書くことを「スクリプティング」、保存されたファイルを「スクリプトファイル」と呼びます。



▶ 参考：

- **Blender 2.80 Python API Documentation**
  https://docs.blender.org/api/current/index.html

JavaScriptもアニメーションツールなどの専用言語として「スクリプトを書く」ための言語として利用されており、Pythonと同様に「スクリプト言語」と記されることがあります（例えば、Adobe社の製品の多くはJavaScriptで制御することができます）。

専門領域に特化している場合は「スクリプト言語」や「スクリプティング」、「スクリプトファイル」などの用語が使われますので、理解しておきましょう。

Step 03

なぜAIの分野で利用されているのか?

Python がなぜ AI の分野（特に機械学習と呼ばれる手法）で使われているのか、学習します。

Pythonと機械学習

近年、話題になっている「機械学習」は人工知能の中の1つの手法です。1980年代から続く研究分野ですが、特に注目されているのが2012年以降の「深層学習（ディープラーニング）」です。

機械学習がさまざまな業界で利用されるようになったのは、大量のデータを収集しやすくなり、高速なGPU（Graphics Processing Unit）環境が容易に利用できるようになったこと、そして機械学習のためのライブラリが充実してきたことなどが影響しています。

機械学習の領域では、Pythonが世界中で使われており、事実上の標準と言っても大げさではありません。Webアプリ（フロントエンド）やGUIなどはJavaScriptが適していますが、データ分析の分野はPythonが圧倒的です。

以下は、機械学習で使用されるPythonのライブラリです。

ライブラリ	読み用途	
NumPy	ナムパイ	数値計算や多言語配列、画像処理、音声処理など
SciPy	サイパイ	科学技術計算（信号処理や統計計算など）
Pandas	パンダス	表形式のデータ構造を操作（抽出や加工、集計など）
scikit-learn	サイキット・ラーン	機械学習に必要なアルゴリズムが実装されている定番ライブラリ
Matplotlib	マットプロットリブ	データのグラフ描画（統計量の可視化など）

学習目標のチェックシート：

Chapter 1の達成目標を確認して自己評価してください。理解できた／習得できた場合は評価「5」になります。この後の学習で必要な前提知識・前提スキルになりますので、理解できなかった／習得できていない学習項目は必ず復習をしてください。

達成目標	自己評価（5段階評価）
「プログラミング」や「プログラム」の言葉の意味について説明できる	5・4・3・2・1
プログラミング言語の歴史を大まかに理解することができる	5・4・3・2・1
Pythonがなぜ「書きやすい言語」と言われているのか理解できる	5・4・3・2・1
「スクリプト言語」や「スクリプティング」の言葉の違いについて説明できる	5・4・3・2・1
なぜAIの分野で利用されているのか説明できる	5・4・3・2・1

Chapter 1 のまとめ

- プログラミングとは人間がコンピューターを動かすために命令を書くことである
- 人間が書く命令文のことを「ソースコード」と呼び、ソースコードのまとまりが「プログラム」である
- プログラミング言語は、機械語から「低水準言語」、そして「高水準言語（高級言語）」へと進化してきた
- Pythonは「書きやすく、読みやすい」言語として知られている
- Pythonはインデント（字下げ）で処理の区切りを表す。誰が書いても同じような体裁の記述になるため、他の人が書いたプログラムを容易に理解できる
- 歴史はJavaよりも古く、開発が始まったのは1989年である
- Pythonの現在の人気は、ディープラーニング／機械学習の世界的なブームが追い風となった
- 「スクリプト言語」とは、ある特定の作業の「スクリプト」を書くための専用言語。例えば、3Dソフトの拡張機能を開発するための言語としてPythonが使われた場合、スクリプト言語と記される

Chapter

2

学ぶための準備と基本文法

Chapter 2 では、最初に Python の学習に必要な専門用語やプログラミング言語の種類（インタプリタ型とコンパイラ型）について学習します。次に、Python のプログラムを書くためのツールについて学び、Python を学習に使用するパソコンにインストールしてもらいます。実際に簡単なコードを記述しながら、基本構文や Python の作法について学びます。

本章で学ぶこと

▶ 達成目標：

1. プログラミングとコーディングの言葉の意味について理解できる
2. インタプリタ型とコンパイラ型の違いを理解できる
3. GUIとCUIの違いを理解できる
4. 最新のPythonをインストールすることができる
5. Python独特の書き方を理解して実践できる

▶ 学習ポイント：

- 曖昧な専門用語を明確にします
- インタプリタ型とコンパイラ型の働きや関係について解説します
- Pythonプログラミングで使うツールの種類と特徴について解説します
- Pythonのバージョンの違いやインストール方法を学べます
- Pythonのプログラミングでは特別な意味を持つインデントについて解説します

▶ 学習の流れ：

- Step-01 プログラミングの流れ
- Step-02 Pythonのインストール
- Step-03 プログラムの書き方について

Step 01

プログラミングの流れ

Pythonのプログラムを書くためのツール（シェルやIDLE、コードエディタ）について学習します。

専門用語について

プログラミングの流れを確認する前に、Pythonを使うときに出てくる用語について学習しておきましょう。

まずChapter 1の復習です。「プログラミング」とは人間がコンピューターを動かすために命令を書くこと、「プログラム」とは人間が書く命令文をまとめたものです。この命令文を「ソースコード」と呼び、コードを書くことを「コーディング」と呼びます。

Webアプリの設計や全体像について解説するときは「プログラム」、その中の処理を個別に解説するときは「ソースコード（もしくは単にコード）」と表現しますが、意味に大きな違いはないことを理解してください。

Chapter 1では「Pythonはスクリプト言語」であることも学習しました。Pythonのファイル（拡張子が「.py」）をスクリプトファイルと呼ぶこともあります。

スクリプト言語は、ある特定の作業（3Dツールやアニメーションツールなど）の「スクリプト」を書くための専用言語という扱いで使用されますが、JavaやCなどのプログラミング言語より敷居の低い言語として紹介されることもあります。

文脈の違いによって、コード、スクリプト、コーディング、スクリプティングなど表現が変わりますので注意しましょう。

- **プログラミング（プログラムを書くこと）**
- **プログラム（ソースコードのまとまり）**
- **ソースコード（命令文）**
- **コーディング（命令文を書くこと）**
- **スクリプト（スクリプト言語のコード）**
- **スクリプティング（スクリプトを書くこと）**

▶ インタプリタ型とコンパイラ型

プログラミング言語には「インタプリタ型」と「コンパイラ型」があります。PythonやJavaScript、Ruby、PHPなどはインタプリタ型のプログラミング言語です。

インタプリタ型は機械語に翻訳しながら使うことができますので、1行ごとコードを逐次実行してすぐに結果を確認でき、何度も試行錯誤しながらプログラミングを進めていくことが可能です。初心者にとっては学びやすい言語だと言えるでしょう。

一方、コンパイル型はソースコードを機械語に一括変換（コンパイル）してから実行します。1つでもエラーが発生すると実行できませんので、プログラムを見直す必要があります。また、コンパイルされたコードは機械語に変換されていますので、人間は読めません。

これからプログラミングを学ぶ人にとっては取っ付きにくいかもしれませんが、インタプリタ型より高速に処理されますので、実行時の処理速度が要求されるプログラムに適しています。

JavaやCなどが代表的なコンパイル型の言語です。

比較項目	インタプリタ型	コンパイル型
処理方法	逐次変換	一括変換
実行速度	あまり速くはない	高速
配布されるコード	読める	読めない

Pythonのインストールについては Step-02 で詳しく解説しますので、ここではプログラミングの全体像を把握してください。

Pythonのプログラムを書くツール

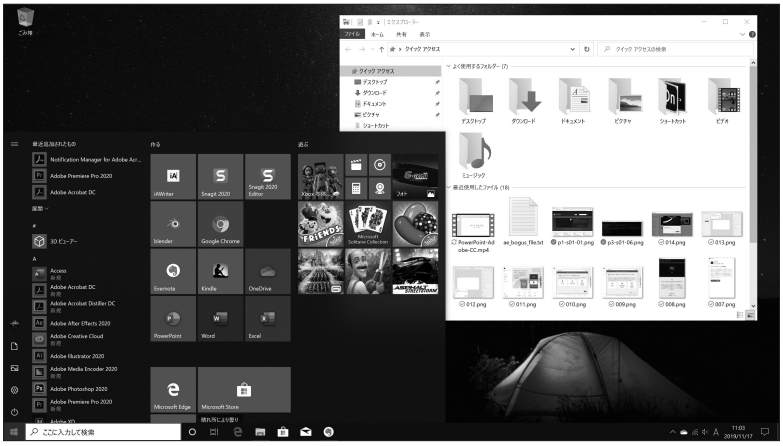
私たちが使っているWindowsやMacなどのパソコンは、GUI（Graphical User Interface：グラフィカルユーザーインターフェース）が採用されています。マウスやキーボードを使って、デスクトップ上のアイコンやメニューなどをクリックしたり、ドラッグしたりしながら直感的に操作しています。

▶ GUI（グラフィカルユーザーインターフェース）

WindowsやmacOSなどのOS（Operating System：オペレーティングシステム）は、机の上をモデルにしたデスクトップメタファを採用しています。

※OSとは利用者がコンピューターを動かすための基本ソフトウェアのことです。

※メタファは隠喩（いんゆ）を表しています。デスクトップメタファとは、机の上にノートやペンなどの道具が置かれている状況を、パソコンの画面上で表現するデザイン概念のことです。



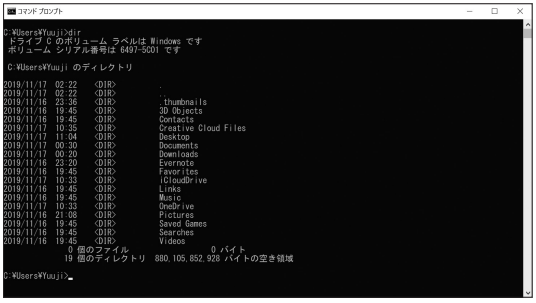
GUI以外のコンピューター操作方法としてCUI（Character User Interface：キャラクタユーザーインターフェース）があります。名前のとおり、コンピューターの画面に文字を入力して操作を行います。

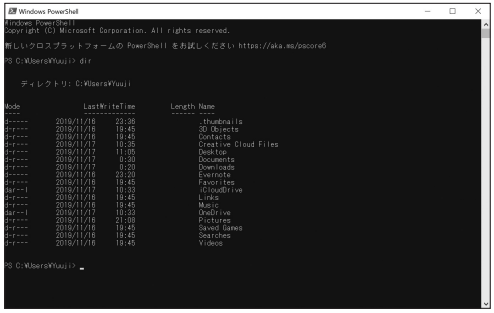
GUIが普及する前（1980年代）は「MS-DOS」などのCUIが主流でした。世界的なヒット製品となった「Windows 95」がリリースされたのは1995年8月です（日本版のリリースは11月）。

※MS-DOSは、Microsoftが開発したOSです。1981年に発売されました。コマンドラインインターフェースとも呼ばれ、黒い画面にコマンドを入力してコンピューターを操作するシステムです。

▶ CUI（キャラクタユーザーインターフェース）

CUIはGUIのように、直感的に操作することはできません。オペレーティングシステムの階層構造の仕組みや入力するコマンドの知識が必要となります。





▶ シェル（shell）

Pythonのプログラミングでは、OSに標準搭載されているシェル（shell）と呼ばれるアプリケーションソフトを使います。シェルは文字を入力しながらコンピューターを操作するCUIツールです。

一般のパソコン利用者が使い慣れているGUIとは異なり、コマンド（命令）を打ち込んで操作を実行させなければいけませんのである程度の練習が必要になります。

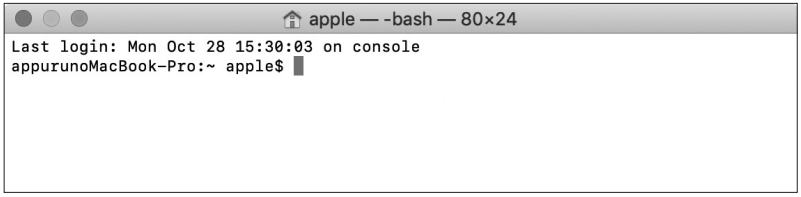
プログラミングの学習を始める前に、まずはCUIの操作に慣れましょう。

Windowsは「コマンドプロンプト」、macOSは「ターミナル」を使います。本書では基本的にWindowsのコマンドプロンプトを使って解説していきますが、使い方に大きな違いはありません。

操作が異なる場合のみ補足していきます。

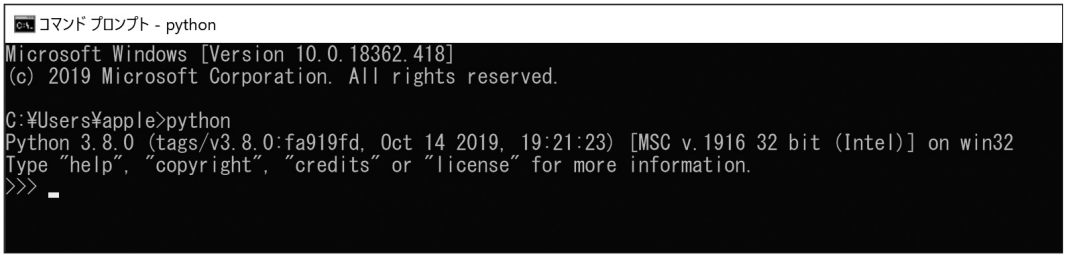
- **コマンドプロンプト（Windows）**

- **ターミナル（macOS）**



Pythonでプログラミングを始めるときは「python」と入力してEnterキーを押します。Pythonのバージョンが表示されていれば、準備完了です。

この状態を「対話モード」と呼びます。



macOSの場合は「python3」と入力する必要があります。「python」と入力してしまうと、標準インストールされているPython 2.7.xの方が起動してしまいますので注意しましょう。

```
apple — Python — 80×24
Last login: Mon Oct 28 15:30:03 on console
appurunoMacBook-Pro:~ apple$ python3
Python 3.8.0 (v3.8.0:fa919fdf25, Oct 14 2019, 10:23:27)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Pythonは、インタプリタ型のプログラミング言語なので、1行ごとにプログラムを実行しながらプログラムを作成していくことができます。入力方法はコマンドプロンプトとターミナルで大きな違いはありません。

対話モードを終了する場合は「exit()」と入力してEnterキーを押します（「quit()」でもかまいません）。

```
コマンド プロンプト - python
Microsoft Windows [Version 10.0.18362.418]
(c) 2019 Microsoft Corporation. All rights reserved.

C:¥Users¥apple>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print('おはようございます！')
おはようございます！
>>>
```

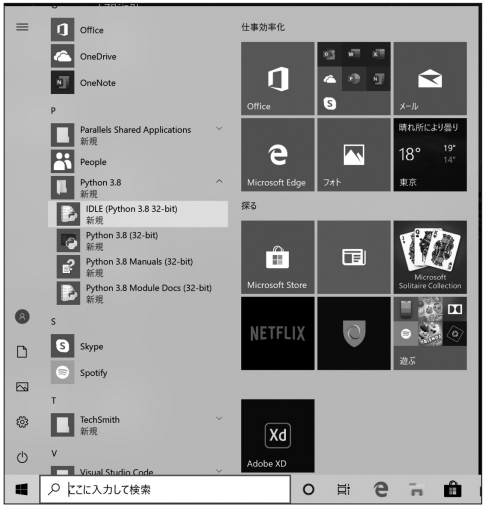
Windowsには「Windows PowerShell」というシェルもインストールされています。PowerShellは、Windows 7から標準搭載されている開発者向けのシェルです。すでにこちらのツールを使いこなしている人は、コマンドプロンプトを使う必要はありません。

```
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

新しいクロスプラットフォームの PowerShell をお試しください https://aka.ms/pscore6

PS C:¥Users¥apple> python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print('おはようございます！')
おはようございます！
>>> exit()
PS C:¥Users¥apple>
```

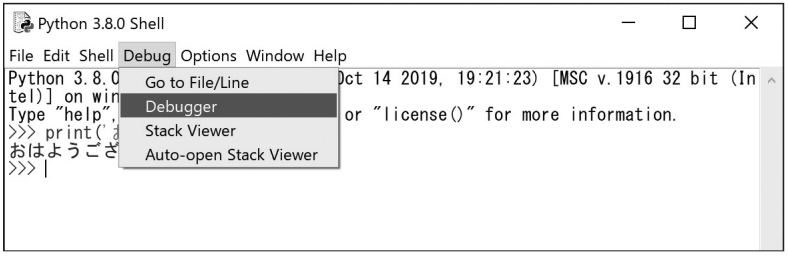
▶ IDLE

Pythonには「IDLE（アイドル）」という開発ツールが付属しています。IDLEはPythonで作られたアプリケーションソフトです。

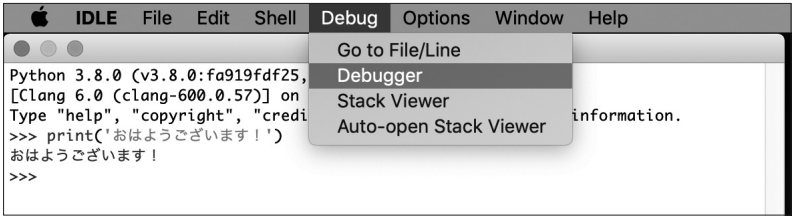
起動すると、シェルウィンドウが表示され、コマンドプロンプトの対話モードと同じように使うことができます。プログラムを保存したり、後で呼び出して修正したりなど、ファイルを扱うときはIDLEを使います。コマンドプロンプトでプログラムを保存することはできません。



FileメニューからNew Fileを選ぶと、エディタウィンドウが表示され、プログラムを書くだけでなくPythonファイルを保存することができます。プログラムを実行した結果はシェルウィンドウに表示されますので、2つのウィンドウを行き来することになります。



macOSの場合は、アプリケーションのフォルダーの中にありますので、ダブルクリックして起動します（Spotlight検索で「IDLE」と入力して起動することもできます）。

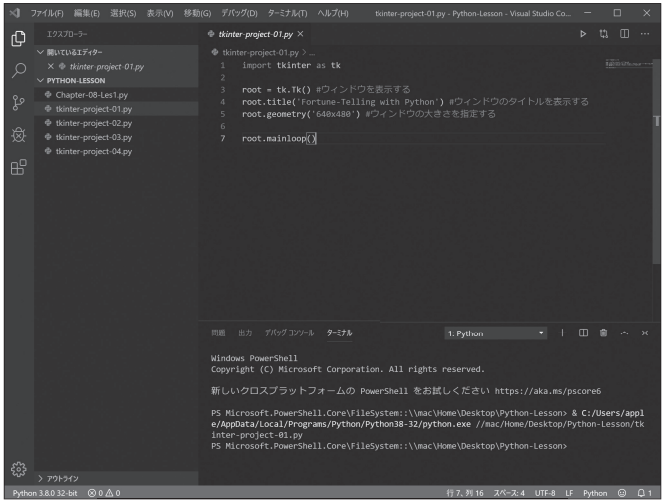


本書では、コマンドプロンプト（ターミナル）をメインで使用していきますが、ファイルの保存／読み込み／修正などが必要な作業ではIDLEを使います。

▶ Visual Studio Code

Chapter 8からは複数のライブラリを活用した本格的なプログラミング作業を行いますので「Visual Studio Code」というMicrosoftが無償で提供しているコードエディタを使います。

Visual Studio Codeには、Pythonを使うための優れた機能拡張が用意されていますので、効率的なプログラミングが可能になっています。



Step 02

Pythonのインストール

Python のバージョン（2.x、3.x）、およびインストールの準備と実行方法について学習します。

公式サイトからPythonの最新バージョンをインストールする

Pythonの公式サイトで配布されているインストーラーをダウンロードして、最新バージョンのPythonを学習に使用するパソコンにインストールします。

本書は、Windiows 10の画面で手順を解説していきますが、macOSについても補足解説します。

Pythonには2種類のバージョンがあります。「2.x」と「3.x」です。

バージョンごとのリリースノートを見ると「2.x」と「3.x」が混在しています。互換性レベルが低いため「2.x」で作成したプログラムは「3.x」の環境ではうまく動きません。企業の業務システムなどでは「3.x」への書き換えにコストがかかるため、現在でも「2.x」が使われています。

Pythonのバージョン	リリース日
Python 3.9.0	2020年10月5日
Python 3.8.6	2020年9月24日
Python 3.5.10	2020年9月5日
Python 3.7.9	2020年8月17日
Python 3.6.12	2020年8月17日
Python 3.8.5	2020年7月20日
Python 3.8.4	2020年7月13日
Python 3.7.8	2020年6月27日
Python 3.6.11	2020年6月27日

「2.x」は2020年1月1日にサポートが終了しましたので、各種ライブラリも対応しなくなる可能性が高いため「3.x」への移行が推奨されています。

※バージョン3.8.5の後に3.6.12がリリースされていますが、これはバグ修正リリースです。