Python ライブラリの使い方 第2版

GUIから機械学習プログラミングまで

松田晃一◉著

CUTT カットシステム

■**サンプルファイルのダウンロードについて**

本書掲載のサンプルファイルは、一部を除いてインターネット上のダウンロードサービスからダウンロードすることができます。詳しい手順については、本書の巻末にある袋とじの内容をご覧ください。

なお、ダウンロードサービスのご利用にはユーザー登録と袋とじ内に記されている番号が必要です。そのため、本書を中古書店から購入されたり、他者から貸与、譲渡された場合にはサービスをご利用いただけないことがあります。あらかじめご承知おきください。

・本書の内容についてのご意見、ご質問は、お名前、ご連絡先を明記のうえ、小社出版部宛文書（郵送またはE-mail）でお送りください。
・電話によるお問い合わせはお受けできません。
・本書の解説範囲を越える内容のご質問や、本書の内容と無関係なご質問にはお答えできません。
・匿名のフリーメールアドレスからのお問い合わせには返信しかねます。

前書き

Google の機械学習（人工知能）で使用されているプログラミング言語として有名になった Python は、その簡潔なコーディングや高い機能性などから、今日、最も注目されている言語の 1 つであり、初心者向きとも言われています。

「初心者向き」という形容は、個人的には疑問に思う点もありますが、近年プログラミング言語の善し悪しは、言語そのものの仕様ではなく、言語に関する Web ページなどの情報量、その言語から使用できるライブラリの豊富さ、それを支えるフォーラムなどコミュニティを含めた「エコシステム」（生態系）の方が重要になってきています。Python は、前述の機械学習以外にも、画像処理、Web スクレイピングや、自然言語処理、科学技術計算、ロボットのアプリケーション開発（ROS）※ 1、IoT や組み込みシステムなどさまざまな分野で利用されてきており、それを支えるライブラリやコミュニティの豊富さなどからさまざまなシステムやアプリケーションが開発できるようになってきています。また、既存のソフトウェアへの組み込みとしては、統合 3DCG ソフト（Maya、Blender、LightWave 等）でデファクトスタンダードのスクリプト言語にもなっています。

本書ではそのような Python のライブラリの中から比較的手軽に利用でき、知っておくと便利そうなものを選び、サンプルプログラムとともに説明したものです。サンプルプログラムは 200 個以上あり、ダウンロードサービスからすべて取得できます。全部で 12 種類のライブラリを扱っており、第 1 章はそのライブラリを用いたサンプルプログラムを読むのに必要な Python の知識を概説しています。それ以降は各ライブラリの説明になりますが、それぞれのライブラリをなるべく等しく扱うために各章を 20 ～ 40 ページとしてあります。基本的にはどの章から読まれても大丈夫だと思いますが、第 10 ～ 13 章は互いに関連するため続けて読まれることをお勧めします。また、本書を大学の講義などで使う場合は長い章を 2 回に分けるなどすると 14 ～ 15 回とすることができると思いますし、いくつかの章を飛ばしてもよいでしょう※ 2。

※ 1　Pepper の開発環境でも Python が使えます。

※ 2　参考までに、筆者が話をするときは、自然言語処理とデータベースが人気がありません。

本書の執筆にあたっては、弓林司君（4、10、12、13 章）、竹畑柚木さん（5 章、6 章）の協力を得ることができました。また、草稿のチェックはこれまでも大変お世話になっている小沼千絵さん（NEC）、山本美咲季さんにお願いできました。この場を借りて感謝します。編集はカットシステムの武井智裕さんのお世話になりました。二つ返事で本書の執筆の機会を与えていただけたカットシステムの石塚勝敏さんに感謝するとともに遅筆をお詫びします。

令和 4 年 11 月吉日
自由が丘にでけりますと
松田晃一

0.1 新しそうで古い言語 Python

Python は近年話題になることが多いことから新しい言語だと思いがちですが、実は 1991 年に登場した古いプログラミング言語です。Java が登場したのが 1995 年なのでそれよりも 4 年も古いことになります。当時のスクリプト言語としては Tcl/Tk があり、筆者もその頃、開発していたシステムのスクリプト言語を決める際に、Tcl/Tk か Python で迷った記憶があります。当時は Python がまだ不安定だったせいもあり、前者を採用しました。このように歴史の長い言語であるため、そこここで古さを感じることがあります※3。

Python の特徴としては、波括弧を排除し言語仕様で字下げが必須であるため自然とプログラムが読みやすくなる、他の言語に比べて同じ機能のプログラムが少ない行数で書けるという点があります。ライブラリという本書の観点からはインストールが簡単というのも大きな特徴でしょう（特に OpenCV を勉強される方は Python をお勧めします）。その反面、他の言語を経験された方には逆に自由度が少ないと感じられることもあるかもしれませんし、クラスなどは分かりにくいと思われるかもしれません。初心者向きとも言われていますが、これは人によるでしょう。以下に 1 から 10 までの整数の和を求めるプログラムを Python と C 言語で示します。

```
sum = 0
for i in range(1, 11):
    sum = sum + i
print (sum)
```

```
int main(){
    int sum = 0;
    for (int i = 1; i < 11; i++){
        sum = sum + i;
    }
    printf ("%d¥n", sum);
    return 0;
}
```

Python は C 言語の半分くらいの行数であることが分かります。また、C 言語をご存じの方は、（これまでのプログラミング言語とは）かなり違うという印象をもたれたかもしれません。

※3　Tcl/Tk の Tk が Python の tkinter で利用できることを知ったときは当時を思い出しました。

0.2 Python のインストールとプログラムの実行

Python のインストール方法と、インストールした Python を使ってプログラムを作成して動作させる方法を一通り確認しましょう。

Python は以下に示す Python のホームページで公開されています。この他にも Anaconda などの Python と他のライブラリをセットにしたディストリビューションもありますが、本書ではどのような環境でも使用できるように最もプリミティブな環境である、本家の「すっぴん」の Python を用います。

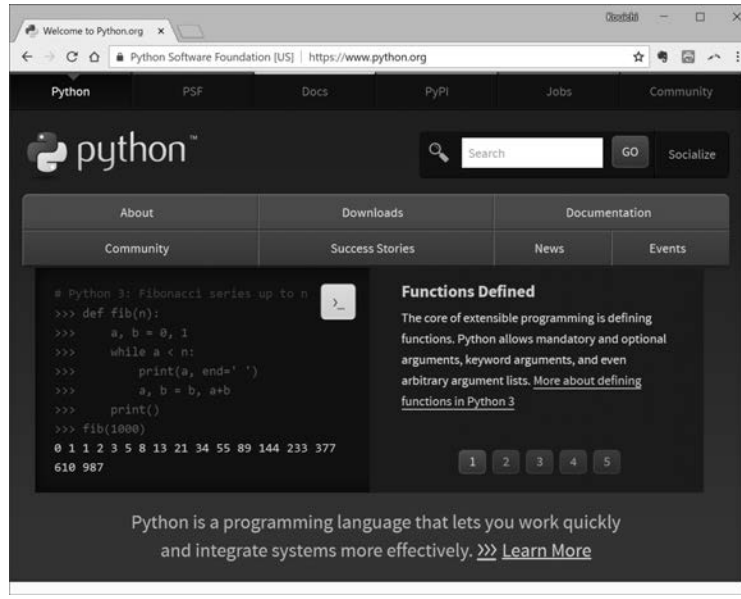

図0.1●Pythonのホームページ（http://www.python.org）

ここで画面真ん中上の Downloads をクリックすると以下のページが表示されます。

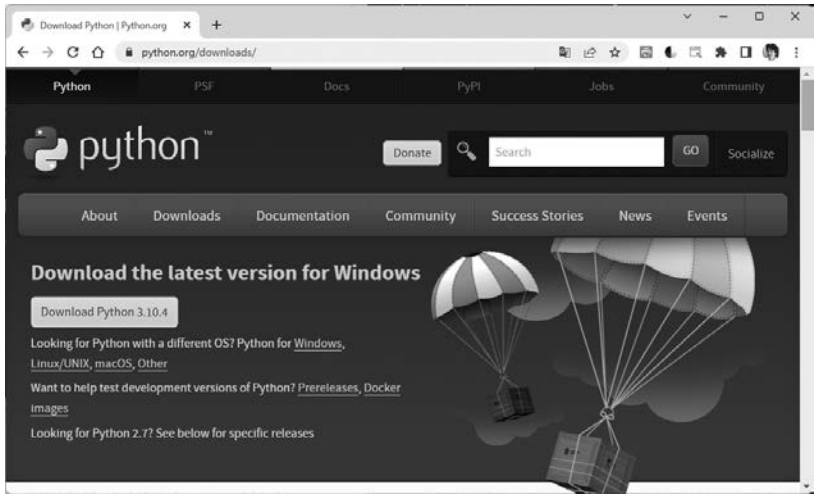

図0.2●Downloadsページ

ここで「Downloads Python 3.10.x」をダウンロードして、ダウンロードしたファイルをダブルクリックして実行してください※4。Windowsの場合は、次のようなウィンドウが表示されます。

図0.3●インストーラの表示するウィンドウ

ここで下の「Add Python 3.10 to PATH」のチェックボックスをクリックしてから、上の方にある「Install Now」をクリックしてください。デバイスの変更の許可などは「はい」を選

※4　2022年11月現在、第13章のtensorflow以外は3.11に対応されており、サンプルプログラムも3.11で動きます。

択してください。Mac の場合も同様のウィンドウが表示されますが、PATH は自動的に設定されるので、そのまま「続ける」をクリックしてください。最後に「インストール」をクリックします。これでインストールが始まります。次のウィンドウが表示されればインストール完了です。

図0.4●インストールの完了を示すウィンドウ

これで Python を使用する準備が整いました。それではさっそくインストールした Python を使ってみましょう。Python のプログラムの実行方法は、(1) インタラクティブシェルで実行する方法と、(2) ファイルにプログラムを書きそれを実行する方法の 2 つの方法があります。

0.3 インタラクティブシェル

「インタラクティブシェル」という言葉はあまり聞き慣れない言葉だと思います。シェルという言葉には太古の UNIX の時代から「ユーザからのコマンド入力を受け付け、解釈するプログラム」という特殊な意味があります。つまり、インタラクティブシェルとは「対話型（インタラクティブ）で、ユーザからのコマンドを受け付け、解釈するプログラム」という意味となります。

Python プログラムにおけるインタラクティブシェルとは、「対話型、すなわち、コンピュータ（この場合は、Python）と向かい合って話し合う形式で、ユーザからの Python のプログラ

ムやコマンドを受け付け、解釈するプログラム」のこととなります。では、インタラクティブシェルを実行して Python を使ってみましょう。

0.4 インタラクティブシェルでの実行

インタラクティブシェルの実行方法には 2 つの方法があります。1 つはコマンドプロンプトやターミナルで「python」と入力して起動する方法、もう 1 つは Windows に Python をインストールすると一緒にインストールされる python 3.10 というアプリケーションを利用する方法です。以下に、Windows のコマンドプロンプトで「python」と入力した場合のウィンドウを示します。実行するとバージョンが表示され入力待ちになります。Mac ではターミナルで同様に行います。Mac の場合は Python 2 が最初からインストールされているので、それが起動してしまう場合は **python3** と入力してください。本書では、「python」と表記します。

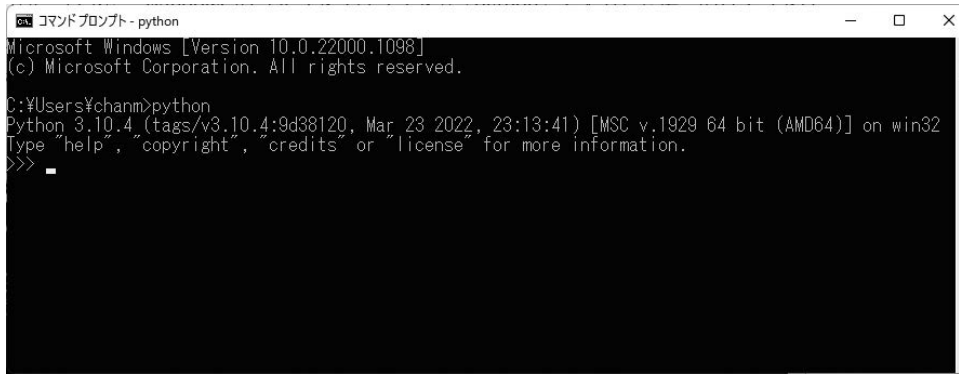

図0.5●コマンドプロンプトでpythonを実行した様子

これに対し、Windows の Python 3.10 というアプリケーションでは、スタートメニューで表示される以下のようなアイコンを選択することで起動できます。こちらの場合は、先ほどと異なり「python」と入力しなくても、すぐに Python のコマンドを試してみることができます。

図0.6●Python 3.7アプリケーション

Python 3.10 を起動した後の画面を以下に示します。

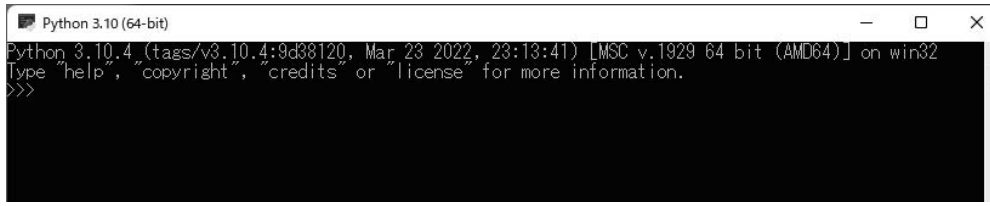

図0.7●Python 3.10アプリケーションの起動画面

どちらの方法でも画面上に「>>>」という記号が表示されます。ここで表示される「>>>」は Python がコマンドの入力を待っていることを示す表示です。これは「コマンドプロンプト」と言いますが、この名称は Windows のコマンドプロンプトと名前が同じなので紛らわしいかもしれません。

インタラクティブシェルを終了するには「quit()」と入力して Enter キーを押してください。インタラクティブシェルで Python のプログラムを入力して実行することができます。例えば、簡単な数式を入力して Enter キーを押すと、Python がその式を計算して答えを表示してくれます。以下のように入力してみてください。

```
>>> 4 + 6
10
>>> 4 * 6
24
>>>
```

ここで２つ目の「*」は掛け算を表します（第 1 章参照）。数式を入力すると Python が受け取り計算結果を表示してくれます。それぞれ入力し Enter キーを押し、結果が表示されると、再度「>>>」が表示されることが分かります。

実際には、Python は計算以外の処理も行い、その結果も表示するので、「計算」ではなく「評価」という言葉が使われます。先ほどの例は、数式を「評価」している、と言うことになります。つまり、数式を評価することは、数式を計算することを意味します。

このようにインタラクティブシェルは、入力をすぐに Python がチェックし評価してくれるので、何かを試してみるのには非常に便利ですが、複雑なプログラムを入力するのには大変です。このような場合に、プログラムをエディタで入力し、それをファイルに保存、Python に

実行させる方法があります。

0.5 プログラムの作成、保存、実行

Python のプログラムを作成するのは簡単です。みなさんが普段使っているテキストエディタでテキストファイルを作成し、そこに Python に実行してほしい内容を書いておくだけです。テキストエディタは何でも構いません。

ここでは Python をインストールするとついてくる IDLE（Integrated Development Environment）を使って説明します。これは、さまざまな OS で使用することができ、プログラム入力支援機能（キーワードをハイライトする機能や、自動字下げ機能など）、エディタからプログラムを実行する機能などがサポートされています。

図0.8●IDLE

IDLE は Windows ではスタートメニューから実行することができます。Mac の場合は、SpotLight で「IDLE」を検索してみてください。実行すると次のようなウィンドウが表示されます。

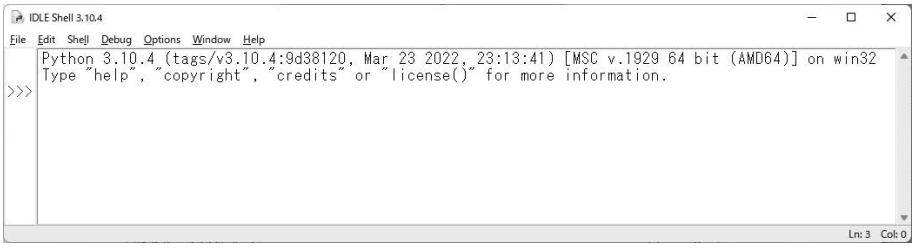

図0.9●IDLEの画面

「>>>」が表示されていることから分かるように、これは先ほど説明したインタラクティブシェルです。ここにコマンドを入力すると、前の節の説明と同様に実行できます。

IDLE を使って新しく Python のプログラムを書くには、［File］メニューから［New File］を選んでください。次のようなウィンドウが表示されます。

図0.10●IDLEのエディタ画面

これが Python のプログラムを入力するエディタになります。ここに次のように入力してみましょう。

図0.11●IDLEにプログラムを入力している様子

print という文字列を入力し終わると文字列が紫色に変わることが分かります。これは「print」が Python で使われる特別なキーワードであることを示しています。例えば、これを間違えて、「printo」と入力すると黒字のままになります。このような機能をキーワードのハイライト機能と言います。

入力が終わったら保存します。その前に分かりやすいように Windows の場合は C: ドライブ（Mac の場合はホームディレクトリなど）に python3 というフォルダを作っておいてください。［File］メニューから［Save］を選択してください。次のようなウィンドウが表示されるので、C ドライブの python3 フォルダに移動し、test という名前をつけて保存してください。

ここでは表示されていませんが、このファイルの拡張子は「py」となります。Python では拡張子に py を用いるのが一般的です。

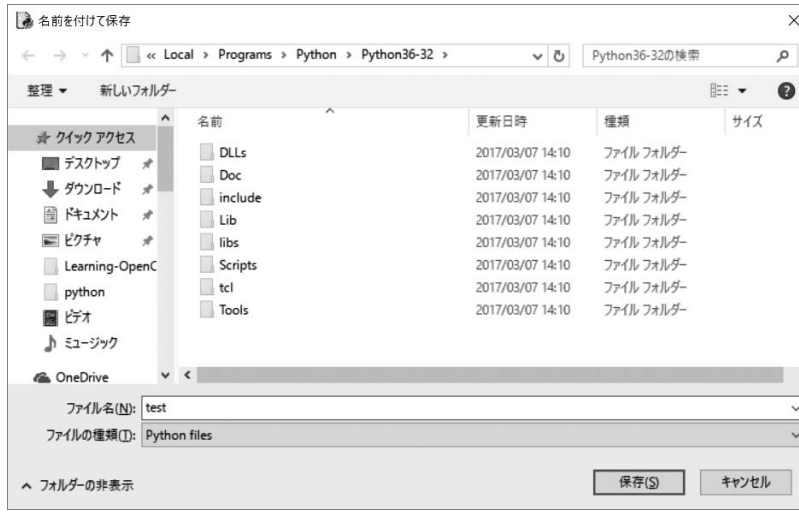

図0.12●IDLEのファイルダイアログボックス

また、既に保存してあるプログラムを編集するには、［File］メニューから［Open...］を選択して読み込んでください。ファイルの作成や保存、再度開く場合の操作は他のアプリケーションと同じです。

これでプログラムの保存ができたので、さっそく実行してみましょう。test.py を作成した画面上の［Run］メニューから［Run Module］を選択してください。これで今保存したプログラムを実行することができます。以下に結果を示します。

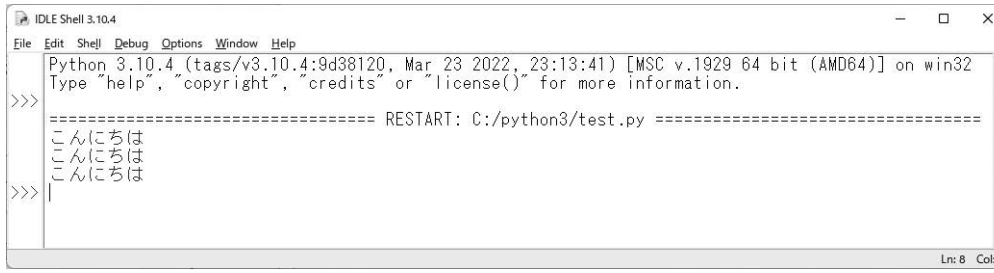

図0.13●IDLEによるプログラムの実行とその結果

先ほど書いた 3 つの print(' こんにちは ') が実行され、「こんにちは」が 3 回表示されている

ことが分かります。

IDLE を終了する場合は Windows では［File］メニューから［Exit］、Mac では［IDLE］メニューから［Quit IDLE］を選択してください。ウィンドウの「×」を押して終了しても構いません。

このようにして作成したプログラムは前の節で説明した Windows のコマンドプロンプトや Mac のターミナルでも実行できます。コマンドプロンプトやターミナルを立ち上げて、test.py が置かれているフォルダに移動します。これには cd コマンドを用います。Windows では C:/python3 においたのでそこに移動します。

図0.14●コマンドプロンプトからのプログラムの実行（フォルダの移動）

移動した後は、python コマンドの引数に先ほど作成した test.py を指定し Enter キーを押すと実行することができます。

図0.15●コマンドプロンプトからのプログラムの実行

本書では、内容に応じて Python のインタラクティブシェルでの実行とファイルでの実行を切り替えて用います。インタラクティブシェルでの実行は

```
>>>
```

を用いて表し、ファイルでの実行は

```
> python プログラム名
```

という形で表記してあります。

なお、本書のダウンロードサービスで提供されるサンプルプログラムにはすべて 1 行目にコメントが入っているので、本文中の行番号とは 1 行ずれるので注意してください。

最後に、本書で使用しているライブラリの一覧を載せておきます。

- beautifulsoup4
- Django
- Janome
- Keras
- matplotlib
- numpy
- opencv-contrib-python
- opencv-python
- requests
- scikit-learn
- tensorflow

0.6 まとめ

ここまで、Python の紹介に加えて、Python のインストール方法、インストールした Python を使ってプログラムを作成し実行する方法について説明しました。Python にはキーボードでプログラムを入力し、1 行 1 行実行する方法（インタラクティブシェル）と、ファイ

ルに入力してから実行する方法の 2 つがあります。インタラクティブシェルでは、プログラムを入力するとすぐに結果を返してくれるので、最初に勉強する場合や、何かをちょっと確認したい場合に便利でしょう。ファイルに入力する方法では、テキストエディタであればどれでも使用できますが、ここでは IDLE を紹介しました。この方法では、長めのプログラムを書いたり、少しずつ変更しながら何度も実行して結果を確認したい場合に役に立ちます。

目次

第1章

Python 概説

本章では、本書を読む上で必要な Python の基本的な構文の説明をします。すでに Python をご存じの方は読み飛ばされても構いません。基本的には以降の章を読むのに必要な、共通な内容を説明し、特定の章にしか出てこないものはその場で説明することにします。

1.1 変数とデータ

Python で「1+1」を計算させてみます。以下の説明では Python のインタラクティブシェルが起動されており、コマンドプロンプト(>>>)が表示されているものとします。>>> の後に、1+1 を入力して、Enter キーを押すと次のように表示されます。

```
>>> 1+1
2
```

ご覧のように Python が + を解釈して足し算を行い、その結果を表示しています。また、このような計算は小数同士でも行うことができます。

```
>>> 3.14+1.1
4.24
```

Python は 1 や 100、123 のような整数、3.14 や 1.1 のような小数で計算を行えます。このようなデータの種類のことを「データ型」と言います。これから、整数型と小数型（浮動小数点数型）の 2 つがあることが分かります。整数や小数以外にも Python にはさまざまなデータ型がありますが、このようなデータ型は type 関数※1 で調べることができます。

```
>>> type(4.24)
<class 'float'>
```

これより 4.24 が float（浮動小数点数型）であることが分かります※2。

1.1.1 算術演算子

Python での計算は、演算子を用いて行います。使用できる算術演算子を以下に示します。

表1.1●算術演算子

演算子	意味	例	説明
+	加算	x + y	x に y を加える
-	減算	x - y	x から y を引く
*	乗算	x * y	x と y を掛ける
/	除算	x / y	x を y で割る
//	切り捨て除算	x//y	x を y で割り、小数点以下を切り捨てる
%	剰余	x % y	x を y で割った場合の余り
**	冪乗	x**y	x の y 乗

数学などと異なるものがあります。乗算ですが、×という記号がキーボードにはないのでアスタリスク（*）を代わりに使用します。また、割り算の除算記号である÷もキーボードにはないのでスラッシュ（/）を使用します。その次の剰余（%）は余りを計算する演算子です。

※1　関数に関しては、1.10 節参照。

※2　後で説明するように Python は明示的にデータ型を指定しない動的型付け言語です。明示的に型を書かないため、型が分からなくなる場合があります。そのような場合に便利です。

冪乗は、3^2 のような計算を行うときに用います。使用例を以下に示します。

```
>>> 8 + 5
13
>>> 8 / 5
1.6
>>> 8 % 5
3
>>> 10 // 3
3
>>> 2**10
1024
```

「8%5」は、8 を 5 で割ると商は 1 なので、8 – 5 × 1 = 3 となり 3 が表示されます。「2**10」は 1024 です。2 × 2 ×…× 2 という感じで 2 を 10 回乗算したものです。これらの演算子は小数に対しても使用できます。

1.1.2 代入演算子と変数

ここまでの演算子を使うことでいろいろな計算ができます。例えば、運動時に消費されるカロリー（kcal）は次の式で計算することができます※3。

メッツ × 体重 (Kg) × 運動時間 × 1.05

メッツ（METs）値は「Metabolic equivalents」の略で、安静時は 1.0 メッツ、散歩時は 2.5、エアロビクスが 6.5 というように値が決まっており、体重 60 kg の人が 1 時間散歩したときの消費カロリーは次のように計算できます。

```
>>> 2.5 * 60 * 1 * 1.05
157.5
```

この結果から消費カロリーが 157.5kcal であることが分かりました。これがショートケーキ何個分かを計算してみましょう。ショートケーキは 8 等分したものが 366 kcal だそうです。

※3　国立健康栄養研究所 https://www.nibiohn.go.jp/eiken/programs/program_kenko.html

```
>>> 157.5 / 366
0.430327868852459
```

約 0.43 個、つまり、ショートケーキ半分未満なことが分かります。このように何かを計算するときは、ある計算を行い、その結果を用いて別の計算を行うことがよくあります。このような時に便利なのが「変数」です。変数はデータを取っておくための入れ物です（図 1.1 参照）。

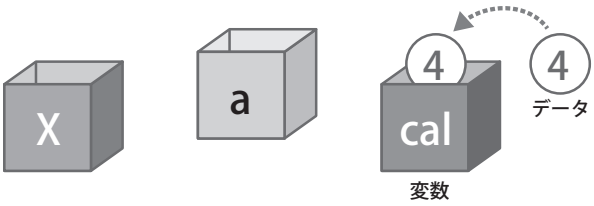

図1.1●変数はデータの入れ物

入れ物なので、入れ物（変数）にデータを入れる機能が必要になります。これが、代入演算子のイコール（=）です。cal という変数を用意して、そこに計算結果を代入してみます※ 4。

```
>>> cal = 2.5 * 60 * 1 * 1.05
>>>
```

これで cal という変数にカロリーの計算結果（157.5）が代入されました。変数に代入するだけなので何も表示されません。代入されたものは、cal と Enter キーで表示されます。

```
>>> cal
157.5
```

また print 関数を用いることでも表示することができます（関数に関しては 1.10 節参照）。print 関数は表示したいものを ( ) の中に書くことで表示させることができます。

※ 4　他の言語をご存じの方への注意 : Python では、変数は宣言する必要はありませんが、逆に言うと「変数を宣言する」ことだけを行うことはできません。必ず代入が伴います。また、型を書く必要もありません。型は変数にデータが代入された時点で決まります（動的型付け言語）。

```
>>> print(cal)
157.5
```

cal に代入された値を使用する場合は、変数名をそのまま使い、次のようにします。

```
>>> cal / 366
0.430327868852459
```

先ほど計算したのと同じ値が表示されました。

1.2 変数

前節では、cal という変数に 157.5 という数値を代入し、簡単な使い方を見てきました。変数の内容は新しい値が代入されるまで変わりませんが、「変数」という言葉の「変」が表すようにいつでも内容を変更できます。ここでは変数名の付け方、使い方を説明します。

1.2.1 変数名の付け方

変数名には、x、y、z、width、red など好きなものが使用できますが、以下のような命名規則があります。

(1) 使える文字は、半角英数字（a ～ z、A ～ Z、0 ～ 9）、下線文字（_）、ドル記号（$）、全角文字です。
(2) 最初の文字に数字は使えません。例えば、100niku はエラーになります。
(3) Python で使用する用途が決まっている予約語は変数に使えません。Python では次のように入力するとどのような予約語があるか分かります。

```
>>> import keyword
>>> keyword.kwlist
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class',
'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from',
'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or',
'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

なお、予約語が変数名の一部なっていても構いません。例えば、変数名として if は使えませんが、gif は（if という文字列が含まれていますが）使えます。

1.2.2 変数の使い方

まず前の節で説明した変数の使い方を再度見てみましょう。以下のようなものでした。

```
>>> cal = 2.5 * 60 * 1 * 1.05
>>> cal / 366
0.430327868852459
```

このカロリーがご飯何杯分かを知りたくなったとしましょう。ご飯のカロリーは 100 g で 168 kcal だそうです。これは、次のように計算できます。

```
>>> cal / 168
0.9375
```

このように変数の値は新しい値を代入するまで保持されます。では、別の値を代入してみましょう。2 時間散歩したときのカロリーです。

```
>>> cal
157.5
>>> cal = 2.5 * 60 * 2 * 1.05
>>> cal
315.0
```

変数の内容が変わっています。このような変数には、次に示すように計算結果と同様にデータそのものも代入できます。

```
>>> x = 8
>>> y = 5
>>> x + y
13
```

ここで注意が必要なのはデータが代入されていない変数を使うとエラーになることです。

```
>>> x = 5
>>> x + z
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'z' is not defined
```

Python は、変数が使われるとその中に入っているものを取り出します。何も入っていないと取り出せなくエラーになります。

最後に、コンピュータ独特な変数の使い方を見てみましょう。

```
>>> x = 8
>>> x = x + 1
>>> x
9
```

2 行目の式は数学的には変ですが、最初に代入演算子の右側の x + 1 が計算され、このとき x が 8 なので 8 + 1 で 9 となります。その結果が、代入演算子の左側の x に代入されます。実世界でも、預金の計算などで似たような計算をします。例えば、1000 円の預金をしていて、年に利子が 5 % つくとします。1 年後の預金額は次のようになります。

```
>>> 1000 * 1 + 0.05
1050.0
```

次の年の利子は、元金と利子を合わせたものを元金にして付くとすると、次の年には以下のような金額になります。

```
>>> 1050.0 * 1 + 0.05
1102.5
```

このような計算は、代入演算子の左辺と右辺で同じ変数を使うと簡単に書けます。

```
>>> x = 1000
>>> x = x * (1 + 0.05)
>>> x = x * (1 + 0.05)
>>> x
1102.5
```

分かりにくい場合は x の値がどのようになっているかを print 関数などで確認してみてください。このように変数に算術演算を行った結果をもとの変数に代入する場合には次のような略記ができます。

```
>>> x *= (1 + 0.05)
```

これは他の演算子も同じです。

```
>>> x += 10 # x = x + 10 と同じ
>>> x -= 10 # x = x - 10 と同じ
```

ここで「#」はコメントを表します。Python では、# が書かれていると、その記号も含め行末までを無視します（つまり、# 以降に書かれたものは何も書かれていないのと同じ扱いになります）。