Pythonで学ぶやさしい数学

Pythonプログラミングに
役立つ数学の基礎が
わかる!!

日向俊二◉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、下記 URL からダウンロードできます。

https://----------

- 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。

はじめに

コンピューターは数を扱うマシンです。文字や画像・映像でさえも、数値として扱います。そのため、コンピューターのプログラミングは、数と関連が深いだけでなく、数学の概念をさまざまなところで利用しています。

本書では、Python というプログラミング言語を使って数学に関連するプログラミングについてやさしく解説します。主な説明の後には、やさしく解ける問題と、その解答と解説を示すので、知識を確実に身に着けることができます。

また、本書では、単に問題を解決する方法を示すだけでなく、数式の意味や実社会との関係などについても可能な限り具体的に説明することにも焦点を当てるので、いままでわからなかったことや、意味も分からないまま使っていた式などについて、より深く知ることができるでしょう。

最近特に脚光を浴びている AI の主要な構成要素である、自然言語処理、エキスパートシステム、画像認識、そして機械学習などには、数学そのものや数学の概念が使われています。本書を通して学習することでそれらの基礎となる数学的な考え方と、数学と AI プログラムとの関係も垣間見ることができます。

本書の表記

`>`	Windows のコマンドプロンプトを表します。
`$`	Linux や WSL など UNIX 系 OS のコマンドプロンプトを表します。
`>>>`	Python の一次プロンプトを表します。
`...`	Python の二次プロンプトを表します。
`()`	ひとまとまりの実行可能なコードブロックである関数であることを示します。たとえば、`main` という関数を表すときに、「main という名前の関数」や「関数 `main()`」と表記しないで、単に「`main()`」と表記することがあります。

Italic	そこに具体的な文字や数値、変数、式などが入ることを表します。たとえば「Python *m.n*」は、「Python 3.11」などとなることを表します。また、慣例としてイタリックで表記される集合や確率などを表すときにもイタリック体を使うことがあります。ただし、実行例などリストの場合はイタリックにしません。
Bold	ユーザー（プログラマ）が入力する式や値などであることを表します。
0x*n*	「0x」で始まる表記は 16 進数表現の整数であることを表します。たとえば、0x41 は 10 進数で 65 であることを表します。
0o*n*	「0o」で始まる数値の表記は 8 進数表現の整数であることを表します。たとえば、0o41 は 10 進数で 33 であることを表します。
0b*n*	「0b」で始まる数値の表記は 2 進数表現の整数であることを表します。たとえば、0b0110 は 10 進数で 6 であることを表します。
...	書式の説明において任意の個数を記述できることを示します。
[...]	書式の説明において省略可能であることを示します。
X	キーボードのキーを押すことを示します。たとえば、F5 は F5 キーを押すことを意味します。
S + X	キーボードの S キーを押したまま X キー押すことを示します。Ctrl + F5 は、Ctrl キーを押したまま F5 キーを押すことを意味します。
Note	本文を補足するような説明や、知っておくとよい話題です。
n/m	分数の表記は $\frac{n}{m}$ と表記する場合と、n/m と表記する場合があります。

ご注意

- 本書の内容は本書執筆時の状態で記述しています。Python のバージョンによっては本書の記述と実際とが異なる結果となる可能性があります。
- 本書は数学の全領域を網羅しているものではありません。また、Python のすべてのことについて完全に解説するものではありません。必要に応じて他の資料を参照してください。
- 「問題」に対する「解説と解答」で示すプログラムは典型的な例にすぎません。ひとつの目的に対するプログラミングの方法は複数ある場合があるので、本書で示すコードと違っていても問題となっている目的を達成できていれば正解とします。
- 本書のサンプルは、プログラミングを理解するために掲載するものです。実用的なプログラムとして提供するものではありませんので、ユーザーのエラーへの対処やセキュリティー、その他の面で省略してあるところがあります。

本書に関するお問い合わせについて

本書に関するお問い合わせは、sales@cutt.co.jp にメールでご連絡ください。

なお、お問い合わせは本書に記述されている範囲に限らせていただきます。特定の環境や特定の目的に対するお問い合わせ等にはお答えできませんので、あらかじめご了承ください。特に、特定の環境における特定の開発ツールのインストールや設定、使い方、読者固有の環境におけるエラーなどについてご質問いただいてもお答えできませんのでご了承ください。

お問い合わせの際には下記事項を明記してくださいますようお願いいたします。

- 氏名
- 連絡先メールアドレス
- 書名
- 記載ページ
- 問い合わせ内容
- 実行環境

目次

第1章

はじめてのPython

Python のプログラムの主な実行方法には、2 種類あります。ひとつは、Python の対話モードで実行する方法、もうひとつは、Python のプログラムファイル（スクリプトファイル）を作成して実行する方法です。この章では簡単なプログラムの実行のしかたを学びます。

Python のプログラムの実行方法について良く知っている場合は、この章は飛ばして第 2 章に進んでもかまいません。

1.1 Python との対話

ここでは対話モードで Python を使い始めるために必要なことを説明します。Python のインストールについては付録 A を参照してください。

1.1.1 Python の起動

Python を対話モードで起動する方法は複数あります。

- Python のアイコンがデスクトップにある場合は Python のアイコンをクリックします。
- （Windows の場合）スタートパネル（**図 1.1**）やアプリのリストまたはスタートメニュー（**図 1.2**）から、「Python *X.Y*」→「Python *X.Y*」を選択してクリックします（*X.Y* には具体的な Python のバージョン番号が入ります）。

図 1.1　Windows 11 のスタートパネルの例

図 1.2　Windows 10 のスタートメニューの例

Python のバージョンによっては、「Python *X.Y*」→「Python(command line)」を選択してクリックします。

- （Windows の場合）「検索」フィールドに python と入力して Python を探すこともできます。
- Python のアイコンが見当たらない場合は、コマンドプロンプトや PowerShell などのプロンプトから「`python`」と入力して Python の対話モードを起動することもできます（**図 1.3**）。

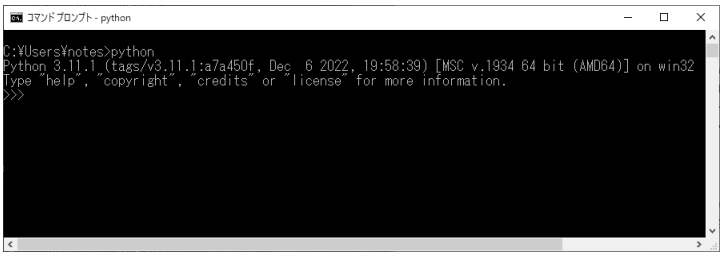

図 1.3　コマンドプロンプトで Python の対話モードを起動した様子

Note　システムによっては、「python」の代わりに「python3」や「python3.7」などバージョンを含めた名前を入力します。また、「py」、「bpython」、「bpython3」などで Python を起動できる場合もあります。さらに、コンソールから idle と入力して Python を使うことができる場合もあります（インストールする環境と Python のバージョンによって異なります）。

Python が起動すると、Python のメッセージと一次プロンプトと呼ばれる「>>>」が表示されます。これが Python の（対話モードの）プロンプトです。

```
>python
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec 6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

これは Windows で Python 3.11.1 を起動した場合の例です。表示されるバージョン番号やそのあとの情報（Python をコンパイルしたコンパイラやプラットフォームの名前など）は、この例と違っていても構いません。

Linux なら、たとえば次のように表示されることがあります。

```
$ python3
Python 3.10.6 (main, Mar 10 2023, 10:55:28) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

いずれにしても、「Type "help", …for more information.」を含む Python のメッセージとプロンプト（>>>）が表示されれば、対話モードが起動したことがわかります。

Note 何かうまくいかない場合は、付録 A「Python のインストールと環境設定」や付録 B「トラブルシューティング」を参照してください。なお、本書では Python 3.7 以降のバージョンを使うことを前提としています。

1.1.2 プロンプト

Python の対話モードでは、入力された Python の命令や式などを 1 行ずつ読み込んで、その結果を必要に応じて出力します。**プロンプト**は、入力を受け付けられる状態になった環境が、そのことをユーザー（プログラマ）に示して入力を促すために表示します。プロンプトに対して命令や計算式などを入力することで、後で説明するようなさまざまなことを行うことができます。

Python を使っているときには、OS（コマンドウィンドウ、ターミナルウィンドウなど）のプロンプトである「>」や「#」、「$」などと、Python のプロンプトである「>>>」を使います。この 2 種類のプロンプトは役割が異なるので区別してください。

1.1.3 単純な計算

最初に Python で計算をしてみましょう。Python のプロンプトに対して、2 + 3 Enter と入力してみます。

```
>>> 2+3
5
>>>
```

上に示したように、2 + 3 の結果である 5 が表示されたあとで、新しいプロンプトが表示されるはずです（以降の例では、結果の後に表示される>>>は省略します）。

Note Web ブラウザや IDE のようなツールを使ってプログラムを実行するときには、プログラムコードを入力するための入力フィールドにコードを入力して、プログラムを実行するためのメニューコマンドやボタンをクリックします。なお、Web ブラウザや IDE を使う実行環境で実行するときには、「`print(2+3)`」のように `print()` を使わないと結果が出力されない場合があります。

引き算や掛け算、割り算を行うこともできます。引き算の記号は「-」（マイナス）ですが、掛け算の記号は数学と違って「*」（アスタリスク）、割り算の記号は「/」（スラッシュ）です。以下に、いくつかの式を続けて入力した様子を示します。

```
>>> 12-5                        # 引き算の例
7
>>> 6*7                         # 掛け算の例
42
>>> 8/2                         # 割り算の例
4.0
>>> 123.45*(2+7.5)-12.5/3       # 複雑な式もOK
1168.6083333333333
```

Note Python のコードで # よりあとはコメント（注釈）とみなされます。コメントはプログラムの実行に影響を与えません。実行結果を確認するだけであれば、上記の # 以降を入力する必要はありません。

Python の対話モードを終了するときには、プロンプトに対して exit()、または、quit() を入力します。

```
>>> exit()
（または）
>>> quit()
```

プログラムが無限ループに入るなどしてこれらの入力を受け付けない状態になったときには、Windows では Ctrl + Z + Enter を実行してみてください。Linux のような UNIX 系 OS では Ctrl + D を実行してみてください。

1.1.4 文字列の表示

C 言語の最初の解説書である『プログラミング言語 C』以来、プログラミングの最初のステップは伝統的に「Hello world!」と表示するコードを示すことになっています。ここでは Python で文字列を表示する方法を説明します。

Python の対話モードでは、単に「'」（アポストロフィ）で囲った文字列を入力しても、入力した文字列が出力されます。

```
>>> 'Hello, Python!'
'Hello, Python!'
```

文字列を「'」で囲まないで次のようにすると、エラーになってしまいます。

```
>>> Hello, Python!
  File "<stdin>", line 1
    Hello, Python!
                ^
SyntaxError: invalid syntax
```

これは、文字列「Hello, Python!」の最後の「!」という文字が文法的に間違っている（syntax error）ということを表しています。

文字列「Hello, Python!」の最後の!を除いて次のようにしてもエラーになります。

```
>>> Hello, Python
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Hello' is not defined
```

これは、「Hello, Python」の Hello の部分を変数とみなして出力しようとしたものの、Hello の値が定義されていない（not defined）ということを表しています。

文字列を出力するときには、文字列の前後を「'」で囲むのを忘れないようにしましょう。

Note 文字列を「'」で囲む代わりに「"」（引用符）で囲んでもかまいません。

```
>>> "Hello, Python!"
'Hello, Python!'
```

1.1.5 print() を使った出力

電卓のように式の値を計算して表示したり、文字列をそのまま表示するのではなく、「プログラムコードを実行した」と感じられることをやってみましょう。

値を出力するために、Python には `print()` が定義されています。

ここでは「Hello, Python!」と出力するプログラムコード「`print ('Hello, Python!')`」を実行してみましょう。プログラムの意味はあとで考えることにします。

```
>>> print ('Hello, Python!')
Hello, Python!
>>>
```

入力したプログラムコードは「print('Hello, Python!')」です。次の行の「Hello, Python!」は、プログラムコードを実行した結果として Python が出力した情報です。

「print ('Hello, Python!')」の「print」は、そのあとのかっこ () の内容を出力する命令です。

Note 「print()」のような何らかの結果をもたらす命令を関数といいます。関数についてはあとの章で説明します。

出力する内容は「Hello, Python!」なのですが、これが文字列であることを Python に知らせるために、「'」または「"」で囲みます。

```
>>> print ("Hello, Python!")
Hello, Python!
>>>
```

同じようにして、計算式を出力することもできます。

```
>>> print (2*3+4*5)
26
>>>
```

今度は文字列ではなく式を計算した結果である数値を出力したので、かっこの中を「'」や「"」で囲っていないことに注意してください。

print() を使う場合と使わない場合で結果がまったく同じであるわけではありません。「print ('Hello, Python!')」を実行すると、「'」で囲まれていない「Hello, Python!」という文字列だけが出力されます。

```
>>> print ('Hello, Python!')
Hello, Python!
```

これは単に「Hello, Python!」という文字列を出力したことを表しています。

一方、>>>に対して print() を使わないで「'Hello, Python!'」と入力すると、「'Hello, Python!'」のように「'」で囲まれた文字列が出力されます。

```
>>> 'Hello, Python!'
'Hello, Python!'
```

これは、Python が値を出力するときに、それが文字列であることを「'」で囲って示すためです。

［問題］ あなたの BMI を計算してみましょう。BMI は次の式で計算します。

$$\mathrm{BMI} = \frac{\text{体重}}{\text{身長} \times \text{身長}}$$

体重は kg 単位、身長は m 単位です。

［解説と解答］ 身長が 1.72 m で体重が 58 kg の場合、Python の対話モードで次のように計算します。

```
>>> 58.0 / (1.72 * 1.72)
19.60519199567334
```

［終］

次に進む前に、Python のプロンプトに対して「`exit()`」または「`quit()`」を入力して対話モードをいったん終了し、OS のコマンドプロンプトに戻ります。

1.2 スクリプトファイル

Python のプログラムは、対話モードで入力しながら実行するほかに、ファイルに保存しておいたコードを実行することもできます。

1.2.1 ファイルの作成と保存

次の 1 行だけのプログラムのファイル（スクリプトファイル）を作成してみましょう。コードをテキストエディタに入力した様子を**図 1.4**、**図 1.5** に示します。

```
print ('Hello, Python!')
```

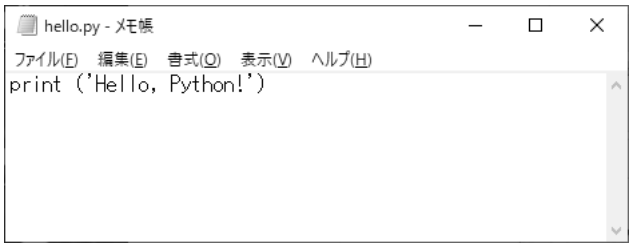

図 1.4　Windows のメモ帳で編集した例

図 1.5　gedit で編集した例

そして、これを hello.py というファイル名で保存します。こうしてできたファイルが Python のプログラムファイルであり、スクリプトファイルともいいます。

Note　Windows のようなデフォルトではファイルの拡張子が表示されないシステムの場合、ファイルの拡張子が表示されるように設定してください。また、自動的に txt のような拡張子が付けられるエディタでは、hello.txt や hello.py.txt というファイル名にならないように注意する必要があります。

ファイルを保存する場所には注意を払う必要があります。あとで.py ファイルを容易に（パスを指定しないで）実行できるようにするには、適切なディレクトリを用意してからそこに保存するとよいでしょう。

Windows ならば、c:¥mathpyex¥ch01 といったフォルダーを作って、そこに保存します。Linux などの UNIX 系 OS ならば、ユーザーのホームディレクトリ以下に mathpyex/ch01 といったディレクトリを作って、そこに保存します。

1.2.2　スクリプトの実行

次に、作成したスクリプト（Python のプログラムファイル）を実行してみます。

まず、コマンドプロンプトや PowerShell などを開き、プロンプト（>、$、% など）が表示さ

れているようにします。

スクリプトファイルを¥mathpyex¥ch01 に保存したのであれば、コマンドラインで「cd ¥mathpyex¥ch01」を実行してカレントディレクトリを変更します。そして、プロンプトに対して「python hello.py」と入力してください。

Note 「python hello.py」の「python」の部分は、インストールされている Python の種類によって「python3」、「py」、「python3.10」などに変えます。

プログラムが実行されて、次のように結果の文字列「Hello, Python!」が表示されるはずです。

```
>python hello.py
Hello, Python!
```

パスを指定して実行するなら、次のようにします。

```
>python c:¥mypython¥ch01¥hello.py
Hello, Python!
```

1.2.3 スクリプトの出力

対話モードでの実行と違って、スクリプトファイルを実行するときには print() を使わないと何も出力されません。たとえば、次のような内容のファイルを作って実行しても何も起きません。

```
'Hello, Python!'
```

これは、対話モードでは、コードが入力されるたびにその実行結果が出力（表示）されるのに対して、スクリプトファイルの実行では、print() を使うなどして明示的に命令しないと値が出力されないからです。

数値計算などでも同じです。対話モードでは、単に「2+3」と入力するだけで「5」という結果が出力されました。しかし、スクリプトファイルの実行で同じ出力結果を得るには、次のようなスクリプトを作成する必要があります。