Ryan Luu
Annotated Copy of Introduction to Pyhon3 translation

Introduction

Python はオランダのプログラマであるグイド・ヴァンロッサム（Guido van Rossum）によって 1991 年に開発されたプログラミング言語であり，言語処理系は基本的にインタプリタである。

Developed as a programming language in 1991 by Dutch programmer Guido van Rossum, Python's language processor is fundamentally an interpreter.

* [Technical]
Language Processors are what transforms source code (Code programmers write) into machine language (1 and 0's) so that the computer can understand what to do. Depending on the programming language, language processors can be any of the following three type; Compilers, Assemblers, and Interpreter. Python just happens to be an Interpreter in this case.

Python は多目的の高水準言語であり，言語そのものの習得とアプリケーション開発に要する労力が比較的に少ないとされる.

Python is a multi-purpose, high-level language that requires relatively little effort to learn the language itself and develop applications.

しかし，実用的なアプリケーションを開発するために必要とされる多くの機能が提供されており，この言語の有用性の評価が高まっている.

However, since many functions necessary for developing practical applications are already provided, the usefulness of the language is increasing.

* [Translation] It doesn't explicitly say "already" provided, but it's implied.

Python の言語処理系（インタプリタ）は多くのオペレーティングシステム（OS）に向けて用意されている。

Python's language processor (interpreters) are available for many operation systems (OS).

Python で記述されたプログラムはインタプリタ上で実行されるために，C や Java と比べて実行の速度は遅いが，C 言語など他の言語で開発されたプログラムとの連携が容易である。

Ryan Luu
Annotated Copy of Introduction to Pyhon3 translation

Since programs written in Python are executed on the interpreter, execution speed is slower than C and Java, but it is easy to link with other programs developed in other languages such as C.

このため，Python に組み込んで使用するための各種の高速なプログラムがライブラリ（モジュール，パッケージ）の形で多数提供されており，それらを利用することができる．

Because of this, a number of high-speed programs are provided in the form of libraries (modules and packages), which can be used.

Python 用のライブラリとして利用できるものは幅広く，言語そのものの習得や運用の簡便性と相俟って，情報工学や情報科学とは縁の遠い分野の利用者に対してもアプリケーション開発の敷居を下げている．

There are a wide variety of libraries that can be used for Python. Combined with learning the language itself and the ease of operation, it lowers the threshold for application development even for users in fields far from information engineering or information science.

本書で前提とする Python の版は 3（3.6 以降）である．

The Python version assumed in this book is 3 (3.6 or later).

1.1 What you can do with Python

Python は汎用のプログラミング言語である．また，世界中の開発コミュニティから多数のライブラリが提供されている．このため，Python が利用できる分野は広く，特に科学，工学系分野のためのライブラリが豊富である．

Python is a general-purpose programming language with a number of libraries provided by development communities around the world. For this reason, Python can be used in a wide range of fields.

*[Translation] Combined sentences together to make it flow nicer.

それらライブラリを利用することで，情報処理技術を専門としない領域の利用者も手軽に独自の情報処理を実現することができる．特に次に挙げるような処理を実現する上で有用なライブラリが揃っている．

• 機械学習
• ニューラルネット
• データサイエンス
• Web アプリケーション開発

By using these libraries, users who not specialize in information processing technologies can easily implement their own information processing. In particular, libraries useful for realizing the following are available.

- Machine Learning
- Neural Network
- Data Science
- Web Application Development

これに加えて，グラフィックス（GUI 含む）を作成するためのライブラリも豊富であり，デスクトップ PC や携帯情報端末のためのアプリケーションプログラムの開発においても簡便な手段を与える．

In addition to this, there are abundant libraries for creating graphics (including GUI's), which provide a convenient means for developing applications for PC's and personal digital assistants.

1.2 This book's contents

本書は Python でアプリケーションプログラムを開発するために必要な最小限の事柄について述べる．本書の読者としては，Python 以外の言語でプログラミングやスクリプティングの基本を学んだ人が望ましい．本書の内容を概略として列挙すると次のようなものとなる．

This book describes the minimum requirements for developing application programs in Python. Readers of this book should be someone who has learned the basics of programming and scripting languages from languages other than Python.

• Python 処理系の導入の方法
インタプリタとライブラリのインストール方法など
• Python の言語としての基本事項
文法など
• 実用的なアプリケーションを作成するために必要なこと

Ryan Luu
Annotated Copy of Introduction to Pyhon3 translation

通信やマルチスレッド，マルチプロセスプログラミングに関することなど
• GUI アプリケーションを作成するために必要なこと
GUI ライブラリの基本的な扱い方など
• サウンドの基本的な扱い
WAV ファイルの入出力や音声のサンプリングと再生

- How to install Python Processor
    - How to install interpreter and libraries
- Basics of the Python Language
    - Syntax
- Things necessary for creating practical applications, telecommunications, multi-threading, multi-process programming etc.
- What you need to create a GUI application
    - Basic handling of GUI libraries
- Basic handling of sound
    - WAV file input/output and audio sampling and playback

*[Translation]

(2nd Bullet Point) : 文法など → Translates to “Grammar”. But in Computer Science what they are referring to is called “syntax”.

Python を初めて学ぶ方には，言語としての基礎的な内容について解説している章.

「1 はじめに」（この章）
「2 Python の基礎」

を読むことをお勧めする．その後，実用的なプログラミングにおいて重要となる事柄に関して解説した章

[4 実用的なアプリケーション開発に必要な事柄]

を読むことをお勧めする．その他の章については，学びの初期においては必ずしも必要とはならないこともあり，読者の事情に合わせて利用されたい.

If you are new to Python, these chapters explain the basics of the language.

[1] Introduction (this chapter)
[2] Python Basics

After that it is recommended that you read the chapter about important matters in practical programming.

[4] Necessary Items for Practical Application Development

Other chapters may not be necessary in the early stages of learning and should be utilized according to the reader's circumstances.

*[Translation]
So I decided to cut out the fragment of [を読むことをお勧めする] since it read awkwardly in English.

1.3 How to install and start Python

Python の処理系（インタプリタ）は Python ソフトウェア財団（以後 PSF と略す）が開発，維持しており，当該財団の公式インターネットサイト https://www.python.org/ から入手することができる

The Python processing system (interpreter) is developed and maintained by the Python Software Foundation (hereinafter abbreviated as PSF) and can be obtained from the foundation's official internet site: https://www.python.org./

*[Translation]

The title actually is → 1.3) 処理系の導入（インストール）と起動の方法.

But I changed it to "How to install and start Python" since it's more intuitive than "How to install and start the processing system" for readers.

Microsoft 社の Windows や Apple 社の Mac には専用のインストールパッケージが用意されており，それらを先のサイトからダウンロードしてインストールすることで Python 処理系が使用可能となる.

Microsoft Windows and Apple's Mac has its own separate installation packages, and you can download and install them from the previous site to start using the Python processor.

*[Translation]

Took a bit of creative liberty to make the sentence flow together more nicely. I think I managed to keep the main idea of the sentence intact though.

Linux をはじめとする UNIX 系 OS においては，OS のパッケージ管理機能を介して Python をインストールできる場合が多いが，先のサイトから Python のソースコードを入手して処理系をビルドすることもできる．

In UNIX-like OS such as Linux, Python can often be installed via the OS package management function, but you can also obtain the Python source code from the previous site and build the processor from the ground up.

* [Technical]

There's a lot of complicated words and terms here that is not beginner friendly. The main idea is that Python can be installed differently on different computers.

1.3.1 Python Processor's distribution (shape of distribution)

Python 処理系は C 言語により記述されており，C 言語処理系によってビルドすることができるが，コンパイル済みのインストールパッケージ（前述）を利用するのが便利である．インストールパッケージとしては PSF が配布するもの以外に Anaconda, Inc.2 が配布する Anaconda が有名である．

The Python processor is written in the C language and can be built by the C language processor, but it is more convenient to use the compiled installation package described above. In addition to those distributed by the PSF, "Anaconda", distributed by Anaconda Inc. 2, is also famous as an installation package.

PSF のインストールパッケージや Anaconda で Python 処理系を導入すると，各種のライブラリを管理するためのツールも同時に導入されるので，Python を中心としたソフトウェア開発環境を整えるための利便性が大きい．

If you install a Python processor in the PSF installation package or through Anaconda, you can manage various libraries. Since the tools will be introduced at the same time, it is very convenient to set up a software development environment centered on Python.

ただし注意すべき点として，PSF の管理ツールと Anaconda ではライブラリの管理方法や，Python 自体のバージョン管理の方法が異なるため，Python の処理環境を導入する際は，ディストリビューション（PSF か Anaconda か）を統一すべきである．

However, it should be noted that PSF management tools and Anaconda use different library management methods from Pythons management methods. So, when installing the Python processing environment, they should first be unified together.

*[Translation] It was hard to translate this literally, so I took the main ideas away and put them into two separate sentences.

※ 巻末付録「A.2 Python のインストール作業の例」（p.195）でインストール方法を概略的に紹介している.

※ The appendix “A.2 Example of Python Installation” (p. 195) provides an overview of the installation method.

本書では PSF のインストールパッケージによって Python を導入した処理環境を前提とするが，言語としての Python の文法はディストリビューションに依らず共通であり，各種のライブラリも導入済みの状態であれば，使用方法は原則として同じである.

Although this book assumes a processing environment in which Python is installed using the PSF installation package, the syntax of Python as a language is common regardless of the distribution and if various libraries are already installed, its usage is in principle the same.

1.3.2 Starting the Python Processor

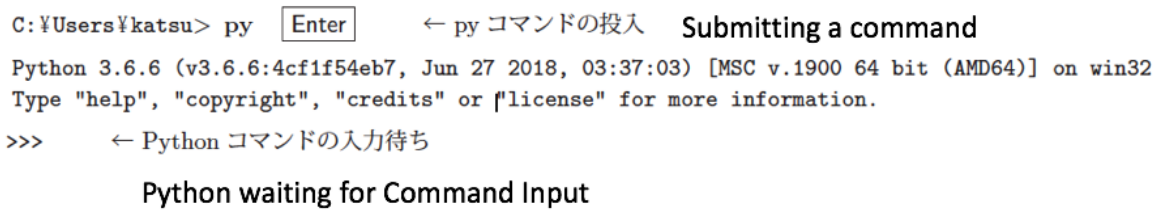
コマンド（PSF 版 Python）を，Linux や Mac ではターミナルウィンドウから python コマンドを投入 3 することで次の例のように Python インタプリタが起動する.

If you enter the command (PSF version of Python) Python 3 into the terminal window in Linux of Mac, the Python interpreter starts up as shown in the following example.

例．Windows のコマンドプロンプトから Python を起動する例

Example) An example of starting Python from the Windows command prompt.

Ryan Luu
Annotated Copy of Introduction to Pyhon3 translation



*[Translation] I figured the easiest way to translate these segments are retain their meaning/instruction would be to just include the picture itself with textboxes translating the important bits. Otherwise I would be losing a lot of context if I were just to show it on my machine or just retype it out.

注1） 本書で例示する操作を Anaconda 環境下で実行するには，巻末付録「A.3.3 Anaconda Prompt の起動」（p.198）に示す方法で Python 処理系を起動する．

注2） 本書では JupyterLab や IPython といった対話環境については言及しない．JupyterLab の基本的な使用方法に関しては別冊の「Python3 ライブラリブック」で解説しているのでそちらを参照のこと．

Note 1) To execute the operations illustrated as illustrated in the manual in the Anaconda environment, see “Appendix A.3.3 Starting the Anaconda Prompt”

- You can then start the Python processor using the method shown on (p 198).

Note 2) This book does not mention interactive environments such as JupyterLab and IPython. The basic usage of JupyterLab is explained in the separate “Python 3 Library Book” so please refer to that.

1.3.3 Interactive Mode

先の例の最後の行で「>>>」と表示されているが，これは Python のプロンプトであり，これに続いて Python の文や式を入力することができる．この状態を Python の対話モードと呼ぶ．処理系を終了するには exit() と入力する．

The last line in the previous example displays “>>>”, which is a Python prompt. This is followed by a Python statement or expression. This state is called the Python interactive mode. To exit the processing system, enter exit().

*[Translation] Though there were two sentences, I broke them up into three to make the ideas flow better.

例． Python の終了

Example) Exiting Python

```
>>> exit()  Enter    ← Python 終了   Exiting Python
C:¥Users¥katsu>      ← Windows のコマンドプロンプトに戻る  Return to Windows Command Prompt
```

対話モードでは，入力された Python の文や式が逐一実行されて結果が表示される．そして再度プロンプト「>>>」が表示されて，次の入力を受け付ける．また，テキストファイルに書き並べた Python の文や式をスクリプトとしてまとめて実行することもできるが，この方法については「2.1 スクリプトの実行」（p.4）で説明する．

In interactive mode, entered Python statements and expressions are executed one by one and the results are displayed. The prompt “>>>” is displayed again and the next input is accepted. In addition, Python statements and expressions written in a text file can be executed together as a script, but this method is explained in “2.1 Executing Scripts” (p4).

異なる複数のバージョンの Python 処理系を導入し，それらを選択して起動することもできる．これに関しては「A.3 Python 起動のしくみ」（p.196）で解説する．

You can also install different versions of the Python processor, select them, and launch those. This is explained in “A.3 How Python starts” (p196).

1.4 Using the GUI library

グラフィカルユーザインターフェース（GUI）を備えたアプリケーションプログラムを構築するには GUI を実現するためのプログラムライブラリを入手して用いる必要がある．

To build an application program with a graphical user interface (GUI), it is necessary to obtain and use a program library to implement the GUI.

GUI のためのライブラリには様々なものがあり，Python 処理系に標準的に提供されている Tkinter をはじめ，GNOME Foundation が開発した GTK+，Qt Development Frameworks が開発した Qt，Julian Smart 氏（英）が開発した wxWidgets，Kivy Organization が開発した Kivy など多くのものが存在している．

There are various libraries for GUIs out there. Here is a list that includes but is not limited to list of available GUI libraries:

- Tkinter -- which is provided as a standard in Python processing systems.
- GTK+ -- developed by the GNOME Foundation.

- Qt -- developed by Qt Development Framworks.
- wxWidgets – Developed by British developer Julian Smart.
- Kivy – Developed by Kivy Organization (Open Source).

*[Translation] This sentence was quite difficult to translate into English since it just lists a bunch of stuff. Though I may have ruined the flow of the authors voice, I feel that breaking it into a list is optimal way to get his ideas across.

具体的には，それら GUI のライブラリを Python から使用するために必要となるライブラリを導入し，Python のプログラムから GUI ライブラリを呼び出すための API を介して GUI を構築する．本書では，比較的新しく開発され，携帯情報端末（スマートフォンやタブレットなど）でも動作する GUI アプリケーションを構築することができる Kivy を主として 4 取り上げる．

Specifically, after the GUI libraries necessary to use are installed, the GUI is constructed via an API calling the GUI library from inside a Python program. In this document, we focus on Kivy, which is a relatively new development and can build GUI applications that can be used on mobile information terminals (smartphones, tablets, etc).

*[Translation] Once again his first sentence in particular is quite a mouthful once literally translated.

1.5 More Information about Python

本書で説明していない情報に関しては，参考となる情報源（インターネットサイトなど）を巻末の付録に挙げるので，そちらを参照のこと．

For information that is not described in this document, refer to the appendix at the end of the book for reference information sources (Internet sites, etc).

---

2 Python Basics

他のプログラミング言語の場合と同様に，Python でも変数，制御構造，入出力の扱いを基本とする．Python の文や式はインタプリタのプロンプトに直接与えること（対話モードでの実行）もできるが，文や式をテキストファイルに書き並べて（スクリプトとして作成して）まとめて実行することもできる．

As in other programming languages, Python is basically based around the handling of variables, control structures, and input/output. Python statements and expressions can be given directly to the interpreter prompt (execution in interactive mode), but statements and expressions can also be written together in a text file (created as a script) and executed together.

2.1 Script execution

例えば次のようなプログラムが test01.py というファイル名のテキストデータとして作成されていたとする．

For example, the following program is created as text data with the file name test01.py

Program: test01.py

```
1 # coding: utf-8
2
3 print( '私の名前はPythonです. ' )
```

print ("My name is Python.")

To do this from the OS terminal window

```
python test01.py
```

Enter (For Windows, Linux, Anaconda)

Or type

```
py test01.py
```

Enter (For PSF Versions of Windows)

と入力する．するとターミナルウィンドウに「私の名前は Python です．」と表示される．

Then if entered, "My name is Python" is displayed to the terminal window.

このプログラム例 test01.py の中に print で始まる行がある．これは print 関数というもので端末画面（標準出力）への出力の際に使用する関数であり，今後頻繁に使用する．

In this program example test01.py, there is a line that starts with print. This is a print function that is used for output to the terminal screen (standard output) and will be used frequently in the future.

Ryan Luu
Annotated Copy of Introduction to Pyhon3 translation

《 print 関数 》
書き方： print(出力内容)
「出力内容」はコンマ「,」で区切って書き並べることができる．出力処理の後は改行されるが,「end=''」を与えると行末の出力を抑止できる.

<< **print function** >>
How to write: **print** (**Output Contents**)
“Output contents” can be separated by a comma “,”. After the output process, a line break will occur, but if you type “end =”, you can suppress the output at the end of the line.

注意） print は関数なので 5 出力内容を必ず括弧’(...)’ で括る（引数として与える）必要がある.print 関数の行末処理に関するサンプルプログラムを printTest01.py に示す.

Note) Since print is a 5 keyword function, it is necessary to enclose the output contents in parentheses ‘(…)’ (given as a argument). A sample program for line ending processing of the print function is shown in printTest01.py.

Program: printTest01.py

Ryan Luu
Annotated Copy of Introduction to Pyhon3 translation

```
1  # coding: utf-8
2  print( 'これは' )
3  print( 'print関数が' )
4  print( '改行する様子の' )
5  print( 'テストです. ', end='\n\n' )
6
7  print( 'しかし, ', end='' )
8  print( 'キーワード引数 end=\'\' を与えると', end='' )
9  print( 'print関数は', end='' )
10 print( '改行しません. ' )
```

1. #coding: utf-8
2. print (“this is”)
3. print (“The print function”)
4. print (“Like a line break”)
5. (“This is a test”, end = ‘\n \n’)
6.
7. print (“But, end = “)
8. print (“If the keyword argument end = \’ \ \ is given”, end = “”)
9. print (‘print , end = “)
10. print (“No line feed”)

‘end=’ はキーワード引数 6 であり，行末に出力するものを指定できる．このプログラムを実行すると次のように表示される．

これは
print 関数が
改行する様子の
テストです.
しかし，キーワード引数 end=’’ を与えると print 関数は改行しません.

“End =” is a 6 keyword argument that can be output at the end of the line. When this program is executed, the follow is displayed.

This is
The print function
of a line break
This is a test

If the keyword argument end = “” is given print no line feed.

*[Translation] So it reads weird because of how Japanese and English are reversed. But I can’t fix that since the literally code itself would produce the output that I’ve shown. It’s quite interesting.

このプログラムの 5 行目に「end=\n\n」という記述があるが，これは改行を意味するエスケープシーケンス 7 であり，「\n」を 2 つ指定することで行末で 2 回改行する．

There is a description “end = \n \n” on the 5th line of this program. This is an escape sequence that means a line feed that results in line 7. By specifying two “\n” characters, the line breaks twice at the end.

*[Translation] Idea was as stated though it was not direct word-to-word translation.

注）表示する端末によってはバックスラッシュ「\」が「￥」と表示されることがある．

Note) Backslash “\” may be displayed as “￥” depending on the display terminal.

2.1.1 Writing comments in the program

「#」で始まる行はコメントであり,Python の文とは解釈されずに無視される．ただし，先のプログラムの冒頭にある

coding: utf-8

は，プログラムを記述するための文字コードを指定するものであり，コメント行が意味を成す特別な例である．文字コードとしては utf-8 が一般的であるが，この他にも表 1 に示すような文字コードを使用してプログラムを記述することができる．

Lines beginning with ‘#’ are comments. They are not interpreted as Python statements and are ignored. However, at the beginning of the previous program

coding: utf-8

specifies a character code for describing a program, and is a special case where comment lines make sense. utf-8 is a common character code, but you can also write programs using the character codes shown in Table 1.

*[Translation] I didn’t want to have two and’s in a single sentence, so I broke up the thought in line 1 into two sentences.

表 1: 指定できる文字コード

文字コード	coding:に指定するもの	文字コード	coding:に指定するもの
UTF-8	utf-8	EUC	euc-jp
シフト JIS	shift-jis, cp932*	JIS	iso2022-jp

* マイクロソフト標準キャラクタセットに基づく指定

Table 1: Character Codes that can be specified

Character Code	What to specify for coding:	Character Code	What to specify for coding:
UTF-8	utf-8	EUC	euc-jp
shift JIS	shift-jis, cp932*	JIS	Iso2022-jp

*Specification based on **Microsoft standard character set**

---

coding:に指定するものは大文字／小文字の区別はなく，ハイフン'-' とアンダースコア' ' のどちらを使用してもよい．また，文字コード指定のコメント行を記述する際,

```
# -*- coding: utf-8 -*-
```

と記述することがあるが，この"-*-" は Emacs 8 で用いられる表現であり，省略しても問題はない．

The coding is not case sensitive, and you can use either a hyphen '-' or an underscore '_'. When describing a comment line with a character code specification,

```
# -*- coding: utf-8-*-
```

the "-*-"is an expression used in Emacs 8, and there shouldn't be a problem even if it was omitted.

2.12 Program Indentation

Python ではプログラムを記述する際のインデント（行頭の空白）に特別な意味があり，不必要なインデントをしてはならない．例えば次のような複数の行からなるプログラム test02.py は正常に動作するが，不必要なインデントを施したプログラム test02-2.py は実行時にエラーとなる．

Ryan Luu
Annotated Copy of Introduction to Pyhon3 translation

In Python, indentation (blank space at the beginning of a line) has a special meaning, and you should refrain from unnecessary indentation. For example, the program test02.py consisting of multiple lines like the following works normally, but the program test-02-2.py, with unnecessary indentation, givens an error at runtime.

正しいプログラム：test02.py

```
1 # coding: utf-8
2
3 print( '1. 私の名前はPythonです. ' )
4 print( '2. 私の名前はPythonです. ' )
5 print( '3. 私の名前はPythonです. ' )
```

間違ったプログラム：test02-2.py

```
1 # coding: utf-8
2
3 print( '1. 私の名前はPythonです. ' )
4     print( '2. 私の名前はPythonです. ' )
5 print( '3. 私の名前はPythonです. ' )
```

Correct Program: test02.py

```
1. #coding: utf-8
2.
3. print(  '1. My name is Python. ')
4. print ( '2. My name is Python. ')
5. print ( '3. My name is Python. ')
```

Wrong Program: test02-2.py

```
1. #coding: utf-8
2.
3. print(  '1. My name is Python. ')
4.          print ( '2. My name is Python. ')
5. print ( '3. My name is Python. ')
```

---

test02-2.py を実行すると次のようになる.

```
File "test02-2.py", line 4
  print( '2. 私の名前は Python です. ' )
IndentationError: unexpected indent
```

インデントが持つ意味については「2.4 制御構造」（p.45）のところで解説する．

When test02-2.py is executed, this is shown as follows.

```
File “test02-2.py”, line 4
   Print (‘2 My name is Python. ’)
IndentationError: unexpected indent
```

The meaning of the indentation is explained in “2.4 Control Structure” (p45)

2.2 Variable and Data Types

変数はデータを保持するものであり，プログラミング言語の最も基本的な要素である．また，変数に格納する値には数値（整数，浮動小数点数）や文字列といった様々な型がある．C 言語や Java では，使用する変数はその使用に先立って明に宣言する必要があり，変数を宣言する際に格納するデータの型を指定しなければならず，宣言した型以外のデータをその変数に格納することはできない．

Variables hold data and are the most basic element of a programming language. There are various types of values stored in variables, such as numbers (integers, floating point numbers) and character strings. In the C language or Java, the variable must be explicitly declared prior to its use, specified when declared, and any other data other than the declared type must also be specified. Otherwise it cannot be stored in a variable.

この様子を指して「C 言語や Java は静的な型付けの言語処理系である」と表現する．これに対して Python は動的な型付けの言語処理系であり，1 つの変数に格納できる値の型に制約は無い．すなわち，ある変数に値を設定した後で，別の型の値でその変数の内容を上書きすることができる．また Python では，変数の使用に先立って変数の確保を明に宣言する必要は無い．例えば次のような例について考える．

This situation is occurs because “The C language and Java are statically typed language processors”. On the other hand, Python is a dynamically typed language processor, and there are no restrictions on the types of values that can be stored in a single variable. That is, after you set a value for a variable, you can overwrite the contents of that variable with another type of value. In Python, it is not necessary to explicitly declare a variable reservation prior to using the variable. For example, consider the following example.

```
>>> x = 2   Enter      ←変数 x に整数の 2 を格納   Stores the integer 2 into the variable x
>>> y = 3   Enter      ←変数 x に整数の 3 を格納   Stores the integer 3 into the variable y
>>> z = x + y   Enter  ← x と y の値を加算したものを変数 z に格納 Stores the sum of x and y into the variable z
>>> print(z)   Enter   ←変数 z の内容を出力する処理 Processing to output the contents of z
5                      出力された内容 Output contents
```

Ryan Luu
Annotated Copy of Introduction to Pyhon3 translation

これは Python の処理系を起動して変数 x, y に整数の値を設定し，それらの加算結果を表示している例である．変数への値の設定はイコール記号'=' を使う．引き続いて次のように Python の文を与える．

This is an example of starting the Python processing system, setting integer values for the variables x and y, and displaying the addition results. The equal sign "=" is used to set the value to the variable. Next, a Python statement is given as follows.

---

*Note I'm stopping here in the translation. I end up on roughly page 6.