Introduction to Python 3

GUI development with Kivy, Sound input and output, and Web scraping

Version 1.1.6

中村勝則

Translated by Ryan Luu without the author's permission for educational purposes

JPN 521 – Graduate Level Japanese Language Course

12/10/2019

1) Introduction

Developed as a programming language in 1991 by Dutch programmer Guido van Rossum, Python's language processor is fundamentally an interpreter. Python is a multi-purpose, high-level language that requires relatively little effort to learn the language itself and develop applications. However, since many functions necessary for developing practical applications are already provided, the usefulness of the language is increasing.

Python's language processor (interpreters) are available for many operation systems (OS). 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. Because of this, a number of high-speed programs are provided in the form of libraries (modules and packages), which can be used. 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.

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

1.1) What you can do with Python

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. 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

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

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.

- 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

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.

1.3) How to install and start Python

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./.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. 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.

1.3.1) Python Processor's distribution

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.

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. 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.

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

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

If you enter the command (PSF version of Python) Python3 into the terminal window in Linux of Mac, the Python interpreter starts up as shown in the following example.

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

```
C:¥Users¥katsu> py  Enter      ← py コマンドの投入   Submitting a command
Python 3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 03:37:03) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>      ← Python コマンドの入力待ち
```

Python waiting for Command Input

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

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().

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
```

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).

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

There are various libraries for GUIs out there. Here is a list of a few of the following 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).

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).

1.5) More Information about Python

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

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.

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)

Then if enter is pressed, “My name is Python” is displayed to the terminal window.

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.

<< **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.

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

Program: printTest01.py

```
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 =” 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
Like a link break
This is a test

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

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.

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

2.1.1 Writing comments in the program

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.

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**

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

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.

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. ')
```

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

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. 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
```

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.