Kinoshita Daisuke
https://www.instagram.com/daisuke23888/

https://s3b.astro.ncu.edu.tw/ipype/

08 May 2025

基礎 Python 程式設計練習

— Session 10 —

https://meet.google.com/wpx-yabb-tka

進來的人請跟我說一聲

有人進來我們就開始

Please talk to me, when you come in.

When you come in, we start the session.

- Date/Time: 08 May 2025
- Google Meet: `https://meet.google.com/wpx-yabb-tka`
- Language: English, or Chinese, or Japanese
- Anyone can join this activity.
- This activity is organised in conjunction with the course “Doing Astrophysics using Python”. If you take the course “Doing Astrophysics using Python” and you would like to study basic Python programming, you are highly encouraged to join this activity.
- Activity web page: `https://s3b.astro.ncu.edu.tw/ipype/`

https://s3b.astro.ncu.edu.tw/ipype/

Some notes for you

- Ask a question to me at any time, if you have any difficulty.
- When you finish writing your code, tell me about it.
- Show me your source code, if you have an error for your code and cannot find a way to fix it.
- Tell me if you want me to provide a sample code for you.
- Be active, then you gain more.

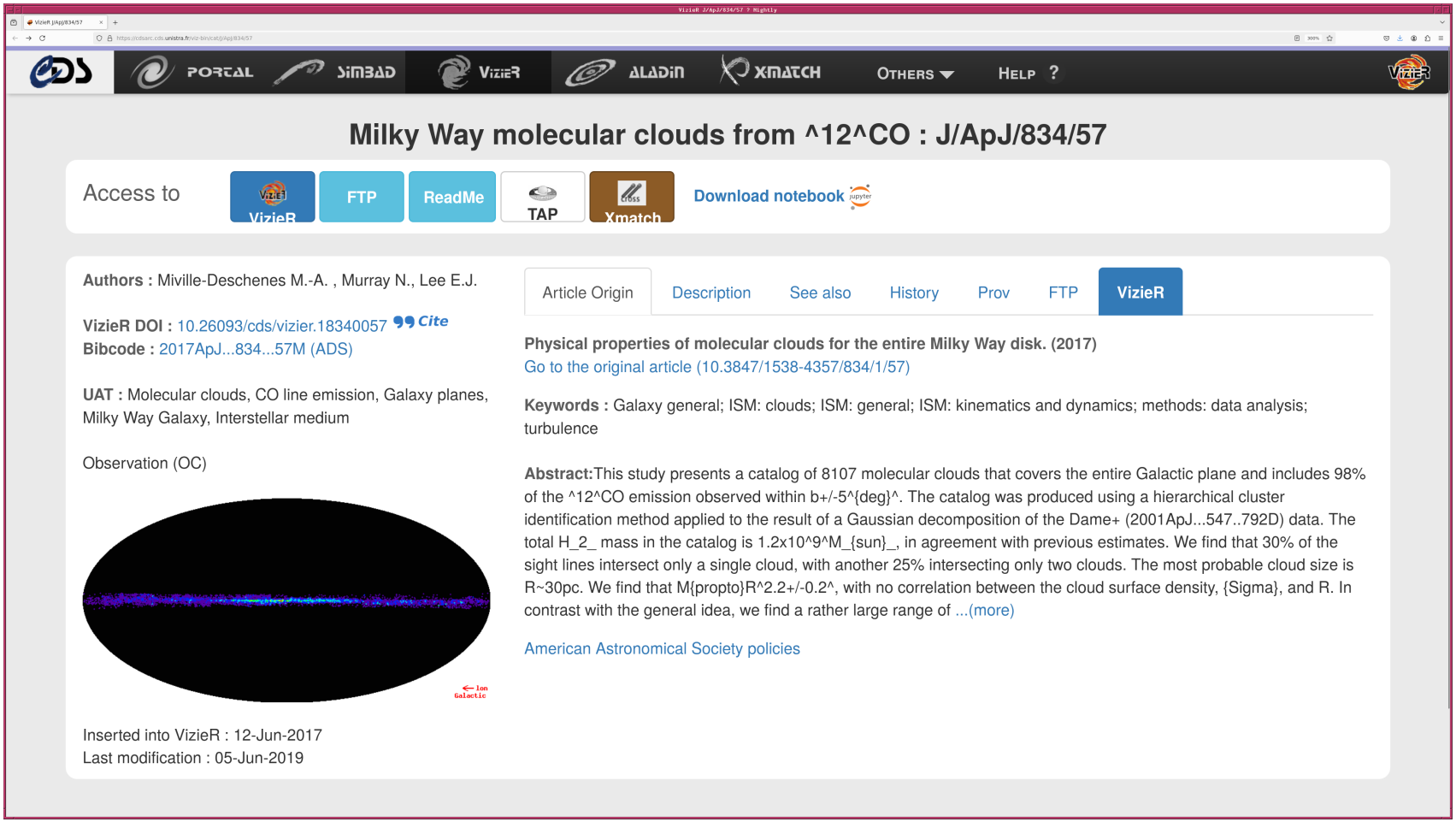
Some notes for you

- 如果想要問問題, 請跟我說. 什麼時候都可以.
- 你的程式寫完的時候, 請跟我說一聲.
- 如果你不確定你自己寫的程式寫得好不好, 請給我看你寫的原始碼. 我跟你說我的意見.
- 如果你想要看我的寫法, 請跟我說. 我寫給你看.
- 比較主動一點, 你的收穫比較多.

Downloading the data file

- Visit the following web page, and download the data file and ReadMe file.
  - "Milky Way molecular clouds from 12CO"
    - `https://cdsarc.cds.unistra.fr/viz-bin/cat/J/ApJ/834/57`

Downloading the data file



https://cdsarc.cds.unistra.fr/viz-bin/cat/J/ApJ/834/57

Exercise s10-01

- Make your own Python script to open the data file "`table1.dat.gz`", extract galactic longitude (GLON), galactic latitude (GLAT), distance flag (INF), near kinematic distance (Dnear), and far kinematic distance (Dfar) for each molecular cloud in the catalogue.

Exercise s10-02

- Make your own Python script to open the data file "`table1.dat.gz`", extract galactic longitude (GLON), galactic latitude (GLAT), distance flag (INF), near kinematic distance (Dnear), and far kinematic distance (Dfar) for each molecular cloud in the catalogue, check the distance flag and find more likely distance (Dnear or Dfar).

Exercise s10-03

- Make your own Python script to open the data file "`table1.dat.gz`", extract galactic longitude (GLON), galactic latitude (GLAT), distance flag (INF), near kinematic distance (Dnear), and far kinematic distance (Dfar) for each molecular cloud in the catalogue, check the distance flag and find more likely distance (Dnear or Dfar), calculate galactocentric coordinate $(x_{gal}, y_{gal}, z_{gal})$.
  - Assume 8.34 kpc for the distance of the Sun from the Galactic centre.
  - Assume 25 pc for the distance of the Sun from the Galactic mid-plane.
  - For the calculation of galactocentric coordinates, read the Section 2.4.1 "Physical Properties" of the paper "A UNIFORM CATALOG OF MOLECULAR CLOUDS IN THE MILKY WAY".
    - `https://iopscience.iop.org/article/10.3847/0004-637X/822/1/52`

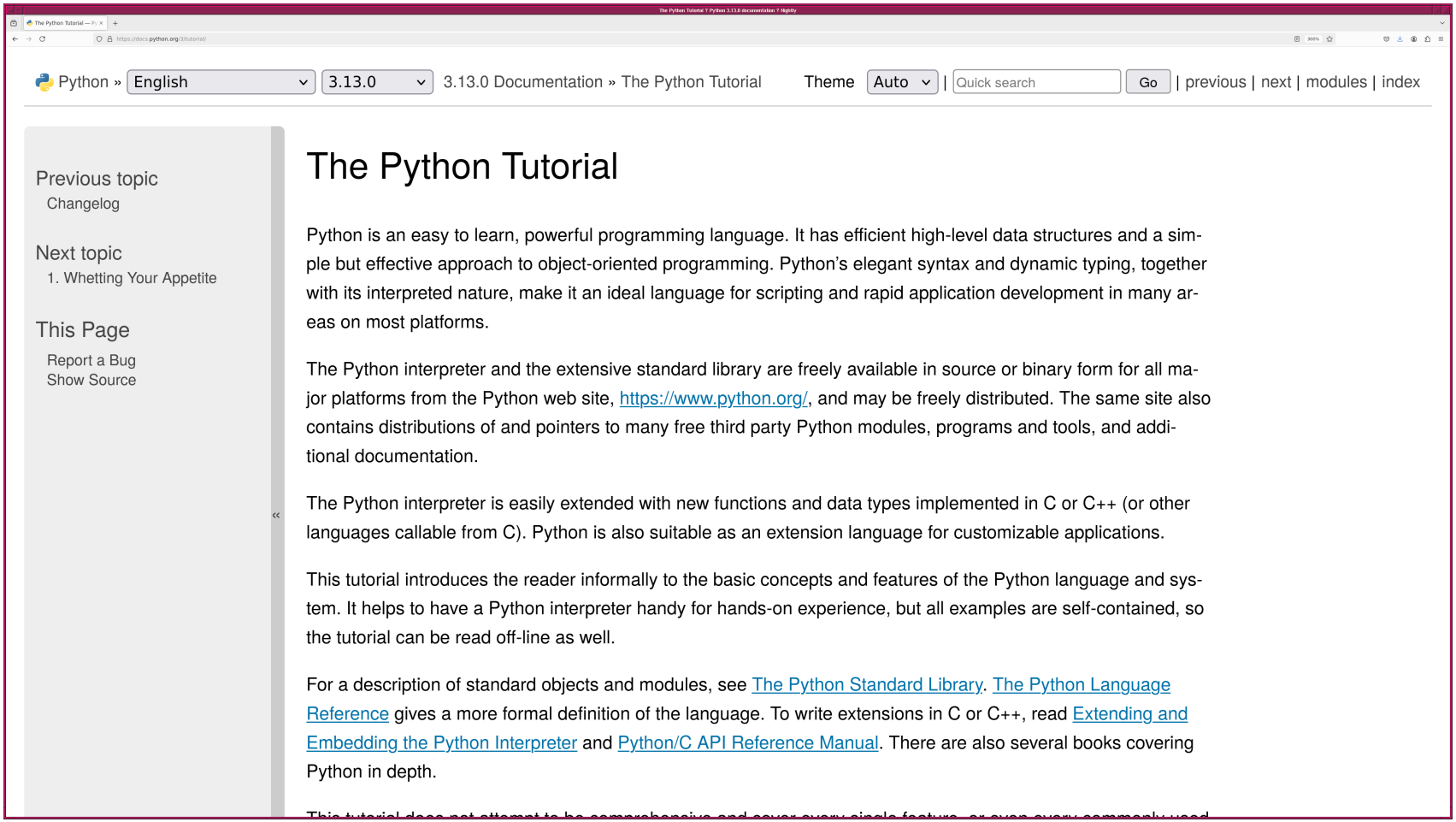
Exercise s10-04

- Make your own Python script to open the data file "`table1.dat.gz`", extract galactic longitude (GLON), galactic latitude (GLAT), distance flag (INF), near kinematic distance (Dnear), and far kinematic distance (Dfar) for each molecular cloud in the catalogue, check the distance flag and find more likely distance (Dnear or Dfar), calculate galactocentric coordinate (x_{gal}, y_{gal}, z_{gal}), and then visualise locations of molecular clouds on (x_{gal}, y_{gal}) plane using Matplotlib.
  - Assume 8.34 kpc for the distance of the Sun from the Galactic centre.
  - Assume 25 pc for the distance of the Sun from the Galactic mid-plane.
  - For the calculation of galactocentric coordinates, read the Section 2.4.1 "Physical Properties" of the paper "A UNIFORM CATALOG OF MOLECULAR CLOUDS IN THE MILKY WAY".
    - `https://iopscience.iop.org/article/10.3847/0004-637X/822/1/52`

Next meeting

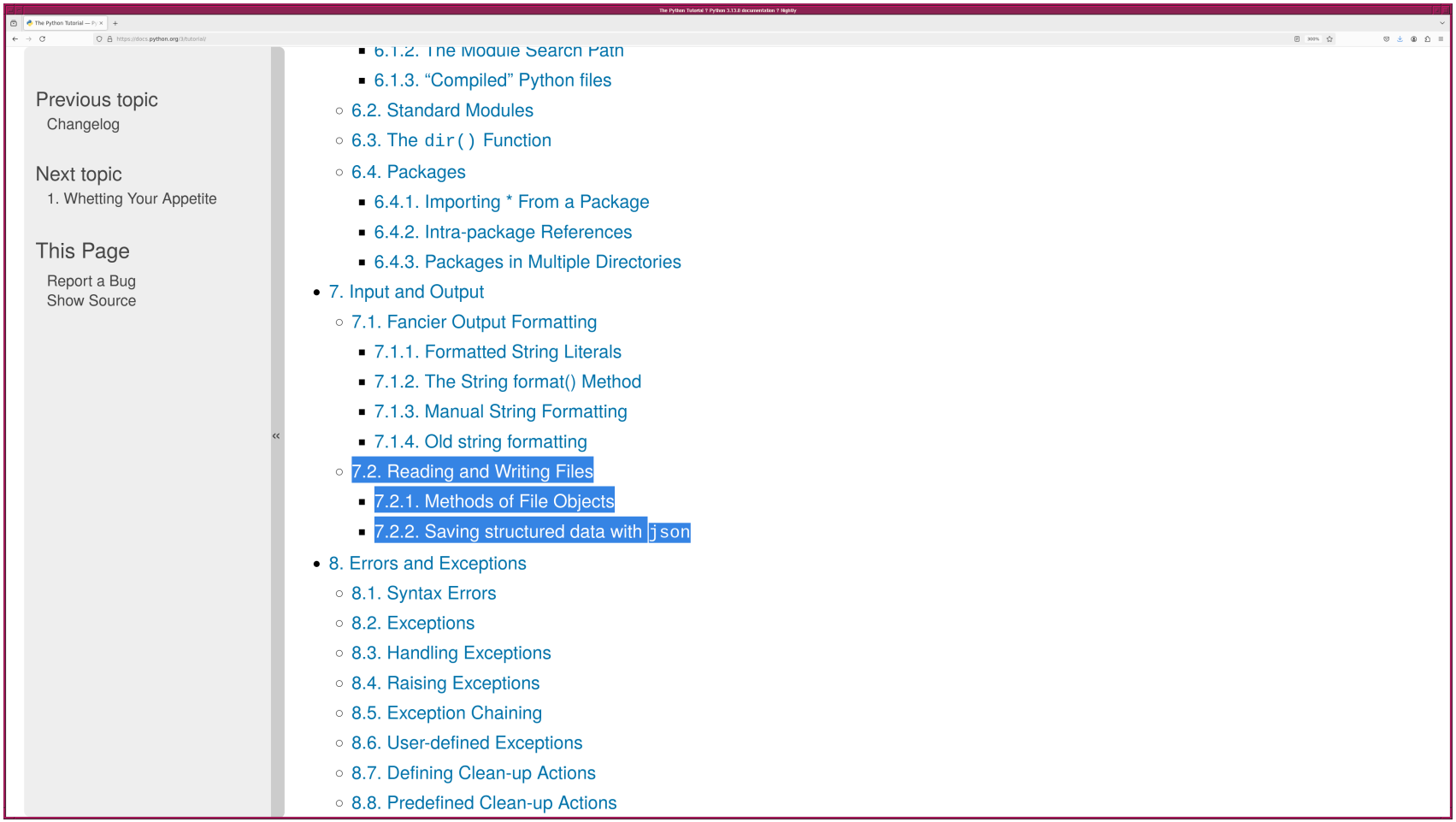
- We will have next meeting on 15 May 2025.
  - Google Meet: `https://meet.google.com/wpx-yabb-tka`
- We start the activity at 20:00.

"The Python Tutorial"



https://docs.python.org/3/tutorial/

"The Python Tutorial"



https://docs.python.org/3/tutorial/

"The Python Tutorial"

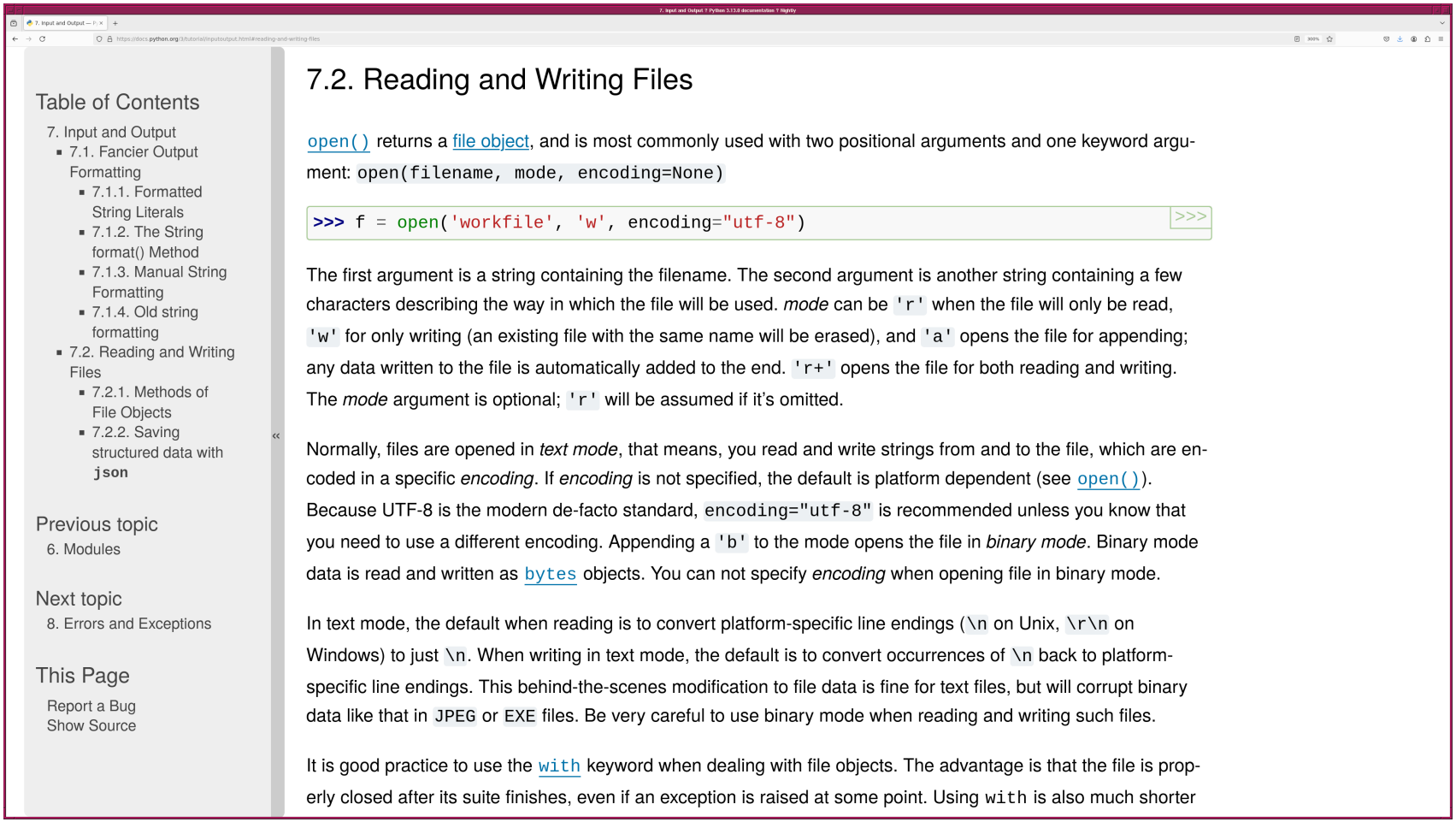


Table of Contents

7. Input and Output
   - 7.1. Fancier Output Formatting
     - 7.1.1. Formatted String Literals
     - 7.1.2. The String format() Method
     - 7.1.3. Manual String Formatting
     - 7.1.4. Old string formatting
   - 7.2. Reading and Writing Files
     - 7.2.1. Methods of File Objects
     - 7.2.2. Saving structured data with `json`

Previous topic
6. Modules

Next topic
8. Errors and Exceptions

This Page
Report a Bug
Show Source

7.2. Reading and Writing Files

`open()` returns a file object, and is most commonly used with two positional arguments and one keyword argument: `open(filename, mode, encoding=None)`

```
>>> f = open('workfile', 'w', encoding="utf-8")
```

The first argument is a string containing the filename. The second argument is another string containing a few characters describing the way in which the file will be used. *mode* can be `'r'` when the file will only be read, `'w'` for only writing (an existing file with the same name will be erased), and `'a'` opens the file for appending; any data written to the file is automatically added to the end. `'r+'` opens the file for both reading and writing. The *mode* argument is optional; `'r'` will be assumed if it's omitted.

Normally, files are opened in *text mode*, that means, you read and write strings from and to the file, which are encoded in a specific *encoding*. If *encoding* is not specified, the default is platform dependent (see `open()`). Because UTF-8 is the modern de-facto standard, `encoding="utf-8"` is recommended unless you know that you need to use a different encoding. Appending a `'b'` to the mode opens the file in *binary mode*. Binary mode data is read and written as `bytes` objects. You can not specify *encoding* when opening file in binary mode.

In text mode, the default when reading is to convert platform-specific line endings (`\n` on Unix, `\r\n` on Windows) to just `\n`. When writing in text mode, the default is to convert occurrences of `\n` back to platform-specific line endings. This behind-the-scenes modification to file data is fine for text files, but will corrupt binary data like that in `JPEG` or `EXE` files. Be very careful to use binary mode when reading and writing such files.

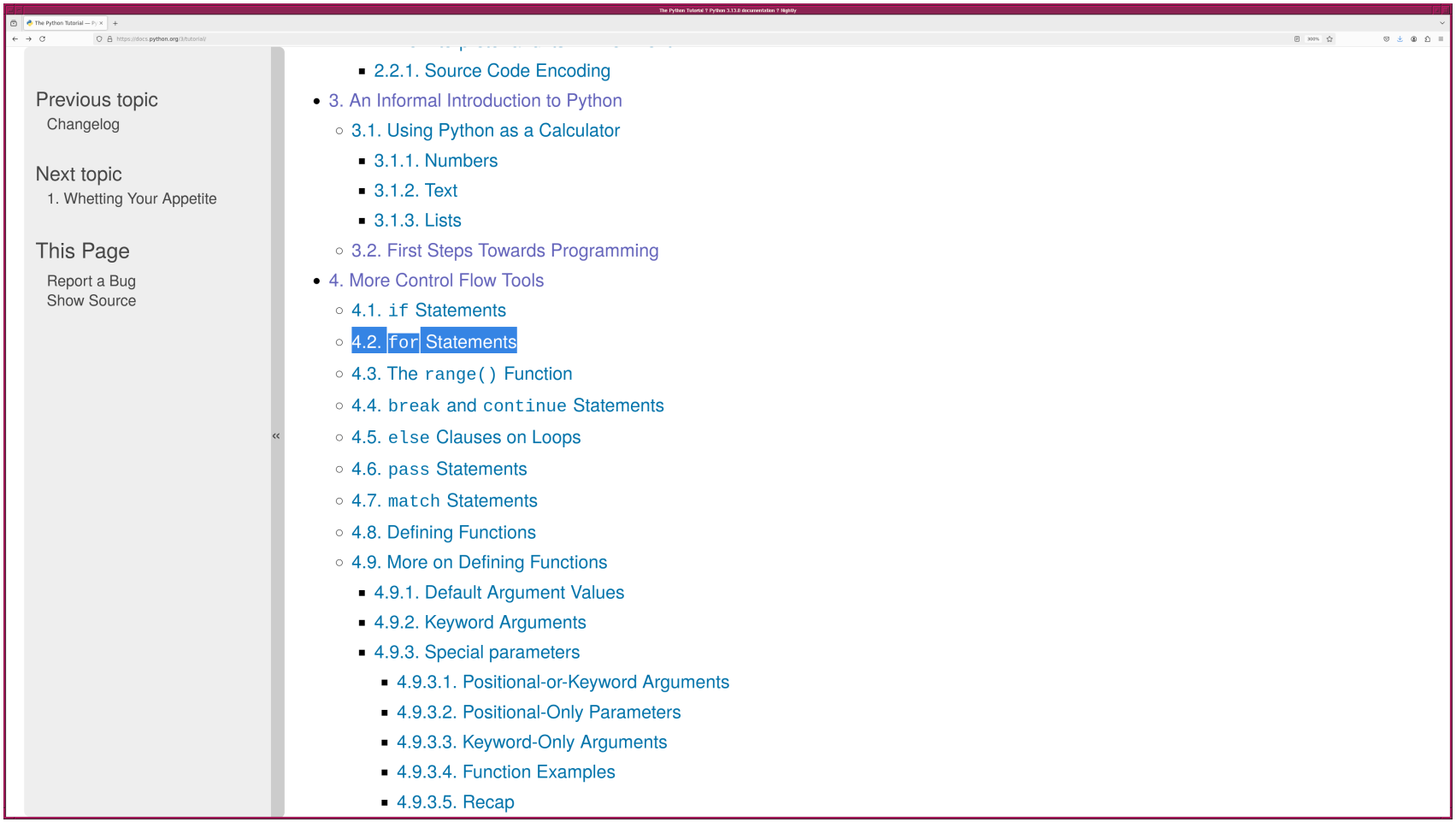
It is good practice to use the `with` keyword when dealing with file objects. The advantage is that the file is properly closed after its suite finishes, even if an exception is raised at some point. Using `with` is also much shorter

https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files

"The Python Tutorial"

- To learn about the "`for`" statement, read the section 4.2 "for Statements".
  - `https://docs.python.org/3/tutorial/controlflow.html#for-statements`

"The Python Tutorial"



https://docs.python.org/3/tutorial/

"The Python Tutorial"

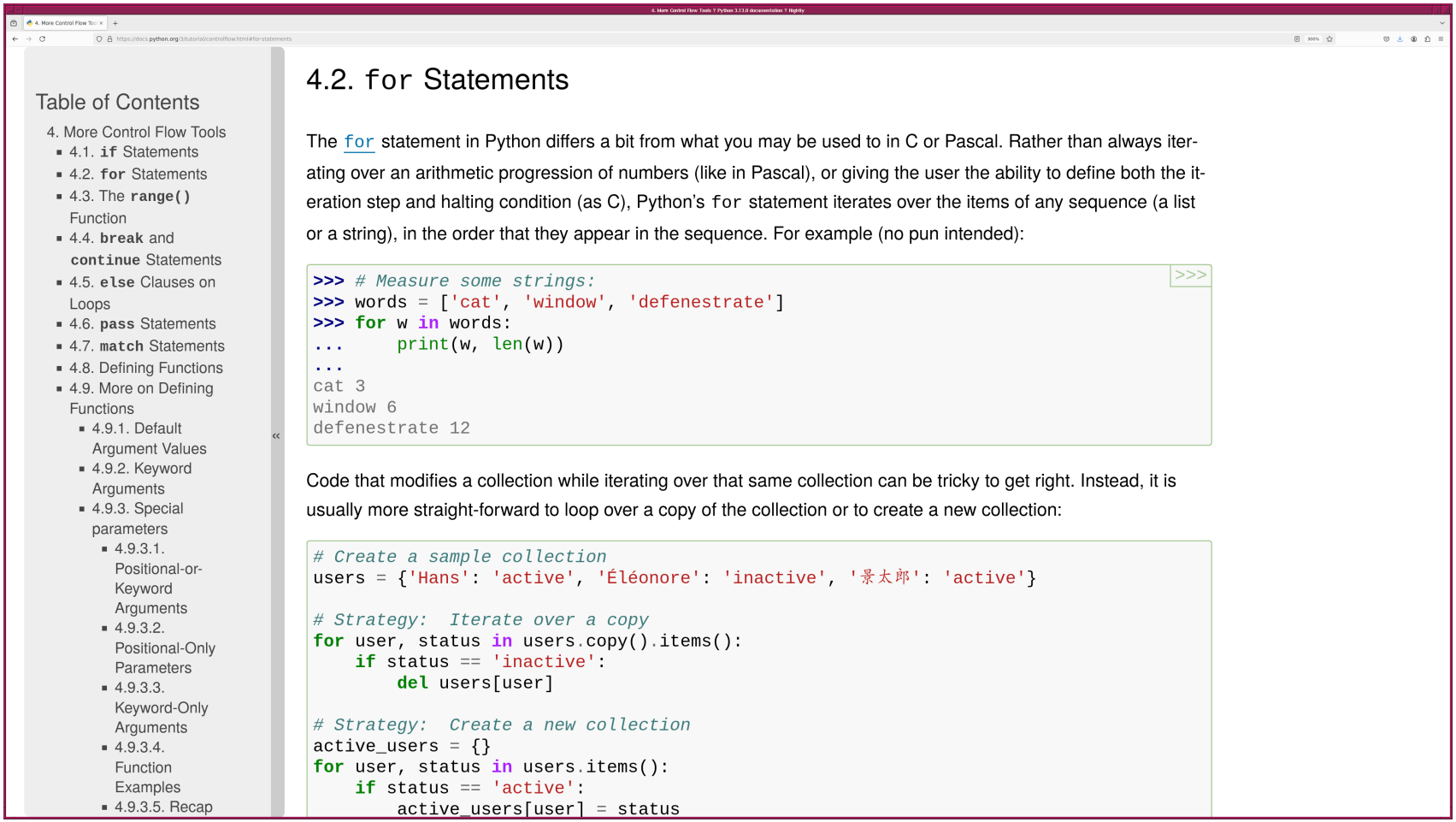


Table of Contents

4. More Control Flow Tools
- 4.1. if Statements
- 4.2. for Statements
- 4.3. The range() Function
- 4.4. break and continue Statements
- 4.5. else Clauses on Loops
- 4.6. pass Statements
- 4.7. match Statements
- 4.8. Defining Functions
- 4.9. More on Defining Functions
  - 4.9.1. Default Argument Values
  - 4.9.2. Keyword Arguments
  - 4.9.3. Special parameters
    - 4.9.3.1. Positional-or-Keyword Arguments
    - 4.9.3.2. Positional-Only Parameters
    - 4.9.3.3. Keyword-Only Arguments
    - 4.9.3.4. Function Examples
    - 4.9.3.5. Recap

4.2. for Statements

The for statement in Python differs a bit from what you may be used to in C or Pascal. Rather than always iterating over an arithmetic progression of numbers (like in Pascal), or giving the user the ability to define both the iteration step and halting condition (as C), Python's for statement iterates over the items of any sequence (a list or a string), in the order that they appear in the sequence. For example (no pun intended):

```
>>> # Measure some strings:
>>> words = ['cat', 'window', 'defenestrate']
>>> for w in words:
...     print(w, len(w))
...
cat 3
window 6
defenestrate 12
```

Code that modifies a collection while iterating over that same collection can be tricky to get right. Instead, it is usually more straight-forward to loop over a copy of the collection or to create a new collection:

```
# Create a sample collection
users = {'Hans': 'active', 'Éléonore': 'inactive', '景太郎': 'active'}

# Strategy:  Iterate over a copy
for user, status in users.copy().items():
    if status == 'inactive':
        del users[user]

# Strategy:  Create a new collection
active_users = {}
for user, status in users.items():
    if status == 'active':
        active_users[user] = status
```

https://docs.python.org/3/tutorial/controlflow.html#for-statements

"The Python Tutorial"

- To learn about the "`while`" statement, read the section 3.2 "First Steps Towards Programming".
  - `https://docs.python.org/3/tutorial/introduction.html#first-steps-towards-programming`

"The Python Tutorial"

https://docs.python.org/3/tutorial/

"The Python Tutorial"

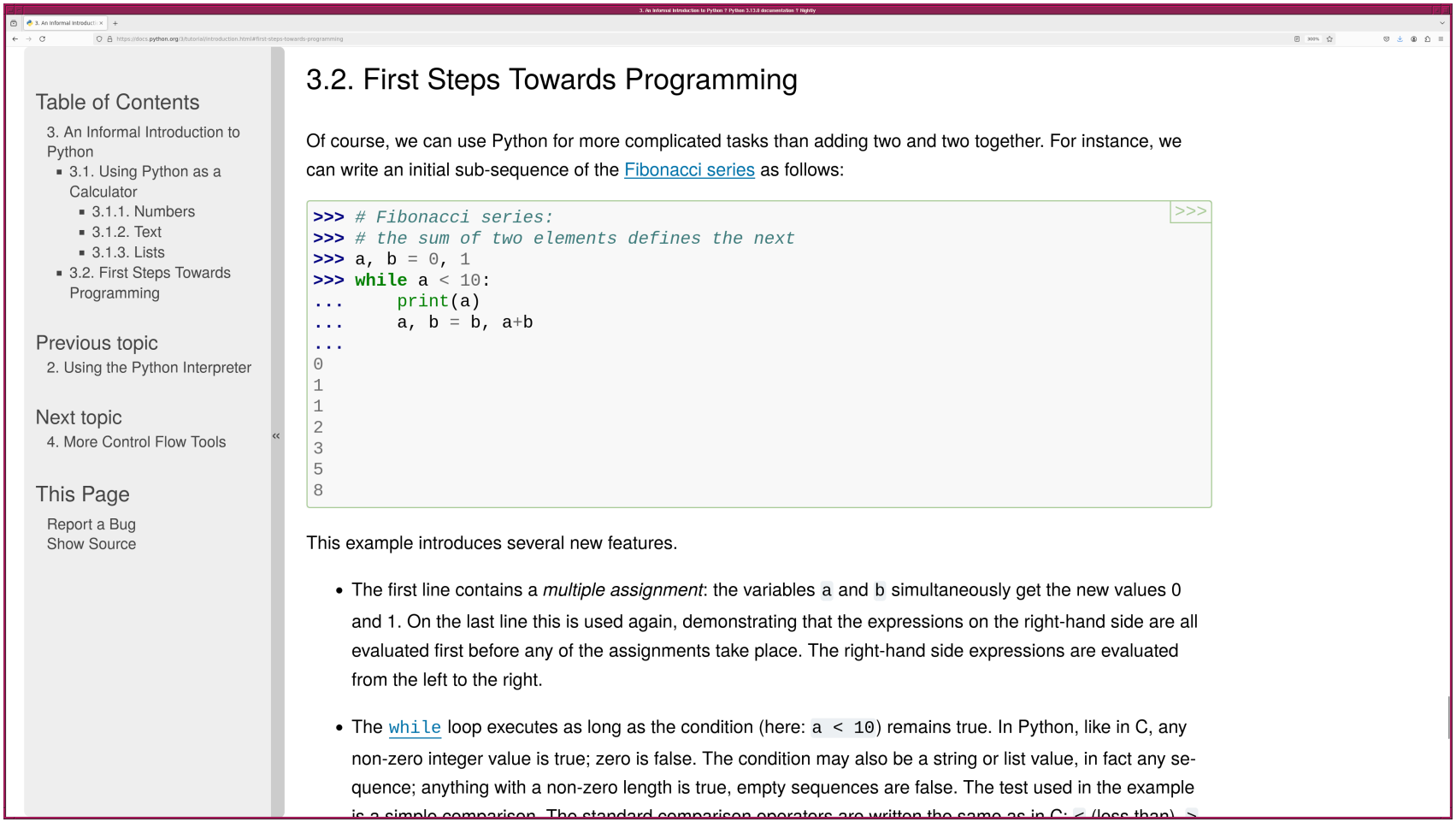


Table of Contents

3. An Informal Introduction to Python
   - 3.1. Using Python as a Calculator
     - 3.1.1. Numbers
     - 3.1.2. Text
     - 3.1.3. Lists
   - 3.2. First Steps Towards Programming

Previous topic
2. Using the Python Interpreter

Next topic
4. More Control Flow Tools

This Page
Report a Bug
Show Source

3.2. First Steps Towards Programming

Of course, we can use Python for more complicated tasks than adding two and two together. For instance, we can write an initial sub-sequence of the Fibonacci series as follows:

```
>>> # Fibonacci series:
>>> # the sum of two elements defines the next
>>> a, b = 0, 1
>>> while a < 10:
...     print(a)
...     a, b = b, a+b
...
0
1
1
2
3
5
8
```

This example introduces several new features.

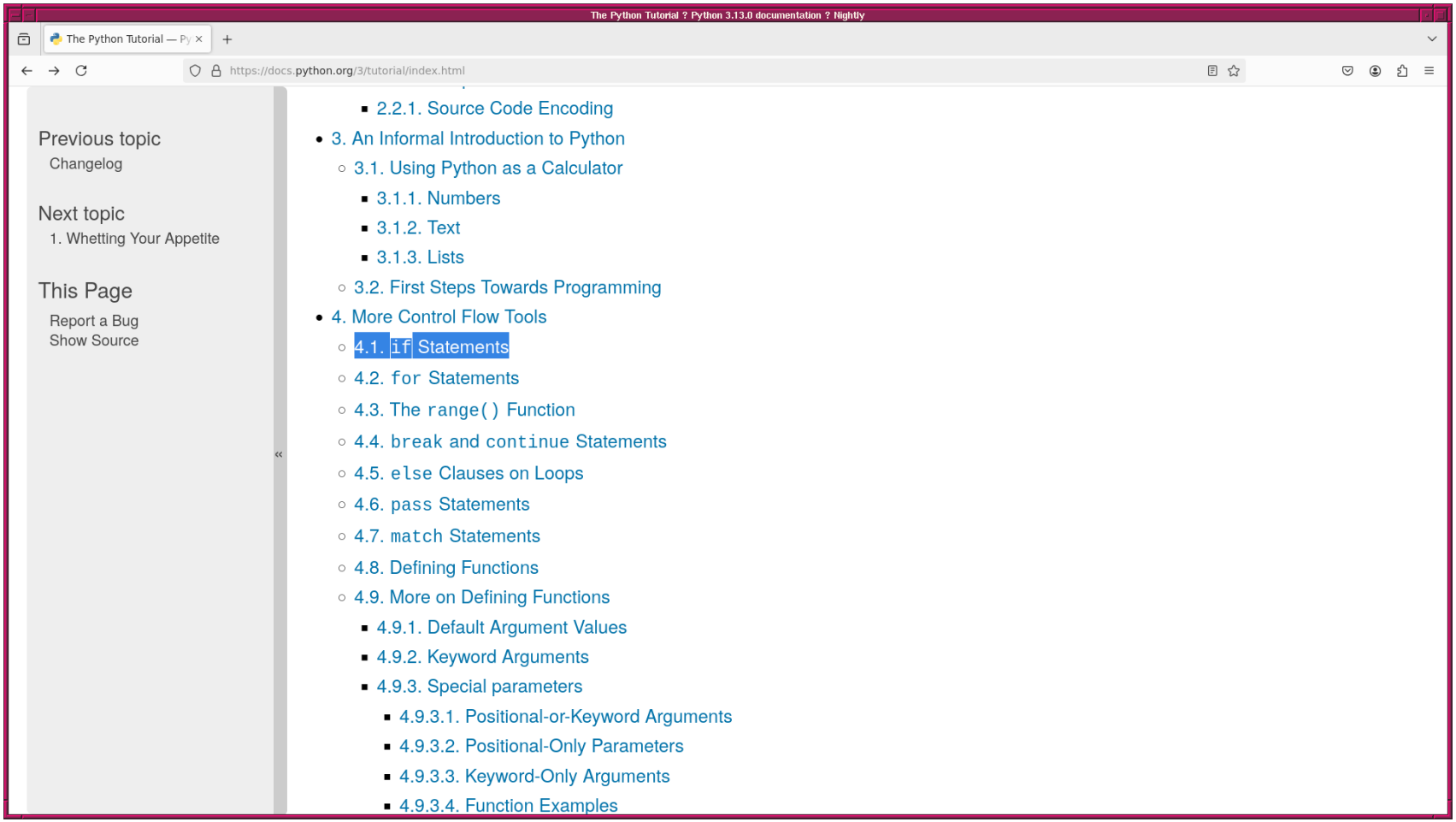
- The first line contains a *multiple assignment*: the variables `a` and `b` simultaneously get the new values 0 and 1. On the last line this is used again, demonstrating that the expressions on the right-hand side are all evaluated first before any of the assignments take place. The right-hand side expressions are evaluated from the left to the right.
- The `while` loop executes as long as the condition (here: `a < 10`) remains true. In Python, like in C, any non-zero integer value is true; zero is false. The condition may also be a string or list value, in fact any sequence; anything with a non-zero length is true, empty sequences are false. The test used in the example is a simple comparison. The standard comparison operators are written the same as in C: `<` (less than), `>`

https://docs.python.org/3/tutorial/introduction.html#first-steps-towards-programming

"The Python Tutorial"

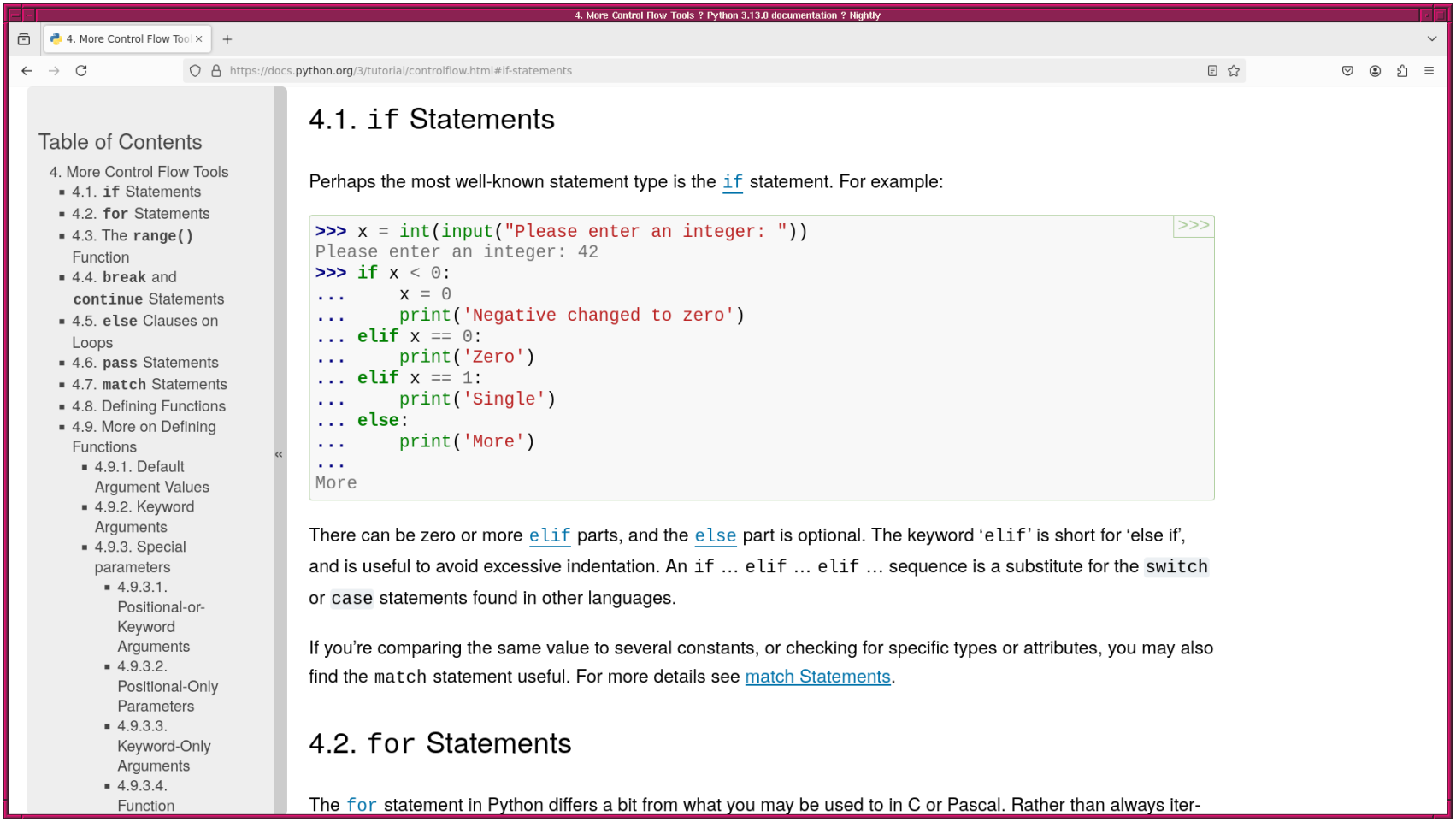
- To learn about the "`if`" statement, read the section 4.1 "`if` Statements".
  - `https://docs.python.org/3/tutorial/controlflow.html#if-statements`

"The Python Tutorial"



https://docs.python.org/3/tutorial/

"The Python Tutorial"

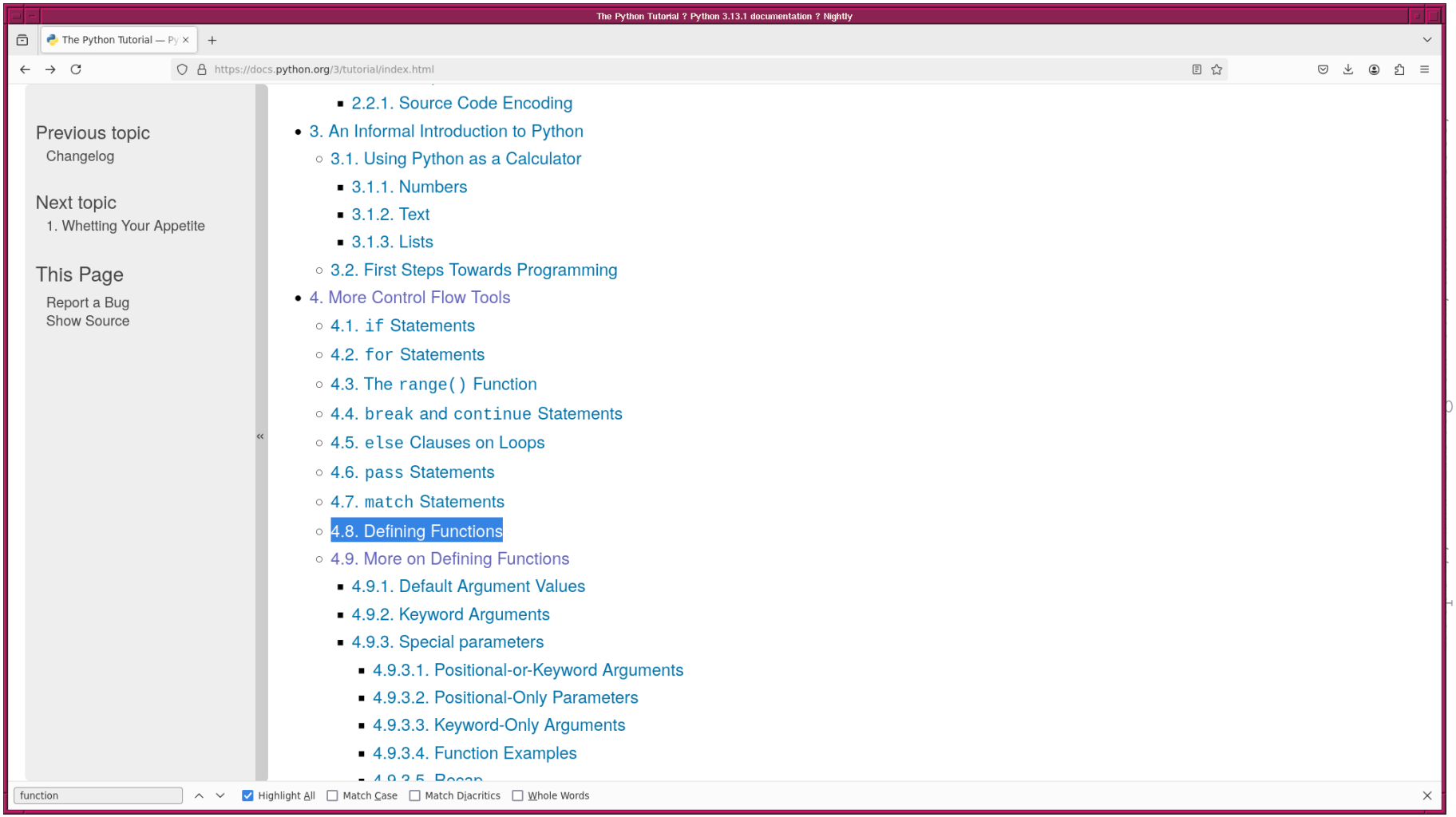


https://docs.python.org/3/tutorial/

"The Python Tutorial"

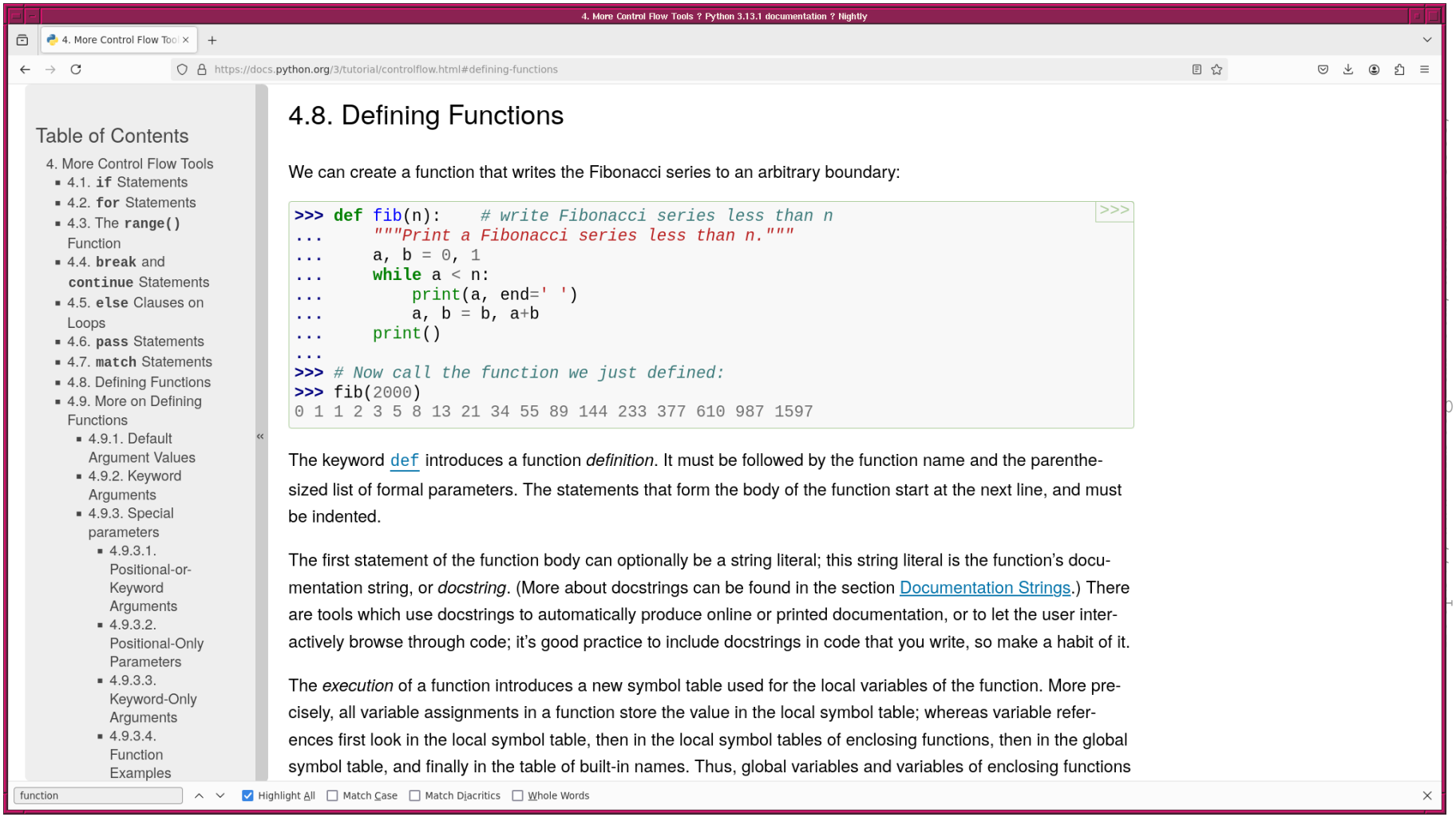
- To learn about the way to define a function, read the sections 4.8 "Defining Functions" and 4.9 "More on Defining Functions".
  - https://docs.python.org/3/tutorial/controlflow.html#defining-functions
  - https://docs.python.org/3/tutorial/controlflow.html#more-on-defining-functions

"The Python Tutorial"



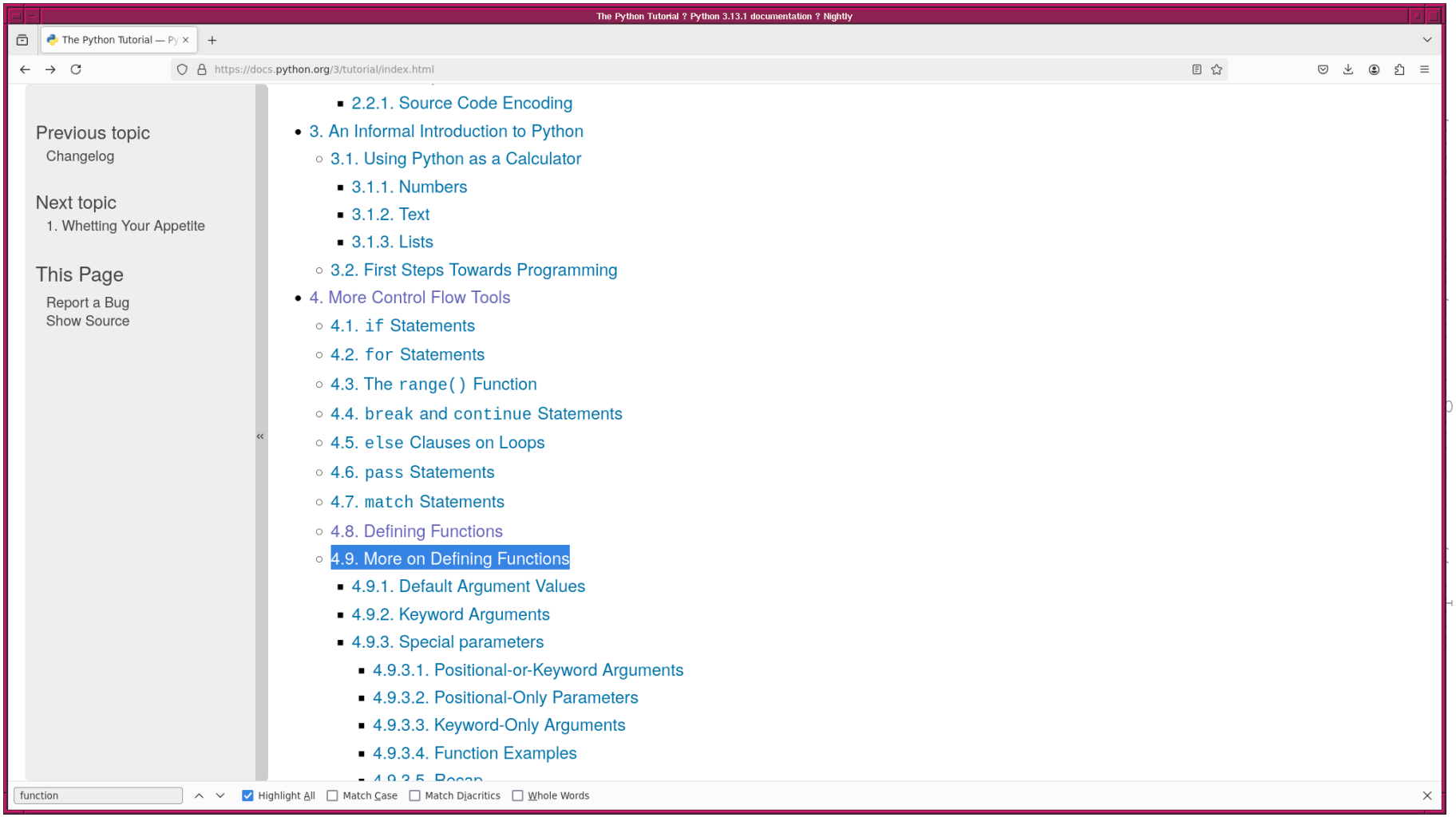
https://docs.python.org/3/tutorial/

"The Python Tutorial"



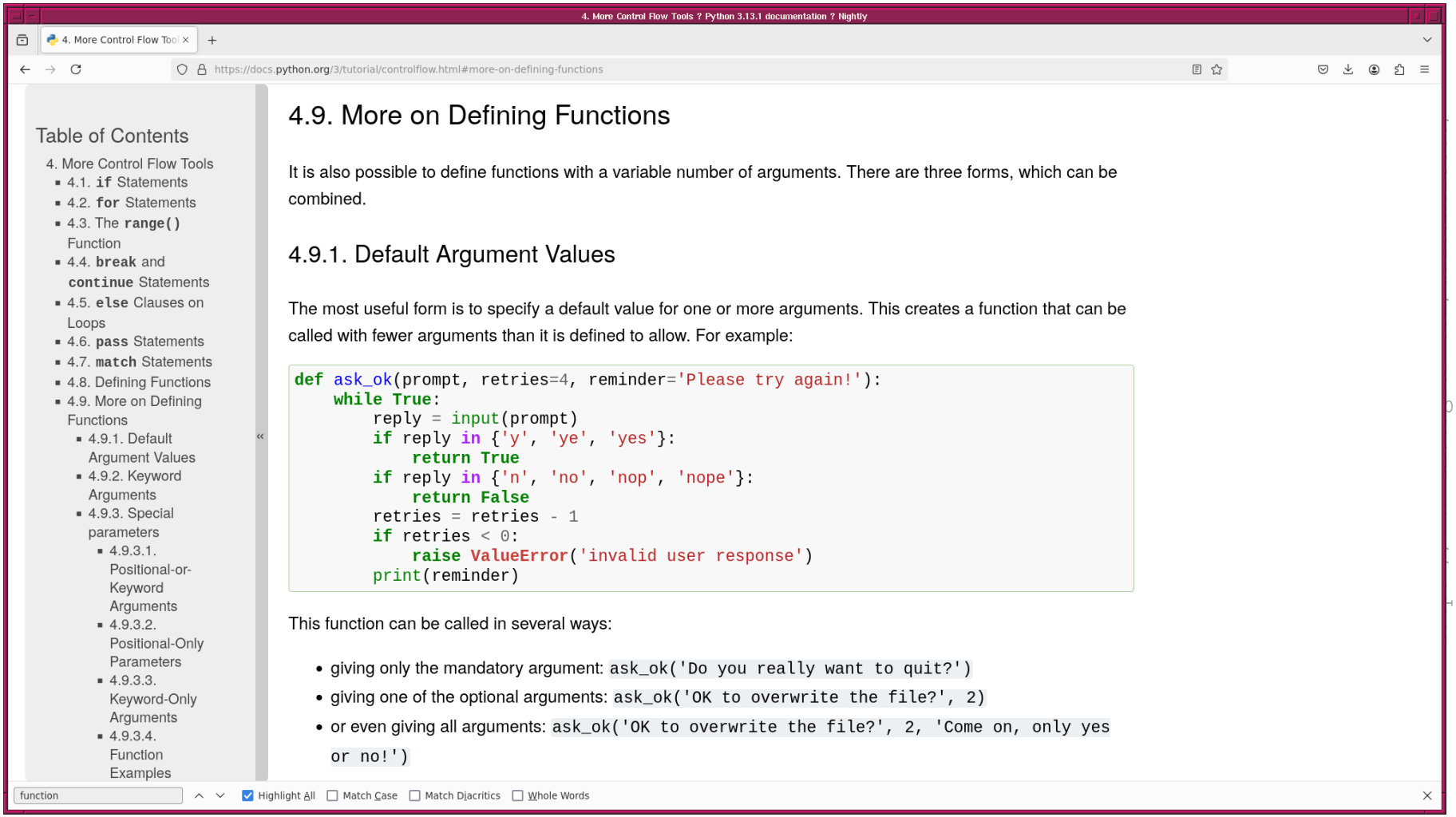
https://docs.python.org/3/tutorial/

"The Python Tutorial"



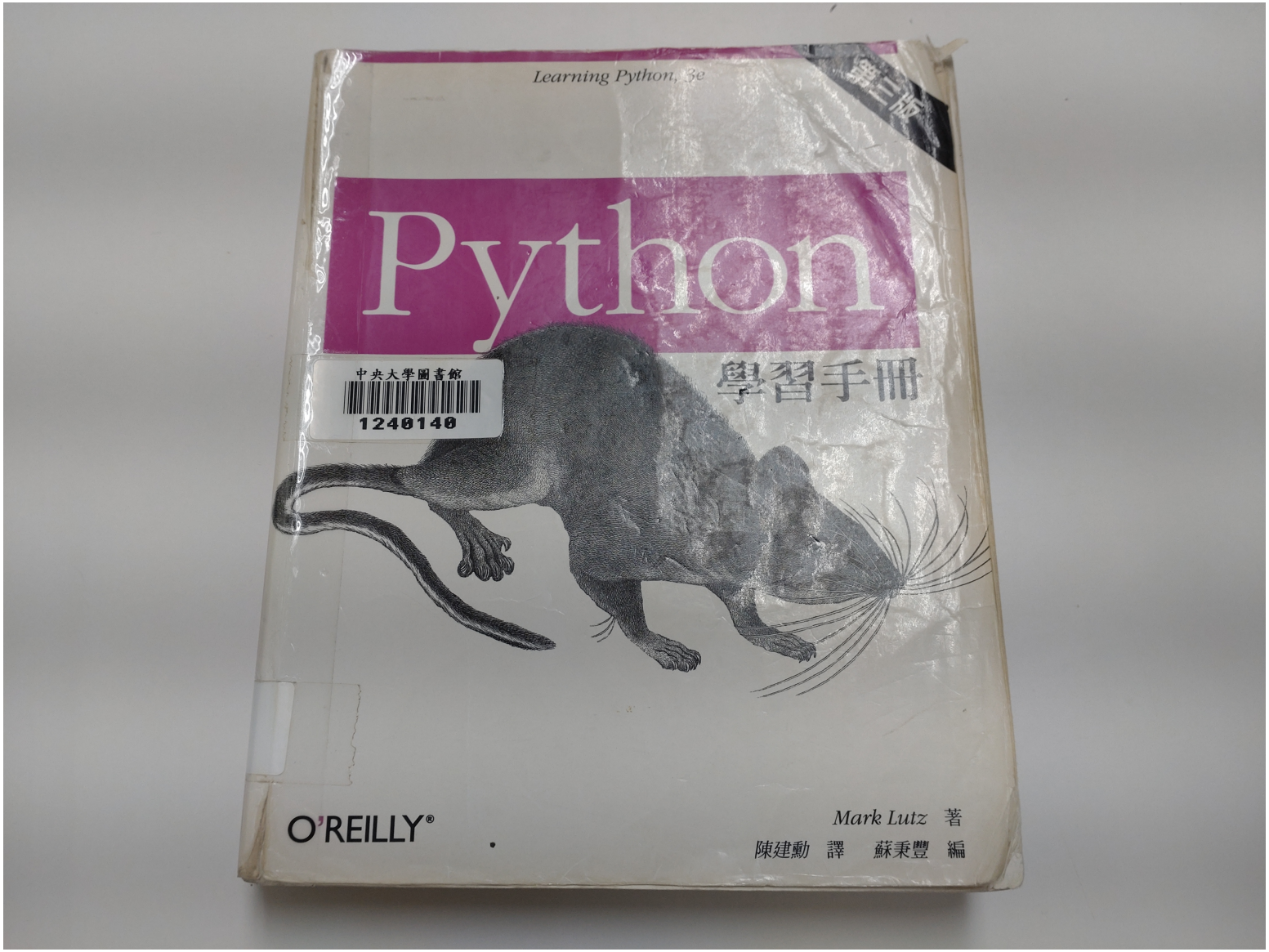
https://docs.python.org/3/tutorial/

"The Python Tutorial"

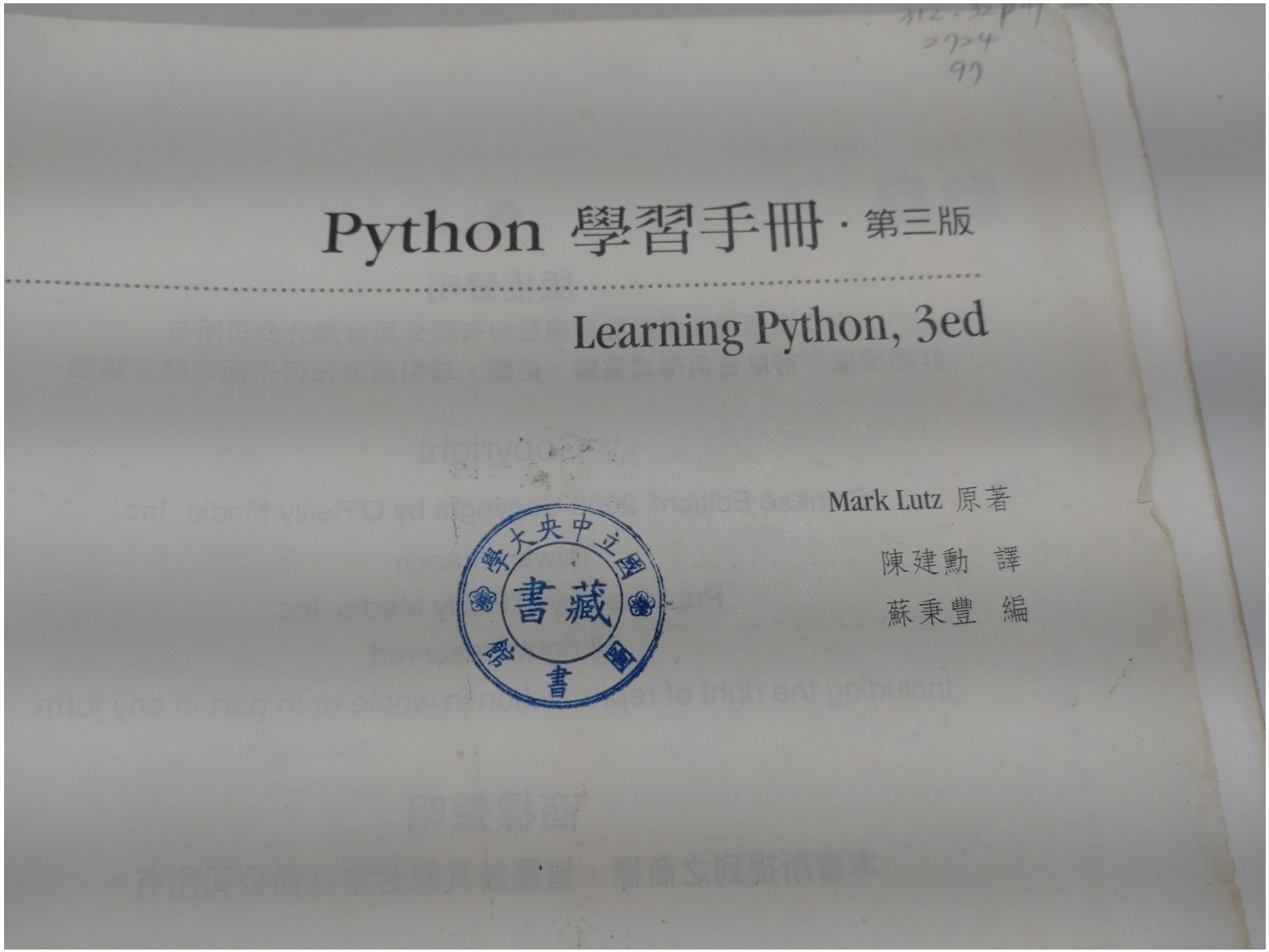


https://docs.python.org/3/tutorial/

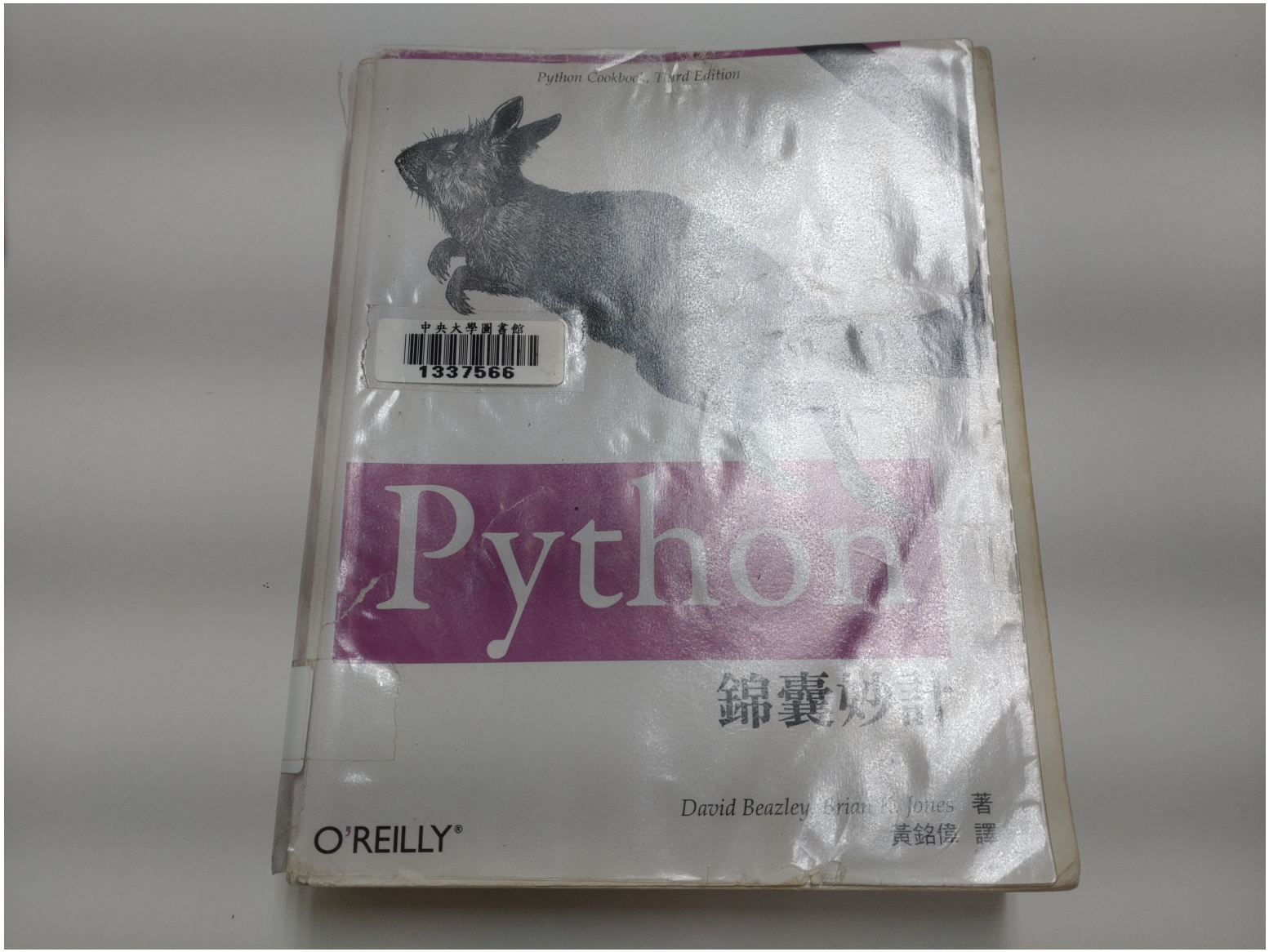
Recommended books



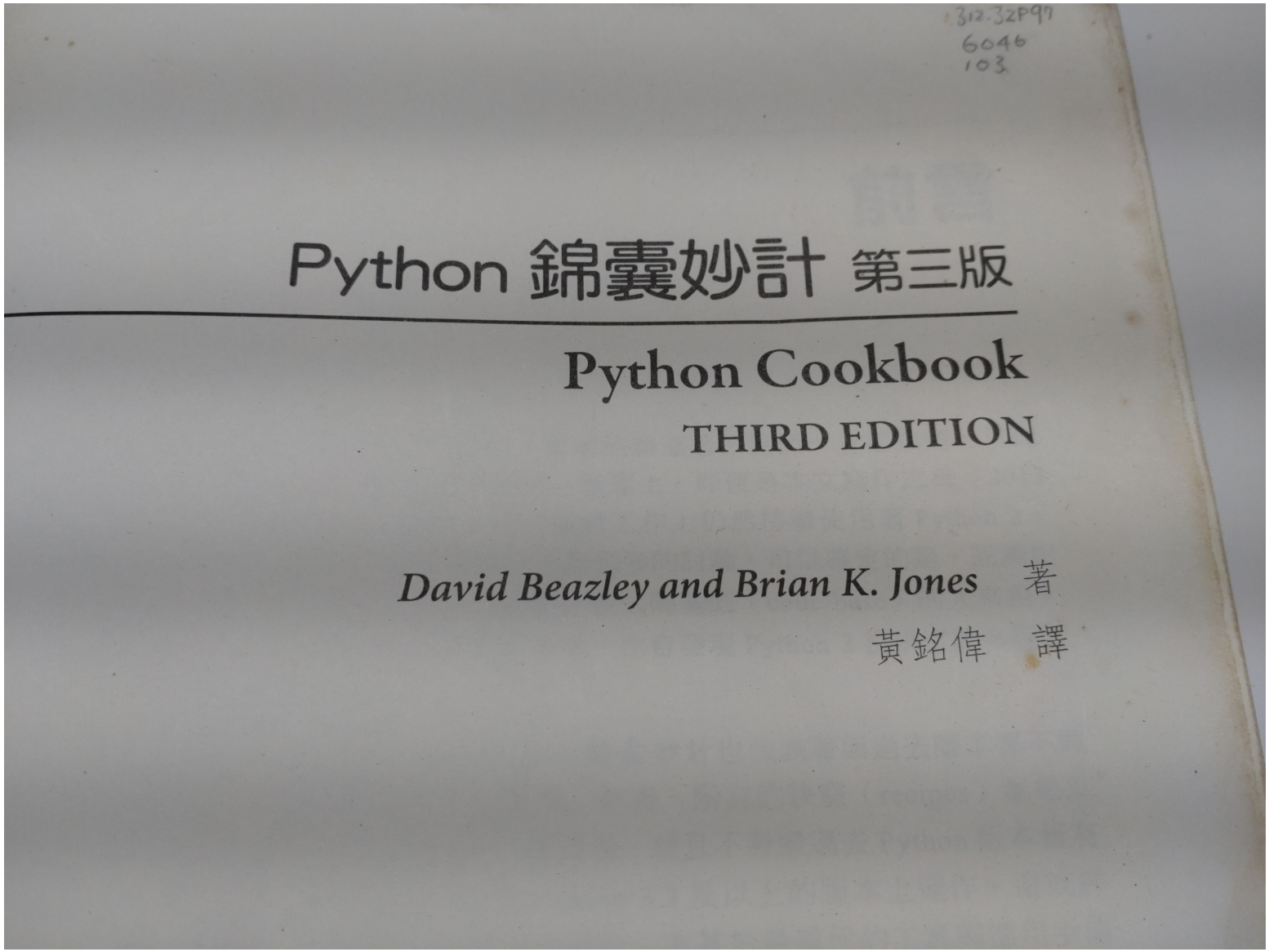
Recommended books



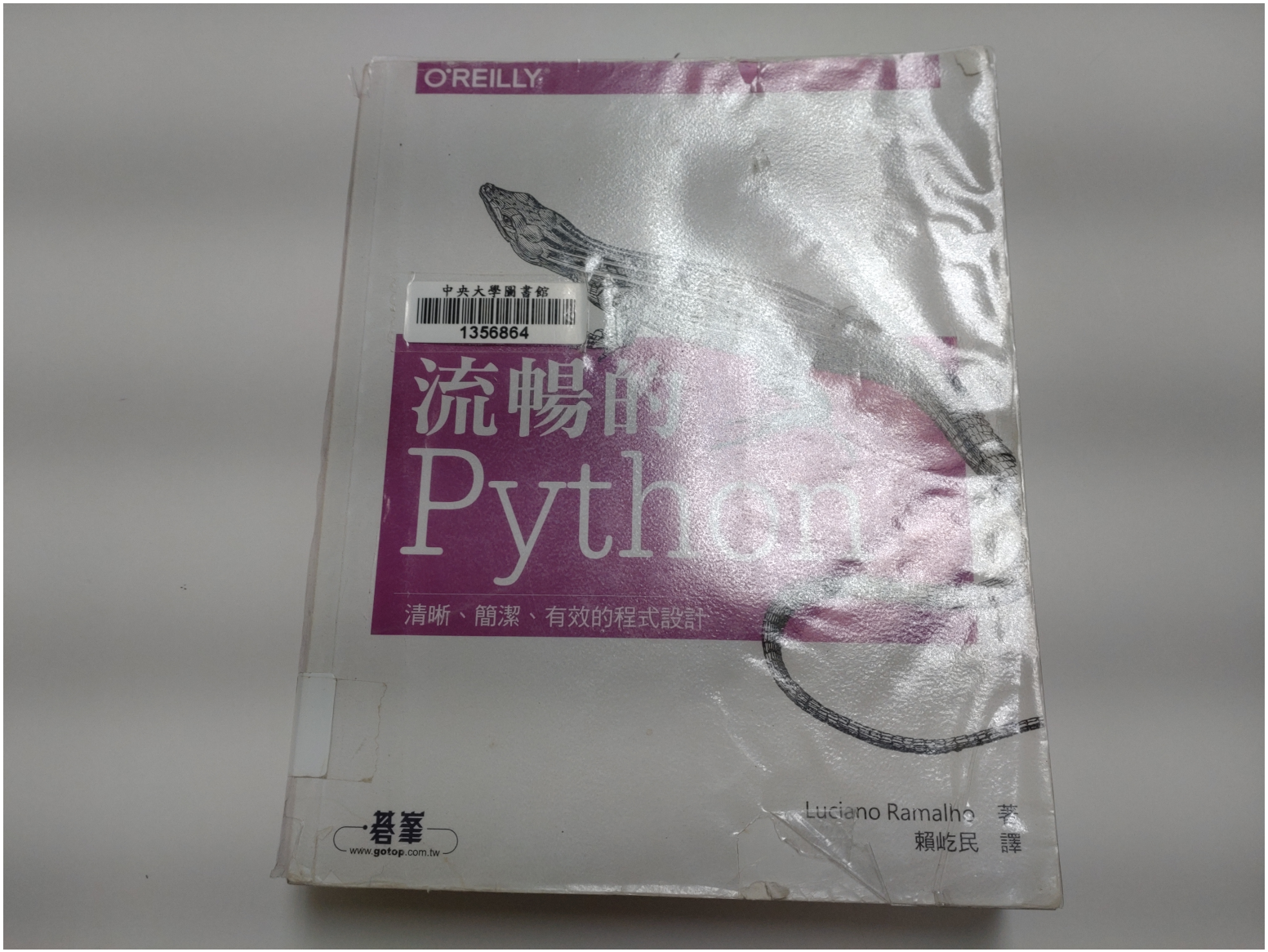
Recommended books



Recommended books



Recommended books



Recommended books