Astroinformatics 2022
Session 01: Basic Python Programming

Kinoshita Daisuke

12 September 2022
publicly accessible version

About this file...

- Important information about this file
  - The author of this file is Kinoshita Daisuke.
  - The original version of this file was used for the course "Astroinformatics" (course ID: AS6095) offered at Institute of Astronomy, National Central University from September 2022 to January 2023.
  - The file is provided in the hope that it will be useful, but there is no guarantee for the correctness. Use this file at your own risk.
  - If you are willing to use this file for your study, please feel free to use. I'll be very happy to receive feedback from you.
  - If you are willing to use this file for your teaching, please contact to Kinoshita Daisuke. When you use this file partly or entirely, please mention clearly that the author of the original version is Kinoshita Daisuke. Please help me to improve the contents of this file by sending your feedback.
  - Contact address: `https://www.instagram.com/daisuke23888/`

For this session, we study basic Python programming.

1 Sample Python scripts for this session

Sample Python scripts for this session can be downloaded from GitHub repository. Visit following GitHub repository.

- `https://github.com/kinoshitadaisuke/ncu_astroinformatics_202209`

1.1 Executing sample Python scripts on a terminal emulator

If you prefer to execute sample Python scripts for this session on a terminal emulator, download `.py` files from GitHub repository.

1.2 Executing sample Python scripts on JupyterLab

If you prefer to execute sample Python scripts for this session on JupyterLab (or Jupyter Notebook), download `.ipynb` file from GitHub repository.

1.3 Executing sample Python scripts using Binder

If you prefer to execute sample Python scripts for this session on Binder, visit following web page.

- `https://mybinder.org/v2/gh/kinoshitadaisuke/ncu_astroinformatics_202209/HEAD`

Start your favourite web browser and go to above web page. (Fig. 1) In a minute or two, you see JupyterLab working on your web browser. (Fig. 2) Go to the directory (folder) "`s01`". (Fig. 3) Choose the file "`ai202209_s01.ipynb`" (Fig. 4 and 5) and open it (Fig. 6).

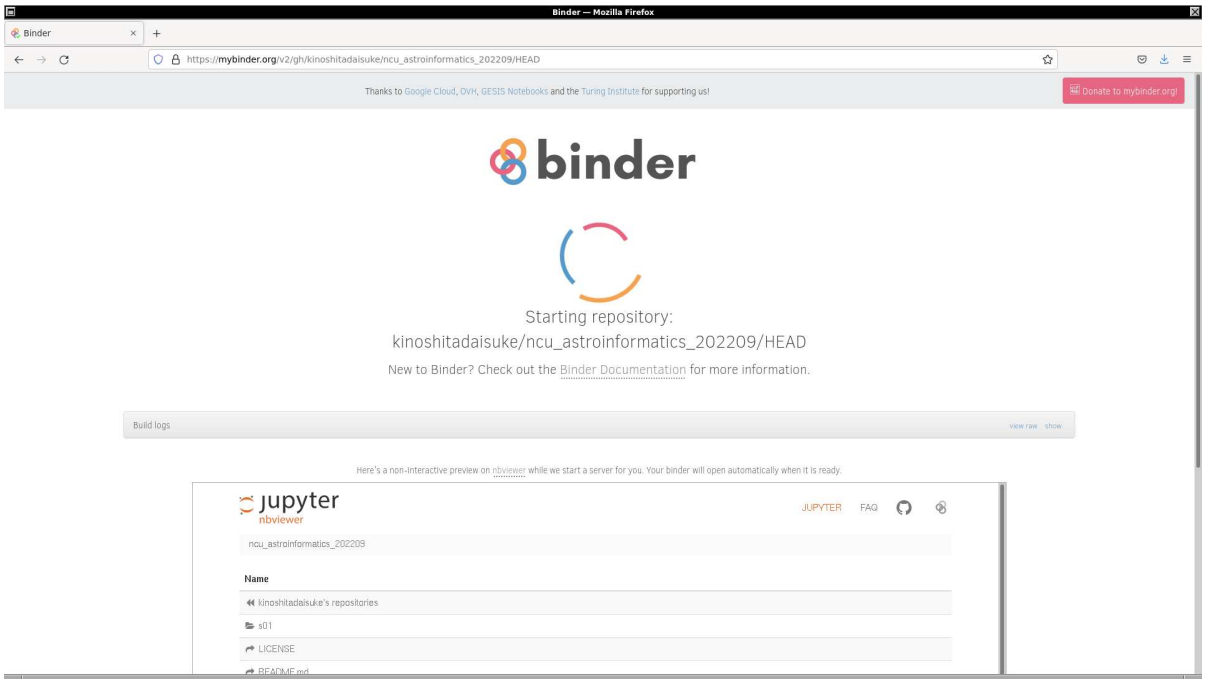

Figure 1: Using Binder to execute sample Python scripts for this session.

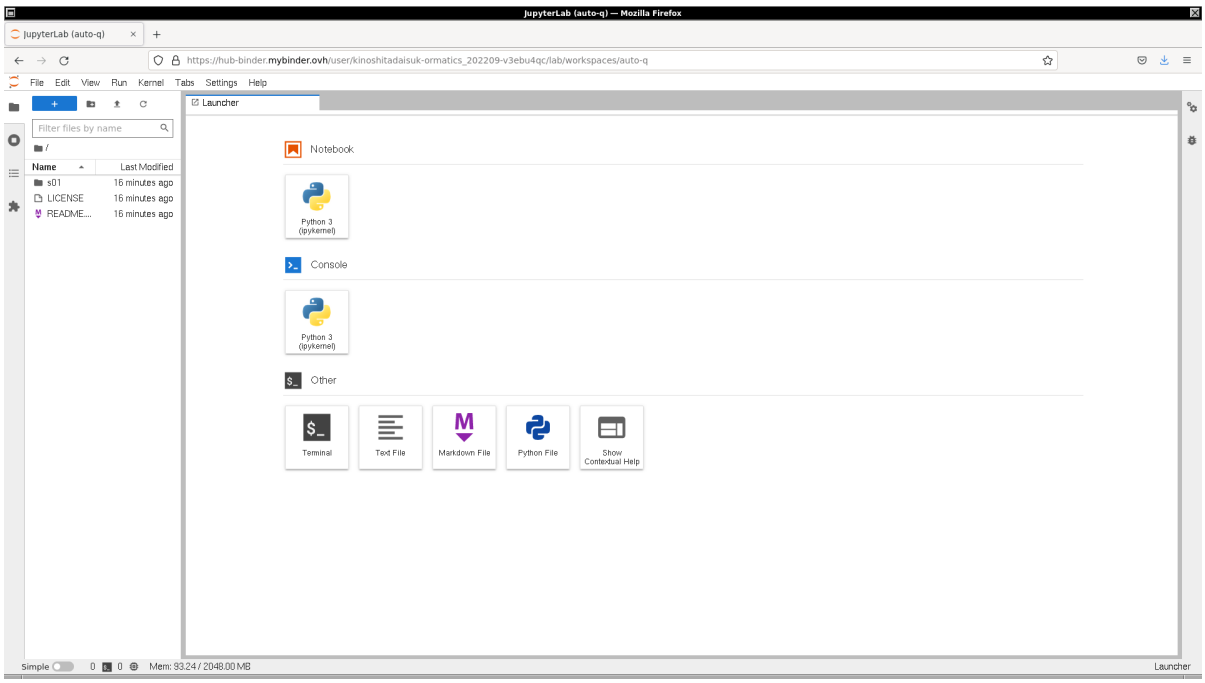

Figure 2: Using Binder to execute sample Python scripts for this session.

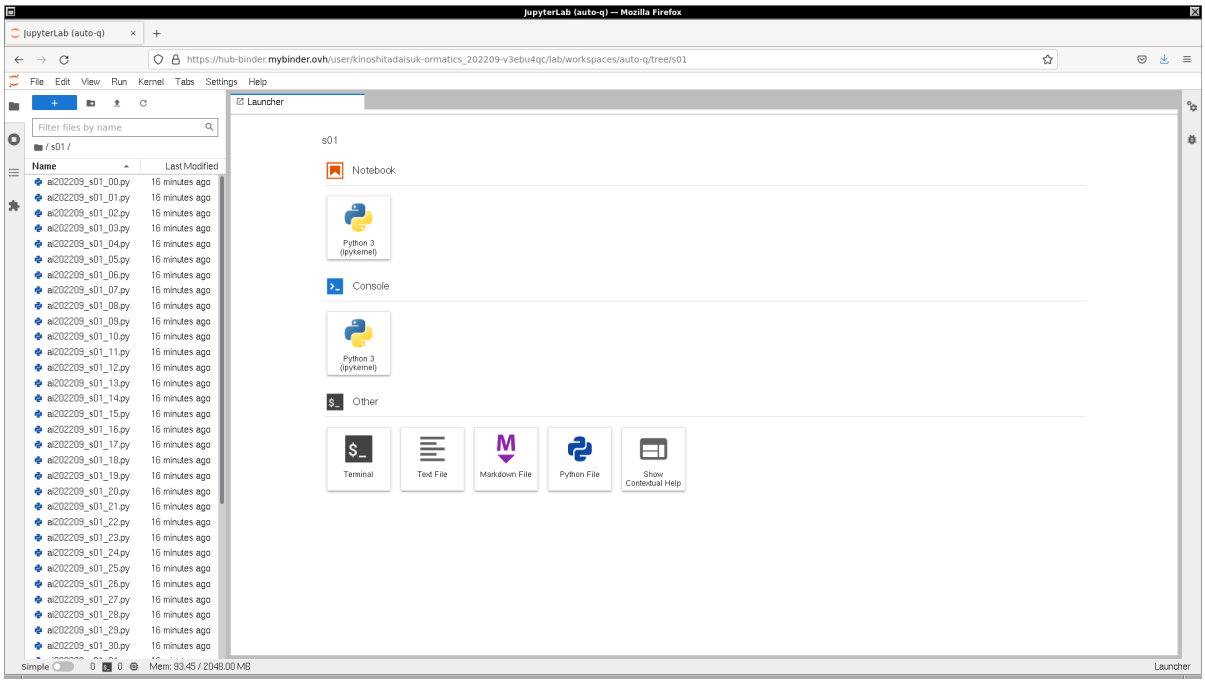

Figure 3: Using Binder to execute sample Python scripts for this session.

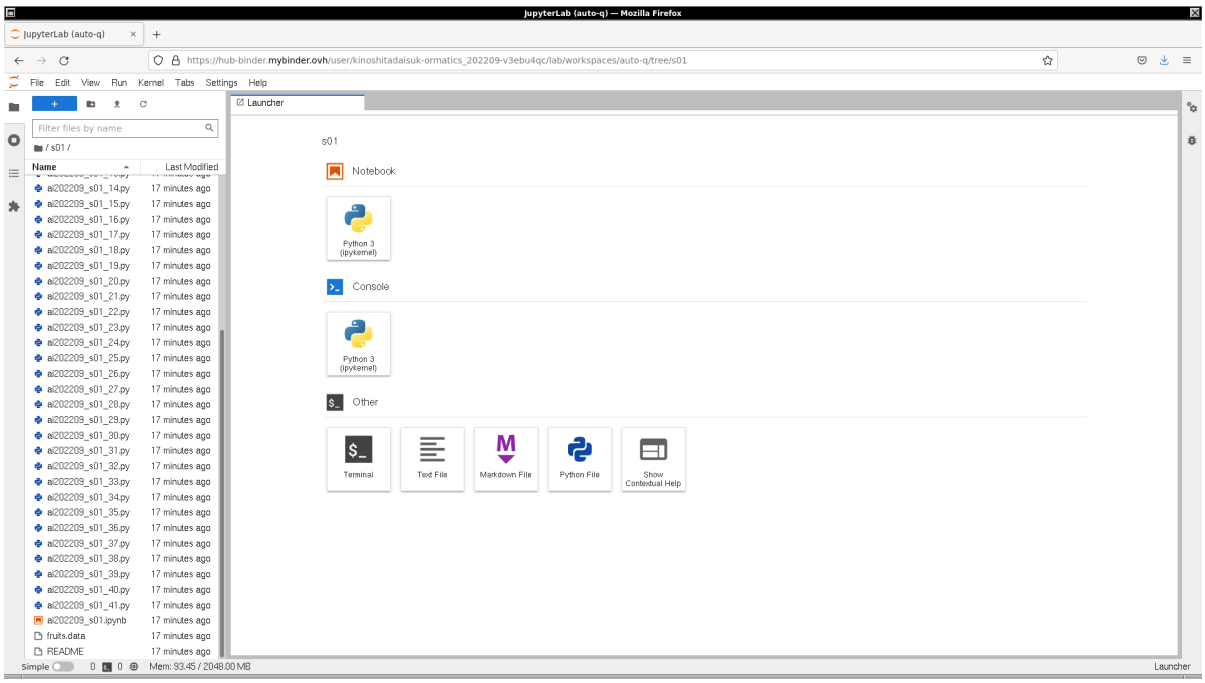

Figure 4: Using Binder to execute sample Python scripts for this session.

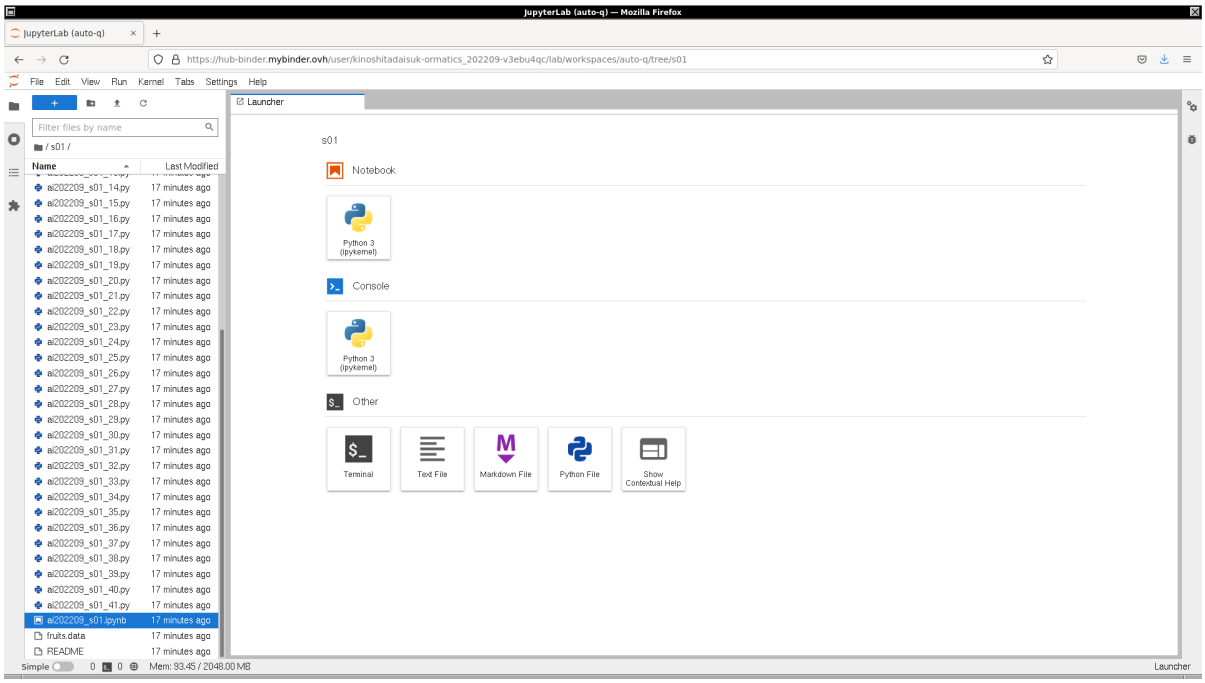

Figure 5: Using Binder to execute sample Python scripts for this session.

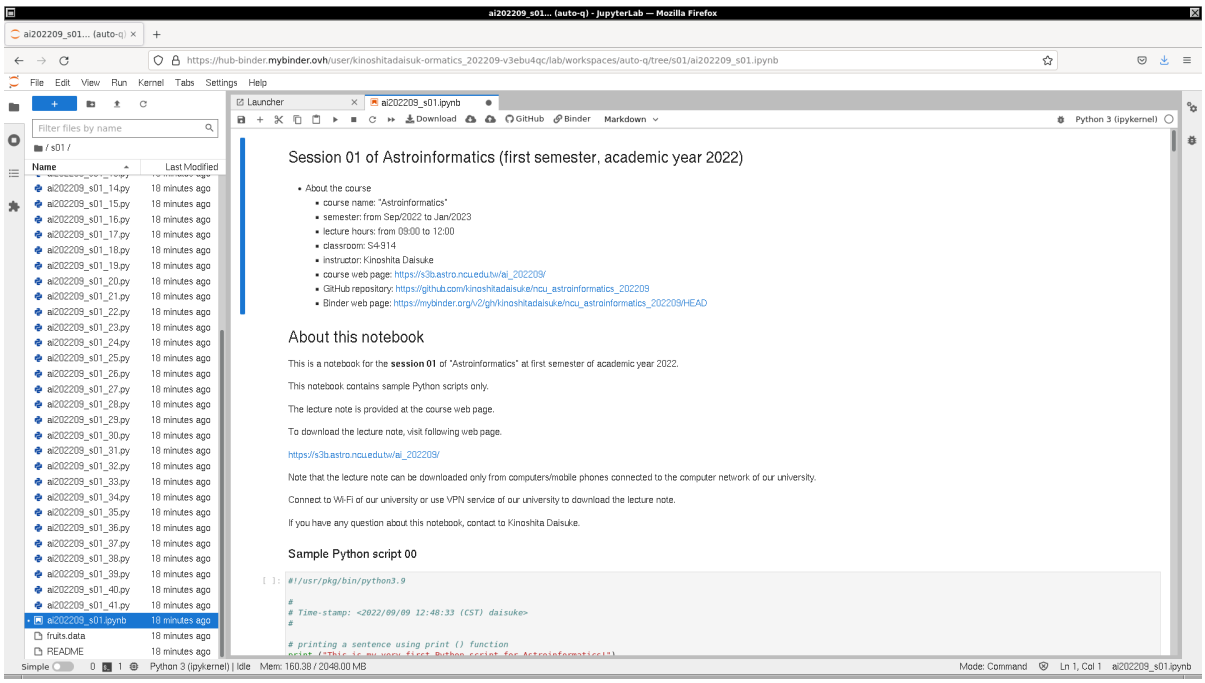

Figure 6: Using Binder to execute sample Python scripts for this session.

2 Printing information

2.1 Printing a sentence

Use Python to print a piece of sentence. Here is a sample script.

Python Code 1: ai202209_s01_00.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 12:48:33 (CST) daisuke>
#

# printing a sentence using print () function
print ("This is my very first Python script for Astroinformatics!")
```

Execute above script.

```
% chmod a+x ai202209_s01_00.py
% ./ai202209_s01_00.py
This is my very first Python script for Astroinformatics!
```

Try following practice.

Practice 01-01

Make your own Python script to print a sentence using "`print ()`" function.

2.2 Printing value of a variable

Make a Python script to print the value of a variable. Here is a sample script.

Python Code 2: ai202209_s01_01.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 13:43:08 (CST) daisuke>
#

# assignment of a variable
message = 'This is my second Python script for Astroinformatics!'

# printing the value of a variable using print () function
print (message)
```

Execute above script.

```
% chmod a+x ai202209_s01_01.py
% ./ai202209_s01_01.py
This is my second Python script for Astroinformatics!
```

Try following practice.

Practice 01-02

Make your own Python script to set a variable using assignment statement and print the value of the variable.

Here is one more example.

Python Code 3: ai202209_s01_02.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 13:53:00 (CST) daisuke>
#

# assignment of a variable
place = 'Taiwan'

# printing the value of a variable using print () function
print ('I live in', place, 'now.')
```

Execute above script.

```
% chmod a+x ai202209_s01_02.py
% ./ai202209_s01_02.py
I live in Taiwan now.
```

Try following practice.

Practice 01-03

Modify Python script "ai202209_s01_02.py" to print your birth place.

2.3 Printing a number

Make a Python script to print a number.

Python Code 4: ai202209_s01_03.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 14:06:04 (CST) daisuke>
#

# assignment of a variable
a = 123.45

# printing the value of a variable using print () function
print ('a =', a)
```

Execute above script.

```
% chmod a+x ai202209_s01_03.py
% ./ai202209_s01_03.py
a = 123.45
```

Try one more example.

Python Code 5: ai202209_s01_04.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 14:08:27 (CST) daisuke>
#
```

```
# assignment of a variable
a = 123.45

# printing the value of a variable using print () function
print ('a = %f' % a)
```

Execute above script.

```
% chmod a+x ai202209_s01_04.py
% ./ai202209_s01_04.py
a = 123.450000
```

Values of two or more variables can be printed using "`print ()`" function. Try following.

Python Code 6: ai202209_s01_05.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 14:14:44 (CST) daisuke>
#

# assignment of variables
a = 1.2
b = 3.4

# calculation
c = a * b

# printing the value of a variable using print () function
print ('%f * %f = %f' % (a, b, c) )
```

Execute above script.

```
% chmod a+x ai202209_s01_05.py
% ./ai202209_s01_05.py
1.200000 * 3.400000 = 4.080000
```

To learn about `printf`-style string format, visit following web page. (Fig. 7 and 8)

- `https://docs.python.org/3/library/stdtypes.html#old-string-formatting`

Try following practice.

Practice 01-04

Make a Python script to print values of solar mass, solar radius, and solar luminosity using printf-style string formatting.

2.4 Fancy formatting using formatted string literals

Try formatted string literals (f-strings) to print values of numbers. Here is an example.

Python Code 7: ai202209_s01_06.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 14:58:00 (CST) daisuke>
#
```

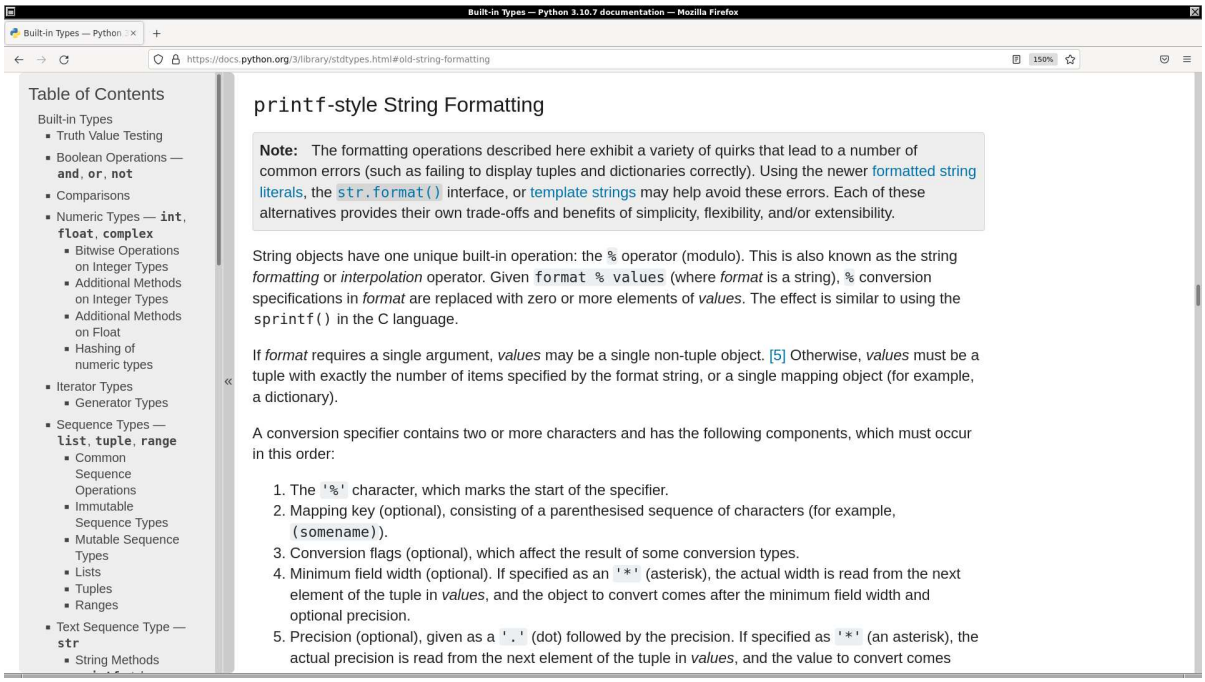

Figure 7: The official document of Python about `printf`-style string format.

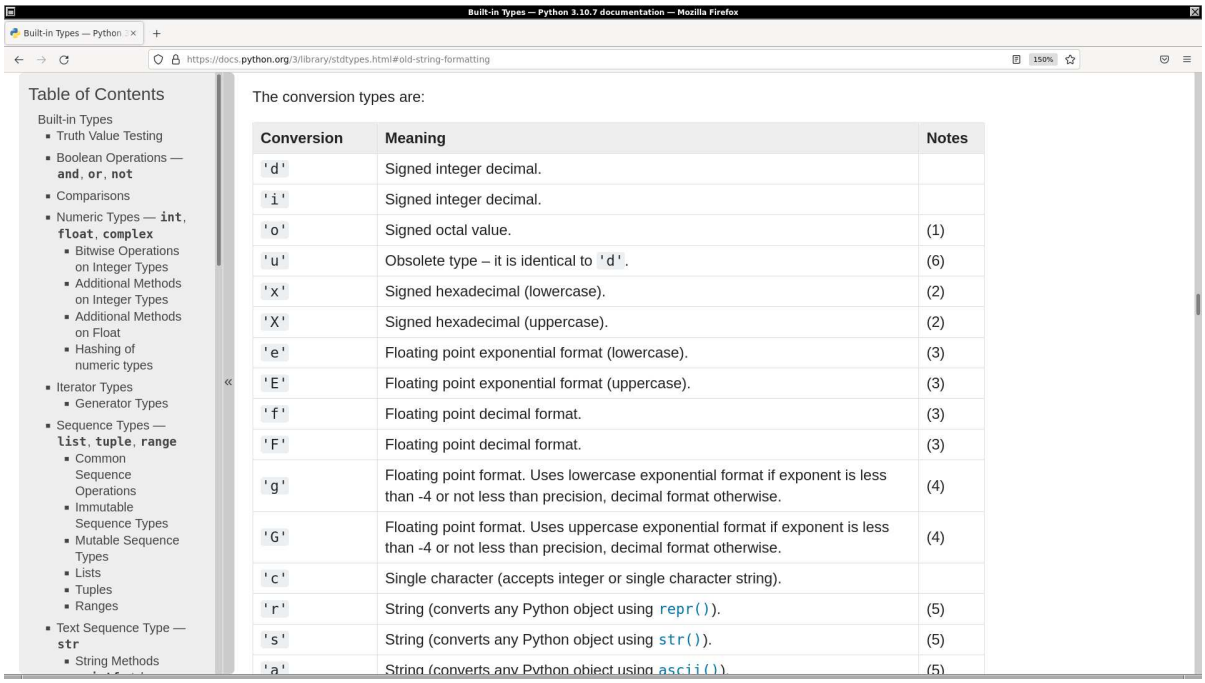

Figure 8: The conversion types of Python's `printf`-style string format.

```
# assignment of variables
pi = 3.141592653589793
e  = 2.718281828459045
c  = 299792458

# fancy formatting using formatted string literals
print (f'pi = {pi:8.6f}\ne  = {e:15.13f}\nc  = {c:g}')
```

Execute above script.

```
% chmod a+x ai202209_s01_06.py
% ./ai202209_s01_06.py
pi = 3.141593
e  = 2.7182818284590
c  = 2.99792e+08
```

To learn about formatted string literals, visit following web page. (Fig. 9)

- `https://docs.python.org/3/tutorial/inputoutput.html#tut-f-strings`

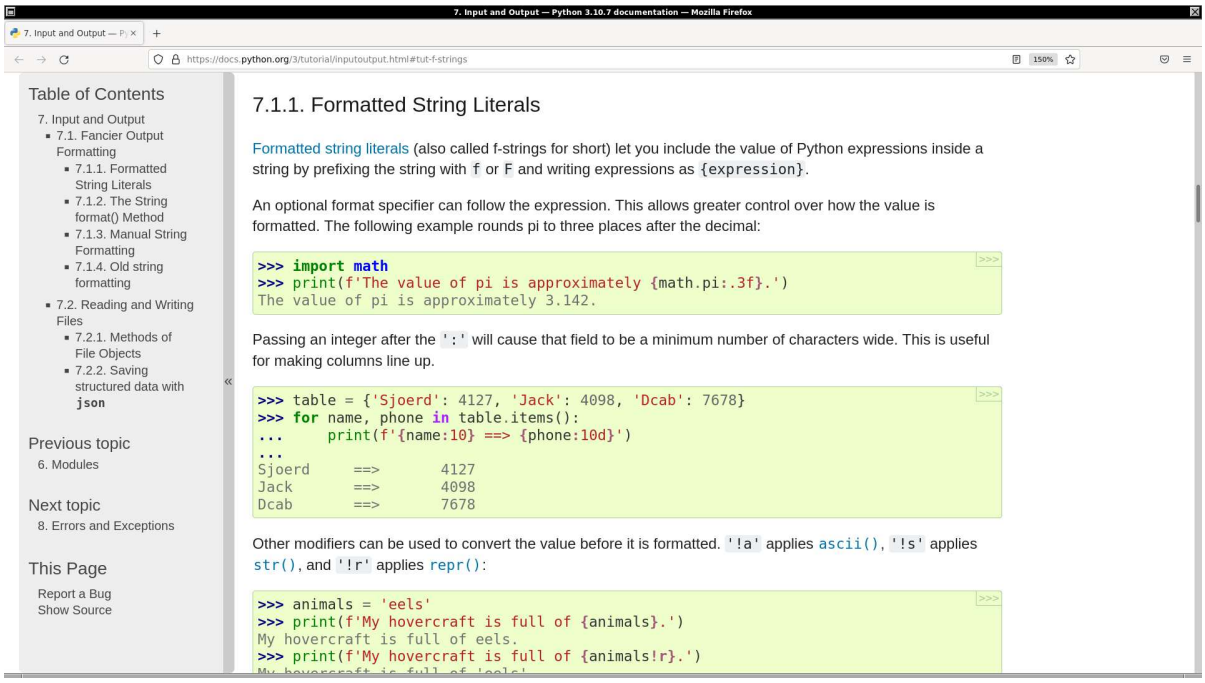

Figure 9: Description of formatted string literals (f-string) on the official document of Python.

Try following practice.

Practice 01-05

Make a Python script to print values of gravitational constant and Planck constant using formatted string literals.

2.5 Fancy formatting using `format ()` method

Make a Python script to print fancy formatted string using `format ()` method. Here is an example.

Python Code 8: ai202209_s01_07.py

```
#!/usr/pkg/bin/python3.9
```

```
#
# Time-stamp: <2022/09/09 14:57:12 (CST) daisuke>
#

# assignment of variables
pi = 3.141592653589793
e  = 2.718281828459045
c  = 299792458

# fancy formatting using .format () method
print ('pi = {:8.6f}\ne  = {:15.13f}\nc  = {:g}'.format (pi, e, c) )
```

Execute above script.

```
% chmod a+x ai202209_s01_07.py
% ./ai202209_s01_07.py
pi = 3.141593
e  = 2.7182818284590
c  = 2.99792e+08
```

To learn about `format ()` method, visit following web page. (Fig. 10)

- `https://docs.python.org/3/tutorial/inputoutput.html#tut-f-strings`

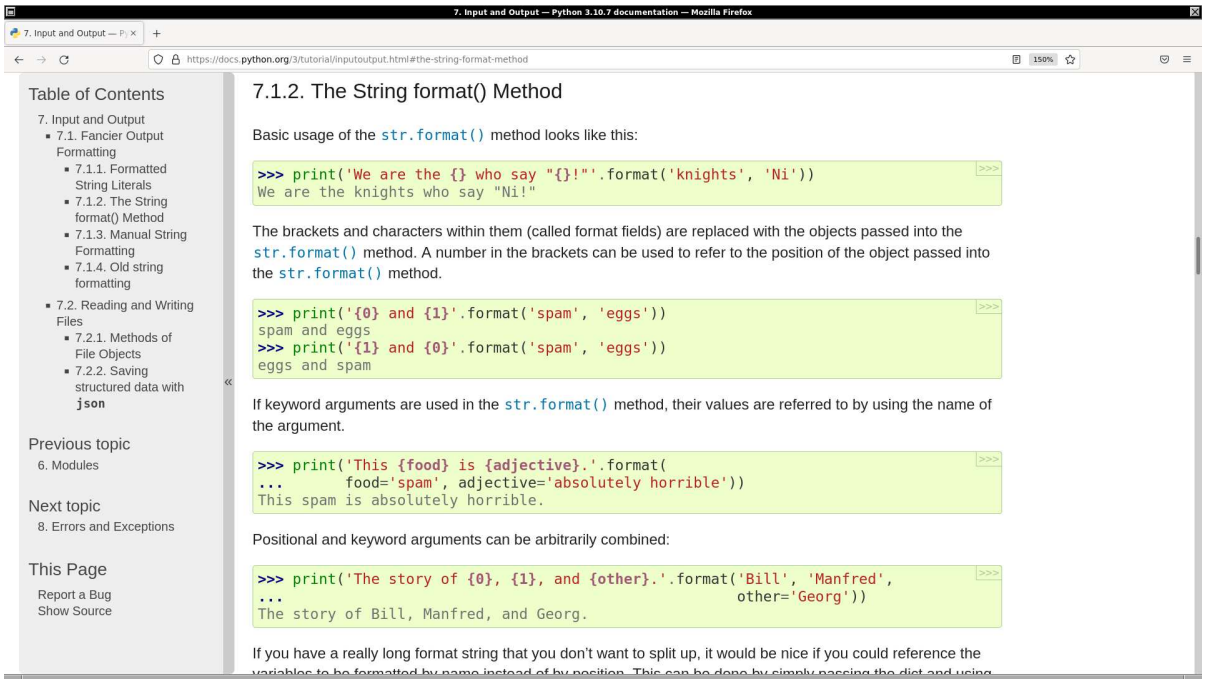

Figure 10: Description of format () method on the official document of Python.

Try following practice.

Practice 01-06

Make a Python script to print length of 1 parsec in metre and mass of a hydrogen atom in kg using format () method.

3 Doing simple calculations

Try simple calculations using Python.

3.1 Simple arithmetic calculations

3.1.1 Addition

Make a Python script to carry out addition of two integers. Here is an example.

Python Code 9: ai202209_s01_08.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 15:48:31 (CST) daisuke>
#

# two numbers
a = 2
b = 3

# calculation
c = a + b

# printing result of calculation
print (f'a = {a}')
print (f'b = {b}')
print (f'c = a + b = {a} + {b} = {c}')
```

Execute above script.

```
% chmod a+x ai202209_s01_08.py
% ./ai202209_s01_08.py
a = 2
b = 3
c = a + b = 2 + 3 = 5
```

Try following practice.

Practice 01-07

Make a Python script to carry out addition of two floating point numbers and print result of calculation.

3.1.2 Subtraction

Make a Python script to carry out subtraction of two integers. Here is an example.

Python Code 10: ai202209_s01_09.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 15:59:37 (CST) daisuke>
#

# two numbers
a = 5
b = 7

# calculation
```

```
c = a - b

# printing result of calculation
print (f'a = {a}')
print (f'b = {b}')
print (f'c = a - b = {a} - {b} = {c}')
```

Execute above script.

```
% chmod a+x ai202209_s01_09.py
% ./ai202209_s01_09.py
a = 5
b = 7
c = a - b = 5 - 7 = -2
```

Try following practice.

Practice 01-08

Make a Python script to carry out subtraction of two floating point numbers and print result of calculation.

3.1.3 Multiplication

Make a Python script to carry out multiplication of two integers. Here is an example.

Python Code 11: ai202209_s01_10.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 16:01:59 (CST) daisuke>
#

# two numbers
a = 9
b = 13

# calculation
c = a * b

# printing result of calculation
print (f'a = {a}')
print (f'b = {b}')
print (f'c = a * b = {a} * {b} = {c}')
```

Execute above script.

```
% chmod a+x ai202209_s01_10.py
% ./ai202209_s01_10.py
a = 9
b = 13
c = a * b = 9 * 13 = 117
```

Try following practice.

Practice 01-09

Make a Python script to carry out multiplication of two floating point numbers and print result of calculation.

3.1.4 Division

Make a Python script to carry out division of two integers. Here is an example.

Python Code 12: ai202209_s01_11.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 16:03:47 (CST) daisuke>
#

# two numbers
a = 15
b = 2

# calculation
c = a / b

# printing result of calculation
print (f'a = {a}')
print (f'b = {b}')
print (f'c = a / b = {a} / {b} = {c}')
```

Execute above script.

```
% chmod a+x ai202209_s01_11.py
% ./ai202209_s01_11.py
a = 15
b = 2
c = a / b = 15 / 2 = 7.5
```

Try following practice.

Practice 01-10

Make a Python script to carry out division of two floating point numbers and print result of calculation.

Also, try following.

Python Code 13: ai202209_s01_12.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 16:08:47 (CST) daisuke>
#

# two numbers
a = 23
b = 7

# calculation
quotient  = a // b
remainder = a % b

# printing result of calculation
print (f'a = {a}')
print (f'b = {b}')
print (f'quotient  = {quotient}')
print (f'remainder = {remainder}')
print (f'{b} * {quotient} + {remainder} = {b * quotient + remainder}')
```

Execute above script.

```
% chmod a+x ai202209_s01_12.py
% ./ai202209_s01_12.py
a = 23
b = 7
quotient  = 3
remainder = 2
7 * 3 + 2 = 23
```

3.2 Power

Make a Python script to calculate power. Here is an example.

Python Code 14: ai202209_s01_13.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 16:21:26 (CST) daisuke>
#

# two numbers
a = 2
b = 8

# calculation
c = a**b

# printing result of calculation
print (f'a = {a}')
print (f'b = {b}')
print (f'c = a^b = {a}^{b} = {c}')
```

Execute above script.

```
% chmod a+x ai202209_s01_13.py
% ./ai202209_s01_13.py
a = 2
b = 8
c = a^b = 2^8 = 256
```

Try following practice.

Practice 01-11

Make a Python script to calculate 10^{-2} and print result of calculation.

For more information about calculations using Python, visit following web page. (Fig. 11)

- `https://docs.python.org/3/tutorial/introduction.html#using-python-as-a-calculator`

3.3 Square root

Make a Python script to carry out calculation of square root. Here is an example.

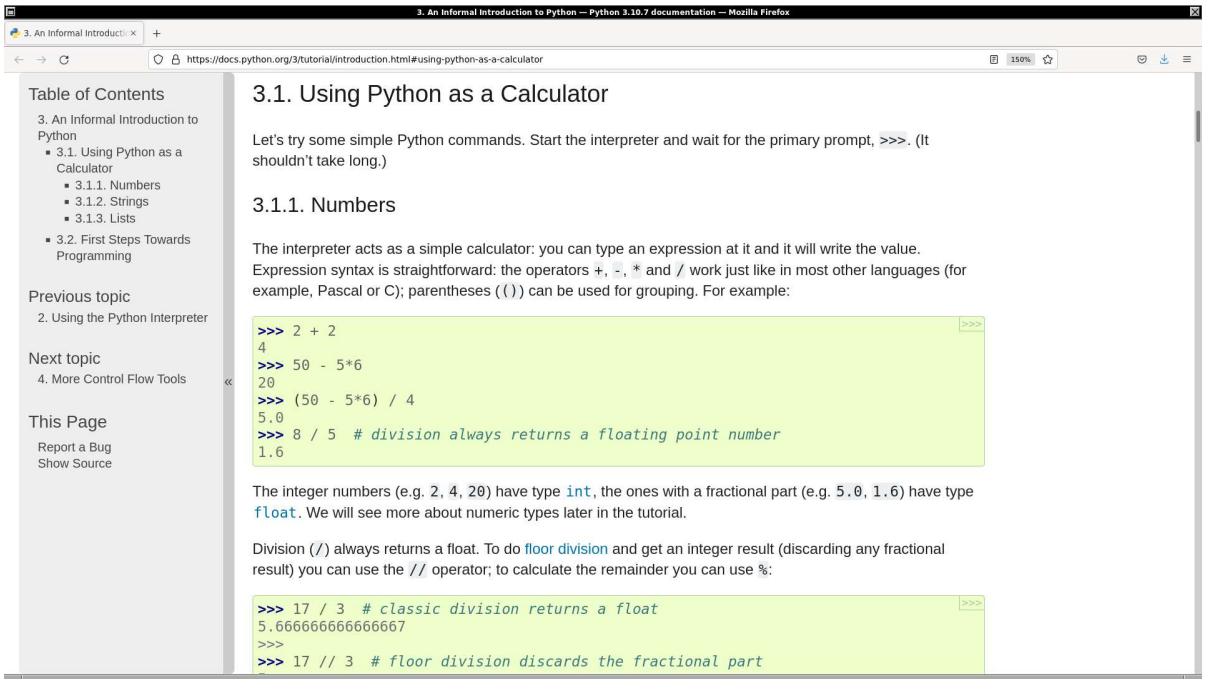

Figure 11: The section 3.1 "Using Python as a Calculator" of "The Python Tutorial".

Python Code 15: ai202209_s01_14.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 16:41:44 (CST) daisuke>
#

# importing math module
import math

# a number
a = 2.0

# calculation of square root
b = math.sqrt (a)

# printing result of calculation
print (f'a = {a}')
print (f'b = a^0.5 = {a}^0.5 = {b}')
```

Execute above script.

```
% chmod a+x ai202209_s01_14.py
% ./ai202209_s01_14.py
a = 2.0
b = a^0.5 = 2.0^0.5 = 1.4142135623730951
```

To learn about `math` module of Python, visit following web page. (Fig. 12)

- `https://docs.python.org/3/library/math.html`

Try following practice.

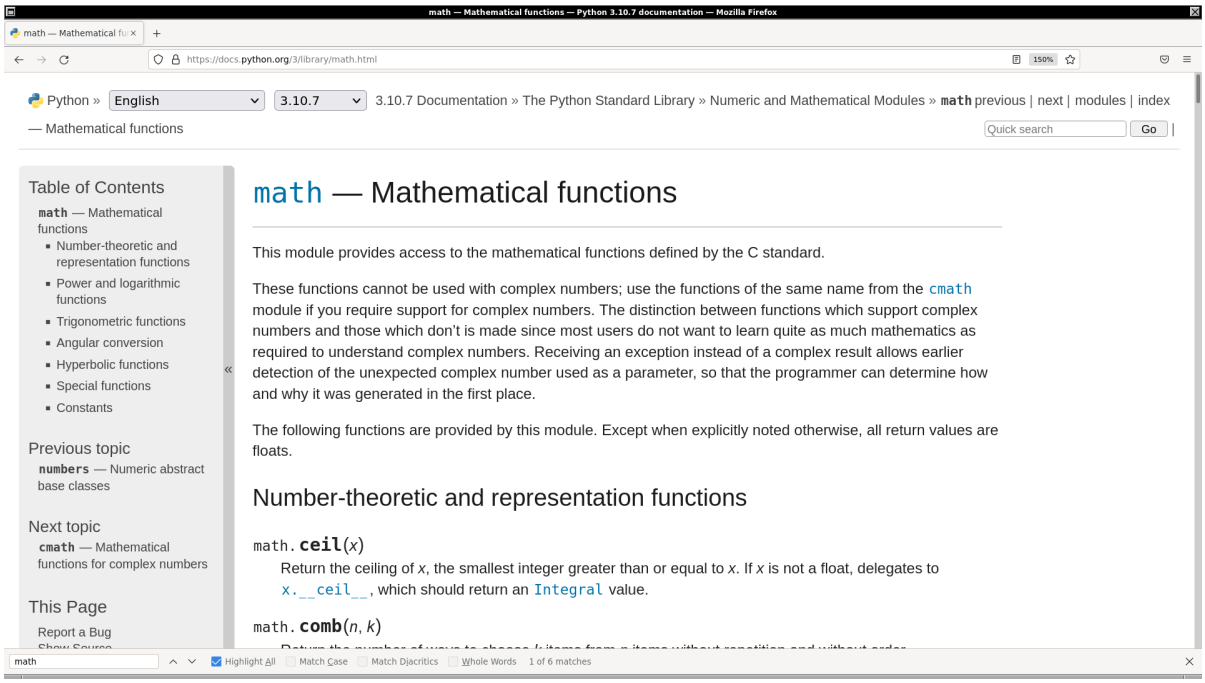

Figure 12: The official document about `math` module of Python.

Practice 01-12

Make a Python script to calculate $\sqrt{3}$ and print result of calculation.

3.4 Exponential function

Make a Python script to carry out calculation of exponential function. Here is an example.

Python Code 16: ai202209_s01_15.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 16:45:46 (CST) daisuke>
#

# importing math module
import math

# a number
a = -1.0

# calculation of exponential function
b = math.exp (a)

# printing result of calculation
print (f'a = {a}')
print (f'b = exp (a) = {math.e}^{a} = {b}')
```

Execute above script.

```
% chmod a+x ai202209_s01_15.py
% ./ai202209_s01_15.py
a = -1.0
```

```
b = exp (a) = 2.718281828459045^-1.0 = 0.36787944117144233
```

Try following practice.

Practice 01-13

Make a Python script to calculate $\exp 5$ and print result of calculation.

3.5 Natural logarithm

Make a Python script to calculate natural logarithm. Here is an example.

Python Code 17: ai202209_s01_16.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 16:57:30 (CST) daisuke>
#

# importing math module
import math

# a number
a = 2.7183

# calculation of natural logarithm
b = math.log (a)

# printing result of calculation
print (f'a = {a}')
print (f'b = log (a) = log ({a}) = {b}')
```

Execute above script.

```
% chmod a+x ai202209_s01_16.py
% ./ai202209_s01_16.py
a = 2.7183
b = log (a) = log (2.7183) = 1.0000066849139877
```

Try following practice.

Practice 01-14

Make a Python script to calculate log(100) and print result of calculation.

3.6 Base-10 logarithm

Make a Python script to calculate base-10 logarithm. Here is an example.

Python Code 18: ai202209_s01_17.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 17:00:42 (CST) daisuke>
#

# importing math module
```

```
import math

# a number
a = 1000

# calculation of natural logarithm
b = math.log10 (a)

# printing result of calculation
print (f'a = {a}')
print (f'b = log10 (a) = log10 ({a}) = {b}')
```

Execute above script.

```
% chmod a+x ai202209_s01_17.py
% ./ai202209_s01_17.py
a = 1000
b = log10 (a) = log10 (1000) = 3.0
```

Try following practice.

Practice 01-15

Make a Python script to calculate $\log_{10}(0.1)$ and print result of calculation.

3.7 Trigonometric functions

Try calculations of trigonometric functions.

3.7.1 Sine function

Make a Python script to calculate sine function. Here is an example.

Python Code 19: ai202209_s01_18.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 17:13:14 (CST) daisuke>
#

# importing math module
import math

# value of pi
pi = math.pi

# angle in degree
a_deg = 60.0

# angle in radian
a_rad = a_deg / 180.0 * pi

# calculation of sine
sin_a = math.sin (a_rad)

# printing result of calculation
print (f'pi          = {pi}')
print (f'a_deg       = {a_deg} deg')
```

```
print (f'a_rad       = {a_rad} rad')
print (f'sin (a_rad) = sin ({a_rad}) = {sin_a}')
```

Execute above script.

```
% chmod a+x ai202209_s01_18.py
% ./ai202209_s01_18.py
pi          = 3.141592653589793
a_deg       = 60.0 deg
a_rad       = 1.0471975511965976 rad
sin (a_rad) = sin (1.0471975511965976) = 0.8660254037844386
```

Try following practice.

Practice 01-16

Make a Python script to calculate sin(45°) and print result of calculation.

3.7.2 Cosine function

Make a Python script to calculate cosine function. Here is an example.

Python Code 20: ai202209_s01_19.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 17:16:55 (CST) daisuke>
#

# importing math module
import math

# value of pi
pi = math.pi

# angle in degree
a_deg = 60.0

# angle in radian
a_rad = a_deg / 180.0 * pi

# calculation of sine
cos_a = math.cos (a_rad)

# printing result of calculation
print (f'pi          = {pi}')
print (f'a_deg       = {a_deg} deg')
print (f'a_rad       = {a_rad} rad')
print (f'cos (a_rad) = cos ({a_rad}) = {cos_a}')
```

Execute above script.

```
% chmod a+x ai202209_s01_19.py
% ./ai202209_s01_19.py
pi          = 3.141592653589793
a_deg       = 60.0 deg
a_rad       = 1.0471975511965976 rad
cos (a_rad) = cos (1.0471975511965976) = 0.5000000000000001
```

Try following practice.

Practice 01-17

Make a Python script to calculate cos(90°) and print result of calculation.

3.7.3 Tangent function

Make a Python script to calculate tangent function. Here is an example.

Python Code 21: ai202209_s01_20.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/11 19:24:51 (CST) daisuke>
#

# importing math module
import math

# value of pi
pi = math.pi

# angle in degree
a_deg = 60.0

# angle in radian
a_rad = a_deg / 180.0 * pi

# calculation of sine
tan_a = math.tan (a_rad)

# printing result of calculation
print (f'pi          = {pi}')
print (f'a_deg       = {a_deg} deg')
print (f'a_rad       = {a_rad} rad')
print (f'tan (a_rad) = tan ({a_rad}) = {tan_a}')
```

Execute above script.

```
% chmod a+x ai202209_s01_20.py
% ./ai202209_s01_20.py
pi          = 3.141592653589793
a_deg       = 60.0 deg
a_rad       = 1.0471975511965976 rad
tan (a_rad) = tan (1.0471975511965976) = 1.7320508075688767
```

Try following practice.

Practice 01-18

Make a Python script to calculate tan(−45°) and print result of calculation.

3.7.4 Arctangent function

Make a Python script to calculate arctangent function. Here is an example.

Python Code 22: ai202209_s01_21.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 17:31:03 (CST) daisuke>
#

# importing math module
import math

# (x, y) coordinate
x = +1.0
y = -1.0

# calculation of arctangent
a_rad = math.atan2 (y, x)

# conversion from radian into degree
a_deg = math.degrees (a_rad)

# printing result of calculation
print (f'x             = {x}')
print (f'y             = {y}')
print (f'atan2 (y, x) = {a_rad} rad = {a_deg} deg')
```

Execute above script.

```
% chmod a+x ai202209_s01_21.py
% ./ai202209_s01_21.py
x             = 1.0
y             = -1.0
atan2 (y, x) = -0.7853981633974483 rad = -45.0 deg
```

Try following practice.

Practice 01-19

Make a Python script to calculate $\mathsf{atan2}(\sqrt{3}, 1)$ and print result of calculation.

4 Using control flow statements

Try some control flow statements.

4.1 `if` statement

Make a Python script using `if` and `else` statements.

Python Code 23: ai202209_s01_22.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 17:59:11 (CST) daisuke>
#

# reading an integer number from keyboard typing
a_str = input ('Type one integer number: ')

# converting a string into integer
```

```
a = int (a_str)

# if and else statements
if (a % 2 == 0):
    print ("The number you type is an even number.")
else:
    print ("The number you type is an odd number.")
```

Execute above script.

```
% chmod a+x ai202209_s01_22.py
% ./ai202209_s01_22.py
Type one integer number: 8
The number you type is an even number.
% ./ai202209_s01_22.py
Type one integer number: 17
The number you type is an odd number.
```

Try following practice.

Practice 01-20

Make a Python script using `if` and `else` statements.

Make a Python script using `if`, `elif`, and `else` statements.

Python Code 24: ai202209_s01_23.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 18:03:00 (CST) daisuke>
#

# reading an integer number from keyboard typing
a_str = input ('Type one integer number: ')

# converting a string into integer
a = int (a_str)

# if and else statements
if (a > 0):
    print ("The number you type is a positive number.")
elif (a < 0):
    print ("The number you type is a negative number.")
else:
    print ("The number you type is zero.")
```

Execute above script.

```
% chmod a+x ai202209_s01_23.py
% ./ai202209_s01_23.py
Type one integer number: 3
The number you type is a positive number.
% ./ai202209_s01_23.py
Type one integer number: -4
The number you type is a negative number.
% ./ai202209_s01_23.py
Type one integer number: 0
The number you type is zero.
```

Try following practice.

Practice 01-21

Make a Python script using `if`, `elif`, and `else` statements.

4.2 `for` statement

Make a Python script using `for` statement. Here is an example.

Python Code 25: ai202209_s01_24.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 18:12:39 (CST) daisuke>
#

# initialisation of a variable "total"
total = 0

# calculating 1 + 2 + 3 + ... + 10 using "for" statement
for i in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
    # adding "i" to "total"
    total = total + i

# printing result of calculation
print (f'1 + 2 + 3 + ... + 10 = {total}')
```

Execute above script.

```
% chmod a+x ai202209_s01_24.py
% ./ai202209_s01_24.py
1 + 2 + 3 + ... + 10 = 55
```

Try following practice.

Practice 01-22

Make a Python script calculating $1 \times 2 \times 3 \times \cdots \times 10$ using `for` statement and printing result of calculation.

Above Python script can also be written like below.

Python Code 26: ai202209_s01_25.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 18:19:01 (CST) daisuke>
#

# initialisation of a variable "total"
total = 0

# calculating 1 + 2 + 3 + ... + 10 using "for" statement
for i in range (1, 11, 1):
    # adding "i" to "total"
    total += i

# printing result of calculation
print (f'1 + 2 + 3 + ... + 10 = {total}')
```

Execute above script.

```
% chmod a+x ai202209_s01_25.py
% ./ai202209_s01_25.py
1 + 2 + 3 + ... + 10 = 55
```

Try following practice.

Practice 01-23

Make a Python script calculating $1+2+3+\cdots+100$ using `for` statement and printing result of calculation.

4.3 `while` and `break` statements

Make a Python script using `while` statement. Here is an example.

Python Code 27: ai202209_s01_26.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 18:28:08 (CST) daisuke>
#

# importing math module
import math

# initialisation of a variable "a"
a_deg = 0.0

# calculating sin (0 deg), sin (15 deg), sin (30 deg), ..., sin (180 deg)
while (a_deg <= 180.0):
    # converting from degree into radian
    a_rad = math.radians (a_deg)
    # calculation of sine
    sin_a = math.sin (a_rad)
    # printing result of calculation
    print (f'sin ({a_deg:5.1f} deg) = {sin_a:8.6f}')
    # incrementing variable "a"
    a_deg += 15.0
```

Execute above script.

```
% chmod a+x ai202209_s01_26.py
% ./ai202209_s01_26.py
sin (  0.0 deg) = 0.000000
sin ( 15.0 deg) = 0.258819
sin ( 30.0 deg) = 0.500000
sin ( 45.0 deg) = 0.707107
sin ( 60.0 deg) = 0.866025
sin ( 75.0 deg) = 0.965926
sin ( 90.0 deg) = 1.000000
sin (105.0 deg) = 0.965926
sin (120.0 deg) = 0.866025
sin (135.0 deg) = 0.707107
sin (150.0 deg) = 0.500000
sin (165.0 deg) = 0.258819
sin (180.0 deg) = 0.000000
```

Same calculations can be carried out by following script using `break` statement.

Python Code 28: ai202209_s01_27.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 18:38:40 (CST) daisuke>
#

# importing math module
import math

# initialisation of a variable "a"
a_deg = 0.0

# calculating sin (0 deg), sin (15 deg), sin (30 deg), ..., sin (180 deg)
while (True):
    # stop "while" loop if "a_deg" is greater than 180.0 deg
    if (a_deg > 180.0):
        break
    # converting from degree into radian
    a_rad = math.radians (a_deg)
    # calculation of sine
    sin_a = math.sin (a_rad)
    # printing result of calculation
    print (f'sin ({a_deg:5.1f} deg) = {sin_a:8.6f}')
    # incrementing variable "a"
    a_deg += 15.0
```

Execute above script.

```
% chmod a+x ai202209_s01_27.py
% ./ai202209_s01_27.py
sin (  0.0 deg) = 0.000000
sin ( 15.0 deg) = 0.258819
sin ( 30.0 deg) = 0.500000
sin ( 45.0 deg) = 0.707107
sin ( 60.0 deg) = 0.866025
sin ( 75.0 deg) = 0.965926
sin ( 90.0 deg) = 1.000000
sin (105.0 deg) = 0.965926
sin (120.0 deg) = 0.866025
sin (135.0 deg) = 0.707107
sin (150.0 deg) = 0.500000
sin (165.0 deg) = 0.258819
sin (180.0 deg) = 0.000000
```

Try following practice.

Practice 01-24

Make a Python script calculating $\cos(0°)$, $\cos(15°)$, $\cos(30°)$, $\cdots$, $\cos(360°)$ using `for` statement and printing result of calculation.

4.4 `continue` statement

Make a Python script using `continue` statement. Here is an example.

Python Code 29: ai202209_s01_28.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 18:45:12 (CST) daisuke>
#

# printing odd number between 0 and 30
for i in range (30):
    # if number is divisible by 2, then skipping to next number
    if (i % 2 == 0):
        continue
    # if not, then it is an odd number
    print (f'{i} is an odd number.')
```

Execute above script.

```
% chmod a+x ai202209_s01_28.py
% ./ai202209_s01_28.py
1 is an odd number.
3 is an odd number.
5 is an odd number.
7 is an odd number.
9 is an odd number.
11 is an odd number.
13 is an odd number.
15 is an odd number.
17 is an odd number.
19 is an odd number.
21 is an odd number.
23 is an odd number.
25 is an odd number.
27 is an odd number.
29 is an odd number.
```

Try following practice.

Practice 01-25

Make a Python script finding odd numbers between 50 and 70 using `for` and `continue` statements and printing those numbers.

To learn more about control flow statements of Python, visit following web page. (Fig. 13)

- `https://docs.python.org/3/tutorial/controlflow.html`

5 Making and using your own function

Make a Python script to define your own function and call it. Here is an example.

Python Code 30: ai202209_s01_29.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 19:44:25 (CST) daisuke>
#
```

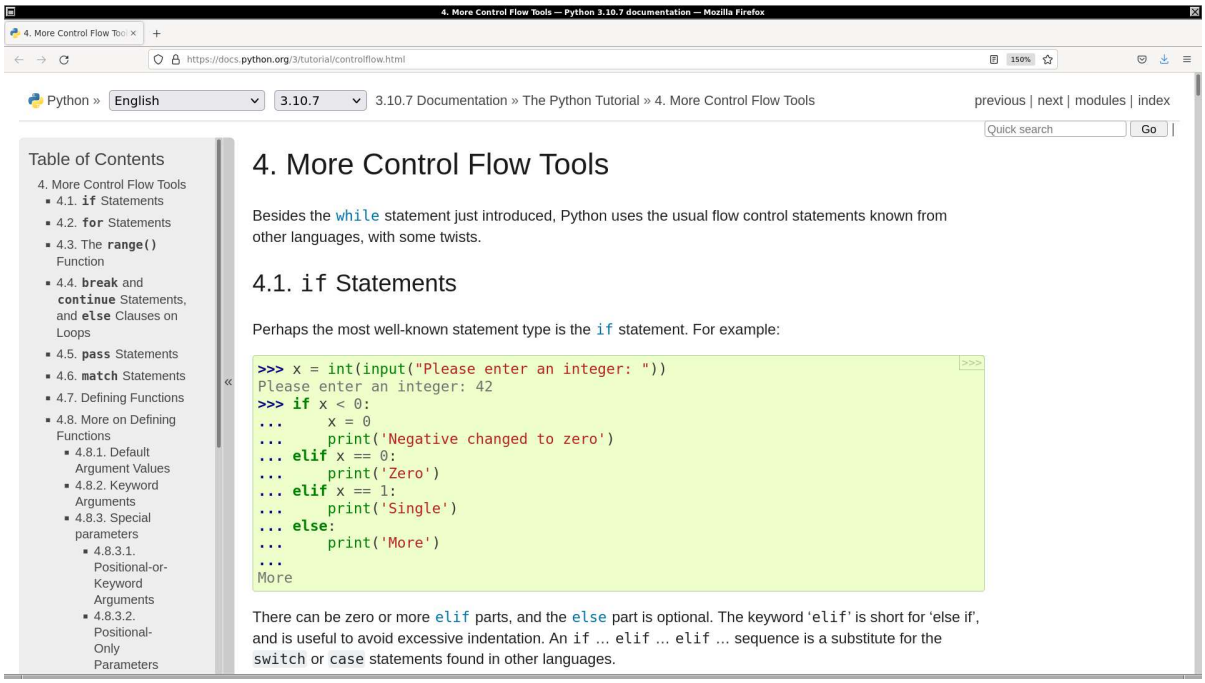

Figure 13: The section 4 “More Control Flow Tools” of “The Python Tutorial”.

```
# defining a function to add two numbers
def add_two_numbers (a, b):
    # adding two numbers
    c = a + b
    # returning result of calculation
    return (c)

# two numbers
n1 = 7
n2 = 8

# using the function "add_two_numbers"
n3 = add_two_numbers (n1, n2)

# printing result
print (f'n1 = {n1}')
print (f'n2 = {n2}')
print (f'n3 = n1 + n2 = {n3}')
```

Execute above script.

```
% chmod a+x ai202209_s01_29.py
% ./ai202209_s01_29.py
n1 = 7
n2 = 8
n3 = n1 + n2 = 15
```

Try following practice.

Practice 01-26

Make a Python script defining your own function and calling the function you have defined.

To learn more about defining and using functions, visit following web pages. (Fig. 14 and 15)

- https://docs.python.org/3/tutorial/controlflow.html#defining-functions
- https://docs.python.org/3/tutorial/controlflow.html#more-on-defining-functions

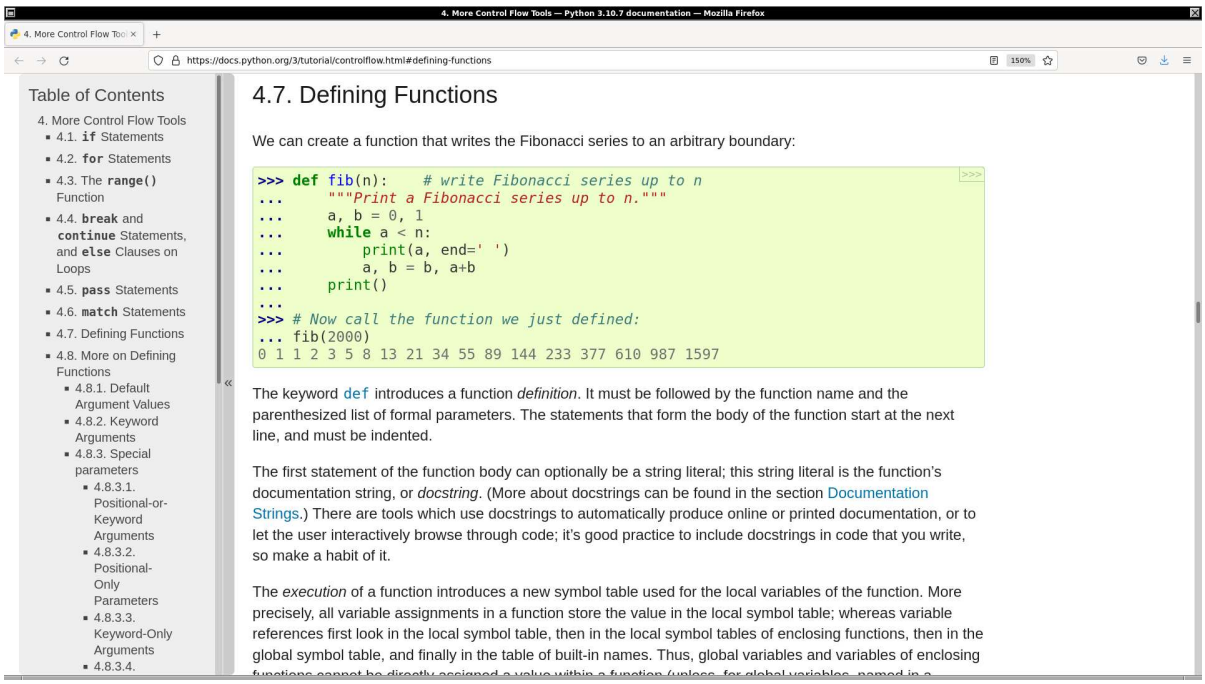

Figure 14: The section 4.7 "Defining Functions" of "The Python Tutorial".

6 Data structure

Python provides number of powerful data structure, such as lists and dictionaries.

6.1 Using lists

Make a Python script to play with a list. Here is an example.

Python Code 31: ai202209_s01_30.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 21:22:09 (CST) daisuke>
#

# initialisation of a list
list_a = [ 0.1, 2.3, 4.5, 5.6, 6.7, 8.9, 10.1 ]

# type of "list_a"
print ("type of list_a:", type (list_a) )

# printing the list "list_a"
print ("list_a       =", list_a)

# counting number of elements in the list "list_a"
n = len (list_a)

# appending a new element to the list
```

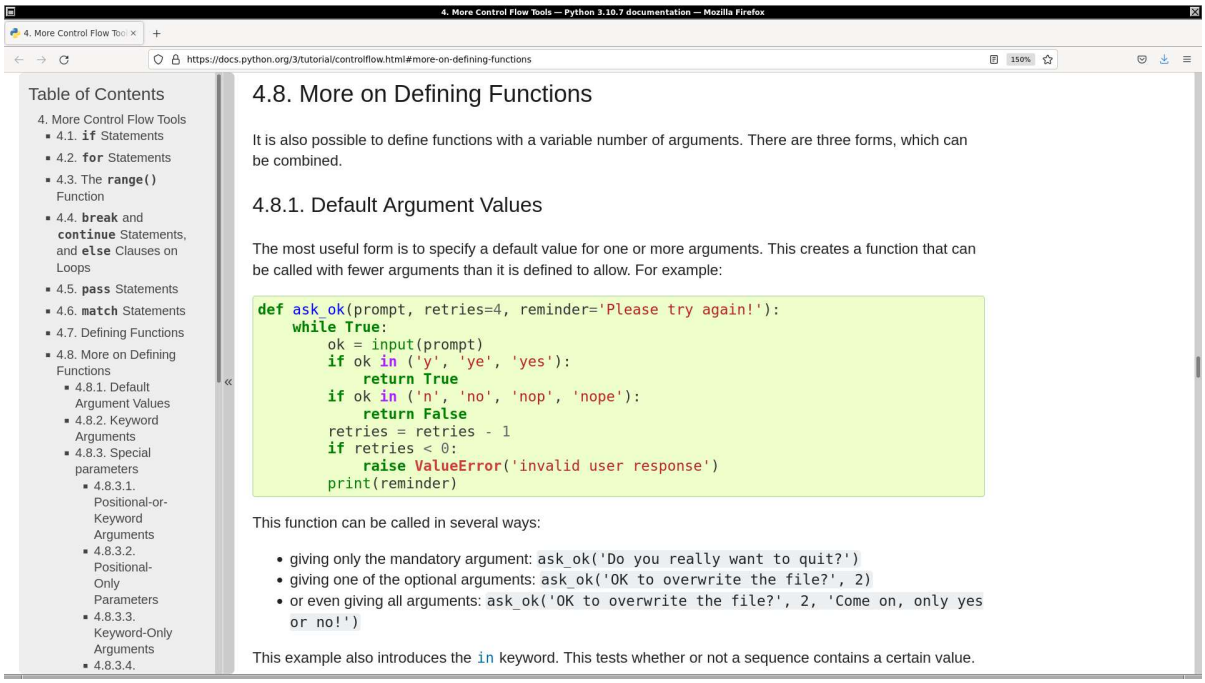

Figure 15: The section 4.8 “More on Defining Functions” of “The Python Tutorial”.

```
list_a.append (11.3)

# printing the list "list_a"
print ("list_a       =", list_a)

# counting number of elements in the list "list_a"
n = len (list_a)

# printing number of elements in the list "list_a"
print ("len (list_a) =", n)

# printing an element of the list "list_a"
print ("list_a[0]    =", list_a[0])
print ("list_a[3]    =", list_a[3])
print ("list_a[-2]   =", list_a[-2])

# printing elements of the list "list_a" using slicing
print ("list_a[2:5]  =", list_a[2:5])

# printing elements of the list "list_a" using slicing
print ("list_a[4:]   =", list_a[4:])

# printing elements of the list "list_a" using slicing
print ("list_a[:3]   =", list_a[:3])

# adding 1.23 to each element of "list_a"
for i in range ( len (list_a) ):
    list_a[i] += 1.23

# printing the list "list_a"
print ("list_a       =", list_a)
```

Execute above script.

```
% chmod a+x ai202209_s01_30.py
% ./ai202209_s01_30.py
type of list_a: <class 'list'>
list_a       = [0.1, 2.3, 4.5, 5.6, 6.7, 8.9, 10.1]
list_a       = [0.1, 2.3, 4.5, 5.6, 6.7, 8.9, 10.1, 11.3]
len (list_a) = 8
list_a[0]    = 0.1
list_a[3]    = 5.6
list_a[-2]   = 10.1
list_a[2:5]  = [4.5, 5.6, 6.7]
list_a[4:]   = [6.7, 8.9, 10.1, 11.3]
list_a[:3]   = [0.1, 2.3, 4.5]
list_a       = [1.33, 3.53, 5.73, 6.83, 7.93, 10.13, 11.33, 12.530000000000001]
```

Try following practice.

Practice 01-27

Make your own Python script to play with lists.

6.2 Using tuples

Make a Python script to play with a tuple. Here is an example.

Python Code 32: ai202209_s01_31.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 21:22:30 (CST) daisuke>
#

# initialisation of a tuple
planet = ( 'Mercury', 'Venus', 'Earth', 'Mars', \
           'Jupiter', 'Saturn', 'Uranus', 'Neptune' )

# type of "planet"
print ("type of planet:", type (planet) )

# printing the tuple "tuple_a"
print ("planet:\n", planet)

# counting number of elements in the tuple "planet"
n = len (planet)

# printing number of elements in the tuple "planet"
print ("len (planet) =", n)

# accessing to an element using index
print ("planet[2]    =", planet[2])
print ("planet[7]    =", planet[7])
print ("planet[-3]   =", planet[-3])

# accessing to elements using slicing
print ("planet[2:5]  =", planet[2:5])
print ("planet[:4]   =", planet[:4])
print ("planet[6:]   =", planet[6:])
```

Execute above script.

```
% chmod a+x ai202209_s01_31.py
% ./ai202209_s01_31.py
type of planet: <class 'tuple'>
planet:
 ('Mercury', 'Venus', 'Earth', 'Mars', 'Jupiter', 'Saturn', 'Uranus', 'Neptune')
len (planet) = 8
planet[2]    = Earth
planet[7]    = Neptune
planet[-3]   = Saturn
planet[2:5]  = ('Earth', 'Mars', 'Jupiter')
planet[:4]   = ('Mercury', 'Venus', 'Earth', 'Mars')
planet[6:]   = ('Uranus', 'Neptune')
```

Try following practice.

Practice 01-28

Make your own Python script to play with tuples.

6.3 Using sets

Make a Python script to play with a tuple. Here is an example.

Python Code 33: ai202209_s01_32.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 21:39:59 (CST) daisuke>
#

# making a set
fruits = { 'apple', 'banana', 'cherry', 'durian', 'grape', \
           'mango', 'orange', 'apple', 'banana', 'cherry' }

# type of "fruits"
print ("type of fruits:", type (fruits) )

# printing "fruits"
print ("fruits:\n", fruits)

# adding an element to "fruits"
fruits.add ('lemon')

# printing "fruits"
print ("fruits:\n", fruits)

# existence check
exist_banana = 'banana' in fruits
print ("banana exists?", exist_banana)

# one more existence check
exist_tomato = 'tomato' in fruits
print ("tomato exists?", exist_tomato)

# making a list
list_cities1 = [ 'Taipei', 'Taoyuan', 'Hsinchu', 'Taichung', 'Tainan', \
                 'Taipei', 'Taoyuan' ]
```

```
# printing type of "list_cities1"
print ("type of list_cities1:", type (list_cities1) )

# printing list "list_cities1"
print ("list_cities1:\n", list_cities1)

# generating a set from a list
set_cities1 = set (list_cities1)

# printing type of "set_cities1"
print ("type of set_cities1:", type (set_cities1) )

# printing set "set_cities1"
print ("set_cities1:\n", set_cities1)

# making a set "set_cities2"
set_cities2 = { 'Taipei', 'Taoyuan', 'Kaohsiung', 'Pingtung', 'Taitung', \
                'Hualien', 'Yilan', 'Taipei', 'Taoyuan' }

# printing type of "set_cities2"
print ("type of set_cities2:", type (set_cities2) )

# printing set "set_cities2"
print ("set_cities2:\n", set_cities2)

# finding elements in both set_cities1 and set_cities2
set_cities_both = set_cities1 & set_cities2

# printing elements in both set_cities1 and set_cities2
print ("cities in both set_cities1 and set_cities2:\n", set_cities_both)
```

Execute above script.

```
% chmod a+x ai202209_s01_32.py
% ./ai202209_s01_32.py
type of fruits: <class 'set'>
fruits:
 {'cherry', 'durian', 'banana', 'apple', 'grape', 'orange', 'mango'}
fruits:
 {'cherry', 'durian', 'banana', 'apple', 'grape', 'lemon', 'orange', 'mango'}
banana exists? True
tomato exists? False
type of list_cities1: <class 'list'>
list_cities1:
 ['Taipei', 'Taoyuan', 'Hsinchu', 'Taichung', 'Tainan', 'Taipei', 'Taoyuan']
type of set_cities1: <class 'set'>
set_cities1:
 {'Taoyuan', 'Taipei', 'Taichung', 'Hsinchu', 'Tainan'}
type of set_cities2: <class 'set'>
set_cities2:
 {'Taoyuan', 'Taitung', 'Kaohsiung', 'Pingtung', 'Yilan', 'Hualien', 'Taipei'}
cities in both set_cities1 and set_cities2:
 {'Taoyuan', 'Taipei'}
```

Try following practice.

Practice 01-29

Make your own Python script to play with sets.

6.4 Using dictionaries

Make a Python script to play with a dictionary. Here is an example.

Python Code 34: ai202209_s01_33.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 21:44:36 (CST) daisuke>
#

# making a dictionary
star_mag = {
    "Sirius": -1.46,
    "Canopus": -0.74,
    "Rigil Kentaurus": -0.27,
    "Arcturus": -0.05,
    "Vega": 0.03,
    "Capella": 0.08,
    "Rigel": 0.13,
    "Procyon": 0.34,
    "Achernar": 0.46,
    "Betelgeuse": 0.50,
}

# type of "star_mag"
print ("type of star_mag:", type (star_mag) )

# printing dictionary "star_mag"
print (star_mag)

# accessing to an element
print ("magnitude of Vega =", star_mag['Vega'])

# printing all the data
for star in sorted (star_mag.keys ()):
    print (f'{star:16} : {star_mag[star]:+5.2f}')
```

Execute above script.

```
% chmod a+x ai202209_s01_33.py
% ./ai202209_s01_33.py
type of star_mag: <class 'dict'>
{'Sirius': -1.46, 'Canopus': -0.74, 'Rigil Kentaurus': -0.27, 'Arcturus': -0.05,
 'Vega': 0.03, 'Capella': 0.08, 'Rigel': 0.13, 'Procyon': 0.34, 'Achernar': 0.46
, 'Betelgeuse': 0.5}
magnitude of Vega = 0.03
Achernar         : +0.46
Arcturus         : -0.05
Betelgeuse       : +0.50
Canopus          : -0.74
Capella          : +0.08
Procyon          : +0.34
Rigel            : +0.13
Rigil Kentaurus  : -0.27
Sirius           : -1.46
Vega             : +0.03
```

Try following practice.

Practice 01-30

Make your own Python script to play with dictionaries.

6.5 Nested sequence structure

Python's sequence data structure, such as lists and dictionaries, can be nested to make a multi-dimensional data structure. Make a Python script to play with a multi-dimensional dictionary.

Python Code 35: ai202209_s01_34.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/09 21:53:51 (CST) daisuke>
#

# making a multi-dimensional dictionary
asteroid = {
    'Ceres': {
        'a': 2.769,
        'e': 0.076,
        'i': 10.594,
        'H': 3.4,
        'diameter': 939.4,
    },
    'Pallas': {
        'a': 2.774,
        'e': 0.230,
        'i': 34.833,
        'H': 4.2,
        'diameter': 545,
    },
    'Juno': {
        'a': 2.668,
        'e': 0.257,
        'i': 12.991,
        'H': 5.33,
        'diameter': 246.596,
    },
    'Vesta': {
        'a': 2.361,
        'e': 0.089,
        'i': 7.142,
        'H': 3.0,
        'diameter': 525.4,
    },
}

# printing dictionary information
print ("Ceres:\n", asteroid['Ceres'])
print ("Vesta:\n", asteroid['Vesta'])
print ("a of Pallas =", asteroid['Pallas']['a'], "au")
print ("e of Juno   =", asteroid['Juno']['e'])
print ("i of Vesta  =", asteroid['Vesta']['i'], "deg")
print ("H of Ceres  =", asteroid['Ceres']['H'], "mag")

# printing rotation period of 4 asteroids
print ('Diameters of asteroids:')
for key in (asteroid.keys ()):
```

```
    print (f'  {key:6} : {asteroid[key]["diameter"]:4.1f} km')
```

Execute above script.

```
% chmod a+x ai202209_s01_34.py
% ./ai202209_s01_34.py
Ceres:
 {'a': 2.769, 'e': 0.076, 'i': 10.594, 'H': 3.4, 'diameter': 939.4}
Vesta:
 {'a': 2.361, 'e': 0.089, 'i': 7.142, 'H': 3.0, 'diameter': 525.4}
a of Pallas = 2.774 au
e of Juno   = 0.257
i of Vesta  = 7.142 deg
H of Ceres  = 3.4 mag
Diameters of asteroids:
  Ceres  : 939.4 km
  Pallas : 545.0 km
  Juno   : 246.6 km
  Vesta  : 525.4 km
```

Try following practice.

Practice 01-31

Make your own Python script to play with multi-dimensional dictionaries.

To learn more about data structures of Python, visit following web page. (Fig. 16)

- `https://docs.python.org/3/tutorial/datastructures.html`

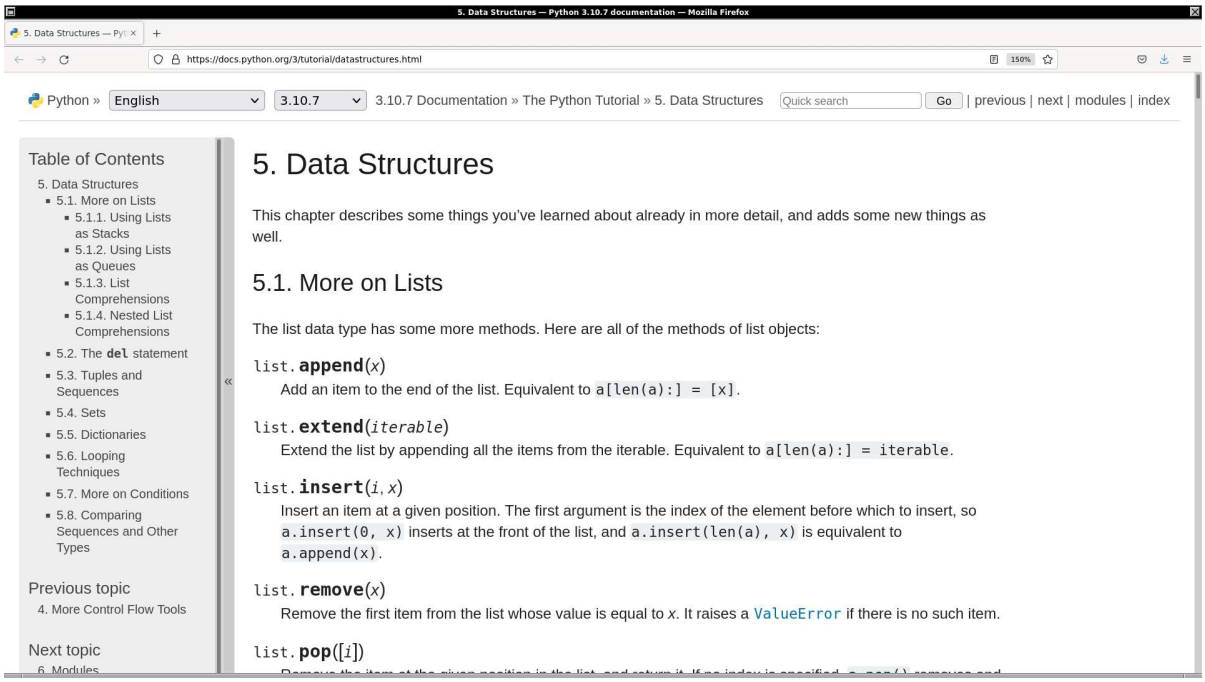

Figure 16: The section 5 "Data Structures" of "The Python Tutorial".

7 Reading and writing files

For astronomical data analysis, we often download data file (e.g. image data, spectrum, catalogue data, table data), read it, do some calculation, and write a new file.

7.1 Reading a file

Start your favourite web browser, such as Firefox or Chrome, and open following file. (Fig. 17)

- `https://s3b.astro.ncu.edu.tw/ai_202209/data/fruits.data`

Save the file on your computer. Move the file to the directory (folder) where you store files for this session.

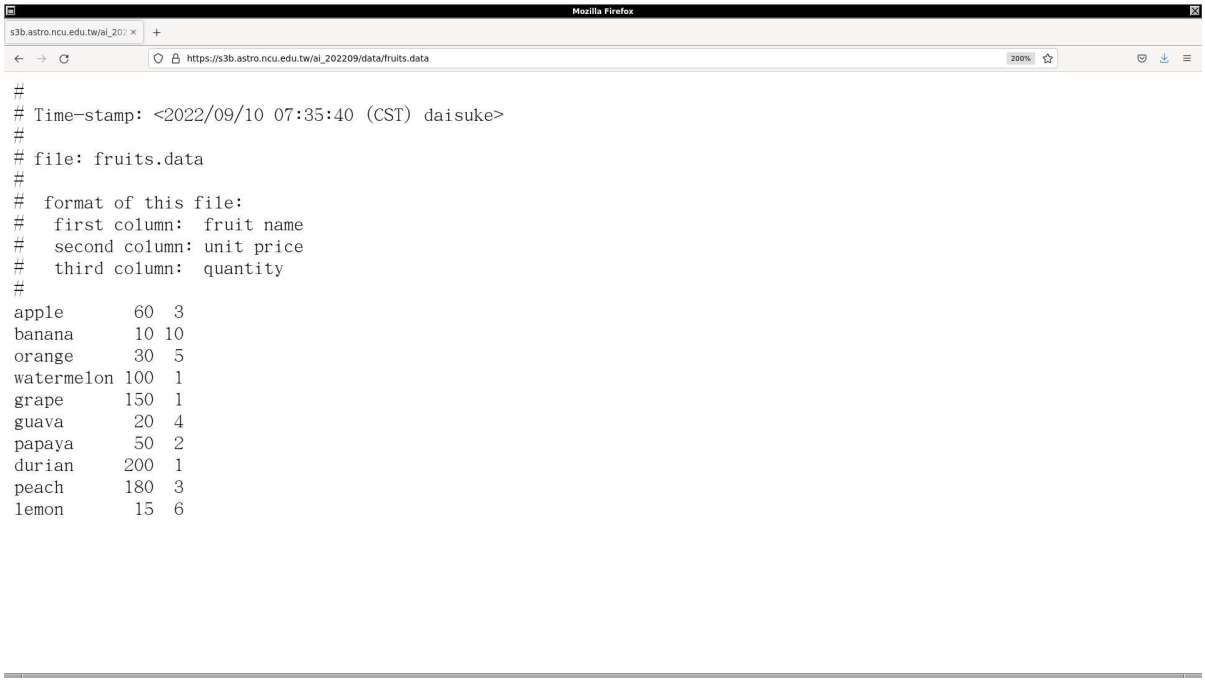

Figure 17: The sample data file.

The file "`fruits.data`" has a header part at the beginning of the file. The header lines start with "`#`". The data part has three columns. The first column is fruit name. The second column is unit price. The third column is quantity. Columns are separated by white spaces.

Make a Python script to read the file "`fruits.data`" and calculate the total price. Here is an example.

Python Code 36: ai202209_s01_35.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/10 07:58:54 (CST) daisuke>
#

# name of data file
file_data = 'fruits.data'

# a variable for total price
total = 0

# opening file with read mode
with open (file_data, 'r') as fh:
    # reading file line-by-line
    for line in fh:
        # if line starts with '#', then it is a header line and skip
        if (line[0] == '#'):
            continue
        # splitting line
        (name, unit_price_str, quantity_str) = line.split ()
```

```
        # converting from string into integer
        unit_price = int (unit_price_str)
        quantity   = int (quantity_str)
        # calculation of sub-total
        subtotal = unit_price * quantity
        # adding "subtotal" to "total"
        total += subtotal
        # printing information
        print (f'{name:10} {unit_price:3d} {quantity:2d} {subtotal:3d}')

# printing total price
print (f'\ntotal price: {total:4d}')
```

Execute above script.

```
% chmod a+x ai202209_s01_35.py
% ./ai202209_s01_35.py
apple       60  3 180
banana      10 10 100
orange      30  5 150
watermelon 100  1 100
grape      150  1 150
guava       20  4  80
papaya      50  2 100
durian     200  1 200
peach      180  3 540
lemon       15  6  90

total price: 1690
```

Try following practice.

Practice 01-32

Make your own Python script to read a file.

7.2 Writing a file

Make a Python script to examine numbers between 2 and 100 whether they are prime numbers or not and write results into a file. Here is an example.

Python Code 37: ai202209_s01_36.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/10 08:16:34 (CST) daisuke>
#

# start and end numbers
n_start = 2
n_end   = 100

# output file name
file_output = "primenumbers.data"

# opening file for writing
with open (file_output, 'w') as fh:
    # checking numbers from n_start to n_end
```

```
    for i in range (n_start, n_end + 1):
        # resetting parameter "count"
        count = 0
        # examining if the number is divisible by numbers between 2 and (i-1)
        for j in range (2, i):
            # if the number is divisible, then adding 1 to "count"
            if (i % j == 0):
                count += 1
                # if count is >= 1, the number is not a prime number
                break
        if (count == 0):
            # if the number is not divisible by numbers between 2 and (i-1),
            # then it is a prime number
            result = f'{i} is a prime number.\n'
        else:
            # if the number is divisible by at least one of numbers between
            # 2 and (i-1), then it is not a prime number
            result = f'{i} is NOT a prime number.\n'
        # writing result into output file
        fh.write (result)
```

Execute above script.

```
% chmod a+x ai202209_s01_36.py
% ./ai202209_s01_36.py
% ls -lF primenumbers.data
-rw-r--r--  1 daisuke  taiwan  2467 Sep 10 08:21 primenumbers.data
% head primenumbers.data
2 is a prime number.
3 is a prime number.
4 is NOT a prime number.
5 is a prime number.
6 is NOT a prime number.
7 is a prime number.
8 is NOT a prime number.
9 is NOT a prime number.
10 is NOT a prime number.
11 is a prime number.
```

Try following practice.

Practice 01-33

Make your own Python script to write a file.

To learn more about reading and writing files, visit following web page. (Fig. 18)

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

8 Strings

Python offers a convenient way to manipulate strings. Here is an example.

Python Code 38: ai202209_s01_37.py

```
#!/usr/pkg/bin/python3.9

#
```

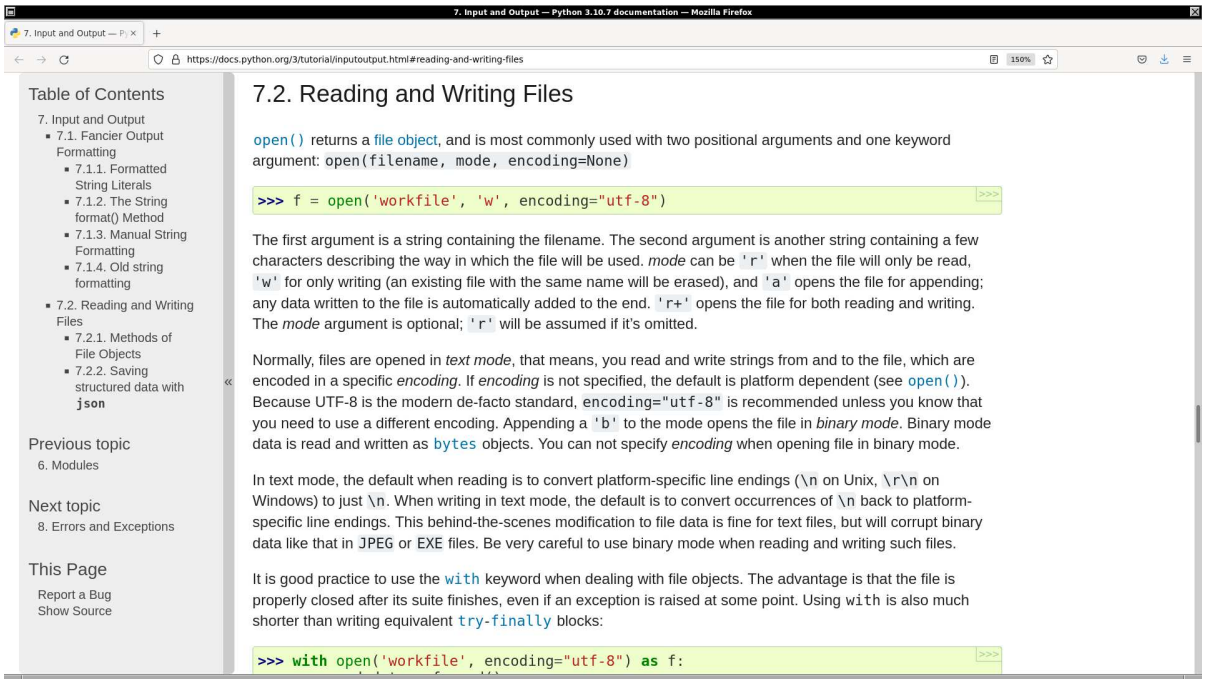

Figure 18: The section 7.2 “Reading and Writing Files” of “The Python Tutorial”.

```
# Time-stamp: <2022/09/11 17:56:00 (CST) daisuke>
#

# sample string
string1 = 'National Central University'

# printing a string
print (f'string1       = {string1}')

# using index for a string
print (f'string1[0]    = {string1[0]}')
print (f'string1[1]    = {string1[1]}')
print (f'string1[2]    = {string1[2]}')
print (f'string1[-1]   = {string1[-1]}')
print (f'string1[-2]   = {string1[-2]}')

# using slice for a string
print (f'string1[9:16] = {string1[9:16]}')
print (f'string1[9:]   = {string1[9:]}')
print (f'string1[:16]  = {string1[:16]}')

# using replace method
string2 = string1.replace ('Central', 'Tsing-Hua')
print (f'string2       = {string2}')

# concatenating strings
string3 = 'Institute'
string4 = 'of'
string5 = 'Astronomy'
string6 = string3 + ' ' + string4 + ' ' + string5
print (f"{string3} + ' ' + {string4} + ' ' + {string5} = {string6}")
```

Execute above script.

```
% chmod a+x ai202209_s01_37.py
% ./ai202209_s01_37.py
string1       = National Central University
string1[0]    = N
string1[1]    = a
string1[2]    = t
string1[-1]   = y
string1[-2]   = t
string1[9:16] = Central
string1[9:]   = Central University
string1[:16]  = National Central
string2       = National Tsing-Hua University
Institute + ' ' + of + ' ' + Astronomy = Institute of Astronomy
```

Try following practice.

Practice 01-34

Make your own Python script to play with strings.

To learn more about string manipulation, visit following web page. (Fig. 19)

- `https://docs.python.org/3/tutorial/introduction.html#strings`

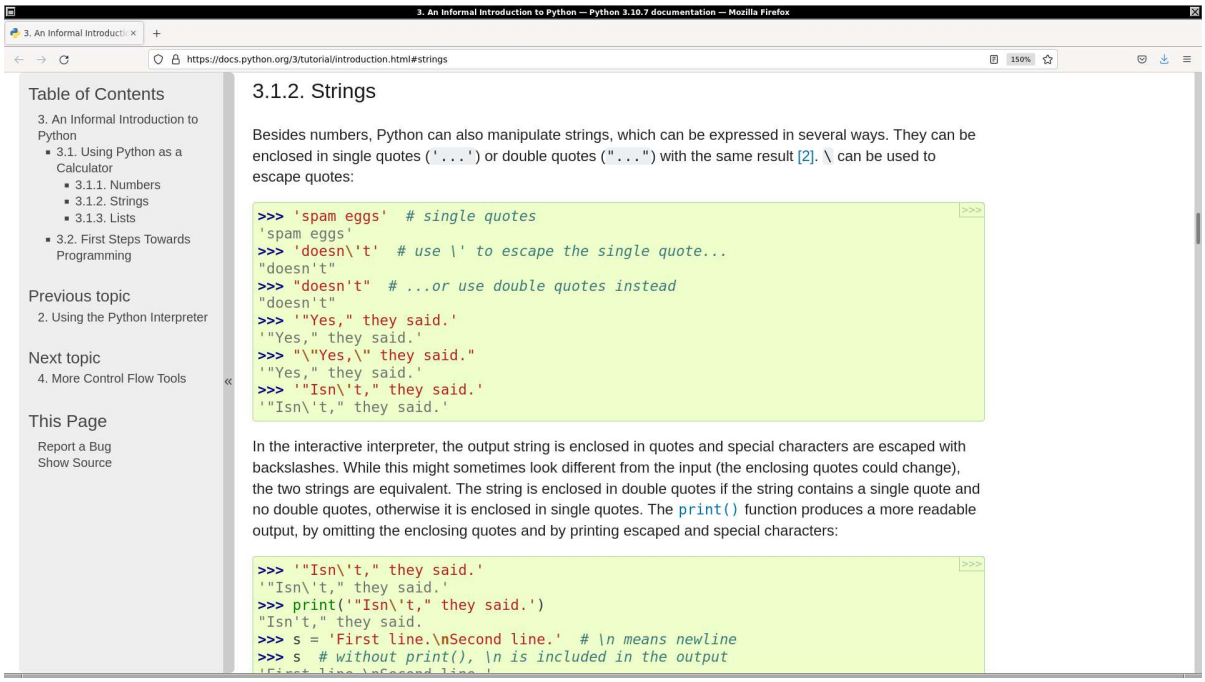

Figure 19: The section 3.1.2 “Strings” of “The Python Tutorial”.

9 Exceptions

Python script sometimes stops with an error. Some errors may be due to incorrect syntax. Some other errors may be caused by exceptions, such as “ZeroDivisionError”, “NameError”, or “TypeError”.

Here is an example of “ZeroDivisionError”.

Python Code 39: ai202209_s01_38.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/11 18:14:27 (CST) daisuke>
#

# two numbers
a = 3
b = 0

# calculation
c = a / b

# printing result of calculation
print (f'{a} / {b} = {c}')
```

Execute above script.

```
% chmod a+x ai202209_s01_38.py
% ./ai202209_s01_38.py
Traceback (most recent call last):
  File "/tmp/./ai202209_s01_38.py", line 12, in <module>
    c = a / b
ZeroDivisionError: division by zero
```

Try following practice.

Practice 01-35

Make your own Python script to cause a "ZeroDivisionError".

Here is an example of "NameError".

Python Code 40: ai202209_s01_39.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/11 18:16:37 (CST) daisuke>
#

# two numbers
a = 3

# calculation
c = a / b

# printing result of calculation
print (f'{a} / {b} = {c}')
```

Execute above script.

```
% chmod a+x ai202209_s01_39.py
% ./ai202209_s01_39.py
Traceback (most recent call last):
  File "/tmp/./ai202209_s01_39.py", line 11, in <module>
    c = a / b
NameError: name 'b' is not defined
```

Try following practice.

Practice 01-36

Make your own Python script to cause a “NameError”.

Here is an example of “TypeError”.

Python Code 41: ai202209_s01_40.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/11 18:19:19 (CST) daisuke>
#

# two numbers
a = 3
b = 'abc'

# calculation
c = a / b

# printing result of calculation
print (f'{a} / {b} = {c}')
```

Execute above script.

```
% chmod a+x ai202209_s01_40.py
% ./ai202209_s01_40.py
Traceback (most recent call last):
  File "/tmp/./ai202209_s01_40.py", line 12, in <module>
    c = a / b
TypeError: unsupported operand type(s) for /: 'int' and 'str'
```

Try following practice.

Practice 01-37

Make your own Python script to cause a “TypeError”.

We may need to handle exceptions in Python script. Here is an example of exception handling.

Python Code 42: ai202209_s01_41.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/09/11 18:30:15 (CST) daisuke>
#

# importing sys module
import sys

# printing a message
print (f'This Python script adds two integers and print result of calculation.')

# receiving two numbers
a = input ('input first integer: ')
b = input ('input second integer: ')
```

```
# conversion of string "a" into integer
try:
    # conversion
    a = int (a)
except:
    # printing a message
    print (f'Error: string "{a}" cannot be converted into an integer.')
    # exit the script
    sys.exit (1)
else:
    print (f'string "{a}" is successfully converted into an integer.')

# conversion of string "b" into integer
try:
    # conversion
    b = int (b)
except:
    # printing a message
    print (f'Error: string "{b}" cannot be converted into an integer.')
    # exit the script
    sys.exit (1)
else:
    print (f'string "{b}" is successfully converted into an integer.')

# calculation
c = a + b

# printing result of calculation
print (f'{a} + {b} = {c}')
```

Execute above script.

```
% chmod a+x ai202209_s01_41.py
% ./ai202209_s01_41.py
This Python script adds two integers and print result of calculation.
input first integer: abc
input second integer: 3
Error: string "abc" cannot be converted into an integer.
% ./ai202209_s01_41.py
This Python script adds two integers and print result of calculation.
input first integer: 2
input second integer: def
string "2" is successfully converted into an integer.
Error: string "def" cannot be converted into an integer.
% ./ai202209_s01_41.py
This Python script adds two integers and print result of calculation.
input first integer: 5
input second integer: 6
string "5" is successfully converted into an integer.
string "6" is successfully converted into an integer.
5 + 6 = 11
```

Try following practice.

Practice 01-38

Make your own Python script to handle exceptions using `try`, `except`, and `else`.

To learn more about exception handling, visit following web page. (Fig. 20)

- `https://docs.python.org/3/tutorial/errors.html`

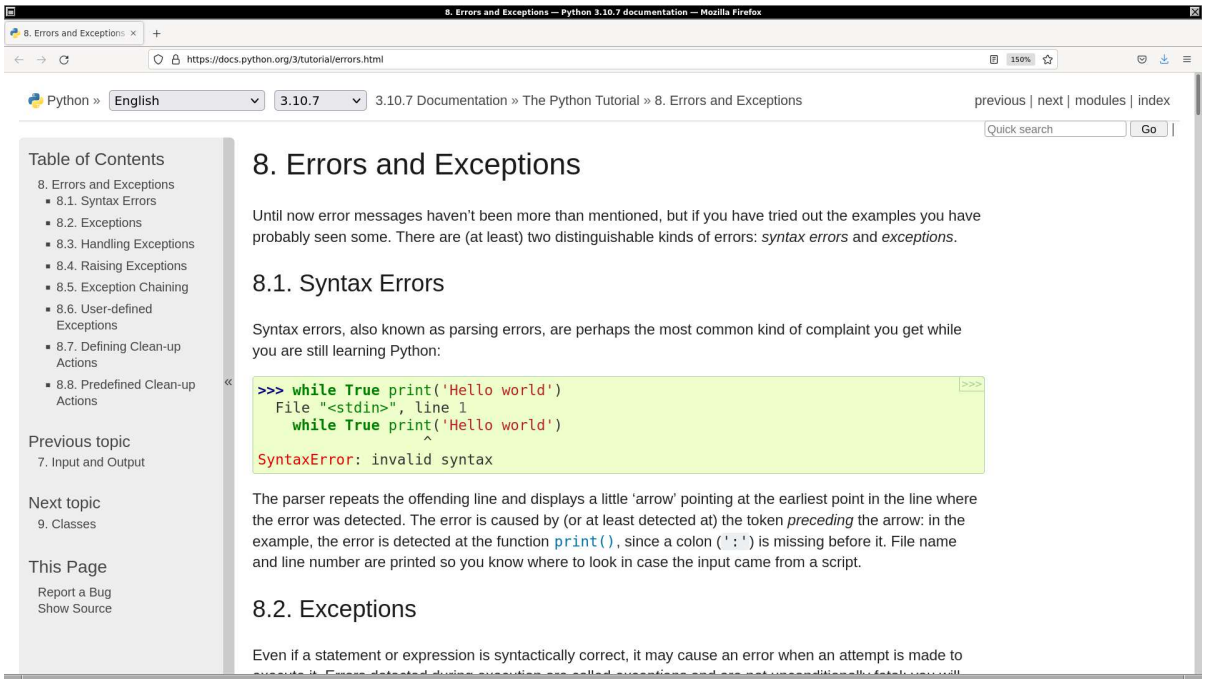

Figure 20: The section 8 "Errors and Exceptions" of "The Python Tutorial".

10 For your further reading

Python's official tutorial "The Python Tutorial" is extremely useful to learn about basic Python programming. If you take the course "Astroinformatics", you are strongly encouraged to read "The Python Tutorial".

- "The Python Tutorial": `https://docs.python.org/3/tutorial/`

To know more about Python, one of following books is highly recommended for your reading.

- "Learning Python": `https://www.oreilly.com/library/view/learning-python-5th/9781449355722/`
- "Programming Python": `https://www.oreilly.com/library/view/programming-python-4th/9781449398712/`
- "Python Cookbook": `https://www.oreilly.com/library/view/python-cookbook-3rd/9781449357337/`
- "Think Python": `https://www.oreilly.com/library/view/think-python-2nd/9781491939406/`
- "Introducing Python": `https://www.oreilly.com/library/view/introducing-python-2nd/9781492051374/`

11 Assignment

1. Read "The Python Tutorial" on the official website of Python from the beginning to the end. Try all the Python codes on "The Python Tutorial". List new things you have learnt from "The Python Tutorial" and briefly describe about them. Explain functionality you have found useful.
   - "The Python Tutorial": `https://docs.python.org/3/tutorial/`
2. A star is located 5.67 million astronomical unit from the Sun. What is the distance to this star in parsec? Write a Python script to find an answer to this question. Show the source code of your Python script. Take a screenshot of your computer display, and show the results of execution of the scripts.

3. In the spectrum of a star, Hα line has a wavelength of 656.03 nm. Find the radial velocity of this star. Write a Python script to find an answer to this question. Show the source code of your Python script. Take a screenshot of your computer display, and show the results of execution of the scripts.

4. The solar constant is 1370 W/m^2. Calculate the solar luminosity. Write a Python script to find an answer to this question. Show the source code of your Python script. Take a screenshot of your computer display, and show the results of execution of the scripts.

5. The surface temperature of Rigel is 1.6 times of that of the Sun. The luminosity of Rigel is 64,000 $L_\odot$. Calculate the radius of Rigel. Write a Python script to find an answer to this question. Show the source code of your Python script. Take a screenshot of your computer display, and show the results of execution of the scripts.

6. Calculate the amount of energy generated from the annihilation of an electron and a positron. Calculate the radius of Rigel. Write a Python script to find an answer to this question. Show the source code of your Python script. Take a screenshot of your computer display, and show the results of execution of the scripts.

7. Count the number of prime numbers up to 1,000,000. Show the source code of your Python script. Take a screenshot of your computer display, and show the results of execution of the scripts.