SWCON104
Web & Python Programming

File

Department of Software Convergence

경희대학교
KYUNG HEE UNIVERSITY

Today

- File I/O

[Textbook]
Practical Programming
(An Introduction to Computer Science Using Python),
by Paul Gries, Jennifer Campbell, Jason Montojo,
The Pragmatic Bookshelf, 2017

Practice

- Practice_13_File.ipynb

How to get data?

$$\text{Body Mass Index (bmi)} = \frac{\text{weight (kg)}}{(\text{height (m)})^2}$$

```
if age < 45:
    if bmi < 22.0:
        risk = 'low'
    else:
        risk = 'medium'
else:
    if bmi < 22.0:
        risk = 'medium'
    else:
        risk = 'high'
```

Mike, 24, 172, 72
Jane, 51, 160, 60
Jason, 35, 180, 80
…

Recommended CSV format

Name: Mike
Age: 24
Height: 172
Weight: 72

Name: Jane
Age: 51
Height: 160
Weight: 60

Name: Jason

Mike, Jane, Jason;
24, 51, 35;
172, 160, 180;
72, 60, 80;
…

Mike 24 172 72; Jane 51 160 60; Jason 35 180 80;
…

What kind of files are there?

- Text files
- Music files
- Videos
- Word processor (docx, hwp)
- Presentation documents (ppt)
- Spread sheets (excel)
- pdf

- Text files
  - Only contain characters
  - Other file formats include formatting information that is specific to that particular file format
    - Ex) You cannot open a ppt file using notepad
    - Ex) Check the size of an empty file of various format

Text files

- Take up little disk space
- Easy to process
- Only letters in a file

- .py file is a text file
  - With a particular syntax
  - Python interpreter can read Python text files and follow the instructions

- Web browsers read and process HTML files
- Spreadsheets read and process comma-separated value files
- Calendar programs read and process calendar data files
- Other programming language applications read and process files written with a particular programming language syntax

Opening a file

- Python assumes that the file you want to read is in the same directory as the current program

1) Make a directory, file_examples
2) Open Visual Code and type the following:

```
First line of text
Second line of text
Third line of text
```

3) Save this file in your file_examples directory as file_example.txt
4) Open new file and type this program:
5) Save this as file_reader.py

   in file_examples directory

6) Run

```
file = open('file_example.txt','r')
contents = file.read()
print(contents)
file.close()
```

Opening a file

file name

file mode

```
file = open('file_example.txt','r')
contents = file.read()
print(contents)
file.close()
```

- Built-in function open opens a file and returns an object that knows…
- How to get information from the file
- How much you've read
- Which part of the file you're about to read next

file cursor

```
First line of text
Second line of text
Third line of text
```

- A file cursor is a marker that keeps track of the current location in the file.
- The file cursor is initially at the beginning of the file.
- As we read or write data, it moves to the end of what we just read or wrote.

The with statement

```
file = open('file_example.txt','r')

contents = file.read()

print(contents)

file.close()
```

```
with open('file_example.txt', 'r') as file:
    contents = file.read()

print(contents)
```

The general form of a with statement is as follows:

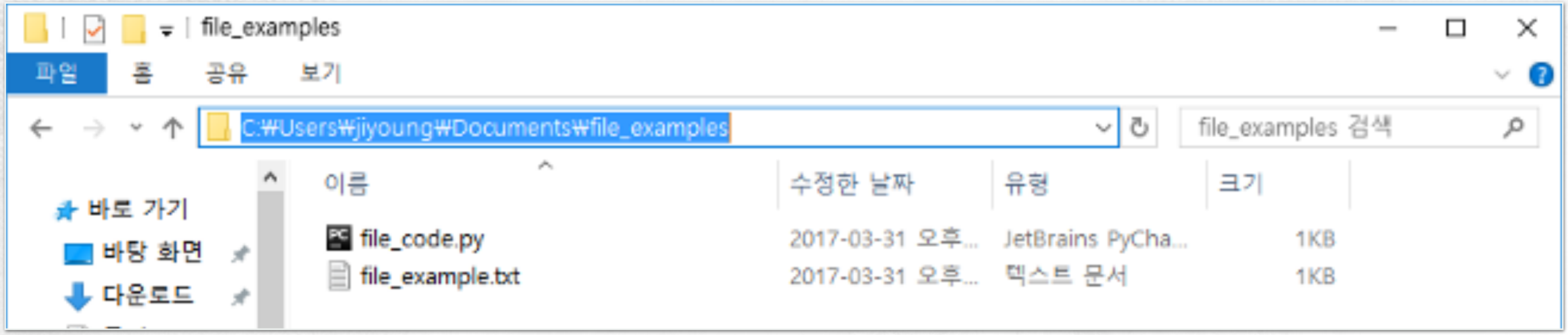
```
with open(«filename», «mode») as «variable»:
    «block»
```

- Automatically closes a file when the end of the block is reached

How files are organized in your computer

- A file path specifies a location in your computer's file system.
- File path for file_example.txt:

  C:\Users\jiyoung\Documents\file_examples\file_example.txt



How files are organized in your computer

```
>>> path = "C:\Users\jiyoung\Documents\file_examples\file_example.txt"
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in positio
n 2-3: truncated \UXXXXXXXX escape
```

```
>>> path = "C:\\Users\\jiyoung\\Documents\\file_examples\\file_example.txt"
>>> path
'C:\\Users\\jiyoung\\Documents\\file_examples\\file_example.txt'
>>> print(path)
C:\Users\jiyoung\Documents\file_examples\file_example.txt
```

```
>>> file = open(path,'r')
>>> c = file.read()
>>> file.close()
>>> c
'first line of text\nsecond line of text\nthird line of text'
```

```
>>> path2 = "C:/Users/jiyoung/Documents/file_examples/file_example.txt"
>>> file = open(path2,'r')
>>> c2 = file.read()
>>> file.close()
>>> print(c2)
first line of text
second line of text
third line of text
```

Specifying which file you want

- Current working directory
- The directory where Python looks for files
- The directory where the current program (.py file) is saved

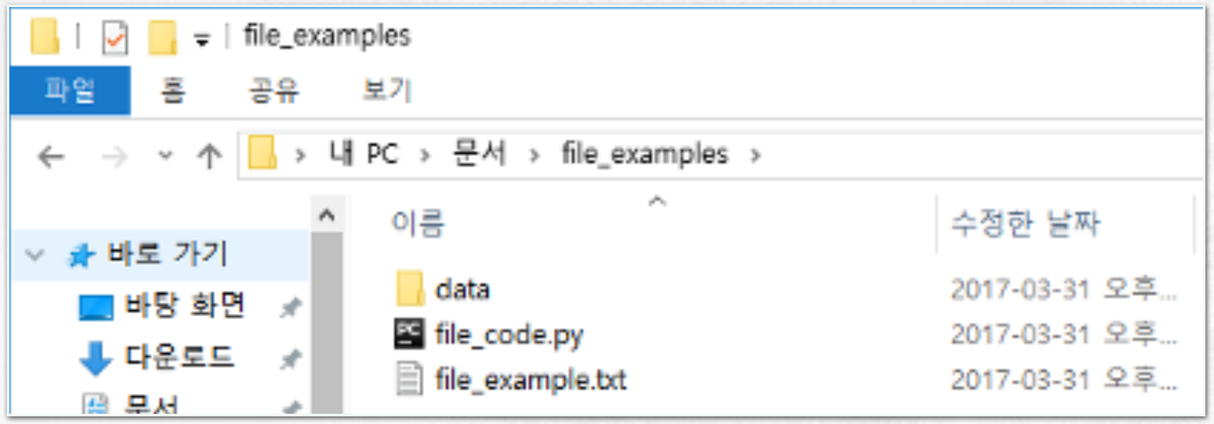
```
>>> import os
>>> os.getcwd()
'C:\\Users\\jiyoung\\AppData\\Local\\Programs\\Python\\Python35'
```

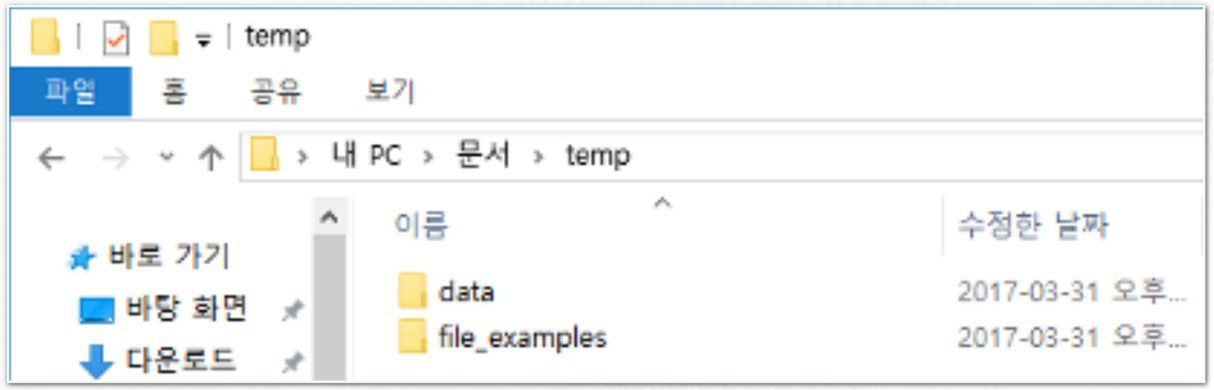
- An absolute path starts at the root directory of the file system

```
>>> import os
>>> os.chdir('F:\\course\\2017s_python\\scripts\\fileio')
>>> os.getcwd()
'F:\\course\\2017s_python\\scripts\\fileio'
```

Specifying which file you want

```
file = open('file_example.txt','r')
contents = file.read()
print(contents)
file.close()
```






```
file = open('data/data1.txt','r')
file = open('../data/data1.txt','r')
file = open('../../../data/data1.txt','r')
```

- Relative path is relative to the current working directory

Techniques for reading files

- read

```
file = open('file_example.txt','r')
contents = file.read()
file.close()
print(contents)
```

- readlines

```
file = open('file_example.txt','r')
contents = file.readlines()
file.close()
print(contents)
```

```
file = open('file_example.txt','r')
first_ten_characters = file.read(10)
the_rest = file.read()
file.close()
print("first:",first_ten_characters)
print("the rest:",the_rest)
```

readlines

planets.txt
Mercury Venus Earth Mars

file = open('planets.txt','r')
planets = file.readlines()
file.close()

```
==== RESTART: C:\Users\jiyoung\Documents\temp\
file_examples\file_code.py ====
>>> planets
['Mercury\n', 'Venus\n', 'Earth\n', 'Mars\n']
```

```
>>> for p in planets:
        print(p.strip())


Mercury
Venus
Earth
Mars
```

```
>>> for p in reversed(planets):
        print(p.strip())


Mars
Earth
Venus
Mercury
```

```
>>> for p in sorted(planets):
        print(p.strip())


Earth
Mars
Mercury
Venus
```

for line in file

planets.txt
Mercury Venus Earth Mars

```
file = open('planets.txt','r')
for line in file:
    print(len(line))
file.close()
```

```
==== RESTART: C:\Users\jiyoung\Documents\temp\
file_examples\file_code.py ====
8
6
6
5
```

```
file = open('planets.txt','r')
for line in file:
    print(len(line.strip()))
file.close()
```

```
==== RESTART: C:\Users\jiyoung\Documents\temp\
file_examples\file_code.py ====
7
5
5
4
```

Skipping the header

Skip the first line in the file

Skip over the comment lines

For each of the remaining lines:

    Process the data on that line

Skip the first line in the file

Find and process the first line of data

For each of the remaining lines:

    Process the data on that line

readline

hopedale.txt

Coloured fox fur production, HOPEDALE, Labrador, 1834-1842
#Source: C. Elton (1942) "Voles, Mice and Lemmings", Oxford Univ. Press
#Table 17, p.265--266
22
29
2
16
12
35
8
83
166

```
hopedale_file = open('hopedale.txt', 'r')

hopedale_file.readline()   # header

data = hopedale_file.readline().strip()
while data.startswith('#'):
    data = hopedale_file.readline().strip()


for data in hopedale_file:
    print(data)

hopedale_file.close()
```

readline

hopedale.txt

```
Coloured fox fur production, HOPEDALE, Labrador, 1834-1842
#Source: C. Elton (1942) "Voles, Mice and Lemmings", Oxford Univ. Press
#Table 17, p.265--266
22
29
2
16
12
35
8
83
166
```

```
hopedale_file = open('hopedale.txt', 'r')

hopedale_file.readline()   # header

data = hopedale_file.readline().strip()
while data.startswith('#'):
    data = hopedale_file.readline().strip()


total_pelts = int(data)

for data in hopedale_file:
    total_pelts = total_pelts + int(data.strip())

hopedale_file.close()

print("Total number of pelts:", total_pelts)
```

readline

hopedale.txt

Coloured fox fur production, HOPEDALE, Labrador, 1834-1842
#Source: C. Elton (1942) "Voles, Mice and Lemmings", Oxford Univ. Press
#Table 17, p.265--266
22
29
2
16
12
35
8
83
166

```
hopedale_file = open('hopedale.txt', 'r')

line = hopedale_file.readline()   # header
```

```
>>> type(hopedale_file)
<class '_io.TextIOWrapper'>
>>> type(line)
<class 'str'>
```

Files over the internet

- Target: http://robjhyndman.com/tsdldata/ecology1/hopedale.dat

```
import urllib.request
url = 'http://robjhyndman.com/tsdldata/ecology1/hopedale.dat'
webpage = urllib.request.urlopen(url)
line = webpage.readline()
```

```
line = line.strip()
line = line.decode('utf-8')
print(line)
```

```
import urllib.request
url = 'http://robjhyndman.com/tsdldata/ecology1/hopedale.dat'
webpage = urllib.request.urlopen(url)
for line in webpage:
    line = line.strip()
    line = line.decode('utf-8')
    print(line)
webpage.close()
```

```
>>> type(webpage)
<class 'http.client.HTTPResponse'>
>>> type(line)
<class 'bytes'>
```

Writing files

```
outfile = open('topics.txt','w')
outfile.write('Computer Science')
outfile.close()

outfile = open('topics.txt','w')
word = outfile.write('Computer Science')
outfile.close()
```

```
outfile = open('topics.txt','a')
outfile.write('Software Engineering')
outfile.close()
```

Computer ScienceSoftware
Engineering

Reading and writing files

total.py

```
def sum_number_pairs(input_file, output_filename):

    """(file open for reading, str) -> NoneType

    Read the data from input_file, which contains two floats
    per line separated by a space. Open file named output_file
    and, for each line in input_file, write a line to the output
    file that contains the two floats from the corresponding
    line of input_file plus a space and the sum of the two floats.
    """
    output_file = open(output_filename,'w')
    for number_pair in input_file:
        number_pair = number_pair.strip()
        operands = number_pair.split()
        total = float(operands[0])+ float(operands[1])
        new_line = '{0} {1}\n'.format(number_pair, total)
        output_file.write(new_line)
    output_file.close()
```

number_pairs.txt

```
1.3 3.4
2 4.2
-1 1
```

out.txt

```
1.3 3.4 4.7
2 4.2 6.2
-1 1 0.0
```

```
>>> import total
>>> total.sum_number_pairs(open('number_pairs.txt','r'),'out.txt')
```

Summary

- When files are opened and read, their contents are commonly stored in lists of strings.
- Data stored in files is usually formatted in one of a small number of ways, from one value per line to multiline records with explicit end-of-record markers. Each format can be processed in a stereotypical way.
- Data processing programs should be broken into input, processing, and output stages so that each can be reused independently.
- Files can be read (content retrieved), written to (content replaced), and added to (new content appended). When a file is opened in writing mode and it doesn't exist, a new file is created.

Summary

- Data files come in many different formats, so custom code is often required, but we can reuse as much as possible by writing helper functions
- To make the functions usable by different types of readers, the reader (for a file or web page) is opened outside the function, passed as an argument to the function, and then closed outside the function.

Thank you

경희대학교
KYUNG HEE UNIVERSITY