Lecture 3—File Reading and Writing

In this lecture, we will explore reading and writing plain text files.

We will learn:

Reading plain text file

We can use `open()` to read and write plain text file. There are 3 modes when opening a file:

- `r` for reading.
- `w` for over-writing.
- `a` for appending.

In [1]:
```
with open("quotes.txt") as file_obj:
    quotes = file_obj.read().splitlines()

quotes
```

Out[1]:
```
['I want to put a ding in the universe.—Steve Jobs',
 'Life is 10% what happens to you and 90% how you react to it.—Charles R. S
windoll',
 "Family is not an important thing. It's everything.—Michael J. Fox",
 "Nothing is impossible, the word itself says 'I'm possible'!—Audrey Hepbur
n",
 'There are two ways of spreading light: to be the candle or the mirror tha
t reflects it.—Edith Wharton',
 "Try to be a rainbow in someone's cloud.—Maya Angelou",
 'Be brave enough to live life creatively. The creative place where no one
else has ever been.—Alan Alda',
 'The secret of getting ahead is getting started.—Mark Twain']
```

In [2]:
```
import random

quote = random.choice(quotes)

print(quote)
```

```
Family is not an important thing. It's everything.—Michael J. Fox
```

Reading CSV file

We can further improve the storage by using CSV file format.

In [3]:
```
import csv

with open("quotes.csv") as file_obj:
    csv_reader = csv.reader(file_obj)
    for line in csv_reader:
        print(line)
```

```
['I want to put a ding in the universe.', 'Steve Jobs']
['Life is 10% what happens to you and 90% how you react to it.', 'Charles R
. Swindoll']
["Family is not an important thing. It's everything.", 'Michael J. Fox']
["Nothing is impossible,the word itself says 'I'm possible'!", 'Audrey Hepb
urn']
['There are two ways of spreading light: to be the candle or the mirror tha
t reflects it.', 'Edith Wharton']
["Try to be a rainbow in someone's cloud.", 'Maya Angelou']
['Be brave enough to live life creatively. The creative place where no one
else has ever been.', 'Alan Alda']
['The secret of getting ahead is getting started.', 'Mark Twain']
```

Using Pandas to read CSV file

When using Pandas, we can read the CSV file into `DataFrame`.

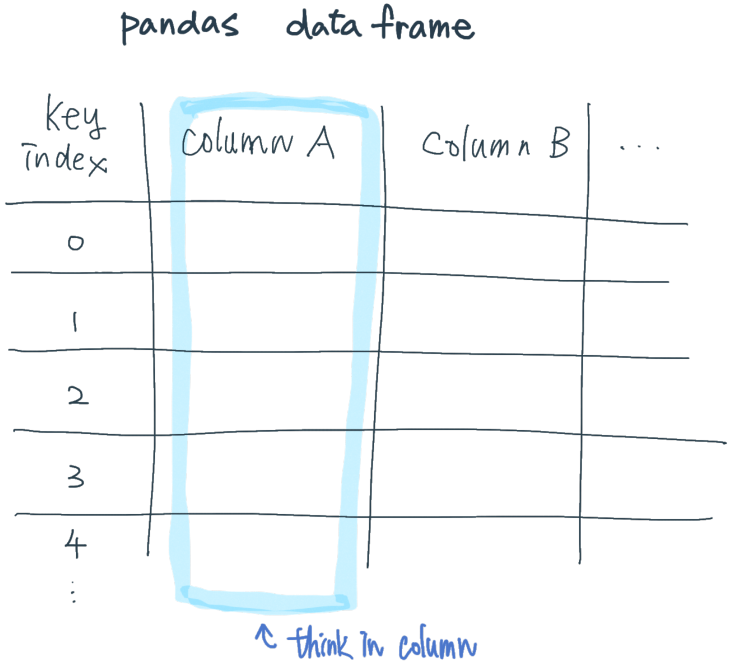
In [4]:
```
import pandas as pd

pd.read_csv("quotes.csv")
```

Out[4]:

	I want to put a ding in the universe.	Steve Jobs
0	Life is 10% what happens to you and 90% how yo...	Charles R. Swindoll
1	Family is not an important thing. It's everyth...	Michael J. Fox
2	Nothing is impossible,the word itself says 'I'...	Audrey Hepburn
3	There are two ways of spreading light: to be t...	Edith Wharton
4	Try to be a rainbow in someone's cloud.	Maya Angelou
5	Be brave enough to live life creatively. The c...	Alan Alda
6	The secret of getting ahead is getting started.	Mark Twain

If you encounter error saying that pandas not found. You may need to install pandas via `pip install pandas`. If you're using Anaconda, it already comes Pandas built-in.



Another examle of reading MPay daily CSV file via Pandas

I have prepared a CSV file named `[ABCD]2020-07-23.csv` . It is the CSV file from MPay every day listing all the transactions.

In [4]:
```
import pandas as pd

df = pd.read_csv('[ABCD]2020-07-23.csv')
```

Let's inspect the result of read_csv. We call it *data frame*.

In [5]:
```
df
```

Out[5]:

	S/N	Merchant No.	Terminal No.	Transaction Channel	Transaction Type	Account No	T
0	1	ABCD	123	MacauPass	Pay(Off Line)	6045543888	
1	2	ABCD	123	MacauPass	Pay(Off Line)	6032888888	
2	3	ABCD	123	MPay	Pay(Off Line)	66****00	202007228888888F1D0
3	4	ABCD	123	MPay	Pay(Off Line)	65****00	202007228888888F1D0
4	5	ABCD	123	MPay	Pay(Off Line)	66****20	202007228888888F1D0
5	6	ABCD	123	MPay	Pay(Off Line)	66****60	202007228888888F1D0
6	7	ABCD	123	MacauPass	Pay(Off Line)	6043633088	
7	8	ABCD	123	MPay	Pay(Off Line)	66****50	202007228888888F1D0
8	9	ABCD	123	MPay	Pay(Off Line)	66****80	202007228888888F1D0
9	10	ABCD	123	MacauPass	Pay(Off Line)	6023249388	
10	11	ABCD	123	MPay	Pay(Off Line)	66****80	202007228888888F1D0
11	12	ABCD	123	MPay	Pay(Off Line)	66****70	202007228888888F1D0
12	13	ABCD	123	MPay	Pay(Off Line)	62****80	202007228888888F1D0

What are the transaction amount?

In [6]:
```
df["Transaction Amount"]
```

Out[6]:
```
0     -22.0
1     -16.0
2     -20.0
3     -24.0
4     -21.6
5     -27.0
6    -115.0
7     -27.0
8     -49.5
9     -44.0
10    -44.0
11    -28.8
12    -72.0
Name: Transaction Amount, dtype: float64
```

What is the sum of the day?

In [7]:
```
df["Transaction Amount"].sum()
```

Out[7]: -510.90000000000003

👆🤔 If you are wondering why the sum is not -510.90. Please refer to the followig documentation:

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

If you really need to output the decimal numbers in particular format, you can use the `format` function.

In [21]:
```
format(df["Transaction Amount"].sum(), '.2f')
```

Out[21]: '-510.90'

How many of them use a physica MacauPass card?

In [10]:
```
mask = df["Transaction Channel"]=="MacauPass"
df[mask]
```

Out[10]:

	S/N	Merchant No.	Terminal No.	Transaction Channel	Transaction Type	Account No	Transaction ID	Channe Transactio I
0	1	ABCD	123	MacauPass	Pay(Off Line)	6045543888	62	Na
1	2	ABCD	123	MacauPass	Pay(Off Line)	6032888888	64	Na
6	7	ABCD	123	MacauPass	Pay(Off Line)	6043633088	69	Na
9	10	ABCD	123	MacauPass	Pay(Off Line)	6023249388	72	Na

In [13]:
```
len(df[mask])
```

Out[13]: 4

In [12]:
```
df[mask]["Transaction Amount"].sum()
```

Out[12]: -197.0

Example of writing plain text file: Diary logging

In [5]:
```
import datetime

content = input("What do you want to say to Mr. Diary? ")
if len(content) > 0:
    with open('diary.txt', "a") as file_obj:
        today = datetime.date.today().isoformat()
        file_obj.write(today + ": " + content + "\n")

with open('diary.txt', "r") as file_obj:
    lines = file_obj.readlines()
    for line in lines[-3:]:
        print(line.rstrip())
```

```
What do you want to say to Mr. Diary? Hello python.
2020-06-11: Hello
2020-06-11: Hello
2020-08-10: Hello python.
```

Writing DOCX

We can use `python-docx` module to write content to DOCX.

First, we need to install the module by calling `pip install python-docx` once in terminal or in Jupyter.

In [6]:
```
pip install python-docx
```

```
Requirement already satisfied: python-docx in /private/var/mobile/Containers/Data/Application/8F879E4C-8D79-4BDE-8FB3-2E6B2D2BD6C6/Library/lib/python3.7/site-packages (0.8.10)
Requirement already satisfied: lxml>=2.3.2 in /private/var/mobile/Containers/Data/Application/8F879E4C-8D79-4BDE-8FB3-2E6B2D2BD6C6/Library/lib/python3.7/site-packages/lxml-4.4.2-py3.7-macosx-10.9-x86_64.egg (from python-docx) (4.4.2)
Note: you may need to restart the kernel to use updated packages.
```

👆🤔 If you're wondering how the above line works. It is a command executed in command line prompt. But Jupyter is smart enough to parse the `pip install` command and execute it right inside the notebook.

In [7]:
```
import datetime
import docx
import os

content = input("What do you want to say to Mr. Diary? ")
if len(content) > 0:
    with open('diary.txt', "a") as file_obj:
        today = str(datetime.date.today())
        file_obj.write(today + ": " + content + "\n")

if os.path.isfile("diary.docx"):
    doc = docx.Document("diary.docx")
else:
    doc = docx.Document()
doc.add_paragraph(content)
doc.save("diary.docx")

print(f"{content} is written to diary.docx")
```

```
What do you want to say to Mr. Diary? Hello
Hello is written to diary.docx
```

Reading DOCX file

Given that we have a DOCX file named `Sample Document.docx` . We can read all the paragrahs in the DOCX file.

In [8]:
```
import docx

doc = docx.Document("Sample Document.docx")
```

In [9]:
```
for paragraph in doc.paragraphs:
    print(paragraph.text)
```

```
Sample Document

This is a sample paragraph.

This is the second paragraph.

Here is the result

Summary

This is the summary of the sample report document.
```

Reading tables in DOCX file

We can also read the tables and the content.

In [10]:
```
doc.tables
```

Out[10]:
```
[<docx.table.Table at 0x1385d0f60>]
```

The following code read the data row by row into 3 lists: `dates` , `morning_visitors` , `evening_visitors` .

In [11]:
```
table = doc.tables[0]

dates = []
morning_visitors = []
evening_visitors = []

for row in table.rows[1:]:
    dates.append(row.cells[0].text)
    morning_visitors.append(int(row.cells[1].text))
    evening_visitors.append(int(row.cells[2].text))

dates
```

Out[11]:
```
['2020-06-01',
 '2020-06-02',
 '2020-06-03',
 '2020-06-04',
 '2020-06-05',
 '2020-06-06',
 '2020-06-07']
```

In [12]:
```
morning_visitors
```

Out[12]:
```
[23, 25, 24, 26, 25, 24, 23]
```

In [13]:
```
sum(morning_visitors)
```

Out[13]:
```
170
```

In [14]:
```
evening_visitors
```

Out[14]:
```
[17, 16, 16, 15, 16, 17, 18]
```

In [15]:
```
sum(morning_visitors) + sum(evening_visitors)
```

Out[15]:
```
285
```

Optional: Zipping multiple-list into a multi-columns list

The following section about zipping and data frame is a preview of data processing. We will go through detail usage of pandas and data frame in lecture 8.

Have 3 separated lists for the same purpose of data makes it hard to maintain. Usually we want to have a tabular data to hold all related data into rows and columns.

In [16]:
```
zipped_list = list(zip(dates, morning_visitors, evening_visitors))

zipped_list
```

Out[16]:
```
[('2020-06-01', 23, 17),
 ('2020-06-02', 25, 16),
 ('2020-06-03', 24, 16),
 ('2020-06-04', 26, 15),
 ('2020-06-05', 25, 16),
 ('2020-06-06', 24, 17),
 ('2020-06-07', 23, 18)]
```

In [17]:
```
zipped_list[3]
```

Out[17]:
```
('2020-06-04', 26, 15)
```

Optional: Converting the multi-columns list into pandas DataFrame

We will learn pandas and DataFrame in lecture 8. But let's have a glimpse on how we can process multi-columns of data in handy way by using it.

In [18]:
```
import pandas as pd

df = pd.DataFrame(zipped_list, columns=['Date', 'Morning Visitors', 'Evenir
```

In [19]:
```
df
```

Out[19]:

	Date	Morning Visitors	Evening Visitors
0	2020-06-01	23	17
1	2020-06-02	25	16
2	2020-06-03	24	16
3	2020-06-04	26	15
4	2020-06-05	25	16
5	2020-06-06	24	17
6	2020-06-07	23	18

What're the Benefits of using data frame? We can perform column-based calcuations to all data at once.

In [20]:
```
df['Total'] = df['Morning Visitors'] + df['Evening Visitors']
```

In [21]:
```
df
```

Out[21]:

	Date	Morning Visitors	Evening Visitors	Total
0	2020-06-01	23	17	40
1	2020-06-02	25	16	41
2	2020-06-03	24	16	40
3	2020-06-04	26	15	41
4	2020-06-05	25	16	41
5	2020-06-06	24	17	41
6	2020-06-07	23	18	41

Optional: Process Text with Regular Expression

We can use Regular Expression to process text with patterns.

In [13]:
```
with open("sample-years.txt") as file_obj:
    lines = file_obj.read().splitlines()

lines
```

Out[13]:
```
['Steven Hawking was born in 1942.',
 'Albert Einstein was born in 1879',
 'Albert Einstein won Nobel Prize in 1921.',
 'Stephen Curry wear No. 30.',
 'Stephen Curry went into NBA in 2009',
 'Stephen Curry won NBA MVP in 2015 and 2016.',
 'Micheal Jordan was born in 1963.']
```

The following code finds all the years in the text document.

In [14]:
```
import re

for line in lines:
    pattern = '\d{4}'
    print(re.findall(pattern, line))
```

```
['1942']
['1879']
['1921']
[]
['2009']
['2015', '2016']
['1963']
```

What if we only want the first year found?

Let's try using `[0]` to get the first result for each line. And then, we have an error:

In [18]:
```
import re

for line in lines:
    pattern = '\d{4}'
    print(re.findall(pattern, line)[0])
```

```
1942
1879
1921
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-18-f0bcda632400> in <module>
      3 for line in lines:
      4     pattern = '\d{4}'
----> 5     print(re.findall(pattern, line)[0])

IndexError: list index out of range
```

The error occurs because there is one line that failed to find any year result.

We can ensure there is empty result by searching the ending of line too. This result in an extra result in every reuslt:

In [20]:
```
import re

for line in lines:
    pattern = '\d{4}|$'
    print(re.findall(pattern, line))
```

```
['1942', '']
['1879', '']
['1921', '']
['']
['2009', '']
['2015', '2016', '']
['1963', '']
```

But it is useful if we need to ensure the first result.

In [21]:
```
import re

for line in lines:
    pattern = '\d{4}|$'
    print(re.findall(pattern, line)[0])
```

```
1942
1879
1921

2009
2015
1963
```

The following code finds all the names in the text document

In [15]:
```
import re

for line in lines:
    pattern = '[A-Z][a-z]* [A-Z][a-z]*'
    print(re.findall(pattern, line))
```

```
['Steven Hawking']
['Albert Einstein']
['Albert Einstein', 'Nobel Prize']
['Stephen Curry']
['Stephen Curry']
['Stephen Curry', 'A M']
['Micheal Jordan']
```

In [16]:
```
import re

for line in lines:
    pattern = '[A-Z][a-z]+ [A-Z][a-z]+'
    print(re.findall(pattern, line))
```

```
['Steven Hawking']
['Albert Einstein']
['Albert Einstein', 'Nobel Prize']
['Stephen Curry']
['Stephen Curry']
['Stephen Curry']
['Micheal Jordan']
```

In [17]:
```
import re

for line in lines:
    pattern = '^[A-Z][a-z]+ [A-Z][a-z]+'
    print(re.findall(pattern, line))
```

```
['Steven Hawking']
['Albert Einstein']
['Albert Einstein']
['Stephen Curry']
['Stephen Curry']
['Stephen Curry']
['Micheal Jordan']
```

You can read more examples of using Regular Expression on Programiz.com.

Optional: Combining DataFrame and Regex

We can combine data frame and regular expression to perform column-based operation to all data at once.

In [3]:
```
import pandas as pd

df = pd.read_csv('sample-years.txt', header=None, names=['Original Text'])

df
```

Out[3]:

	Original Text
0	Steven Hawking was born in 1942.
1	Albert Einstein was born in 1879
2	Albert Einstein won Nobel Prize in 1921.
3	Stephen Curry wear No. 30.
4	Stephen Curry went into NBA in 2009
5	Stephen Curry won NBA MVP in 2015 and 2016.
6	Micheal Jordan was born in 1963.

Now that we loaded the text into a column, we can create a new column that applies our own transformation.

We define the function that find first year and name given the string parameter input.

In [4]:
```
def find_first_year(string):
    pattern = '\d{4}|$'
    return re.findall(pattern, string)[0]

def find_first_name(string):
    pattern = '^[A-Z][a-z]+ [A-Z][a-z]+|$'
    return re.findall(pattern, string)[0]
```

In [5]:
```
import re

df["Years"] = df['Original Text'].apply(find_first_year)
df["Name"] = df['Original Text'].apply(find_first_name)

df
```

Out[5]:

	Original Text	Years	Name
0	Steven Hawking was born in 1942.	1942	Steven Hawking
1	Albert Einstein was born in 1879	1879	Albert Einstein
2	Albert Einstein won Nobel Prize in 1921.	1921	Albert Einstein
3	Stephen Curry wear No. 30.		Stephen Curry
4	Stephen Curry went into NBA in 2009	2009	Stephen Curry
5	Stephen Curry won NBA MVP in 2015 and 2016.	2015	Stephen Curry
6	Micheal Jordan was born in 1963.	1963	Micheal Jordan

In [6]:
```
df.sort_values(by="Years")
```

Out[6]:

	Original Text	Years	Name
3	Stephen Curry wear No. 30.		Stephen Curry
1	Albert Einstein was born in 1879	1879	Albert Einstein
2	Albert Einstein won Nobel Prize in 1921.	1921	Albert Einstein
0	Steven Hawking was born in 1942.	1942	Steven Hawking
6	Micheal Jordan was born in 1963.	1963	Micheal Jordan
4	Stephen Curry went into NBA in 2009	2009	Stephen Curry
5	Stephen Curry won NBA MVP in 2015 and 2016.	2015	Stephen Curry

In [7]:
```
df.sort_values(by="Name")
```

Out[7]:

	Original Text	Years	Name
1	Albert Einstein was born in 1879	1879	Albert Einstein
2	Albert Einstein won Nobel Prize in 1921.	1921	Albert Einstein
6	Micheal Jordan was born in 1963.	1963	Micheal Jordan
3	Stephen Curry wear No. 30.		Stephen Curry
4	Stephen Curry went into NBA in 2009	2009	Stephen Curry
5	Stephen Curry won NBA MVP in 2015 and 2016.	2015	Stephen Curry
0	Steven Hawking was born in 1942.	1942	Steven Hawking

More on sort_values on pandas documentation.

Summary

In this section, we learned to read and write plain text file. We also read and write DOCX file. Then, we process the tablular data read from the DOCX file into nested list and even pandas DataFrame.

Furthermore, you may read more examples about reading excel document on AutomateTheBoringStuff.com. And examples about reading PDF/DOCX document.

We will also use pandas to read Excel data into DataFrame in Lecture 8.