Python Programming

Lecture 12 Downloading Data

12.1 Downloading Data (1)

.CSV

```
1  from pathlib import Path
2  import csv
3
4  path = Path('sitka_weather_07-2021_simple.csv')
5  lines = path.read_text().splitlines()
6
7  reader = csv.reader(lines)
8  header_row = next(reader)
9  print(header_row)
10
11 for index, column_header in enumerate(header_row):
12     print(index,column_header)
```

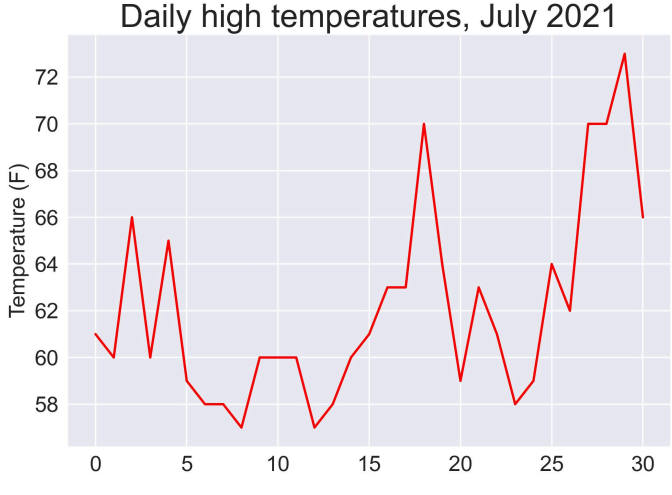
```
['STATION', 'NAME', 'DATE', 'TAVG', 'TMAX', 'TMIN']
0 STATION
1 NAME
2 DATE
3 TAVG
4 TMAX
5 TMIN
```

Extracting and Reading Data

```
 1 from pathlib import Path
 2 import csv
 3
 4 path = Path('sitka_weather_07-2021_simple.csv')
 5 lines = path.read_text().splitlines()
 6
 7 reader = csv.reader(lines)
 8 header_row = next(reader)
 9
10 highs = []
11 for row in reader:
12     high = int(row[4])
13     highs.append(high)
14
15 print(highs)
```

```
[61, 60, 66, 60, 65, 59, 58, 58, 57, 60, 60, 60, 57, 58, 60, 61, 63, 63,
70, 64, 59, 63, 61, 58, 59, 64, 62, 70, 70, 73, 66]
```

```python
from matplotlib import pyplot as plt

# Plot the high temperatures.
plt.style.use('seaborn')
fig, ax = plt.subplots()
ax.plot(highs, c='red')

# Format plot.
ax.set_title("Daily high temperatures, July 2021", fontsize=24)
ax.set_xlabel('', fontsize=16)
ax.set_ylabel("Temperature (F)", fontsize=16)
ax.tick_params(labelsize=16)

#plt.show()
plt.savefig('simple.jpg',dpi=300)
```



The datetime Module

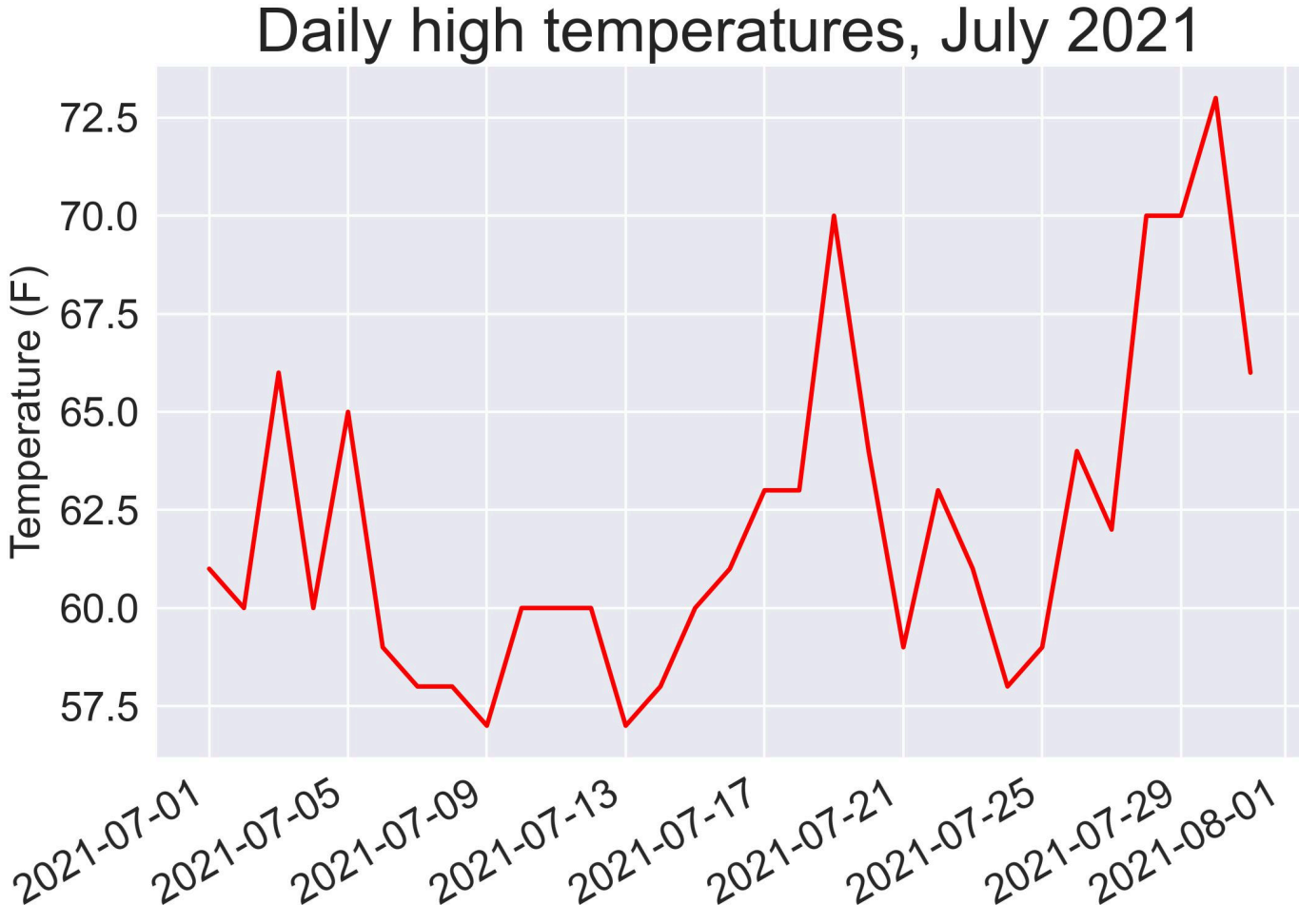
```
from datetime import datetime

first_date = datetime.strptime('2021-7-1', '%Y-%m-%d')
print(type(first_date))
print(first_date.strftime('%B %d %Y'))
print(first_date)
```

```
<class 'datetime.datetime'>
July 01 2021
2021-07-01 00:00:00
```

```
%A Weekday name, such as Monday
%B Month name, such as January
%m Month, as a number (01 to 12)
%d Day of the month, as a number (01 to 31)
%Y Four-digit year, such as 2015
%y Two-digit year, such as 15
%H Hour, in 24-hour format (00 to 23)
%I Hour, in 12-hour format (01 to 12)
%p am or pm
%M Minutes (00 to 59) %S Seconds (00 to 61)
```

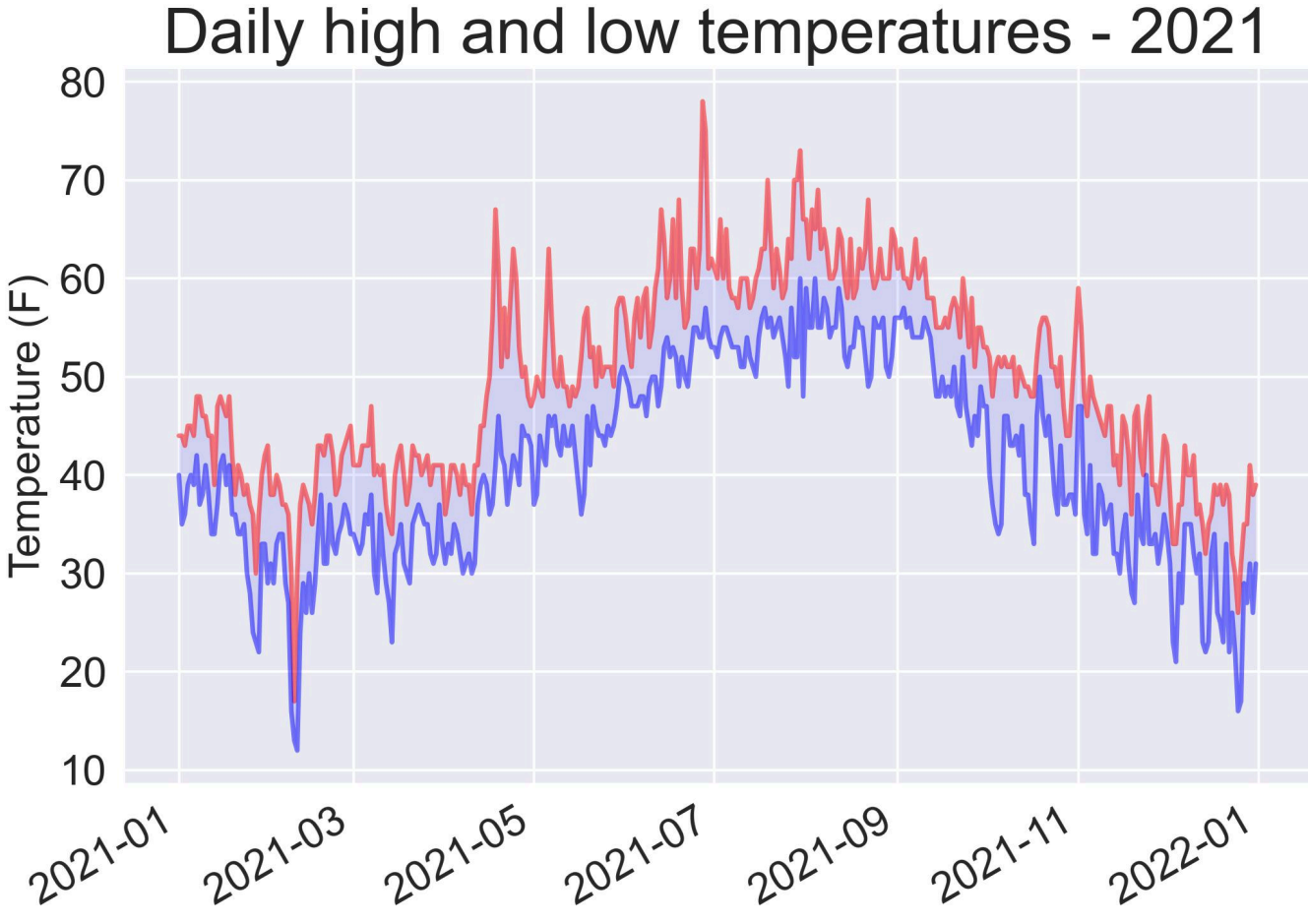
```python
from pathlib import Path
import csv
from datetime import datetime
from matplotlib import pyplot as plt

path = Path('sitka_weather_07-2021_simple.csv')
lines = path.read_text().splitlines()
reader = csv.reader(lines)
header_row = next(reader)

dates, highs = [],[]
for row in reader:
    current_date = datetime.strptime(row[2], "%Y-%m-%d")
    dates.append(current_date)
    high = int(row[4])
    highs.append(high)

plt.style.use('seaborn')
fig, ax = plt.subplots()
ax.plot(dates, highs, c='red')
```

```python
# Format plot.
ax.set_title("Daily high temperatures, July 2021", fontsize=24)
ax.set_xlabel('', fontsize=16)
fig.autofmt_xdate()
ax.set_ylabel("Temperature (F)", fontsize=16)
ax.tick_params(labelsize=16)

plt.savefig('simple.jpg',dpi=300)
```


Daily high temperatures, July 2021
Temperature (F)
72.5
70.0
67.5
65.0
62.5
60.0
57.5
2021-07-01
2021-07-05
2021-07-09
2021-07-13
2021-07-17
2021-07-21
2021-07-25
2021-07-29
2021-08-01

```
from pathlib import Path
import csv
from datetime import datetime
from matplotlib import pyplot as plt

path = Path('sitka_weather_2021_simple.csv')
lines = path.read_text().splitlines()
reader = csv.reader(lines)
header_row = next(reader)

dates, highs, lows= [], [], []
for row in reader:
    current_date = datetime.strptime(row[2], "%Y-%m-%d")
    dates.append(current_date)
    high = int(row[4])
    highs.append(high)
    low = int(row[5])
    lows.append(low)
```

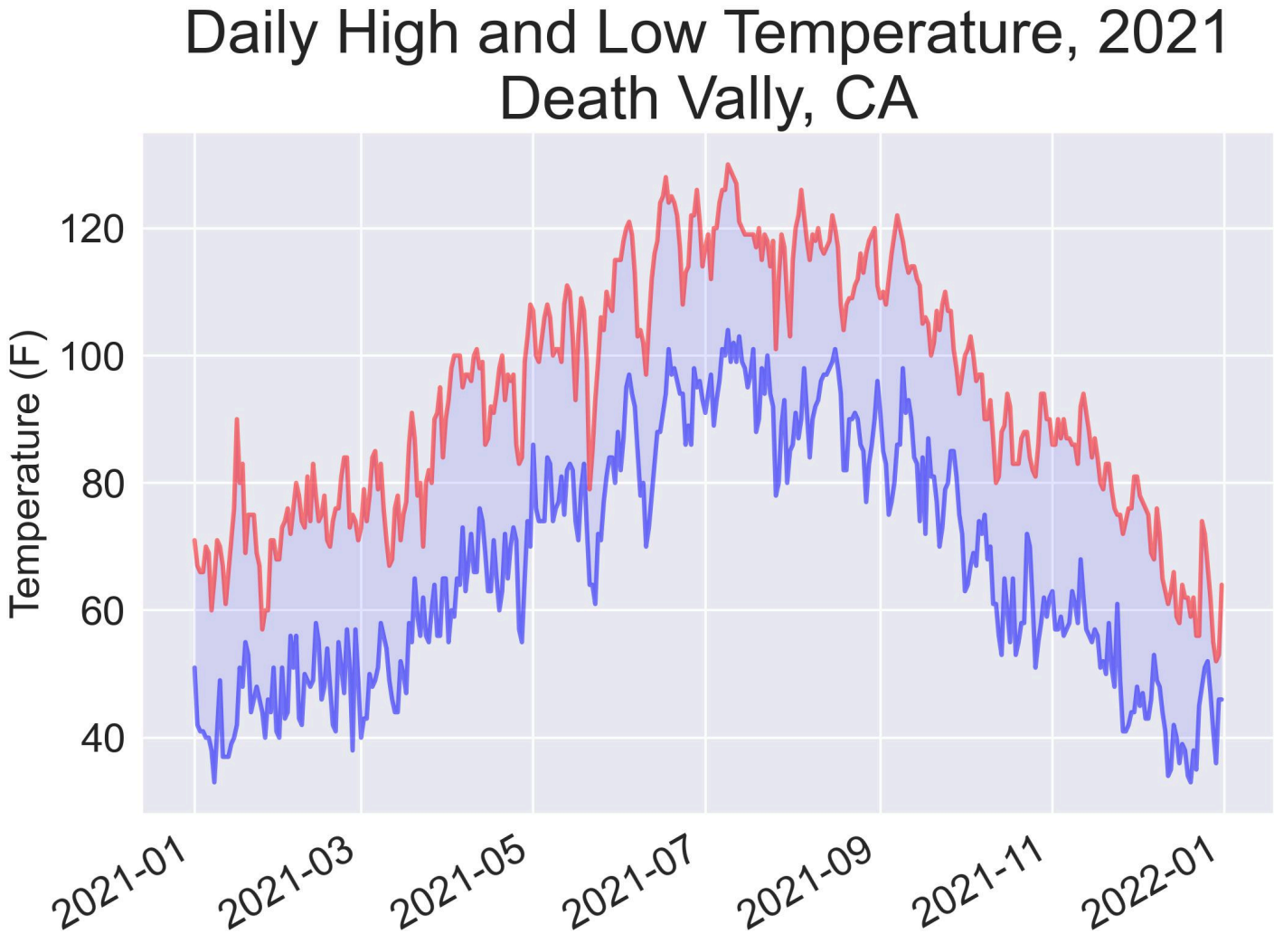
```
# Plot data.
plt.style.use('seaborn')
fig, ax = plt.subplots()
ax.plot(dates, highs, c='red', alpha=0.5)
ax.plot(dates, lows, c='blue', alpha=0.5)
ax.fill_between(dates, highs, lows, facecolor='blue', alpha=0.1)

# Format plot.
ax.set_title("Daily high and low temperatures - 2021", fontsize=24)
ax.set_xlabel('', fontsize=16)
fig.autofmt_xdate()
ax.set_ylabel("Temperature (F)", fontsize=16)
ax.tick_params(labelsize=16)

plt.savefig('simple.jpg',dpi=300)
```


Daily high and low temperatures - 2021
Temperature (F)
80
70
60
50
40
30
20
10
2021-01
2021-03
2021-05
2021-07
2021-09
2021-11
2022-01

```python
from pathlib import Path
import csv
from datetime import datetime
from matplotlib import pyplot as plt

path = Path('death_valley_2021_simple.csv')
lines = path.read_text().splitlines()
reader = csv.reader(lines)
header_row = next(reader)
dates, highs, lows= [], [], []
```

```python
# continue
for row in reader:
    try:
        current_date = datetime.strptime(row[2], "%Y-%m-%d")
        high = int(row[3])
        low = int(row[4])
    except ValueError:
        print(current_date, 'missing data')
    else:
        dates.append(current_date)
        highs.append(high)
        lows.append(low)
```

```
2021-05-04 00:00:00 missing data
```

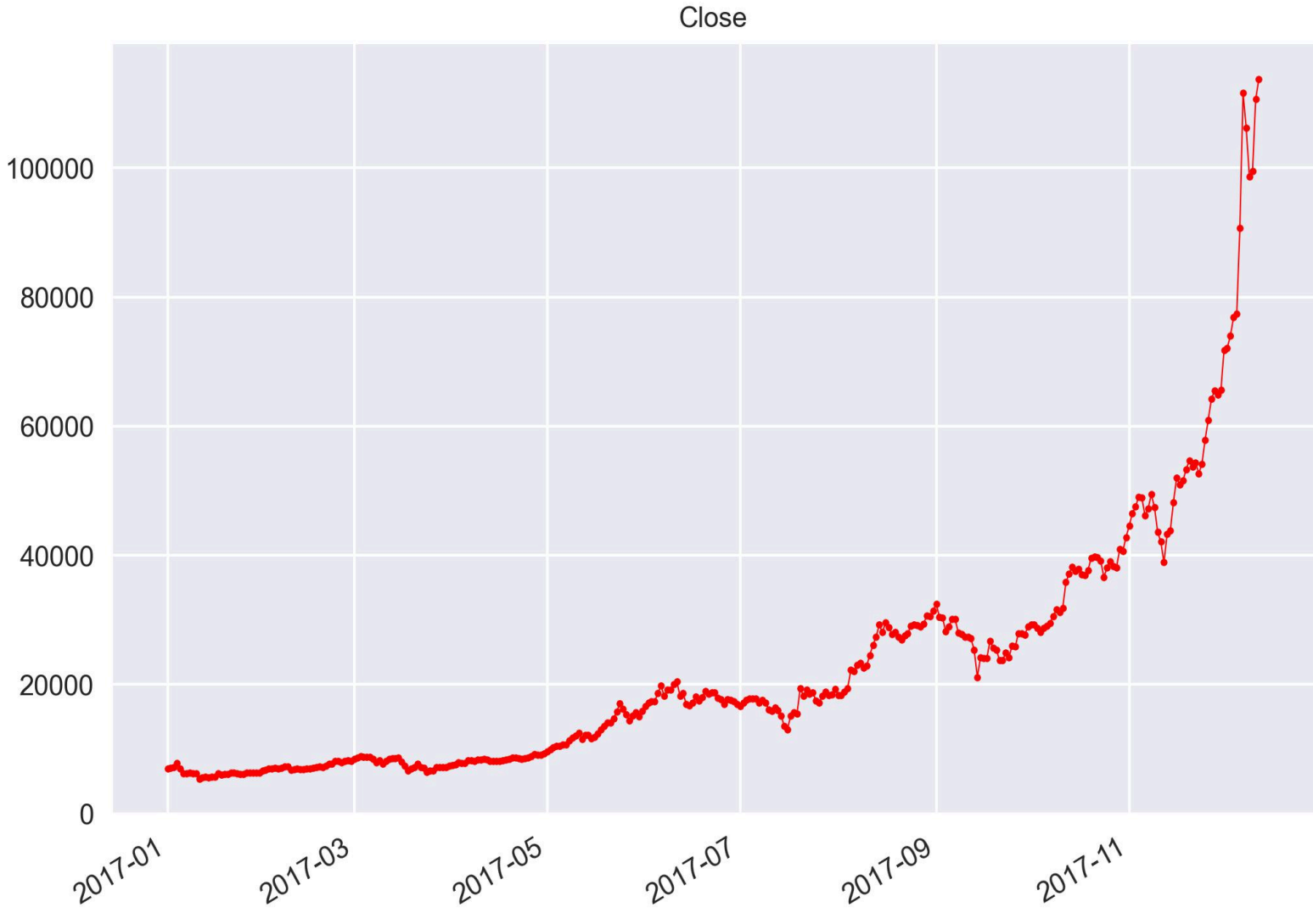

Daily High and Low Temperature, 2021
Death Vally, CA
Temperature (F)
120
100
80
60
40
2021-01
2021-03
2021-05
2021-07
2021-09
2021-11
2022-01

12.2 Downloading Data (2)

.JSON

```
from pathlib import Path
from datetime import datetime
import json

path = Path('btc_close_2017.json')
contents = path.read_text()
btc_data = json.loads(contents)

date=[]; close=[]; months=[]

for btc_dict in btc_data:
    date.append(datetime.strptime(btc_dict['date'], "%Y-%m-%d"))
    months.append(int(btc_dict['month']))
    close.append(int(float(btc_dict['close'])))
```

```
import matplotlib.pyplot as plt

plt.style.use('seaborn')
fig, ax = plt.subplots()

ax.plot(date,close, linewidth=0.5, c='red')
ax.scatter(date,close, s=5, c='red')
ax.set_title('Close',fontsize=10)
fig.autofmt_xdate()

plt.savefig('close.jpg',dpi=300)
```


Close
100000
80000
60000
40000
20000
0
2017-01
2017-03
2017-05
2017-07
2017-09
2017-11

Iterator (迭代器)

在 Python 中，迭代器（Iterator） 是一种用于遍历（或迭代）可迭代对象（Iterable） 的对象。它提供了一种逐个访问元素的方式，而不需要提前加载整个数据结构到内存中。

可迭代对象（Iterable）是指实现了 __iter__() 方法的对象，该方法返回一个迭代器。常见的可迭代对象包括： list、tuple、str、range

- 迭代器是实现了 __iter__() 和 __next__() 方法的对象： __iter__()：返回迭代器自身（迭代器也是可迭代的）。__next__()：返回下一个元素，如果没有更多元素则抛出 StopIteration 异常。
- 迭代器的特点
  - 惰性计算：迭代器按需生成元素，适合处理大数据流（如文件读取）。
  - 一次性使用：迭代器遍历结束后无法再次使用（需重新创建）。
  - 节省内存：不需要一次性加载所有数据（对比列表）
  - 迭代器工具箱：生成抽象数列

推荐方式（低内存）

```python
from pathlib import Path
import csv

file_path = Path("large_file.csv")
with file_path.open("r", encoding="utf-8") as file:  # 指定编码
    csv_reader = csv.reader(file)
    for row in csv_reader:
        print(row)  # 逐行处理
```

不推荐方式（高内存）

```python
from pathlib import Path
import csv

file_path = Path("large_file.csv")
text = file_path.read_text()  # 全部加载到内存!
lines = text.splitlines()     # 再次占用内存
csv_reader = csv.reader(lines)  # 此时数据已完全加载
```

```python
import itertools
nums = itertools.count(0,2)

print(next(nums))
print(next(nums))
print(next(nums))
```

```
0
2
4
```

```python
import itertools

nums = itertools.count(0,2)
for i in nums:
    if i > 6:
        break
    print(i)
```

```
0
2
4
6
```

```python
import itertools

cycle_strings = itertools.cycle('ABC')
i = 1
for string in cycle_strings:
    if i == 7:
        break
    print(string)
    i = i + 1
```

```
A
B
C
A
B
C
```

```python
import itertools
for item in itertools.repeat('hello', 3):
    print(item)
```

```
hello world
hello world
hello world
```

```python
import itertools
nums = itertools.repeat('hello', 3)

print(next(nums))
print(next(nums))
print(next(nums))
print(next(nums))
```

```
hello
hello
hello
Traceback (most recent call last):
    print(next(nums))
StopIteration
```

groupby()

```
1 from itertools import groupby
2
3 for key, value_iter in groupby('aaabbbaaccd'):
4     print(key, ':', list(value_iter))
```

```
a : ['a', 'a', 'a']
b : ['b', 'b', 'b']
a : ['a', 'a']
c : ['c', 'c']
d : ['d']
```

```
1 from itertools import groupby
2
3 data = ['a', 'bb', 'ccc', 'dd', 'eee', 'f']
4 for key, value_iter in groupby(data, len):
5     print(key, ':', list(value_iter))
```

```
1 : ['a']
2 : ['bb']
3 : ['ccc']
2 : ['dd']
3 : ['eee']
1 : ['f']
```

```
from itertools import groupby
data = ['a', 'bb', 'cc', 'ddd', 'eee', 'f']
for key, value_iter in groupby(data, len):
    print(key, ':', list(value_iter))
```

```
1 : ['a']
2 : ['bb', 'cc']
3 : ['ddd', 'eee']
1 : ['f']
```

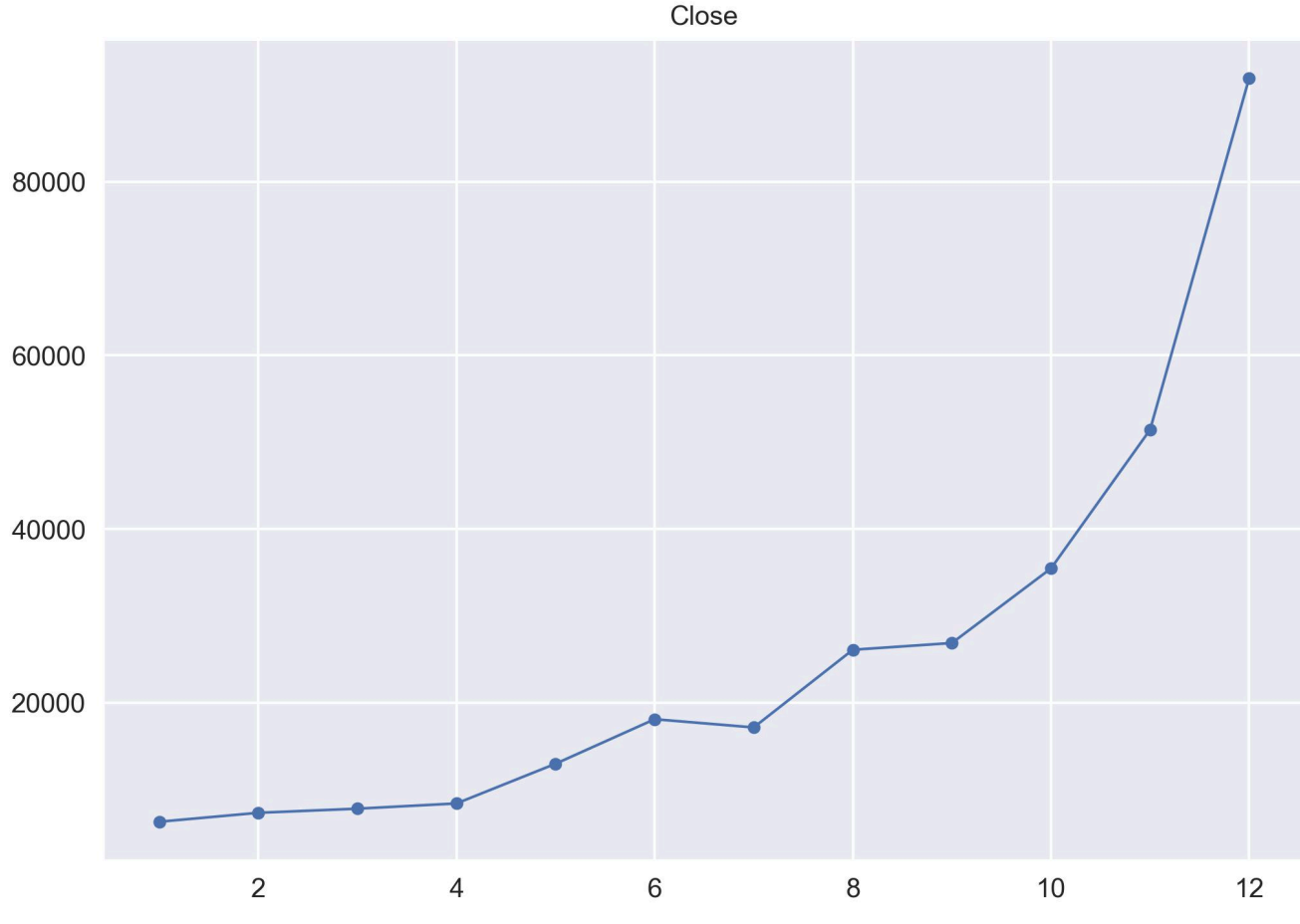
zip()

```
>>> a = [1,2,3]
>>> b = [4,5,6]
>>> c = [4,5,6,7,8]
>>> zipped = zip(a,b)
>>> zipped
 #iterator
>>> list(zipped)
[(1, 4), (2, 5), (3, 6)]
>>> list(zip(a,c))
[(1, 4), (2, 5), (3, 6)]
```

```
>>> a = [1,2,3]
>>> b = [4,5,6]
>>> c = [4,5,6,7,8]
>>> zipped = zip(a,b)
>>> list(zip(*zipped))
[(1, 2, 3), (4, 5, 6)]
>>> zipped = zip(a,b)
>>> x,y = zip(*zipped)
>>> print(x)
(1,2,3)
```

```
from itertools import groupby

xy_map = []
for x, y in groupby(zip(months, close), lambda w: w[0]):
    y_list = []
    for first, second in y:
        y_list.append(second)
    xy_map.append([x, sum(y_list) / len(y_list)])
    x_unique, y_mean = zip(*xy_map)
```

```
import matplotlib.pyplot as plt

plt.style.use('seaborn')
fig, ax = plt.subplots()

ax.plot(x_unique, y_mean, linewidth=1)
ax.scatter(x_unique, y_mean, s=20)
ax.set_title('Close',fontsize=10)

plt.savefig('close.jpg',dpi=300)
```


Close
80000
60000
40000
20000
2
4
6
8
10
12

Summary

- Downloading Data
  - Reading: Python Crash Course, Chapter 16