Kissipo Learning for Deep Learning

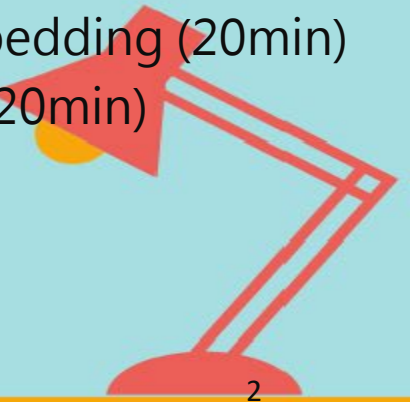
Topic 5: Pandas quick tutorial (15min)

Hsueh-Ting Chu

KLDL-W2-05

Topics

- Topic 01: Introduction to Deep Learning (20min)
- Topic 02: Kissipo Learning for Deep Learning (20min)
- Topic 03: Python quick tutorial (20min)
- Topic 04: Numpy quick tutorial (15min)
- Topic 05: Pandas quick tutorial (15min)
- Topic 06: Scikit-learn quick tutorial (15min)
- Topic 07: OpenCV quick tutorial (15min)
- Topic 08: Image Processing basics (20min)
- Topic 09: Machine Learning basics (20min)
- Topic 10: Deep Learning basics (20min)
- Topic 11: TensorFlow overview (20min)
- Topic 12: CNN with TensorFlow (20min)
- Topic 13: RNN with TensorFlow (20min)
- Topic 14: PyTorch overview (20min)
- Topic 15: CNN with PyTorch (20min)
- Topic 16: RNN with Pytorch (20min)
- Topic 17: Introduction to AOI (20min)
- Topic 18: AOI simple Pipeline (A) (20min)
- Topic 19: AOI simple Pipeline (B) (20min)
- Topic 20: Introduction to Object detection (20min)
- Topic 21: YoloV5 Quick Tutorial (20min)
- Topic 22: Using YoloV5 for RSD (20min)
- Topic 23: Introduction to NLP (20min)
- Topic 24: Introduction to Word Embedding (20min)
- Topic 25: Name prediction project (20min)



Content

- Topic 05: Pandas quick tutorial (20min)
  - Introduction to Pandas
  - Pandas data structure
  - Pandas I/O
  - Pandas Selection
  - Applying functions

Pandas

pandas About us Getting started Documentation Community Contribute

Library Highlights

- A fast and efficient **DataFrame** object for data manipulation with integrated indexing;
- Tools for **reading and writing data** between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format;
- Intelligent **data alignment** and integrated handling of **missing data**: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form;
- Flexible **reshaping** and pivoting of data sets;
- Intelligent label-based **slicing**, **fancy indexing**, and **subsetting** of large data sets;
- Columns can be inserted and deleted from data structures for **size mutability**;
- Aggregating or transforming data with a powerful **group by** engine allowing split-apply-combine operations on data sets;
- High performance **merging and joining** of data sets;
- **Hierarchical axis indexing** provides an intuitive way of working with high-dimensional data in a lower-dimensional data structure;
- **Time series**-functionality: date range generation and frequency conversion, moving window statistics, date shifting and lagging. Even create domain-specific time offsets and join time series without losing data;
- Highly **optimized for performance**, with critical code paths written in Cython or C.
- Python with *pandas* is in use in a wide variety of **academic and commercial** domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more.

https://pandas.pydata.org/

Pandas cheat sheet

Python For Data Science *Cheat Sheet*

Pandas Basics

Learn Python for Data Science Interactively at www.DataCamp.com

Pandas

The Pandas library is built on NumPy and provides easy-to-use data structures and data analysis tools for the Python programming language.

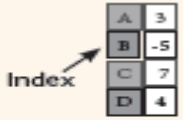
Use the following import convention:

```
>>> import pandas as pd
```

Pandas Data Structures

Series

A one-dimensional labeled array capable of holding any data type




```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

DataFrame

A two-dimensional labeled data structure with columns of potentially different types

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
            'Capital': ['Brussels', 'New Delhi', 'Brasilia'],
            'Population': [11190846, 1303171035, 207847528]}

>>> df = pd.DataFrame(data,
                      columns=['Country', 'Capital', 'Population'])
```

Asking For Help

```
>>> help(pd.Series.loc)
```

Selection

Also see NumPy Arrays

Getting

```
>>> s['b']
-5

>>> df[1:]
  Country     Capital  Population
1   India  New Delhi  1303171035
2  Brazil   Brasilia   207847528
```

Get one element

Get subset of a DataFrame

Selecting, Boolean Indexing & Setting

By Position

```
>>> df.iloc([0],[0])
'Belgium'
>>> df.iat([0],[0])
'Belgium'
```

Select single value by row & column

By Label

```
>>> df.loc([0], ['Country'])
'Belgium'
>>> df.at([0], ['Country'])
'Belgium'
```

Select single value by row & column labels

By Label/Position

```
>>> df.ix[2]
Country       Brazil
Capital     Brasilia
Population  207847528
```

Select single row of subset of rows

```
>>> df.ix[:,'Capital']
0      Brussels
1     New Delhi
2      Brasilia
```

Select a single column of subset of columns

```
>>> df.ix[1,'Capital']
'New Delhi'
```

Select rows and columns

Boolean Indexing

```
>>> s[~(s > 1)]
>>> s[(s < -1) | (s > 2)]
```

Series s where value is not >1
s where value is <-1 or >2

```
>>> df[df['Population']>1200000000]
```

Use filter to adjust DataFrame

Setting

```
>>> s['a'] = 6
```

Set index a of Series s to 6

Dropping

```
>>> s.drop(['a', 'c'])
>>> df.drop('Country', axis=1)
```

Drop values from rows (axis=0)
Drop values from columns(axis=1)

Sort & Rank

```
>>> df.sort_index()
>>> df.sort_values(by='Country')
>>> df.rank()
```

Sort by labels along an axis
Sort by the values along an axis
Assign ranks to entries

Retrieving Series/DataFrame Information

Basic Information

```
>>> df.shape
>>> df.index
>>> df.columns
>>> df.info()
>>> df.count()
```

(rows,columns)
Describe index
Describe DataFrame columns
Info on DataFrame
Number of non-NA values

Summary

```
>>> df.sum()
>>> df.cumsum()
>>> df.min()/df.max()
>>> df.idxmin()/df.idxmax()
>>> df.describe()
>>> df.mean()
>>> df.median()
```

Sum of values
Cummulative sum of values
Minimum/maximum values
Minimum/Maximum index value
Summary statistics
Mean of values
Median of values

Applying Functions

```
>>> f = lambda x: x*2
>>> df.apply(f)
>>> df.applymap(f)
```

Apply function
Apply function element-wise

Data Alignment

Internal Data Alignment

NA values are introduced in the indices that don't overlap:

```
>>> s3 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])
>>> s + s3
a    10.0
b     NaN
c     5.0
d     7.0
```

Arithmetic Operations with Fill Methods

You can also do the internal data alignment yourself with the help of the fill methods:

```
>>> s.add(s3, fill_value=0)
a    10.0
b    -5.0
c     5.0
d     7.0
>>> s.sub(s3, fill_value=2)
>>> s.div(s3, fill_value=4)
>>> s.mul(s3, fill_value=3)
```

I/O

Read and Write to CSV

```
>>> pd.read_csv('file.csv', header=None, nrows=5)
>>> pd.to_csv('myDataFrame.csv')
```

Read and Write to Excel

```
>>> pd.read_excel('file.xlsx')
>>> pd.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
```

Read multiple sheets from the same file

```
>>> xlsx = pd.ExcelFile('file.xls')
>>> df = pd.read_excel(xlsx, 'Sheet1')
```

Read and Write to SQL Query or Database Table

```
>>> from sqlalchemy import create_engine
>>> engine = create_engine('sqlite:///:memory:')
>>> pd.read_sql("SELECT * FROM my_table;", engine)
>>> pd.read_sql_table('my_table', engine)
>>> pd.read_sql_query("SELECT * FROM my_table;", engine)
```

read_sql() is a convenience wrapper around read_sql_table() and read_sql_query()

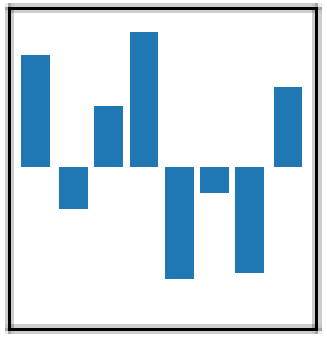
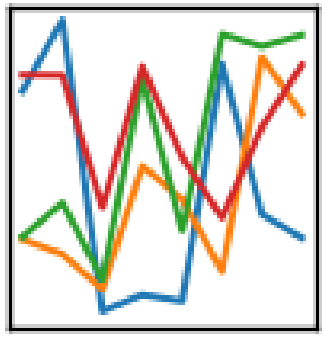
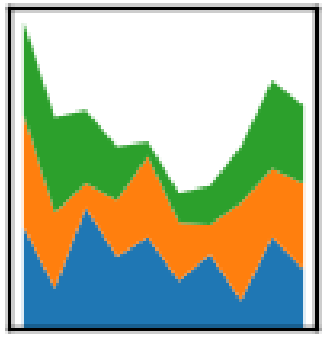
```
>>> pd.to_sql('myDf', engine)
```

DataCamp
Learn Python for Data Science Interactively

Pandas
Python Data Analysis Library

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$

```
df = pd.read_csv(‘xxx.csv')
print(df)
df.head()
df.plot()
```

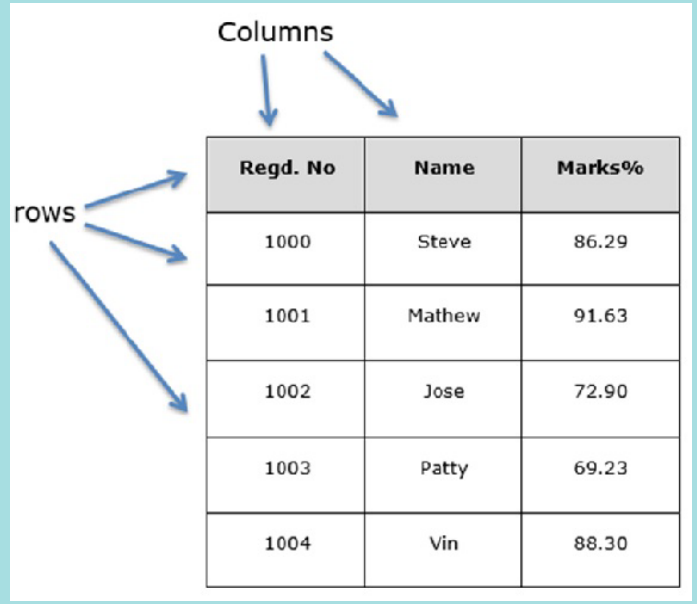
Read data as a DataFrame

```
CALL,FIRSTNAME,LASTNAME,Skywn,IS-100,IS-700,PUBLIC
AC5NT,LARRY,SWANSON,,FALSE,FALSE,TRUE
AC5T,LYNN,JACKSON,,FALSE,FALSE,TRUE
AE5BK,BILL,KRUEGER,,FALSE,FALSE,TRUE
AE5CF,BRIAN,WEIDENMAIER,,FALSE,FALSE,TRUE
AE5FF,DALE,LEWIS,3/26/2011,TRUE,TRUE,TRUE
AE5IT,WALTER,LEMONS,,TRUE,TRUE,TRUE
AE5IV,MIKE,CHITTENDEN,,FALSE,FALSE,TRUE
AE5NS,BRIAN,BAUGH,,FALSE,FALSE,TRUE
K1WL,DONALD,PATTERSON,2/26/2011,TRUE,TRUE,TRUE
K3DHB,DON,BARBER,5/19/2011,TRUE,TRUE,TRUE
K5AGO,JACK,O'TOOLE,,TRUE,FALSE,TRUE
K5AJP,ANDY,PONDER,2/26/2011,TRUE,TRUE,TRUE
K5BP,BERNIE,PARKER,,TRUE,TRUE,TRUE
K5BYX,CRAIG,WAGGONER,,FALSE,FALSE,TRUE
K5EMI,WILLIAM,STEWART,,FALSE,FALSE,FALSE
K5HUD,SETH,HUDSON,,TRUE,TRUE,TRUE
K5IW,BILL,GRUBBS,,FALSE,FALSE,TRUE
```

df = pd.read_csv('xxx.csv')

df.head()

df.tail()



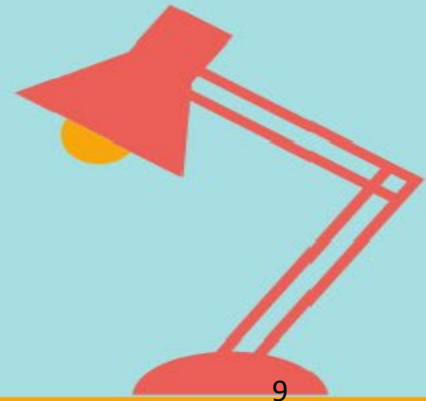
Read a CSV file

- import pandas as pd
- # 讀入 csv 文字檔
- csv_file = "https://storage.googleapis.com/learn_pd_like_tidyverse/df.csv"
- df = pd.read_csv(csv_file)
- print(type(df))
- df.head()

Filtering and select

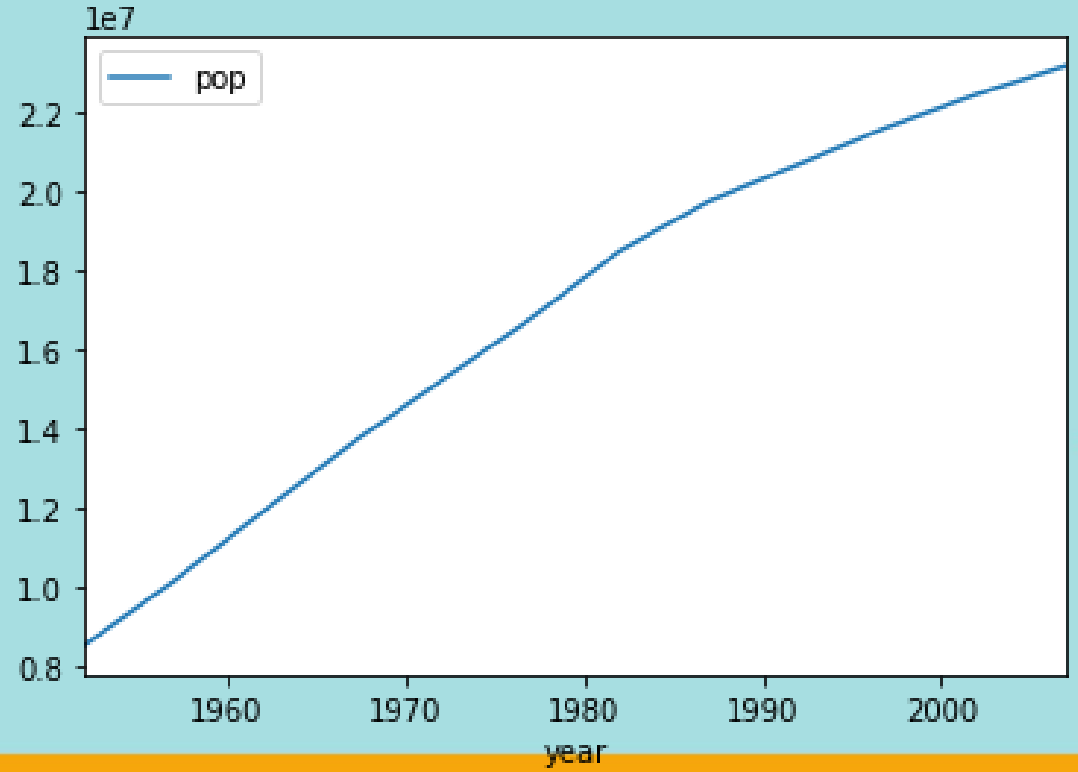
- df[df['country'] == 'Taiwan’]
- df[df[(df['year'] == 2007) & (df['continent'] == 'Asia')]]

- df[['country', 'continent']]



df.Plot()

- df[df['country'] == 'Taiwan'].plot(x='year',y='pop')



df.iterrows() and np.nditer(a)

- for index, row in df.iterrows():
- a = row[1:-1]
- y = row[-1]
- s = ''
- for x in np.nditer(a):
- #print (x,)
- s+=str(x)
- cc = ss[s]

Thanks!

Q&A