

# 數值分析Numerical Analysis

許志宇Chin-Yu Hsu  
October 1, 2018

# 聽眾 Audience

- 碩一 first year graduate
- 基礎課程 Basic curriculum
- 19 students

# Objective

- 引導您了解如何自學。Guide you to know how to learn by yourself.
- 數值分析（NA）的歷史是什麼？
- What is the history of the Numerical Analysis (NA)?
  - Who are the important people NA? Newton, Lagrange, Gauss and Euler.
  - What are the important events of NA?
  - Where is the location of NA?
  - Which objects are related to NA?
  - When is NA popular?

# 材料Materials

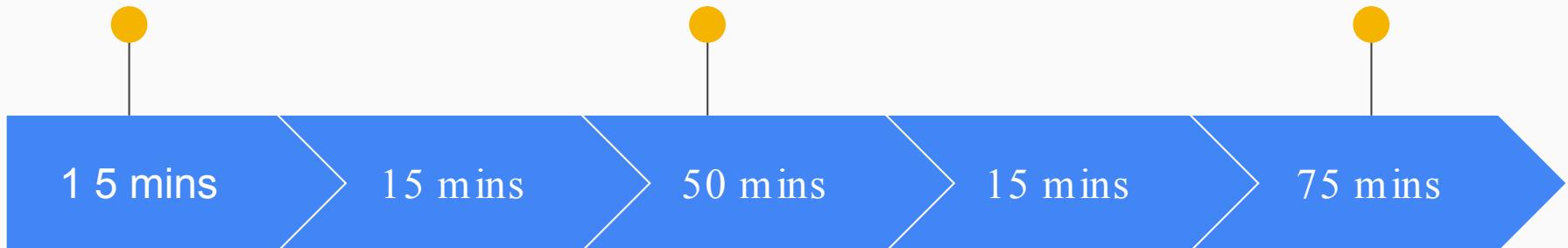
- What is NA
- Python Language
- Tensor flow
- Project and report

# 程序 Procedure

Introduction

Step by step to learn  
small concepts by  
doing

A project and report  
should be done by  
yourself



Demo: How to solve  
problem?

Conclusions

# 家庭作業1/Homework1

Learning efficiency depends on motivation

1. Why do you choose this course?
2. What kind of jobs for NA?
3. What kind of jobs you want?

How to learn knowledge of NA efficiently?

1. Python programming
2. Mathematics
3. Tensorflow, scikit learn and, keras

# Python programming

## Learning topics

1. Using Python as a Calculator
  - a. Arithmetic operations
2. Variables
3. if Statements
4. for Statements
5. The range() Function
6. Executing modules as scripts
7. Mathematics
8. Python Functions - W3Schools

## Learning Resources

1. [Welcome to Python.org](#)
2. [Download](#)
3. [Python For Beginners](#)
4. [The Python Tutorial](#)
5. [IntroductoryBooks](#)
6. [Python Functions - W3Schools](#)
7. [A Visual Introduction to Python](#)
8. [Try Jupyter](#)

# Python programming

## Learning topics

### 1. Python Functions

```
def my_function():
    print("Hello from a function")

my_function()
```

## Learning Resources

1. [Python Functions - W3Schools](#)
2. [Run web python or Anaconda](#)
3. <http://jupyter.org/try>
4. [https://hub.mybinder.org/user/jupyterlab-jupyterlab-demo-yqivt6ba/lab#Integration-\(scipy.integrate\)](https://hub.mybinder.org/user/jupyterlab-jupyterlab-demo-yqivt6ba/lab#Integration-(scipy.integrate))

# Python programming

The screenshot shows a Jupyter Notebook interface running on a web browser. The URL is [https://hub.mybinder.org/user/jupyterlab-jupyterlab-demo-yqivt6ba/lab#Integration-\(scipy.integrate\)](https://hub.mybinder.org/user/jupyterlab-jupyterlab-demo-yqivt6ba/lab#Integration-(scipy.integrate)). The interface includes a top navigation bar with File, Edit, View, Run, Kernel, Tabs, Settings, and Help. Below the navigation is a file browser sidebar showing a directory structure under 'demo'. The main area contains two tabs: 'Lorenz.ipynb' and 'test\_integration.ipynb'. The 'test\_integration.ipynb' tab is active, displaying a section titled 'The computation of integral'. It contains text about calculating the integral of  $f(x) = x$  from 0 to 1, followed by a mathematical equation  $I = \int_0^1 f(x)dx$ . Below this, it says 'Let's calculate the value of  $I$ '. Another section titled 'Learning Latex: Integrals, sums and limits' is present, along with a link to [https://www.overleaf.com/learn/latex/Integrals,\\_sums\\_and\\_lims](https://www.overleaf.com/learn/latex/Integrals,_sums_and_lims). At the bottom, there is more text about exploring definite integration of  $f(x) = x$ , followed by a mathematical derivation of the integral from 0 to 1.

The computation of integral

We calculate the Integral of function  $f(x) = x$  in interval  $(0,1)$ :

$$I = \int_0^1 f(x)dx$$

Let's calculate the value of  $I$ .

### Learning Latex: Integrals, sums and limits

[https://www.overleaf.com/learn/latex/Integrals,\\_sums\\_and\\_lims](https://www.overleaf.com/learn/latex/Integrals,_sums_and_lims)

We explore the definite integration of function  $f(x) = x$ :

$$\begin{aligned} I &= \int_0^1 f(x)dx \\ &= \frac{1}{2}x^2 \Big|_0^1 = \frac{1}{2}1^2 - 0 = \frac{1}{2} \end{aligned}$$

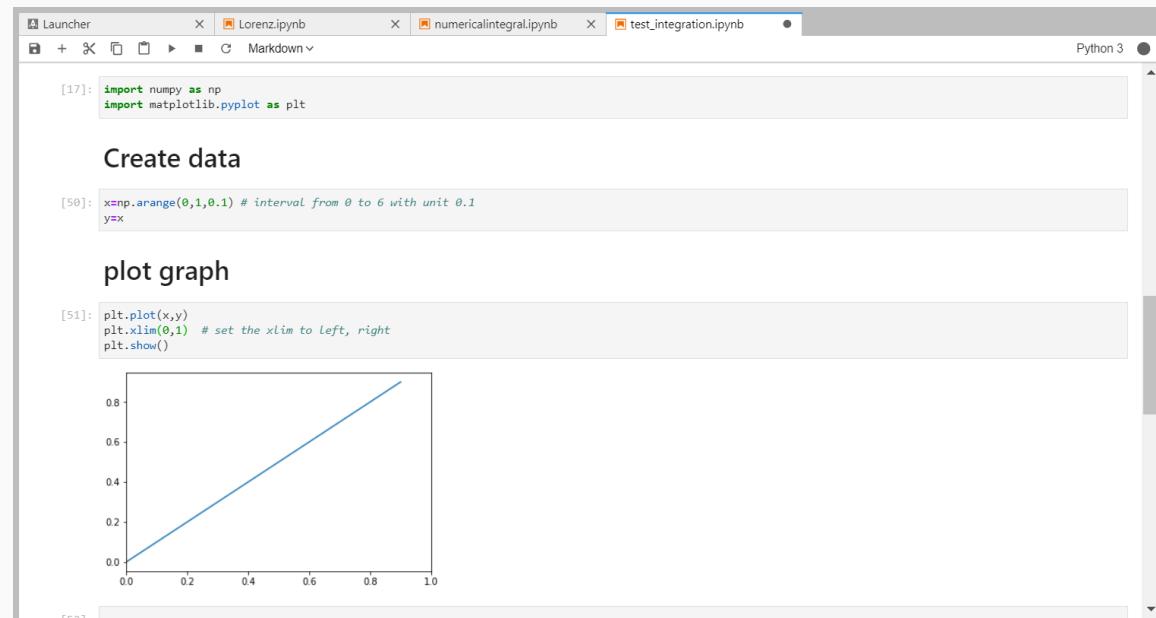
1. <http://jupyter.org/try>

# Python programming

Plot graph curve  $y=f(x)=x$

```
import numpy as np  
import matplotlib.pyplot as plt
```

```
# Create data  
x=np.arange(0,1,0.1) # interval from  
0 to 6 with unit 0.1  
  
y=x  
# plot graph  
plt.plot(x,y)  
plt.xlim(0,1) # set the xlim to left,  
right  
plt.show()
```



The screenshot shows a Jupyter Notebook interface with several tabs at the top: 'Launcher', 'Lorenz.ipynb', 'numericalintegral.ipynb', 'test\_integration.ipynb' (which is the active tab), and 'Markdown'. The notebook contains the following code:

```
[17]: import numpy as np  
import matplotlib.pyplot as plt  
  
Create data  
  
[50]: x=np.arange(0,1,0.1) # interval from 0 to 6 with unit 0.1  
y=x  
  
plot graph  
  
[51]: plt.plot(x,y)  
plt.xlim(0,1) # set the xlim to left, right  
plt.show()
```

Below the code, a plot is displayed showing a straight line segment from (0,0) to (1,1). The x-axis is labeled from 0.0 to 1.0 with increments of 0.2. The y-axis is labeled from 0.0 to 0.8 with increments of 0.2.

# 程式設計 Python programming

Data  
X  
y

1. <http://jupyter.org/try>

The screenshot shows a Jupyter Notebook window with several tabs at the top: 'Launcher', 'Lorenz.ipynb', 'numericalintegral.ipynb', and 'test\_integration.ipynb'. The 'test\_integration.ipynb' tab is active. Below the tabs, there is a plot of a linear function  $y = x$  from 0.0 to 1.0. The x-axis ranges from 0.0 to 1.0 with ticks every 0.2. The y-axis ranges from 0.0 to 1.0 with ticks every 0.2. The plot area is white with a black border.

The notebook contains the following code:

```
[52]: x
[52]: array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9])
[*]: y
[53]: array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9])

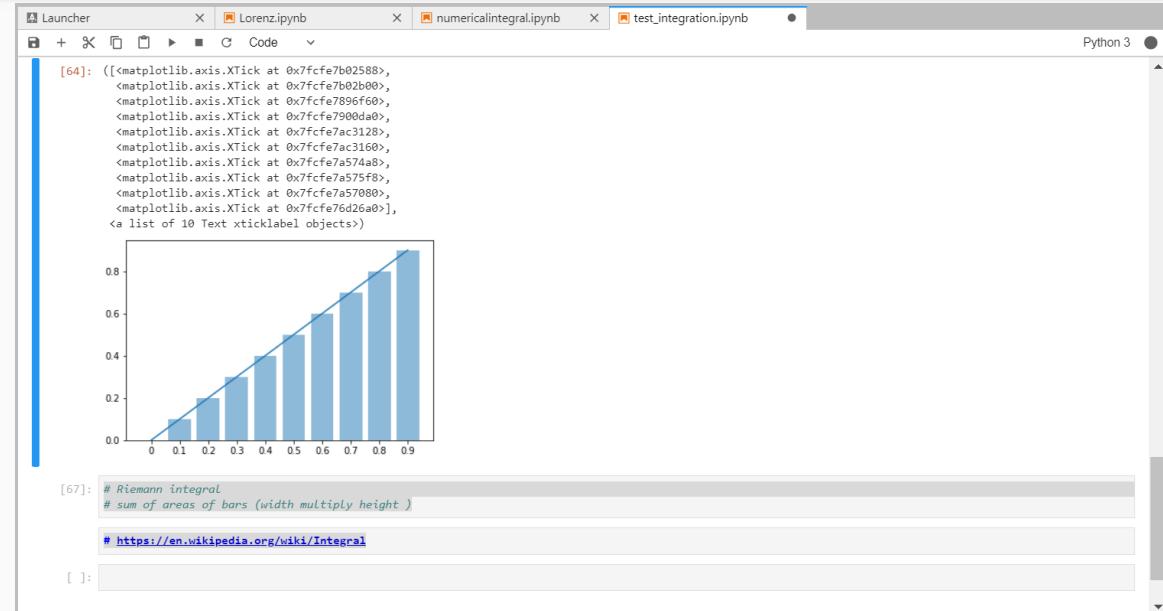
[*]: objects = ('0', '0.1', '0.2', '0.3', '0.4', '0.5', '0.6', '0.7', '0.8', '0.9')
y_pos = np.arange(len(objects))
plt.bar(y_pos, y, align='center', alpha=0.5)
plt.plot(y_pos, y)
plt.xticks(y_pos, objects)

[64]: ([matplotlib.axis.XTick at 0x7fcfe7b02588>,
 <matplotlib.axis.XTick at 0x7fcfe7b02b00>,
 <matplotlib.axis.XTick at 0x7fcfe7896f60>,
 <matplotlib.axis.XTick at 0x7fcfe7900da0>,
 <matplotlib.axis.XTick at 0x7fcfe7ac3128>,
 <matplotlib.axis.XTick at 0x7fcfe7a7c3160>])
```

# 程式設計 Python programming

## Bar Plot

```
# Riemann integral
# sum of areas of bars (width multiply
height )
# https://en.wikipedia.org/wiki/Integral
objects = ('0', '0.1', '0.2',
'0.3','0.4','0.5', '0.6', '0.7', '0.8', '0.9')
y_pos = np.arange(len(objects))
plt.bar(y_pos, y, align='center',
alpha=0.5)
plt.plot(y_pos, y)
plt.xticks(y_pos, objects)
```

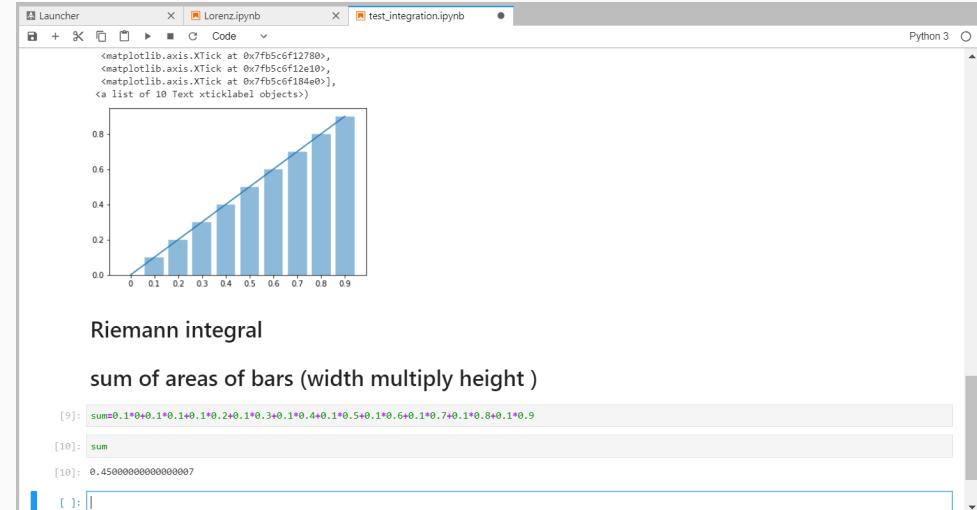


<http://jupyter.org/try>

# Python programming

```
# Riemann integral  
# sum of areas of bars (width multiply height )
```

```
# https://en.wikipedia.org/wiki/Integral
```



[https://en.wikipedia.org/wiki/Numerical\\_integration](https://en.wikipedia.org/wiki/Numerical_integration)

# Python programming

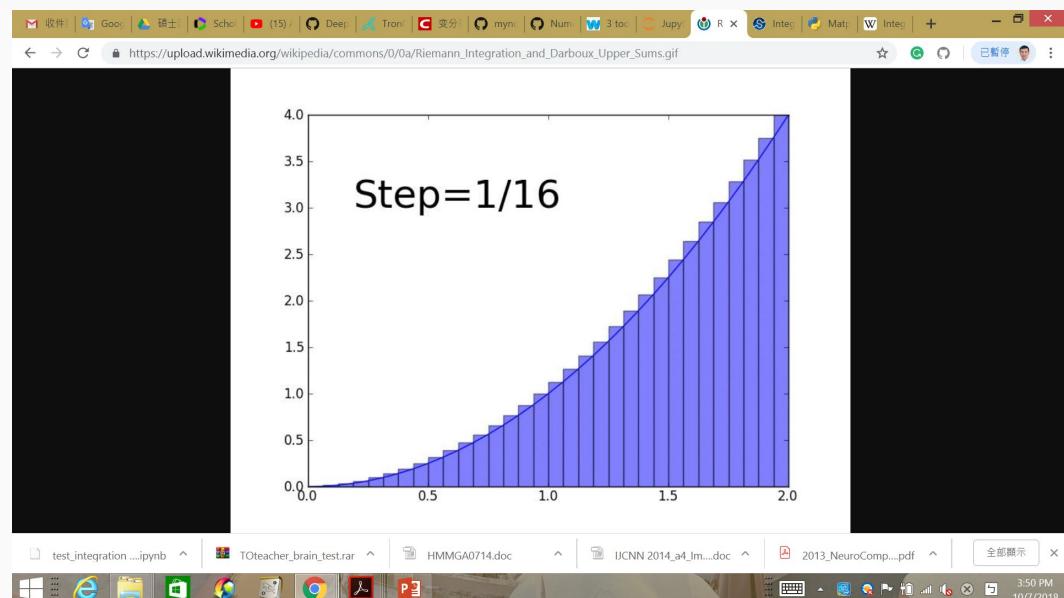
Question:

$$Y=x^{**2} [0 1]$$

Summing the areas of these rectangles, we get a better approximation for the sought integral, namely

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.quad.html>

<https://en.wikipedia.org/wiki/Integral>



# Latex programming

## Learning to Latex to Integrals, sums and limits

The screenshot shows a web browser displaying a LaTeX learning tutorial on Overleaf. The URL in the address bar is [https://www.overleaf.com/learn/latex/integrals,\\_sums\\_and\\_limits](https://www.overleaf.com/learn/latex/integrals,_sums_and_limits). The page has a sidebar on the left with various LaTeX topics and a main content area on the right.

**Creating your first LaTeX document**

- Choosing a LaTeX Compiler
- Paragraphs and new lines
- Bold, italics and underlining
- Lists
- Errors

**Mathematics**

- Mathematical expressions
- Subscripts and superscripts
- Brackets and Parentheses
- Fractions and Binomials
- Aligning Equations
- Operators
- Spacing in math mode
- Integrals, sums and limits

**Integrals**

Integral expression can be added using the `\int_{lower}^{upper}` command.

Note, that integral expression may seems a little different in *inline* and *display* math mode - in *inline* mode the integral symbol and the limits are compressed.

LATEX code	Output
Integral <code>\int_a^b x^2 dx</code> inside text	Integral $\int_a^b x^2 dx$ inside text
<code>\$\$\int_a^b x^2 dx\$\$</code>	$\int_a^b x^2 dx$

[https://www.overleaf.com/learn/latex/integrals,\\_sums\\_and\\_limits](https://www.overleaf.com/learn/latex/integrals,_sums_and_limits)

# Python programming

## Learning to implement the algorithm on wiki

[https://en.wikipedia.org/wiki/Riemann\\_sum](https://en.wikipedia.org/wiki/Riemann_sum)

https://en.wikipedia.org/wiki/Riemann\_sum

Methods [edit]

The four methods of Riemann summation are usually best approached with partitions of equal size. The interval  $[a, b]$  is therefore divided into  $n$  subintervals, each of length

$$\Delta x = \frac{b - a}{n}.$$

The points in the partition will then be

$$a, a + \Delta x, a + 2 \Delta x, \dots, a + (n - 2) \Delta x, a + (n - 1) \Delta x, b.$$

**Left Riemann sum** [edit]

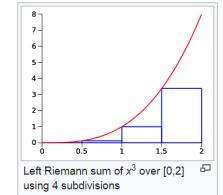
For the left Riemann sum, approximating the function by its value at the left-end point gives multiple rectangles with base  $\Delta x$  and height  $f(a + i\Delta x)$ . Doing this for  $i = 0, 1, \dots, n - 1$ , and adding up the resulting areas gives

$$\Delta x [f(a) + f(a + \Delta x) + f(a + 2 \Delta x) + \dots + f(b - \Delta x)].$$

The left Riemann sum amounts to an overestimation if  $f$  is monotonically decreasing on this interval, and an underestimation if it is monotonically increasing.

**Right Riemann sum** [edit]

$f$  is here approximated by the value at the right endpoint. This gives multiple rectangles with base  $\Delta x$  and height  $f(a + i\Delta x)$ . Doing this for  $i = 1, \dots, n$ , and adding up the resulting areas produces

$$\Delta x [f(a + \Delta x) + f(a + 2 \Delta x) + \dots + f(b)].$$


Left Riemann sum of  $x^3$  over  $[0, 2]$  using 4 subdivisions

# Python programming

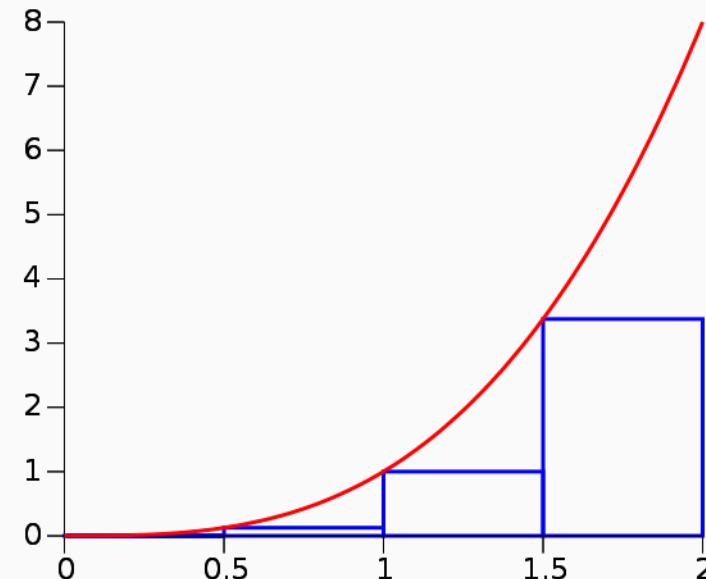
Question:

**Refer to:**

[https://github.com/tccnchsu/Numerical\\_Analysis/blob/master/NA\\_W4\\_Integration.ipynb](https://github.com/tccnchsu/Numerical_Analysis/blob/master/NA_W4_Integration.ipynb)

$F(x)=x^2$

[https://en.wikipedia.org/wiki/Riemann\\_sum](https://en.wikipedia.org/wiki/Riemann_sum)



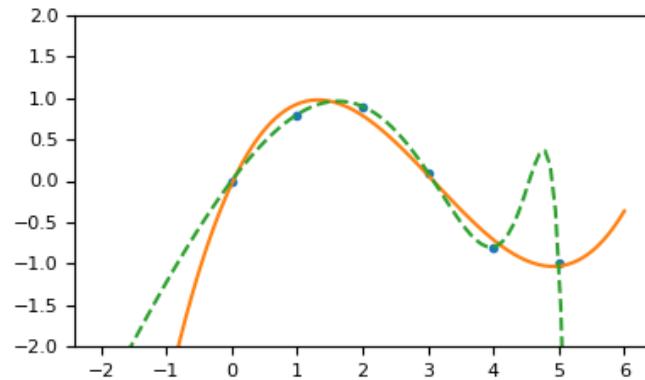
# Python programming

polynomial interpolation

**Two points <-> 2,3,4 degree polynomial**

**Three points <-> 2,3,4 degree polynomial**

**Four points <-> 2,3,4 degree polynomial**



<https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.polyfit.html>

# Python programming

polynomial interpolation

Least squares polynomial fit.

**numpy.polyfit**

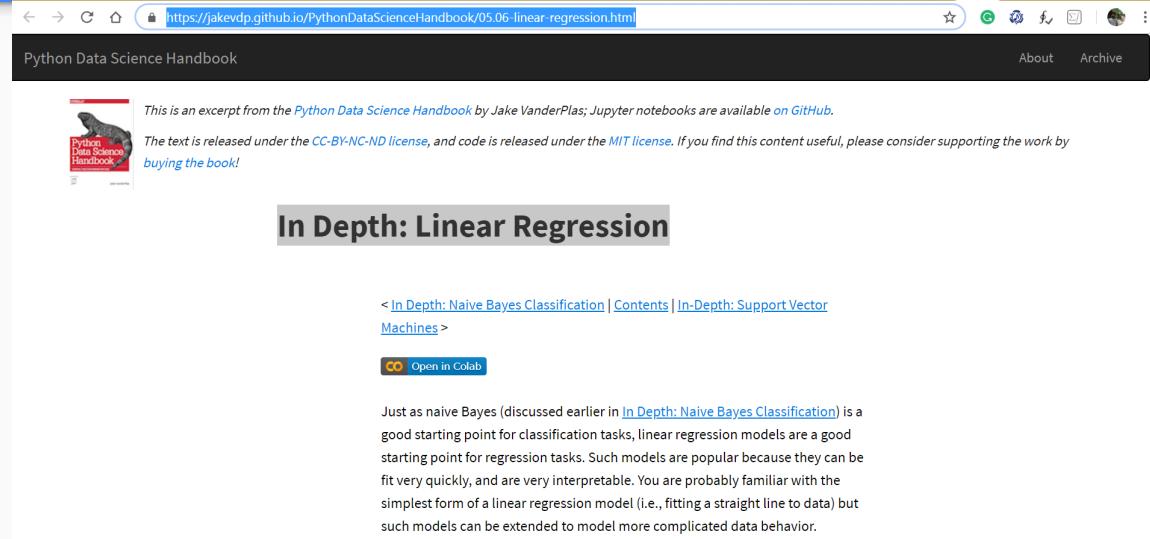
The screenshot shows a web browser displaying the SciPy.org documentation for the `numpy.polyfit` function. The URL in the address bar is `https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.polyfit.html`. The page header includes the SciPy.org logo and navigation links for Scipy.org, Docs, NumPy v1.15 Manual, NumPy Reference, Routines, Polynomials, and Poly1d. On the right side, there are links for index, next, and previous topics, as well as links for Previous topic (`numpy.roots`) and Next topic (`numpy.polyder`). A search bar and a "search" button are also present. The main content area is titled "numpy.polyfit" and contains the function signature `numpy.polyfit(x, y, deg, rcond=None, full=False, w=None, cov=False)`, a "[source]" link, and a detailed description of the least squares polynomial fit. It explains that the function returns a vector of coefficients  $p$  that minimizes the squared error. The parameters are described as follows:

- x**: array\_like, shape (M). x-coordinates of the M sample points ( $x[i]$ ,  $y[i]$ ).
- y**: array\_like, shape (M) or (M, K). y-coordinates of the sample points. Several data sets of sample points sharing the same x-coordinates can be fitted at once by passing in a 2D-array that contains one dataset per column.
- deg**: int. Degree of the fitting polynomial.
- rcond**: float, optional. Relative condition number of the fit. Singular values smaller than this relative to the largest singular value will be ignored. The default value is  $\text{len}(x)*\text{eps}$ , where  $\text{eps}$  is the relative precision of the float type, about 2e-16 in most cases.

<https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.polyfit.html>

# Python programming

## In Depth: Linear Regression



A screenshot of a web browser displaying a page from the Python Data Science Handbook. The URL in the address bar is <https://jakevdp.github.io/PythonDataScienceHandbook/05.06-linear-regression.html>. The page title is "Python Data Science Handbook". On the left, there is a "Open in Colab" button. The main content area features a header "In Depth: Linear Regression". Above the content, there is a note about the book's license: "This is an excerpt from the *Python Data Science Handbook* by Jake VanderPlas; Jupyter notebooks are available on [GitHub](#). The text is released under the [CC-BY-NC-ND license](#), and code is released under the [MIT license](#). If you find this content useful, please consider supporting the work by [buying the book!](#)". Below the content, there is a "Open in Colab" button again and a paragraph of text: "Just as naive Bayes (discussed earlier in [In Depth: Naive Bayes Classification](#)) is a good starting point for classification tasks, linear regression models are a good starting point for regression tasks. Such models are popular because they can be fit very quickly, and are very interpretable. You are probably familiar with the simplest form of a linear regression model (i.e., fitting a straight line to data) but such models can be extended to model more complicated data behavior."

<https://jakevdp.github.io/PythonDataScienceHandbook/05.06-linear-regression.html>

# Python programming

The screenshot shows a web browser window with the following details:

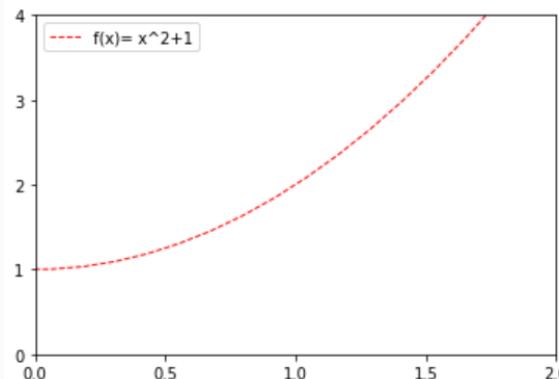
- Title Bar:** https://hk.saowen.com/a/be3251d506c792a17e28882445eb73a04fe79c650c5f0167ea54abf22b005705
- Page Title:** 掃文資訊
- Content Area:**
  - Section Header:** In Depth: Linear Regression  
線性迴歸
  - Date & Source:** 2017-07-02 git.oschina.net
  - Main Title:** Python 數據科學手冊 5.6 線性迴歸
  - Section Header:** 5.6 線性迴歸
  - Text:** 原文：[In Depth: Linear Regression](#)
  - Text:** 譯者：飛龍
  - Text:** 協議：[CC BY-NC-SA 4.0](#)
  - Text:** 譯文沒有得到原作者授權，不保證與原文的意思嚴格一致。

<https://hk.saowen.com/a/be3251d506c792a17e28882445eb73a04fe79c650c5f0167ea54abf22b005705>

# Python programming

How do I compute derivative using Numpy?

<http://jupyter.org/try>



How do I calculate the derivative of a function, for example

$$y = x^2 + 1$$

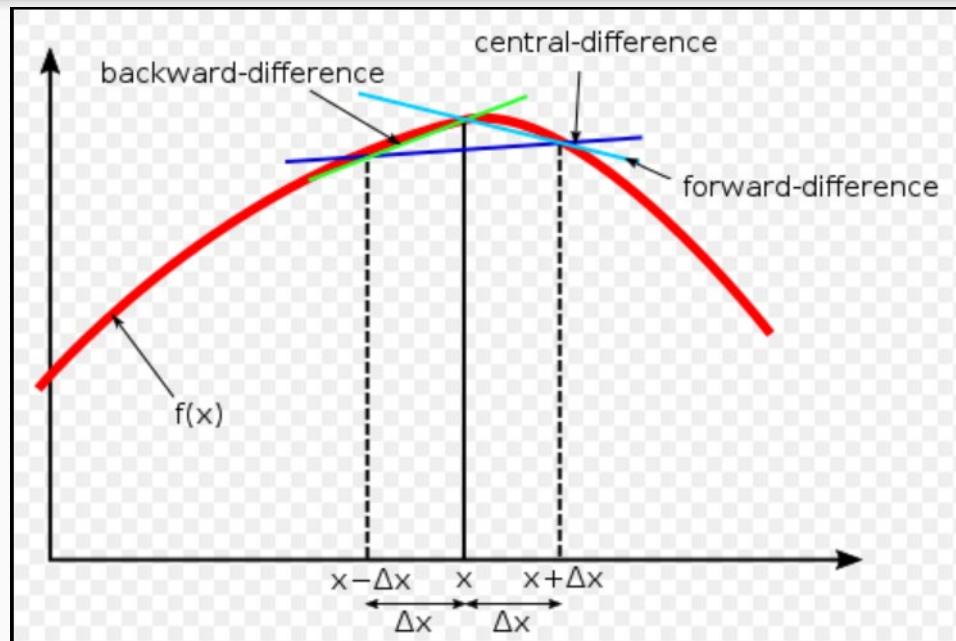
using `numpy` ?

Let's say, I want the value of derivative at  $x = \dots$

<https://stackoverflow.com/questions/9876290/how-do-i-compute-derivative-using-numpy>

# Python programming

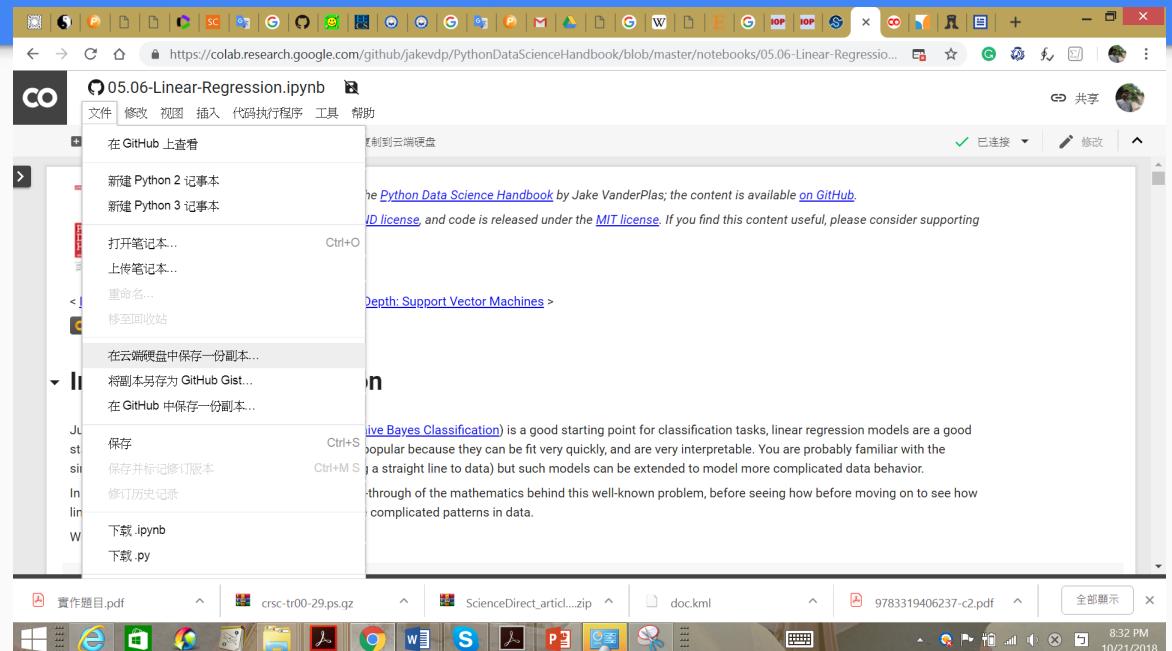
Finite difference



[https://en.wikipedia.org/wiki/Finite\\_difference](https://en.wikipedia.org/wiki/Finite_difference)

# Python programming

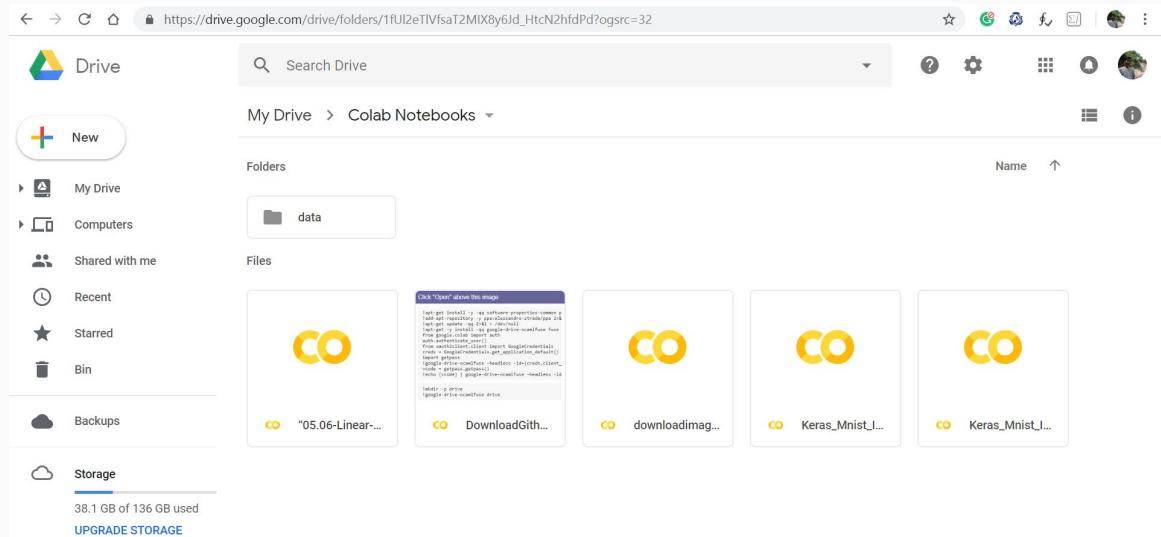
In Depth: Linear Regression  
File copy cloud disk



<https://jakevdp.github.io/PythonDataScienceHandbook/05.06-linear-regression.html>

# Python programming

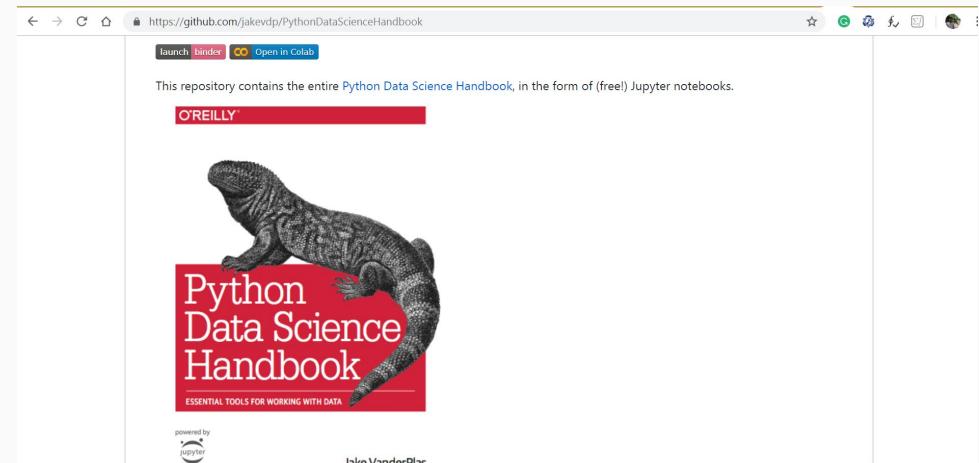
In Depth: Linear Regression  
**File copy cloud disk**



<https://jakevdp.github.io/PythonDataScienceHandbook/05.06-linear-regression.html>

# Python programming

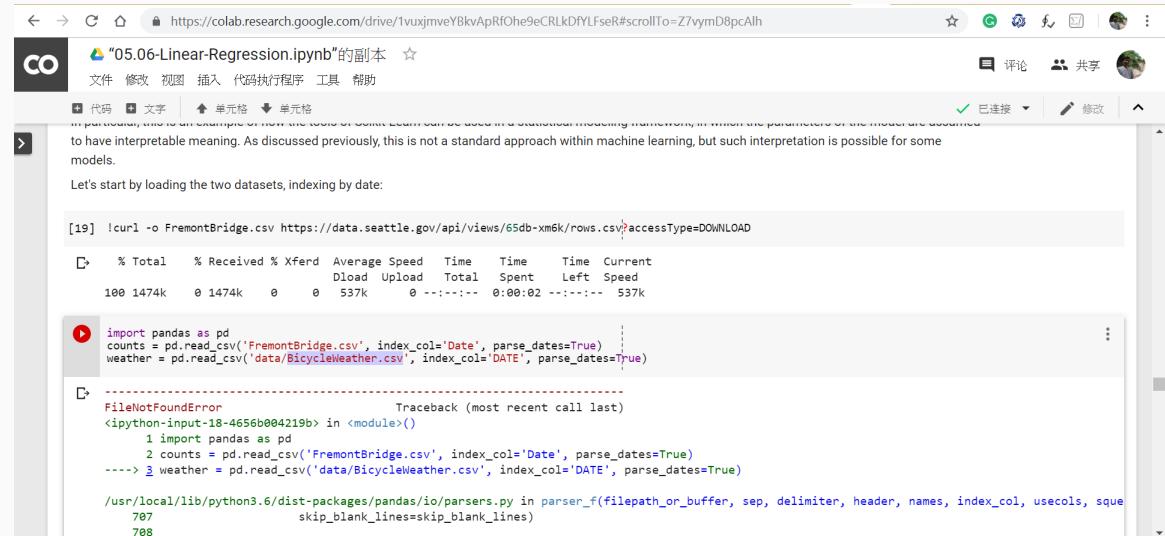
## Python Data Science Handbook



<https://github.com/jakevdp/PythonDataScienceHandbook>

# Python programming

## BicycleWeather.csv



The screenshot shows a Google Colab interface with a notebook titled "05.06-Linear-Regression.ipynb" open. The code cell at the top contains a warning about the use of LassoLars, stating: "In particular, this is an example of how the tools of Scikit-Learn can be used in a statistical modeling framework, in which the parameters of the models are assumed to have interpretable meaning. As discussed previously, this is not a standard approach within machine learning, but such interpretation is possible for some models." Below this, the code cell [19] shows the command to download the FremontBridge.csv file from a Seattle government API:

```
[19] !curl -o FremontBridge.csv https://data.seattle.gov/api/views/65db-xm6k/rows.csv?accessType=DOWNLOAD
```

Following this, the user attempts to import the FremontBridge.csv file and the BicycleWeather.csv file into pandas DataFrames:

```
import pandas as pd
counts = pd.read_csv('FremontBridge.csv', index_col='Date', parse_dates=True)
weather = pd.read_csv('data/BicycleWeather.csv', index_col='DATE', parse_dates=True)
```

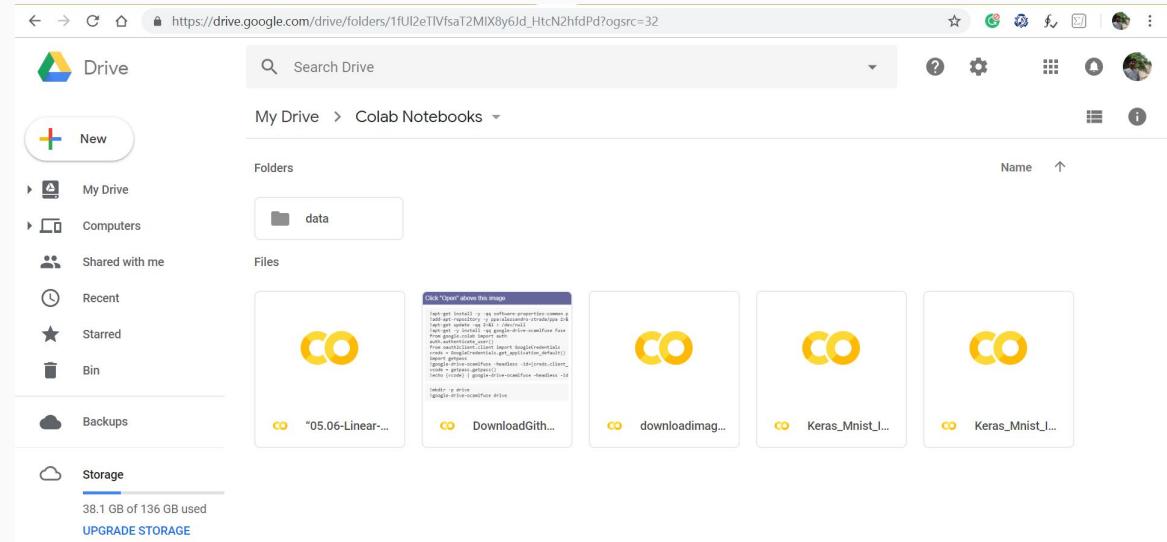
This results in a `FileNotFoundException` for the `BicycleWeather.csv` file, with the full traceback shown:

```
-----  
FileNotFoundException                               Traceback (most recent call last)  
<ipython-input-18-4656b004219b> in <module>()  
      1 import pandas as pd  
      2 counts = pd.read_csv('FremontBridge.csv', index_col='Date', parse_dates=True)  
----> 3 weather = pd.read_csv('data/BicycleWeather.csv', index_col='DATE', parse_dates=True)  
  
/usr/local/lib/python3.6/dist-packages/pandas/io/parsers.py in parser_f(filepath_or_buffer, sep, delimiter, header, names, index_col, usecols, squeeze, skiprows, skip_blank_lines)  
    707  
    708
```

<https://github.com/jakevdp/PythonDataScienceHandbook>

# Python programming

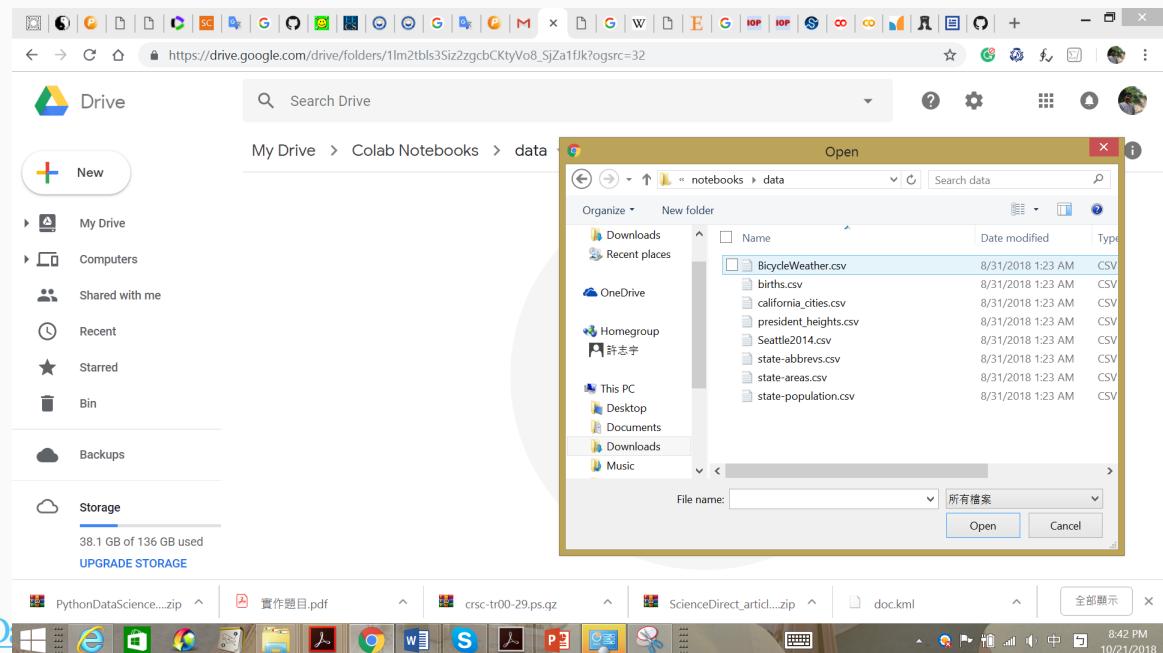
Make directory “data”



<https://github.com/jakevdp/PythonDataScienceHandbook>

# Python programming

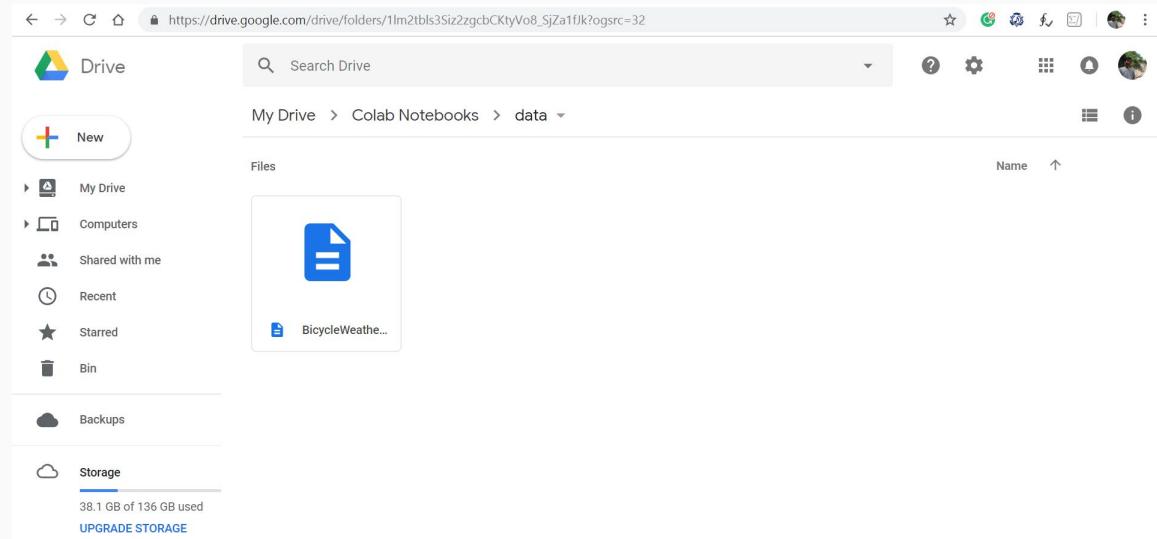
Upload BicycleWeather.csv



<https://github.com/jakevdp/PythonDataScienceHandbook>

# Python programming

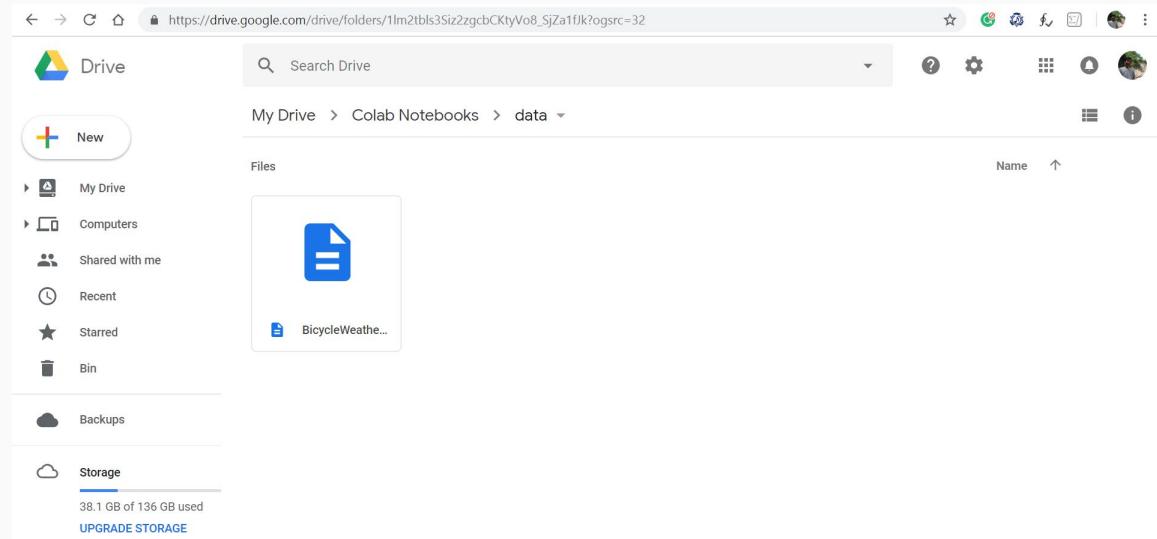
**data/BicycleWeather.csv**



<https://github.com/jakevdp/PythonDataScienceHandbook>

# Python programming

**data/BicycleWeather.csv**



<https://github.com/jakevdp/PythonDataScienceHandbook>

# Python programming

## Run “05.06-Linear-Regression.ipynb”的副本

The screenshot shows a Google Colab interface with the following details:

- Title:** "05.06-Linear-Regression.ipynb"的副本
- Code Cell 19:** !curl -o FremontBridge.csv https://data.seattle.gov/api/views/65db-xm6k/rows.csv?accessType=DOWNLOAD
- Code Cell 20:** import pandas as pd  
counts = pd.read\_csv('FremontBridge.csv', index\_col='Date', parse\_dates=True)  
weather = pd.read\_csv('data/BicycleWeather.csv', index\_col='DATE', parse\_dates=True)
- Error:** FileNotFoundError
- Stack Trace:** /usr/local/lib/python3.6/dist-packages/pandas/io/parsers.py in parser\_f(filepath\_or\_buffer, sep, delimiter, header, names, index\_col, usecols, squeeze, skiprows, skip\_blank\_lines, skip\_footer, thousands, na\_values, keep\_date\_col, parse\_dates, date\_parser, dayfirst, iterator, chunksize, encoding, compression, thousands\_sep, decimal, error\_bad\_lines, warn\_bad\_lines, engine, converters, true\_values, false\_values, skipinitialspace, doublequote, quotechar, lineterminator, quote\_error\_handler, low\_memory, memory\_map, float\_precision, storage\_options, \*\*kwargs)
- Line 707:** skip\_blank\_lines=skip\_blank\_lines)
- Line 708:**
- Line 709:** return \_read(filepath\_or\_buffer, kwds)
- Line 710:**
- Line 711:** parser\_f.\_name\_ = name

<https://github.com/jakevdp/PythonDataScienceHandbook>

# Python programming

```
import os
print(os.getcwd())
os.chdir('..')
print(os.getcwd())
os.listdir()
os.chdir('content')
os.listdir()
```

```
[52] !curl -o FremontBridge.csv https://data.seattle.gov/views/65db-xm6k/rows.csv?accessType=DOWNLOAD
          % Total    % Received % Xferd  Average Speed   Time     Time     Time  Current
                                         Dload  Upload Total   Spent    Left  Speed
 100 1474k    0 1474k      0      495k       0 --:--:--  0:00:02 --:--:--  495k

# https://www.programiz.com/python-programming/directory
import os
print(os.getcwd())
os.chdir('..')
print(os.getcwd())
os.listdir()
os.chdir('content')
os.listdir()

          /content
          /
/bin/bash: google-drive-ocamlfuse: command not found
ls: cannot access 'drive/Colab Notebooks': No such file or directory
```

<https://www.programiz.com/python-programming/directory>

# Python programming

creating or mounting

The screenshot shows a QIITA post with the URL <https://qiita.com/Rowing0914/items/51a770925653c7c528f9>. The post title is "Then try this command!". It contains a snippet of Python code:os.path.isfile("drive/Colab Notebooks/ptb.valid.txt")  
# output  
falseA message below says "Ups..." followed by "We haven't finished creating or mounting anything yet. So try this." It shows a terminal session with the following commands:

```
!mkdir drive  
!google-drive-ocamlfuse drive  
!ls drive/"Colab Notebooks"  
# output  
Test_20180513.ipynb Untitled0.ipynb ptb.valid.txt
```

A message at the bottom says "Congrats, now you have mounted your directory with your environment."

<https://qiita.com/Rowing0914/items/51a770925653c7c528f9>

# Python programming

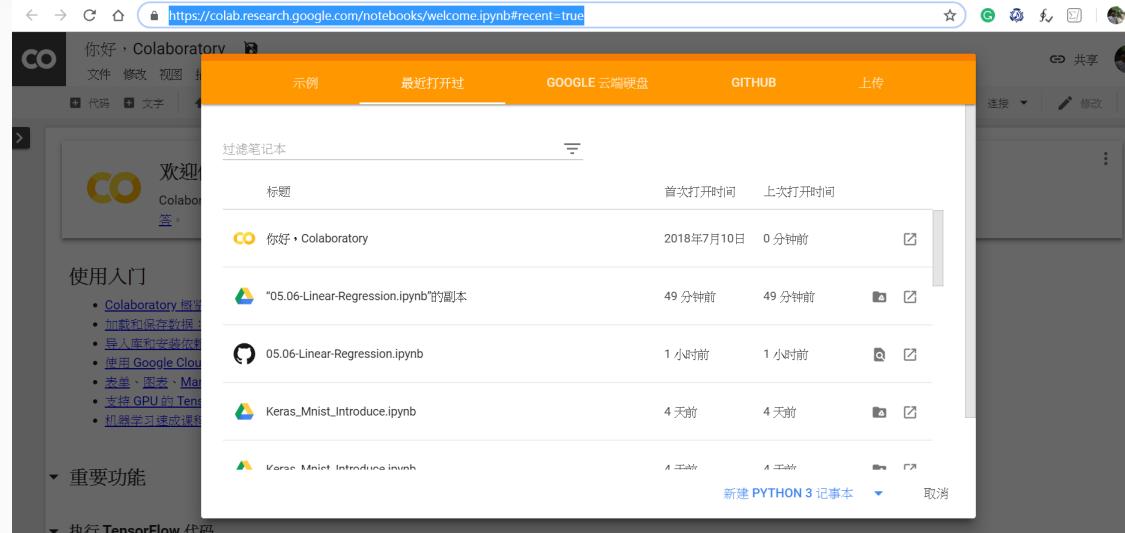
Google Colab Free GPU Tutorial

The screenshot shows a Medium article page. At the top, there's a navigation bar with icons for back, forward, refresh, and search, followed by the URL <https://medium.com/deep-learning-turkey/google-colab-free-gpu-tutorial-e113627b9f5d>. To the right of the URL are links for 'Sign in' and 'Get started'. Below the navigation is a header with a 'M' logo, the text 'DEEP LEARNING TURKEY', a 'Follow' button, and a 'Get started' button. The main navigation menu includes 'HOME', 'HANDS-ON DEEP LEARNING', 'TUTORIALS', and a search icon. The author's profile picture, name 'fuat', and 'Follow' button are displayed. The author's bio reads 'Love to Train Deep Neural Networks' and 'Jan 26 · 8 min read'. The title of the article is 'Google Colab Free GPU Tutorial'. The article content starts with the text: 'Now you can develop **deep learning** applications with [Google Colaboratory](#) - on the **free Tesla K80 GPU**- using [Keras](#), [Tensorflow](#) and [PyTorch](#)'. Below the article is a decorative image of a laptop keyboard with green highlights. At the bottom, there's a call-to-action with the text 'Never miss a story from Deep Learning Turkey, when you sign up for Medium. Learn more' next to a 'GET UPDATES' button.

<https://medium.com/deep-learning-turkey/google-colab-free-gpu-tutorial-e113627b9f5d>

# Python programming

Open colab files



<https://colab.research.google.com/notebooks/welcome.ipynb#recent=true>

# Python programming

## Bike prediction

The screenshot shows a Jupyter Notebook interface with a blue header bar. The main content area has a title "Bike prediction" and a sub-section "Load and prepare the data". It displays Python code for importing libraries and loading data from a CSV file.

Notable styling includes a blue header bar at the top, a light gray background for the main content, and a white box for code snippets. The code is color-coded for syntax highlighting.

```
In [24]: %matplotlib inline  
%config InlineBackend.figure_format = 'retina'  
  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt
```

### Bike prediction

Hi this is Sai. In this project, I will build and train a Neural Network from scratch to predict the number of bikeshare users on a given day.

I will demonstrate how to build a neural network from the ground up, to show the understanding of gradient descent, backpropagation and some basic techniques of data cleansing. And that is the reason that I didn't use some higher level tools such as Tensorflow in this demo.

The data comes from the [UCI Machine Learning Database](#).

```
In [25]: data_path = 'Bike-Sharing-Dataset/hour.csv'  
  
rides = pd.read_csv(data_path)
```

### Load and prepare the data

A critical step in working with neural networks is preparing the data correctly. Variables on different scales make it difficult for the network to efficiently learn the correct weights.

<http://www.sai-tai.com/blog/bikeshare-prediction/>

# Python programming

We will see two types of global (interpolatory) rules:

- Newton-Cotes — interpolatory on uniformly spaced nodes.
- Gauss rules — interpolatory on optimally chosen point sets.

[https://relate.cs.illinois.edu/course/cs450-f17/file-version/f753d2d1a50ecbf2ca66202aa894e6d67c24efdb/lecture-notes/chap08\\_lec2.pdf](https://relate.cs.illinois.edu/course/cs450-f17/file-version/f753d2d1a50ecbf2ca66202aa894e6d67c24efdb/lecture-notes/chap08_lec2.pdf)

Numerical Integration  
Numerical Differentiation  
Richardson Extrapolation

Quadrature Rules  
Adaptive Quadrature  
Other Integration Problems

## Quadrature Rules, continued

- Quadrature rules are based on polynomial interpolation
- Integrand function  $f$  is sampled at finite set of points
- Polynomial interpolating those points is determined
- Integral of interpolant is taken as estimate for integral of original function
- In practice, interpolating polynomial is not determined explicitly but used to determine weights corresponding to nodes
- If Lagrange interpolation is used, then weights are given by

$$w_i = \int_a^b \ell_i(x), \quad i = 1, \dots, n$$



# Python programming

## 5-point Gaussian quadrature rule

$$\begin{aligned}\int_a^b f(x) dx &= \int_{-1}^1 f\left(\frac{b-a}{2}t + \frac{a+b}{2}\right) \frac{b-a}{2} dt \\ &\approx \sum_{i=1}^5 f\left(\frac{b-a}{2}r_{5,i} + \frac{a+b}{2}\right) \frac{b-a}{2} c_{5,i} \\ &\approx \sum_{i=1}^5 f(x_i) w_i,\end{aligned}$$

where

$$x_i = \frac{b-a}{2}r_{5,i} + \frac{a+b}{2}, \quad w_i = \frac{b-a}{2}c_{5,i}, \quad i = 1, \dots, 5.$$

In this example,  $a = 0$  and  $b = 1$ , so the nodes and weights for a [5-point Gaussian quadrature rule](#) for integrating over  $[0, 1]$  are given by

$$x_i = \frac{1}{2}r_{5,i} + \frac{1}{2}, \quad w_i = \frac{1}{2}c_{5,i}, \quad i = 1, \dots, 5,$$

which yields

$i$	Nodes $x_i$	Weights $w_i$
1	0.95308992295	0.11846344250
2	0.76923465505	0.23931433525
3	0.50000000000	0.28444444444
4	0.23076534495	0.23931433525
5	0.04691007705	0.11846344250

It follows that

$$\begin{aligned}\int_0^1 e^{-x^2} dx &\approx \sum_{i=1}^5 e^{-x_i^2} w_i \\ &\approx 0.11846344250e^{-0.95308992295^2} + 0.23931433525e^{-0.76923465505^2} + \\ &\quad 0.2844444444e^{-0.5^2} + 0.23931433525e^{-0.23076534495^2} + \\ &\quad 0.11846344250e^{-0.04691007705^2} \\ &\approx 0.74682412673352.\end{aligned}$$

# Python programming

## Gaussian Quadrature Weights and Abscissae

Weights and Abscissae Tables for n=2 to n=64

n = 2

jump to n = 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,  
64

i	weight - w <sub>i</sub>	abscissa - x <sub>i</sub>
1	1.0000000000000000	-0.5773502691896257
2	1.0000000000000000	0.5773502691896257

n = 3

jump to n = 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,  
64

i	weight - w <sub>i</sub>	abscissa - x <sub>i</sub>
1	0.8888888888888888	0.0000000000000000
2	0.5555555555555556	-0.7745966692414834
3	0.5555555555555556	0.7745966692414834

n = 4

jump to n = 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,  
64

i	weight - w <sub>i</sub>	abscissa - x <sub>i</sub>
1	0.6521451548625461	-0.3399810435848563
2	0.6521451548625461	0.3399810435848563
3	0.3478548451374538	-0.8611363115940526
4	0.3478548451374538	0.8611363115940526

<https://pomax.github.io/bezierinfo/legendre-gauss.html>

# Python programming

## Quadrature rules based on interpolating functions

A large class of quadrature rules can be derived by constructing [interpolating](#) functions that are easy to integrate. Typically these interpolating functions are [polynomials](#). In practice, since polynomials of very high degree tend to oscillate wildly, only polynomials of low degree are used, typically linear and quadratic.

The simplest method of this type is to let the interpolating function be a constant function (a polynomial of degree zero) that passes through the point  $\left(\frac{a+b}{2}, f\left(\frac{a+b}{2}\right)\right)$ . This is called the *midpoint rule* or *rectangle rule*

$$\int_a^b f(x) dx \approx (b-a)f\left(\frac{a+b}{2}\right).$$

The interpolating function may be a straight line (an [affine function](#), i.e. a polynomial of degree 1) passing through the points  $(a, f(a))$  and  $(b, f(b))$ . This is called the *trapezoidal rule*

$$\int_a^b f(x) dx \approx (b-a) \left( \frac{f(a)+f(b)}{2} \right).$$

For either one of these rules, we can make a more accurate approximation by breaking up the interval  $[a, b]$  into some number  $n$  of subintervals, computing an approximation for each subinterval, then adding up all the results. This is called a *composite rule*, *extended rule*, or *iterated rule*. For example, the composite trapezoidal rule can be stated as

$$\int_a^b f(x) dx \approx \frac{b-a}{n} \left( \frac{f(a)}{2} + \sum_{k=1}^{n-1} \left( f\left(a + k \frac{b-a}{n}\right) \right) + \frac{f(b)}{2} \right),$$

where the subintervals have the form  $[a + kh, a + (k+1)h] \subset [a, b]$ , with  $h = \frac{b-a}{n}$  and  $k = 0, \dots, n-1$ . Here we used subintervals of the same length  $h$  but one could also use intervals of varying length  $(h_k)_k$ .

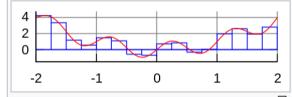


Illustration of the rectangle rule.

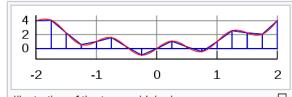


Illustration of the trapezoidal rule.

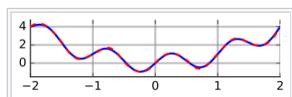


Illustration of Simpson's rule.

[https://en.wikipedia.org/wiki/Numerical\\_integration](https://en.wikipedia.org/wiki/Numerical_integration)