

---

# **Weekly Documentation**

***Release 0.0.1***

**Yen-Kuan Wu**

**Oct 29, 2017**



---

## Contents:

---

<b>1</b>	<b>Welcome</b>	<b>1</b>
1.1	Documentation . . . . .	1
1.2	Learn Python . . . . .	1
1.3	Contributor Guide . . . . .	2
1.4	Coding style . . . . .	3
1.5	Contact us . . . . .	3
<b>2</b>	<b>Mail</b>	<b>5</b>
<b>3</b>	<b>Indices and tables</b>	<b>7</b>



# CHAPTER 1

---

## Welcome

---

**Weekly** is a project which aims to automate all boring stuff in python script without human involved. The primary scenario is about daily life of graduated students in laboratory. Professor asks elder students to do attendance list every week, and sending each member in laboratory mail to inform recent matters and meeting records and so on. That's only interesting for computers ;-)

This project is started by a misanthrope who found life is tough especially in laboratory. All right. Let me change the aim of this project with higher purpose. Make life easy(at least for me).

Here is also a training place for new members who access to our laboratory (unfortunately :-( ). We request basic level of coding skills and commit messages style. Don't be worry about that. Just send us a patch. We don't bite :)

## Documentation

The current implementation is in pure python3.

(... work in progress)

## Learn Python

### Python

As I mentioned before, the primary implementation is in Python. So if you want to fix some bugs or work on new features, you should grasp basic operation of python. Here is a tutorial, [The Python Tutorial](#), written(mostly) by the author of the Python programming language, *Guido van Rossum*.

If you're a non-native English reader and familiar with Traditional Chinese. You could start it by reading the slides or tutorials. As follows:

- **Tutorials**

1. [Python](#)

### 2. Python tutorial - CodeData

- **Slides**

#### 1. Python 3.5 1 - Python

Despite all materials above, if you suggest others, please use **issues** or **PRs** to contribute.

Nevertheless, the way to practice Python is important. It's convenient to examine your assume via [interactive shell of python interpreter](#). However, in order to enhance efficiency, it's really powerful to use IPython and associate with *Jupyter* kernel as a note system which allows you execute python script commands in place. More details could be referred below.

## IPython

(... unfinished)

## Jupyter

(... hang on the air)

## Contributor Guide

Sorry for my chatter. I have to present a guide like a teaching material for variety of people.

## Github and Git

There are three ways to contribute projects on Github. No one is beter, and no one is worst. All of them are necessary for every open source project. As following:

1. Post a message via [Issues](#). This feature, **Issues**, is similiar with other project's trackers. It's useful to keep track of issues you focus on.

- Bug report
- Feature request
- Ask questions
- More details about **Issues**: [Mastering Issues](#)

2. Submit a patch via [Pull Requests](#). The name of **"Pull Request"** may confuse novices for a while. It's shorted for **Send a Pull Request!**

- Fix bugs
- Introduce new features
- Refactor current implementation
- Help documentation
- Learn more about **Pull Request**: [About Pull Requests](#)

3. **Participate in discussion and give some feedbacks.** The two ways I mentioned above, let you take part in conversations and give your opinions.

- Code review

- Answer questions
- Discuss the dis/advantages of features

Moreover, if you want to ask something, please note that asking a good questions is not as easy as you thought. There is a great material, [How To Ask Questions The Smart Way](#), for you to grasp key points of how to ask a question which might be more possible to get replies. There is the Traditional Chinese translation, .

In addition to asking questions, if you want to send PRs, writing clear and meaningful git commit messages are necessary. Please refer to [How to Write a Git Commit Message](#). That's really helpful. Thank *louis lu* for Traditional Chinese translation, [Git Commit Message](#).

## Coding style

Follow [PEP 8 – Style Guide for Python Code](#).

Moreover, there are some built-in tools to check coding style, analyse inconsistent terms with PEP8 and format Python code to conform to PEP8 in place.

By the way, we are pleasure to accept any features which could improve code quality.

## Contact us

- **Current maintainer**
  - Yen-Kuan Wu <[yenwu.tw@gmail.com](mailto:yenwu.tw@gmail.com)>





```
class mail.mail.Mail (prot, *argv)
```

```
    __init__ (prot, *argv)
```

**Args:**

- prot (Instance Object): Protocol instance
- \*argv (Pointer to array): Variable-length argument

```
    login (account, passwd)
```

**Parameters**

- **account** (*string*) – Mail account
- **passwd** (*string*) – Mail password

**Returns** Nothing

**Return type** void

```
    quit ()
```

Close an cleanup

```
    send (frm, to, subject, content)
```

**Args:**

- frm (str): source address of this email
- to (str): target address of this email
- subject (str): Subject of this email
- content (str): Content of this email



## CHAPTER 3

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



## Symbols

`__init__()` (mail.mail.Mail method), 5

## L

`login()` (mail.mail.Mail method), 5

## M

Mail (class in mail.mail), 5

## Q

`quit()` (mail.mail.Mail method), 5

## S

`send()` (mail.mail.Mail method), 5