

# Intro to Robotics with Raspberry Pi!

## Section 1. General Concepts – Linux fundamentals

# Outline

## Linux Fundamentals

### **Getting Started with Linux**

Desktop Environments and Shells

Remote Login with SSH

Remote Login with VNC

Navigating Linux and Using Python Within

### **Using Python on the Raspberry Pi**

From the Mac OS Terminal

From VNC Using the LXTerminal

From VNC Using an IDE

### **Python Applications**

Python Web Application Using Flask

Python Dataviz Application

### **Python Programs to Control Hardware**

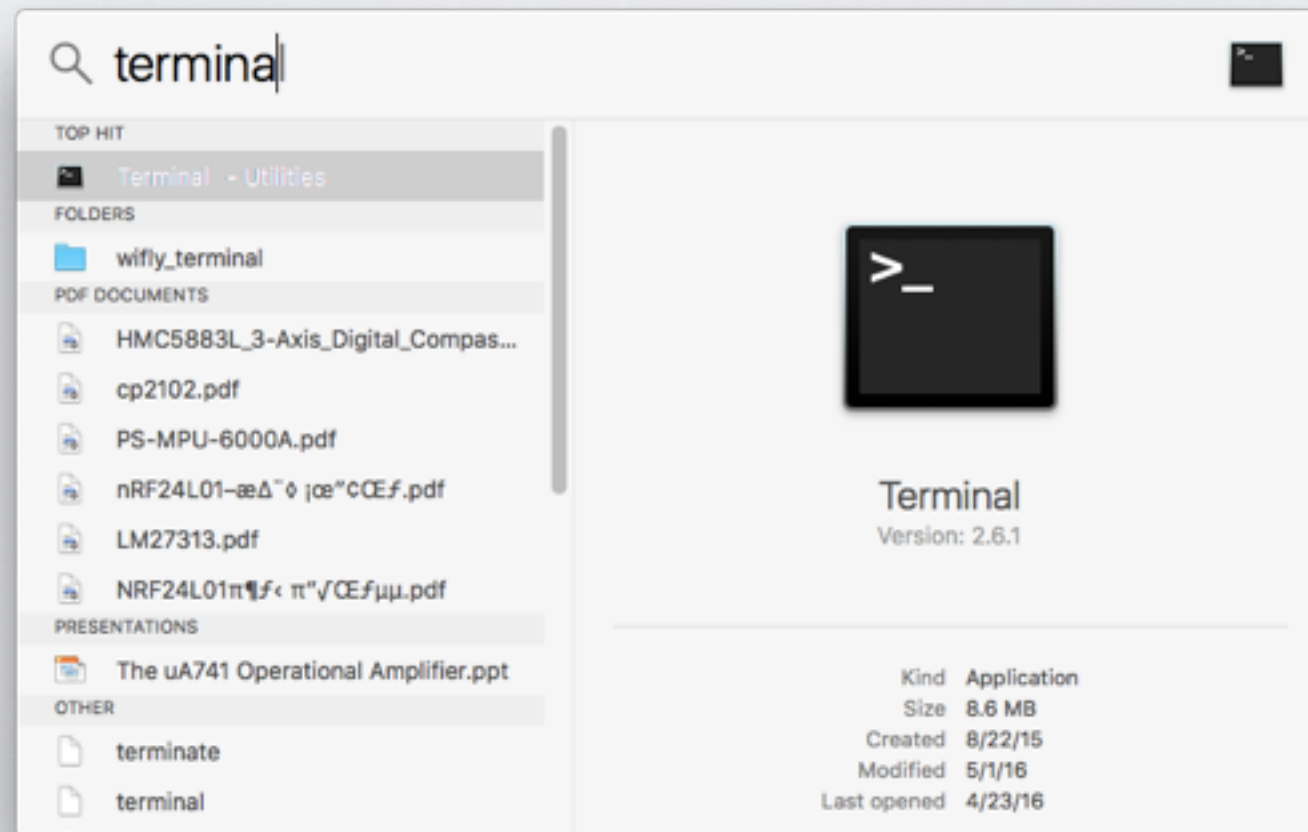
Blinking an LED

Blinking multiple LEDs

# Getting Started with Linux

# Desktop Environments and Shells

- A **desktop environment** is a collection of software that provides a standard look and feel: Mac OS, Windows, GNOME.
- A **shell** is a user interface for access to an operating system's services.
- Shells can either be **graphical** or **text-based**.
- OS X's **Terminal.app** allows us to access the system's text-based shell.



# Example Uses of Text Shells

- Applications (programs) can be used via a graphical or text-based **shell**.
- A **terminal emulator** is an application that allows us to interact with the text-based shell.
- Commonly used programs inside a terminal emulator include:

```
say Hello Disney Interactive!
```

```
vi
```

```
ssh
```

```
wget
```

```
python
```

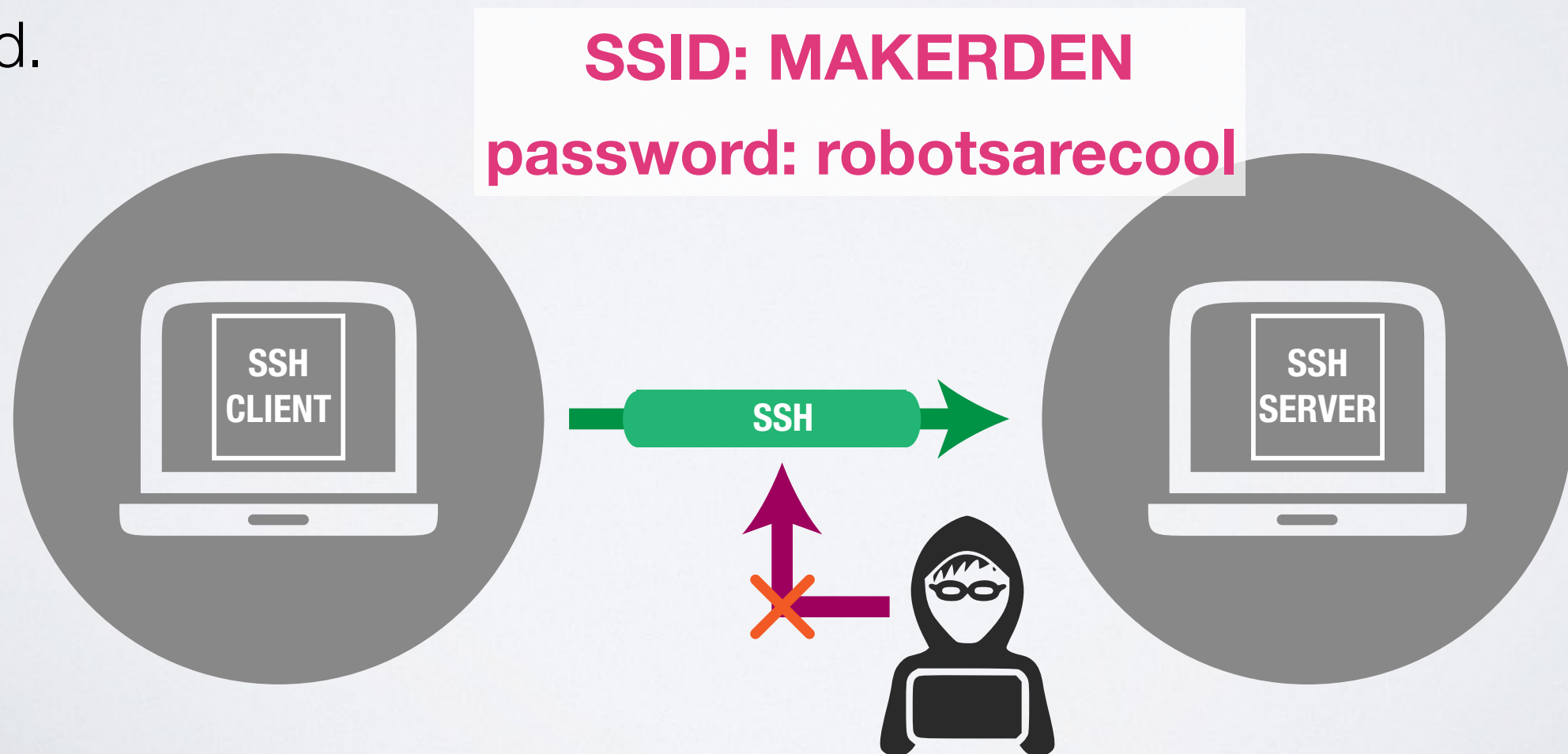
```
python filename.py
```



# Remote Login with SSH

ssh

- Secure Shell (SSH) is an encrypted network protocol that allows remote login and other network services to operate securely over an unsecured network.
- Both OS X and the Raspberry Pi OS have the necessary SSH programs installed.




# Remote Login with SSH

- Using the Terminal app let's log in to our Raspberry Pi:

```
ssh pi@denbotNN.local
```

- **pi** is the name of the default user. Default password is **raspberry**.
- **NN** should be the number of your PyDen Bot!



```
x1sc0 — pi@denbot01: ~ — ssh pi@denbot01.local
~ — pi@denbot01: ~ — ssh pi@denbot01.local

[21:58 $ ssh pi@denbot01.local
The authenticity of host 'denbot01.local (fe80::ba27:ebff:fe08:6cc4%en0)' can't
be established.
ECDSA key fingerprint is SHA256:7+t/Wcf5pkLYwCuUo/aqZEbsxGah084h/KxqXMfP04U.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'denbot01.local,fe80::ba27:ebff:fe08:6cc4%en0' (ECDSA
) to the list of known hosts.
[pi@denbot01.local:~]$
```

SSH password: raspberry  
(no characters are shown as you type it in!)

The programs included with the Debian GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/\*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

Last login: Sun May 8 06:50:18 2016

pi@denbot01:~ \$

# Remote Login with SSH

- You're in! All the commands you now type are executed on the **Raspberry Pi**'s processor!
- Many of the programs we can use on our Mac's OS can also be run on the Raspberry Pi's OS.

```
vi
```

```
ssh
```

```
wget
```

```
python
```

```
python filename.py
```

- We can close the 'session' by entering:

```
exit
```



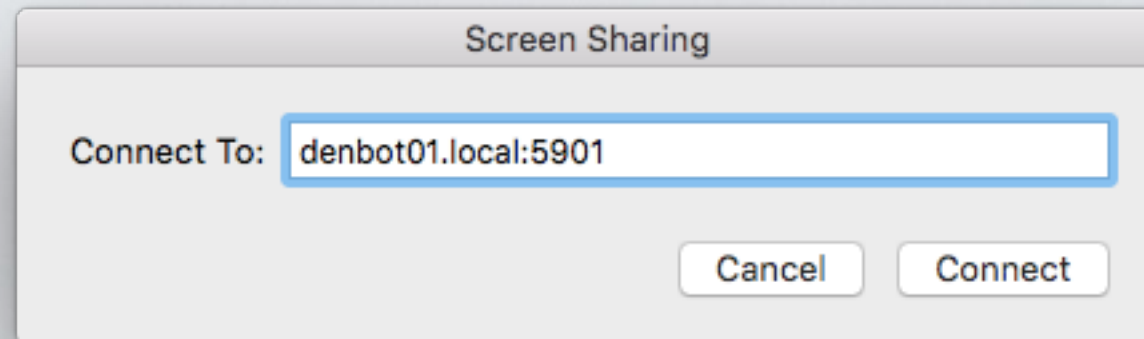
# Remote Login with VNC

- Thankfully, we can also establish a GUI (Graphical User Interface)-based connection!
- Virtual Network Computing (VNC) is a graphical desktop sharing system that allows remote control another computer.
- Both OS X and the Raspberry Pi OS have the necessary VNC programs installed.
- On the terminal enter (ensure you're logged out of your Raspberry Pi!):

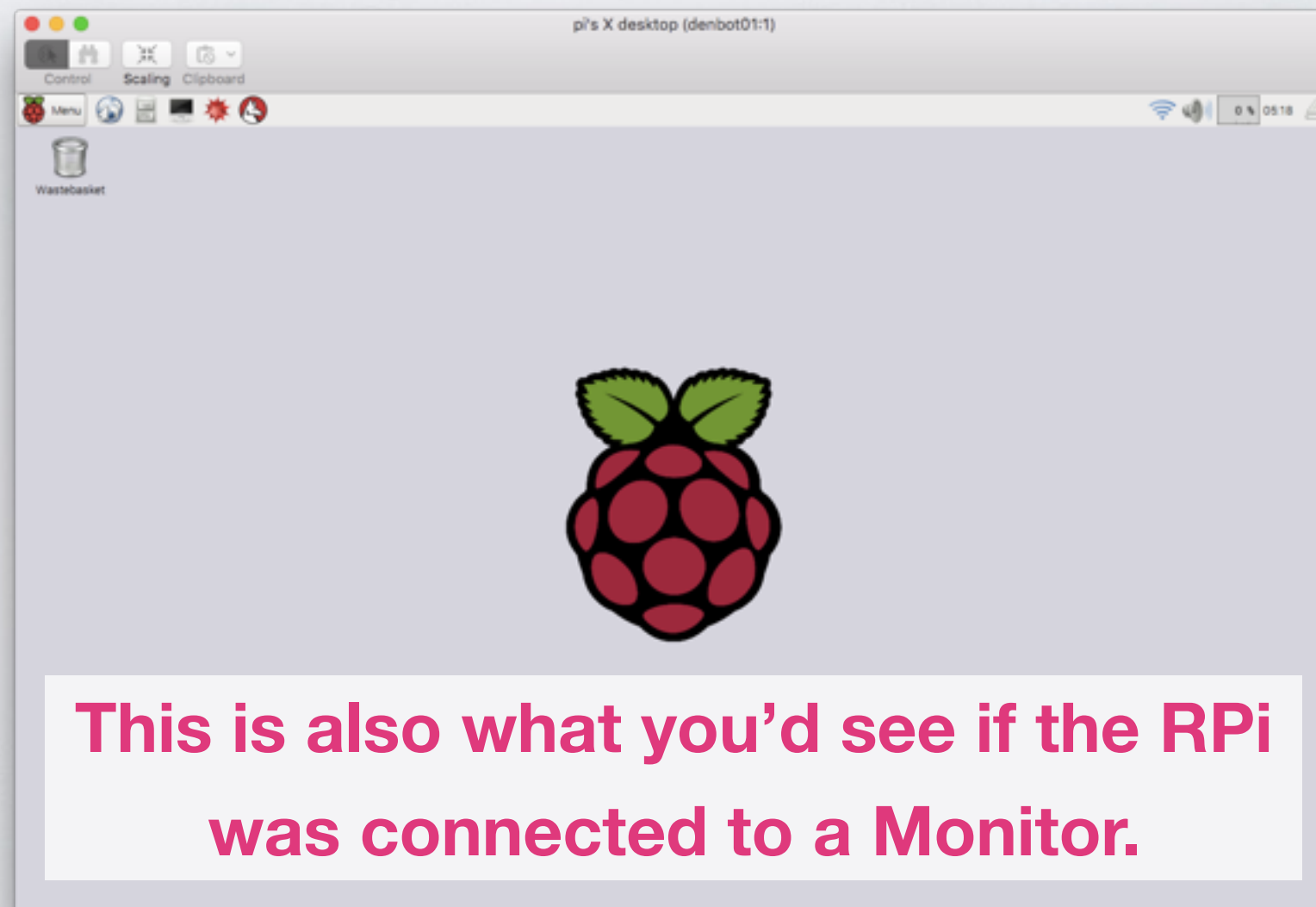
```
open -a /System/Library/CoreServices/Applications/Screen\ Sharing.app
```

# Remote Login with VNC

- Inside the field enter your PyDen Bot name (e.g., denbot01.local:5901):

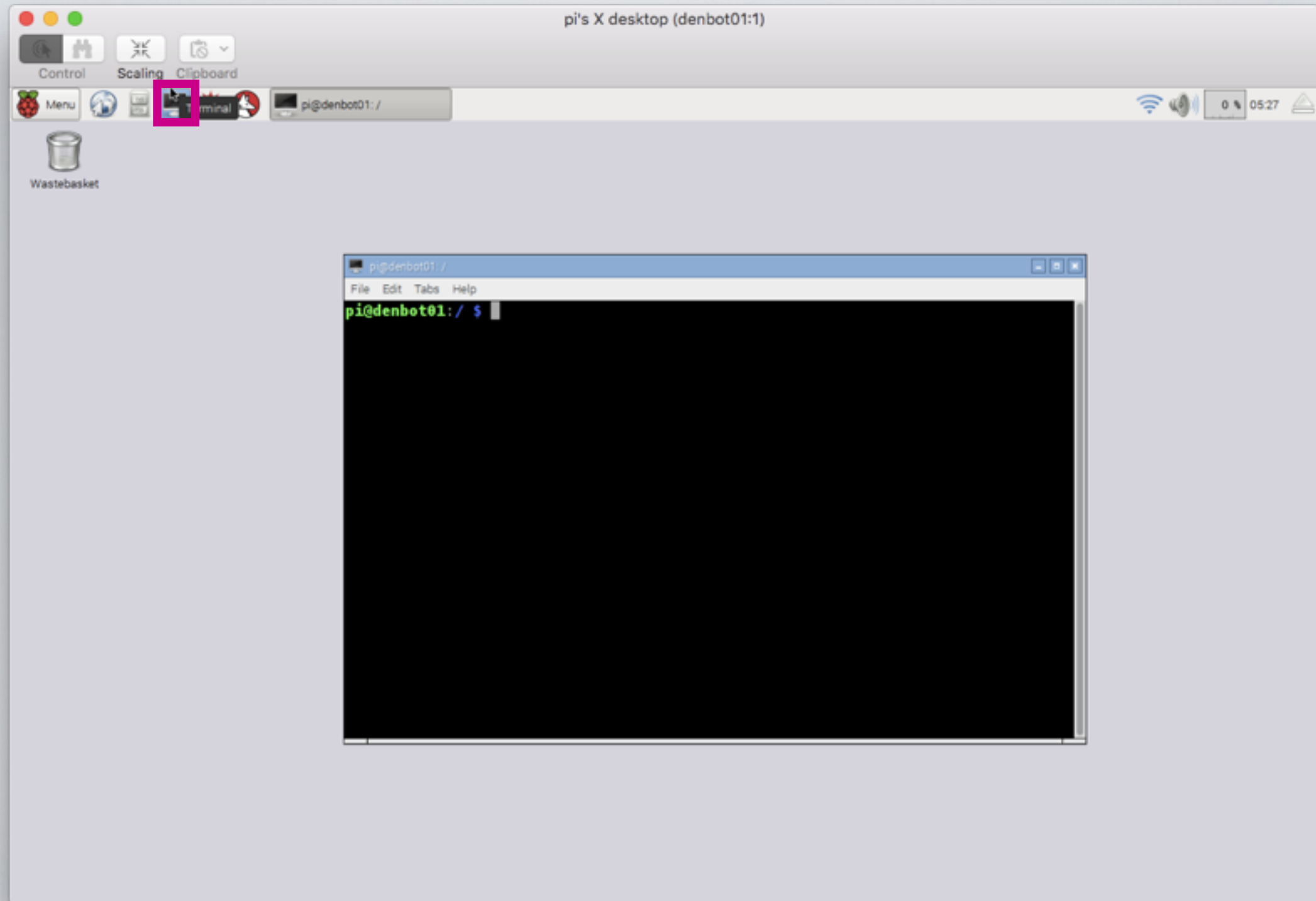


\*5901 is the port where the Raspberry Pi listens for incoming VNC connections. Default password is **pidenbot**.



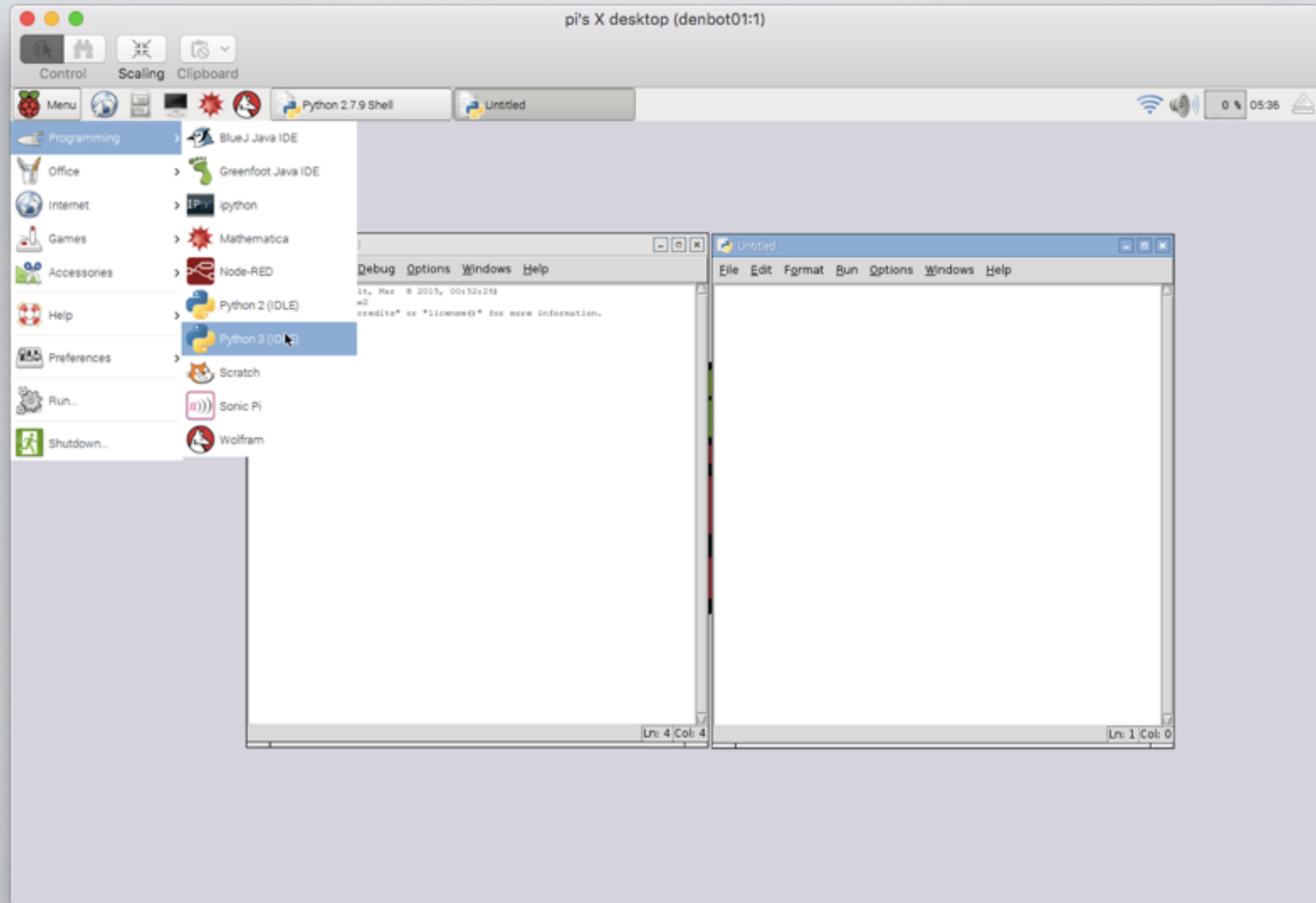
# Linux Fundamentals

- The Raspberry Pi OS (Raspbian) also has a **Terminal emulator** (**LXTerminal**) we can use!



# Using Python in Raspbian

- An Integrated Development Environment for Python (IDLE) is pre-installed in Raspbian (though not used in this class (-:).





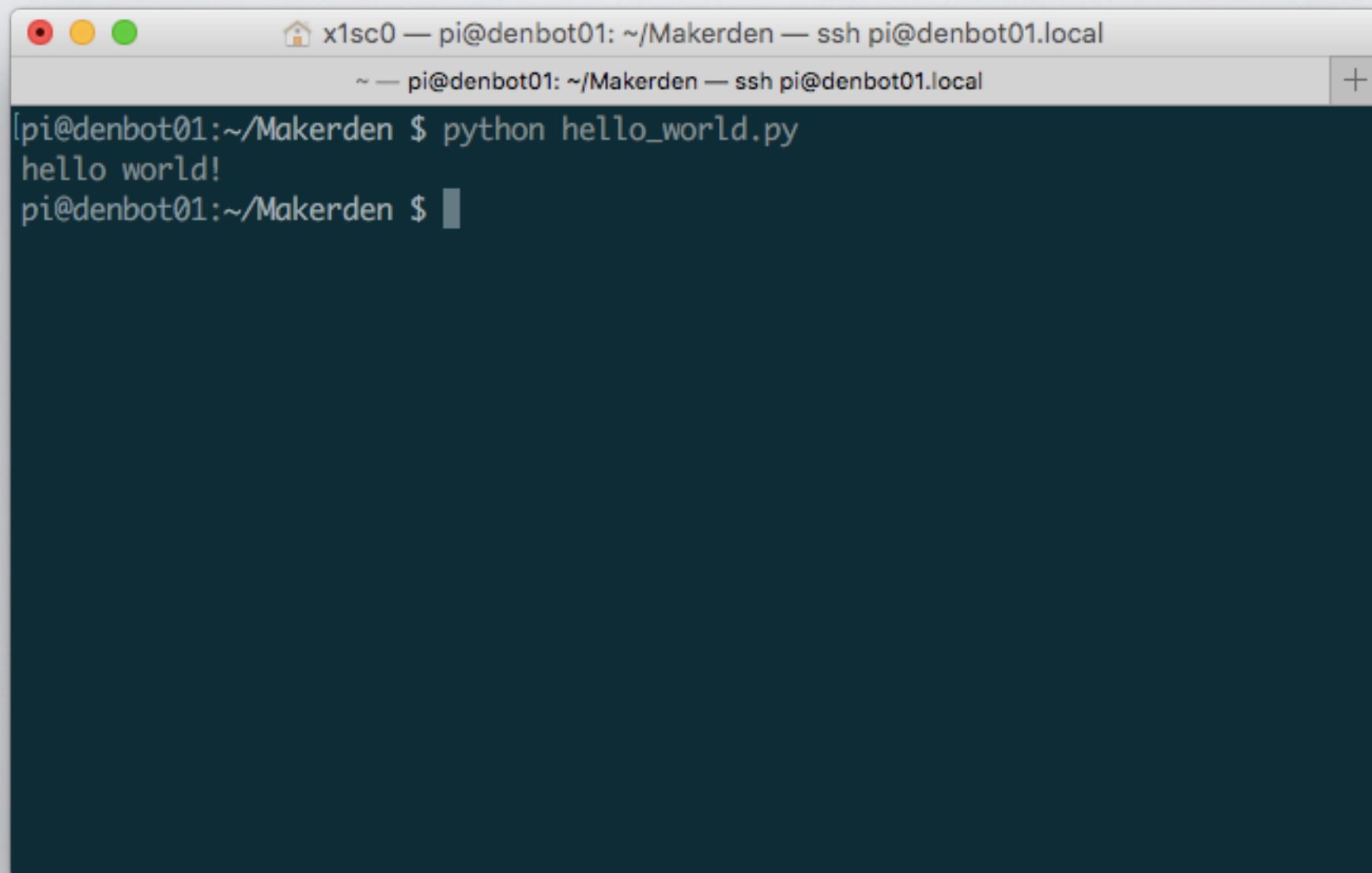
# Using Python on the Raspberry Pi

# Using Python On The Raspberry Pi

- We've got many options for writing and running Python scripts on the Pi.
  1. Connecting via **SSH** and using command-line tools (Mac OS Terminal).

```
cd ~/Makerden
```

```
nano hello_world.py
```



The screenshot shows a Mac OS Terminal window with a dark blue background. The title bar at the top reads "x1sc0 — pi@denbot01: ~/Makerden — ssh pi@denbot01.local". Below the title bar, the terminal shows the following commands and output:

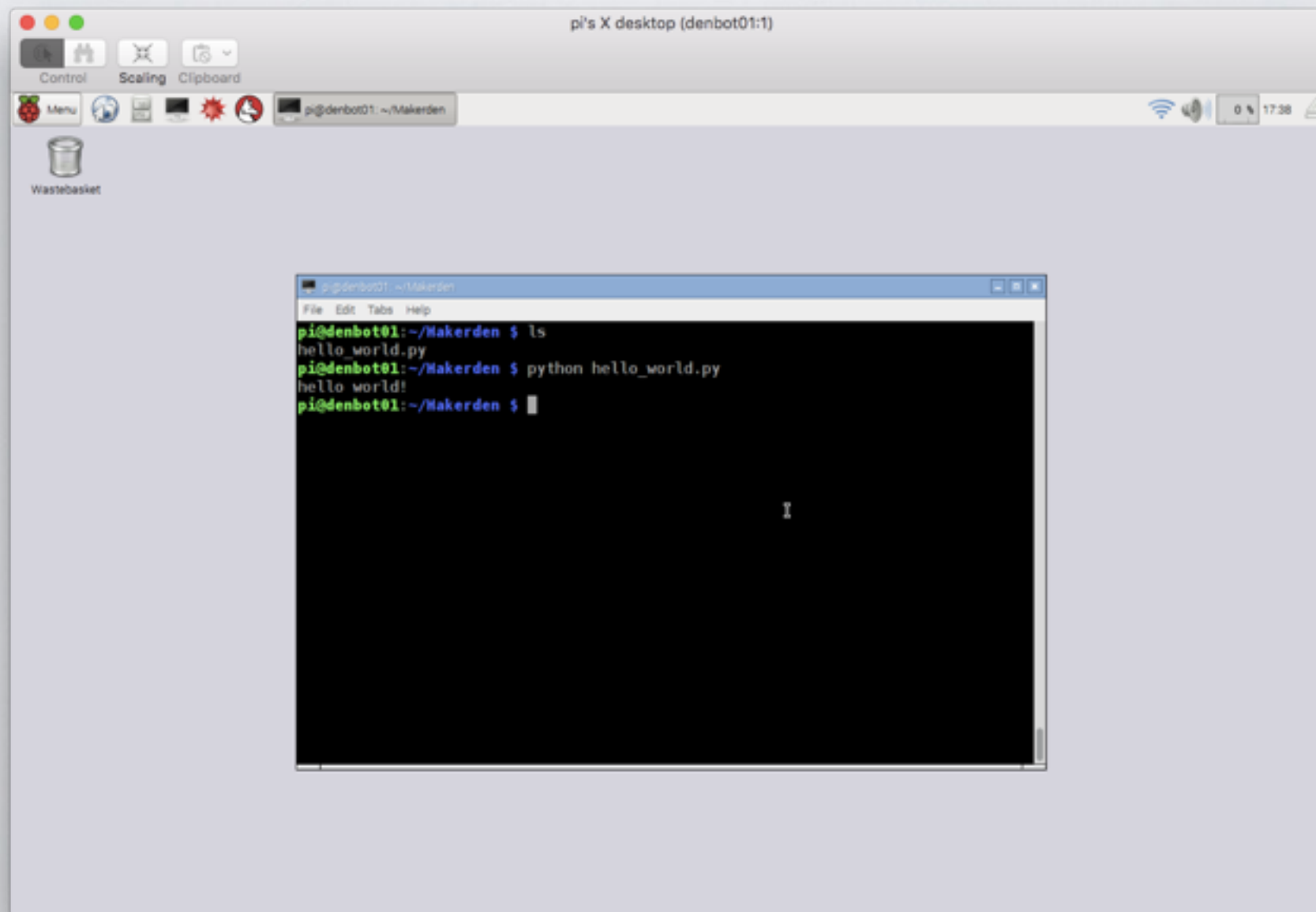
```
[pi@denbot01:~/Makerden $ python hello_world.py  
hello world!  
pi@denbot01:~/Makerden $
```

# Using Python On The Raspberry Pi

- We've got many options for writing and running Python scripts on the Pi.
  1. Connecting via VNC and using graphical tools (Raspbian Desktop).
  2. Connecting via VNC and using command-line tools (Raspbian Terminal).

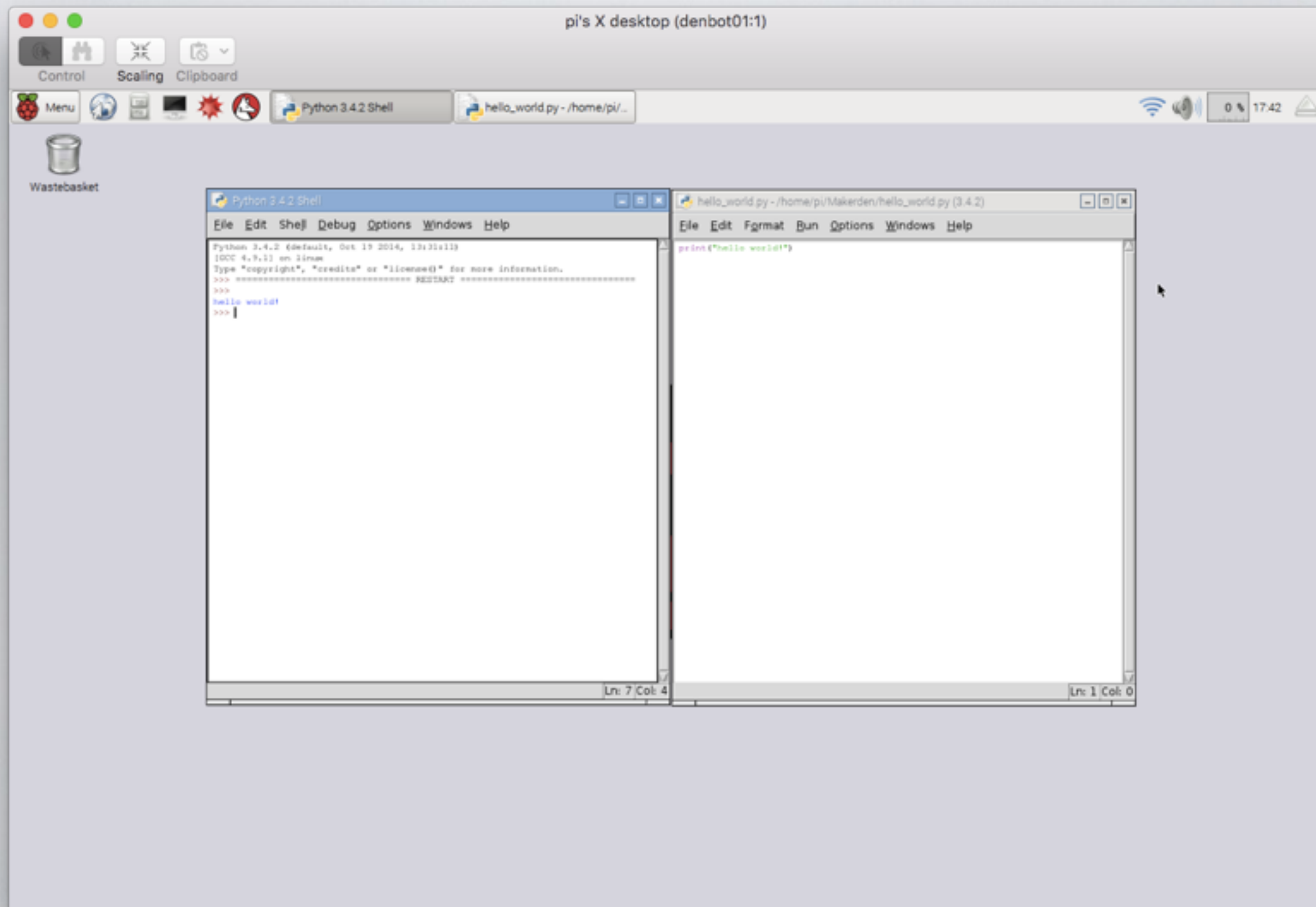
```
cd ~/Makerden
```

```
nano hello_world.py
```



# Using Python On The Raspberry Pi

- We've got many options for writing and running Python scripts on the Pi.
3. Connecting via VNC and using **Python IDE** (Raspbian).

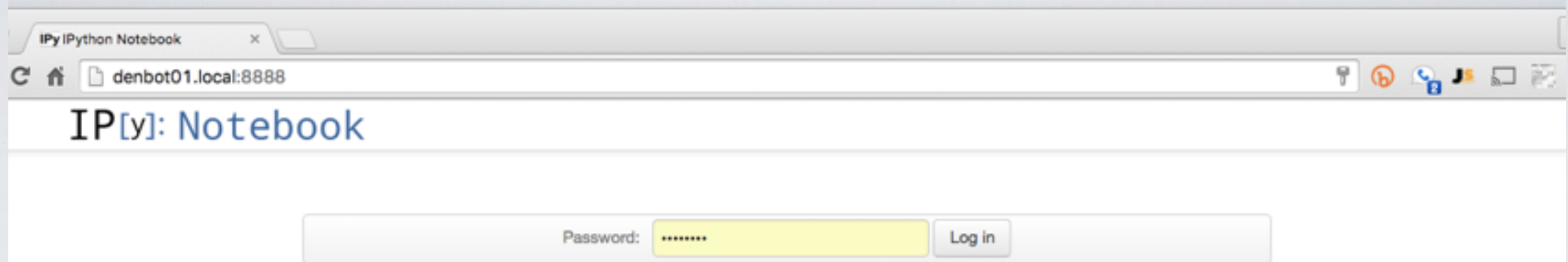




# Using Python On The Raspberry Pi

- We've got many options for writing and running Python scripts on the Pi.

## 4. Using iPython notebook server! (what we'll be using)

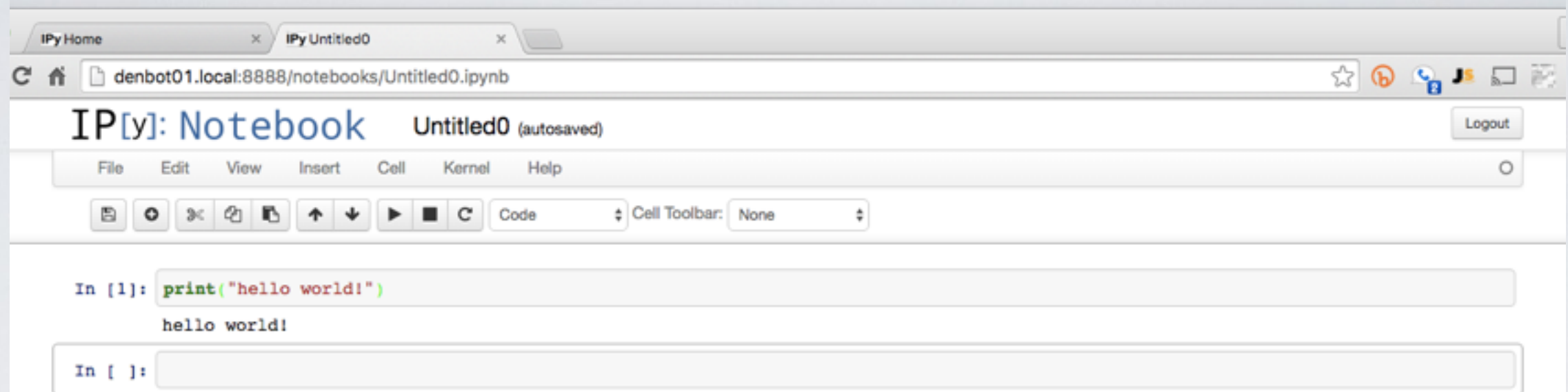


**Navigate to: <http://denbotNN.local:8888>**

**Default password: pidenbot**

# Using Python On The Raspberry Pi

- iPython notebook will work with its own file format (extension .ipynb)
- Choose **File** → **Download as** → **Python (.py)** to convert.



**Click the 'play' button to run a cell!**

# **Writing Python Programs On iPython Notebook**

# The Python Package Index (PyPI)

- The Python Package Index is a repository of software for the Python programming language.

<https://pypi.python.org/pypi>



» Package Index

## PACKAGE INDEX >>

[Browse packages](#)  
[Package submission](#)  
[List trove classifiers](#)  
[List packages](#)  
[RSS \(latest 40 updates\)](#)  
[RSS \(newest 40 packages\)](#)  
[Python 3 Packages](#)  
[PyPI Tutorial](#)  
[PyPI Security](#)  
[PyPI Support](#)  
[PyPI Bug Reports](#)  
[PyPI Discussion](#)  
[PyPI Developer Info](#)

[ABOUT >>](#)

[NEWS >>](#)

[DOCUMENTATION >>](#)

[DOWNLOAD >>](#)

[COMMUNITY >>](#)

[FOUNDATION >>](#)

[CORE DEVELOPMENT >>](#)

## PyPI - the Python Package Index

The Python Package Index is a repository of software for the Python programming language. There are currently **67937** packages here.



To contact the PyPI admins, please use the [Support](#) or [Bug reports](#) links.

### Not Logged In

[Login](#)

[Register](#)

[Lost Login?](#)

Use [OpenID](#)  

### Status

[Nothing to report](#)

### Get Packages

To use a package from this index either "[pip](#) install *package*" ([get pip](#)) or download, unpack and "[python setup.py](#) install" it.

### Package Authors

Submit packages with "[python setup.py upload](#)". The index [hosts package docs](#). You may also use the [web form](#). You must [register](#). Testing? Use [testpypi](#).

### Infrastructure

To interoperate with the index use the [JSON](#), [OAuth](#), [XML-RPC](#) or [HTTP](#) interfaces. Use [local mirroring or caching](#) to make installation more robust.

Updated	Package	Description
2015-10-21	<a href="#">toil 3.1.0a1.dev48</a>	Pipeline management software for clusters.
2015-10-21	<a href="#">django-knob 1.1</a>	A Django reusable application that performs remote configurations on multiple devices, distributing the operations using Celery.
2015-10-20	<a href="#">song2 0.1.0</a>	Typesafe/Immutable schema for dict object
2015-10-20	<a href="#">django-influxdb-metrics 1.2.1</a>	A reusable Django app that sends metrics about your project to InfluxDB
2015-10-20	<a href="#">luigi-monitor 0.2.2</a>	Send summary messages of your Luigi jobs to Slack.
2015-10-20	<a href="#">flask-autorouter 0.1.1</a>	a utility for generating flask URL routing
2015-10-20	<a href="#">django-templatetags 1.1</a>	Custom template tags for notification
2015-10-20	<a href="#">djangorecipe 2.1.2</a>	Buildout recipe for Django
2015-10-20	<a href="#">SciSalt 1.6.1</a>	Tools to make scientific data analysis easier



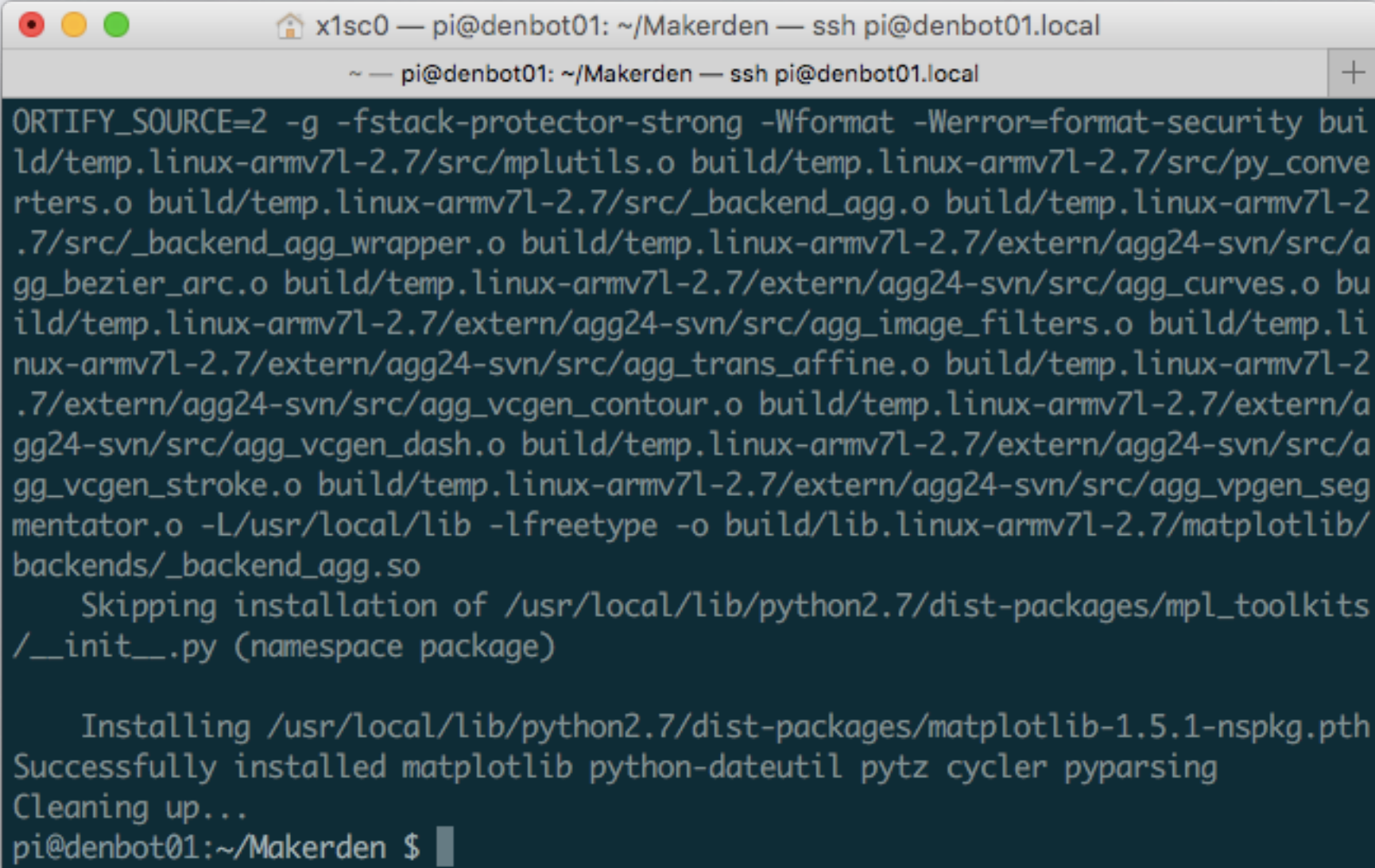
# The Python Package Index (PyPI)

- Let's install a couple of 3rd-party modules using the program **pip**.

```
sudo apt-get update && sudo apt-get install python-dev python-pip
```

```
sudo pip install flask matplotlib werkzeug itsdangerous jinja2
```

\*These steps might've already been completed by your instructor (-:

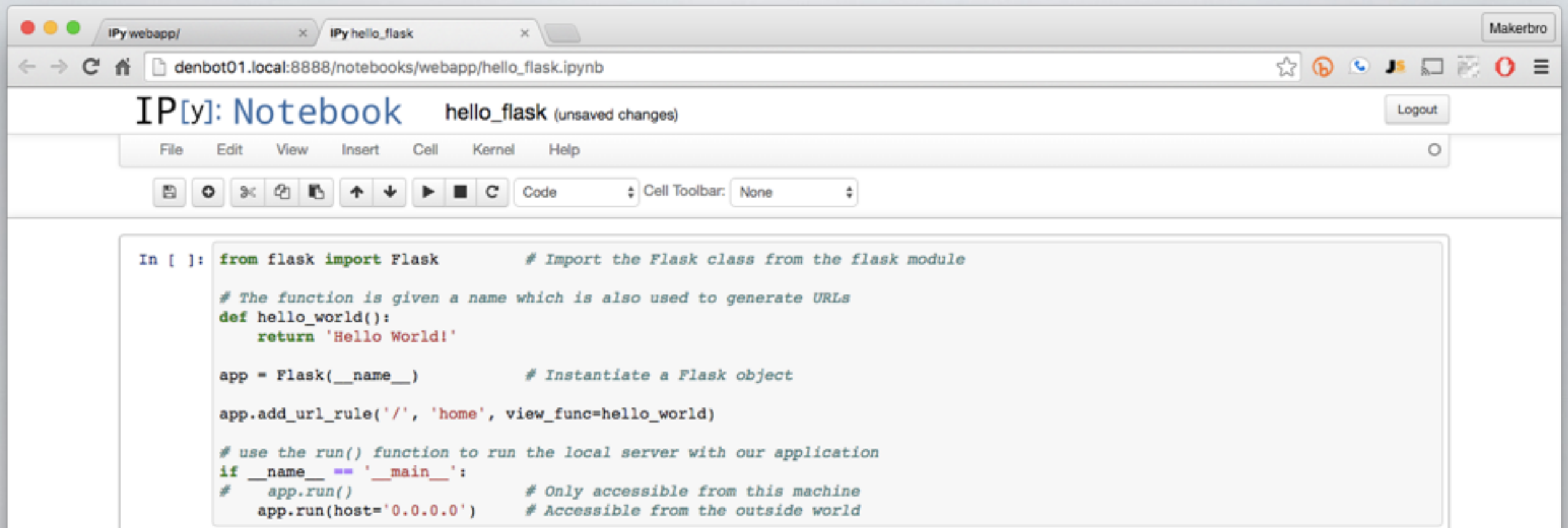


```
x1sc0 — pi@denbot01: ~/Makerden — ssh pi@denbot01.local
~ — pi@denbot01: ~/Makerden — ssh pi@denbot01.local
ORTIFY_SOURCE=2 -g -fstack-protector-strong -Wformat -Werror=format-security bui
ld/temp.linux-armv7l-2.7/src/mplutils.o build/temp.linux-armv7l-2.7/src/py_conve
rters.o build/temp.linux-armv7l-2.7/src/_backend_agg.o build/temp.linux-armv7l-2
.7/src/_backend_agg_wrapper.o build/temp.linux-armv7l-2.7/extern/agg24-svn/src/a
gg_bezier_arc.o build/temp.linux-armv7l-2.7/extern/agg24-svn/src/agg_curves.o bu
ild/temp.linux-armv7l-2.7/extern/agg24-svn/src/agg_image_filters.o build/temp.li
nux-armv7l-2.7/extern/agg24-svn/src/agg_trans_affine.o build/temp.linux-armv7l-2
.7/extern/agg24-svn/src/agg_vcgen_contour.o build/temp.linux-armv7l-2.7/extern/a
gg24-svn/src/agg_vcgen_dash.o build/temp.linux-armv7l-2.7/extern/agg24-svn/src/a
gg_vcgen_stroke.o build/temp.linux-armv7l-2.7/extern/agg24-svn/src/agg_vpgen_seg
mentator.o -L/usr/local/lib -lfreetype -o build/lib.linux-armv7l-2.7/matplotlib/
backends/_backend_agg.so
  Skipping installation of /usr/local/lib/python2.7/dist-packages/mpl_toolkits
/__init__.py (namespace package)

  Installing /usr/local/lib/python2.7/dist-packages/matplotlib-1.5.1-nspkg.pth
Successfully installed matplotlib python-dateutil pytz cyclr pyparsing
Cleaning up...
pi@denbot01:~/Makerden $
```

# Python Web Application Using Flask

- With the necessary modules installed, let's use them to build a simple **web application**!



The screenshot shows a web browser window with two tabs: 'IPy webapp/' and 'IPy hello\_flask'. The address bar shows the URL 'denbot01.local:8888/notebooks/webapp/hello\_flask.ipynb'. The notebook interface has a title bar 'IP[y]: Notebook hello\_flask (unsaved changes)' with a 'Logout' button. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Help'. A toolbar contains various icons for file operations and cell execution. The main area displays a code cell with the following Python code:

```
In [ ]: from flask import Flask          # Import the Flask class from the flask module

# The function is given a name which is also used to generate URLs
def hello_world():
    return 'Hello World!'

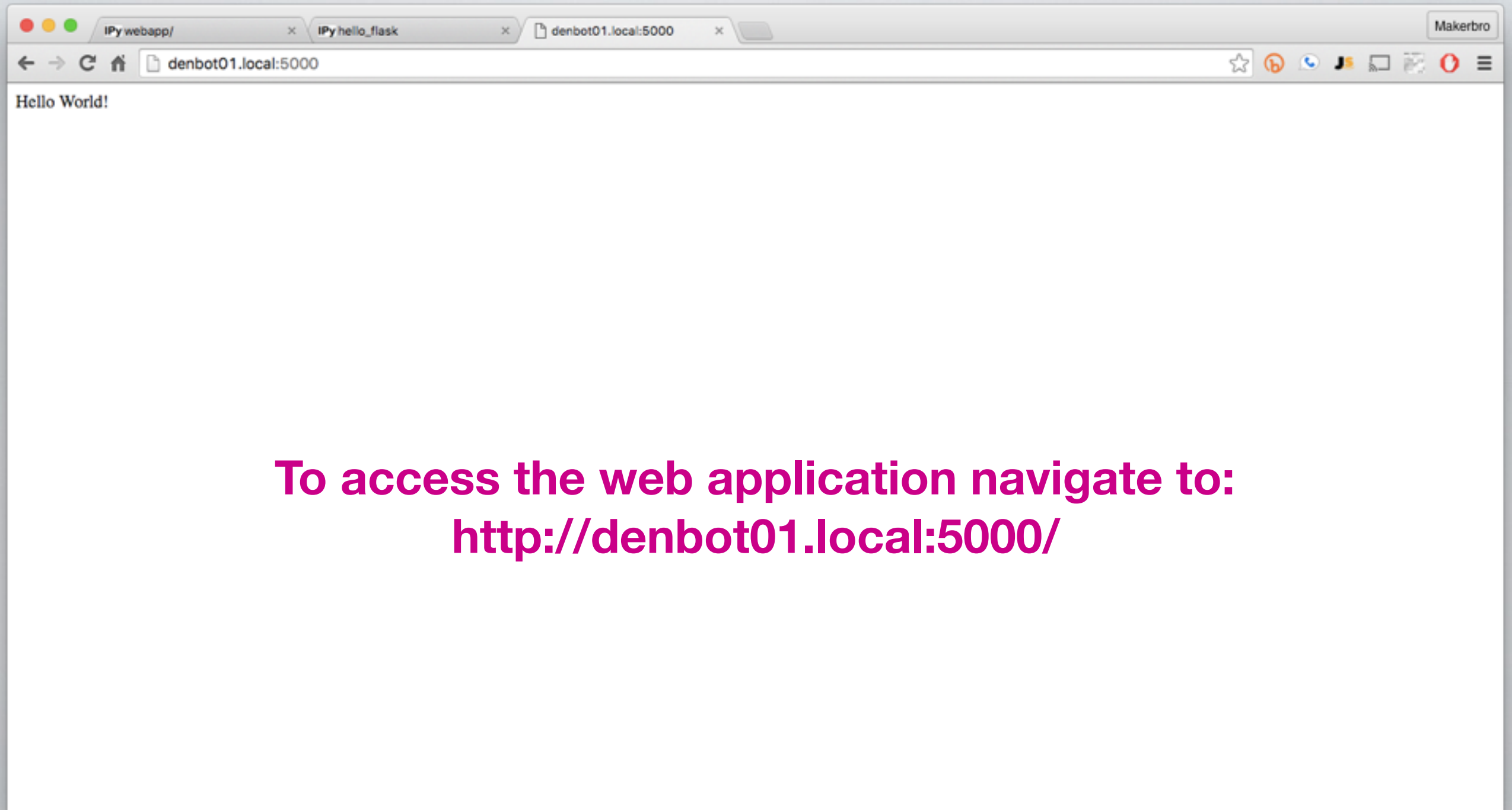
app = Flask(__name__)                    # Instantiate a Flask object
app.add_url_rule('/', 'home', view_func=hello_world)

# use the run() function to run the local server with our application
if __name__ == '__main__':
    # app.run()                        # Only accessible from this machine
    app.run(host='0.0.0.0')            # Accessible from the outside world
```

Create a new notebook “hello\_flask”  
inside the webapp folder.

# Python Web Application Using Flask

- With the necessary modules installed, let's use them to build a simple **web application**!

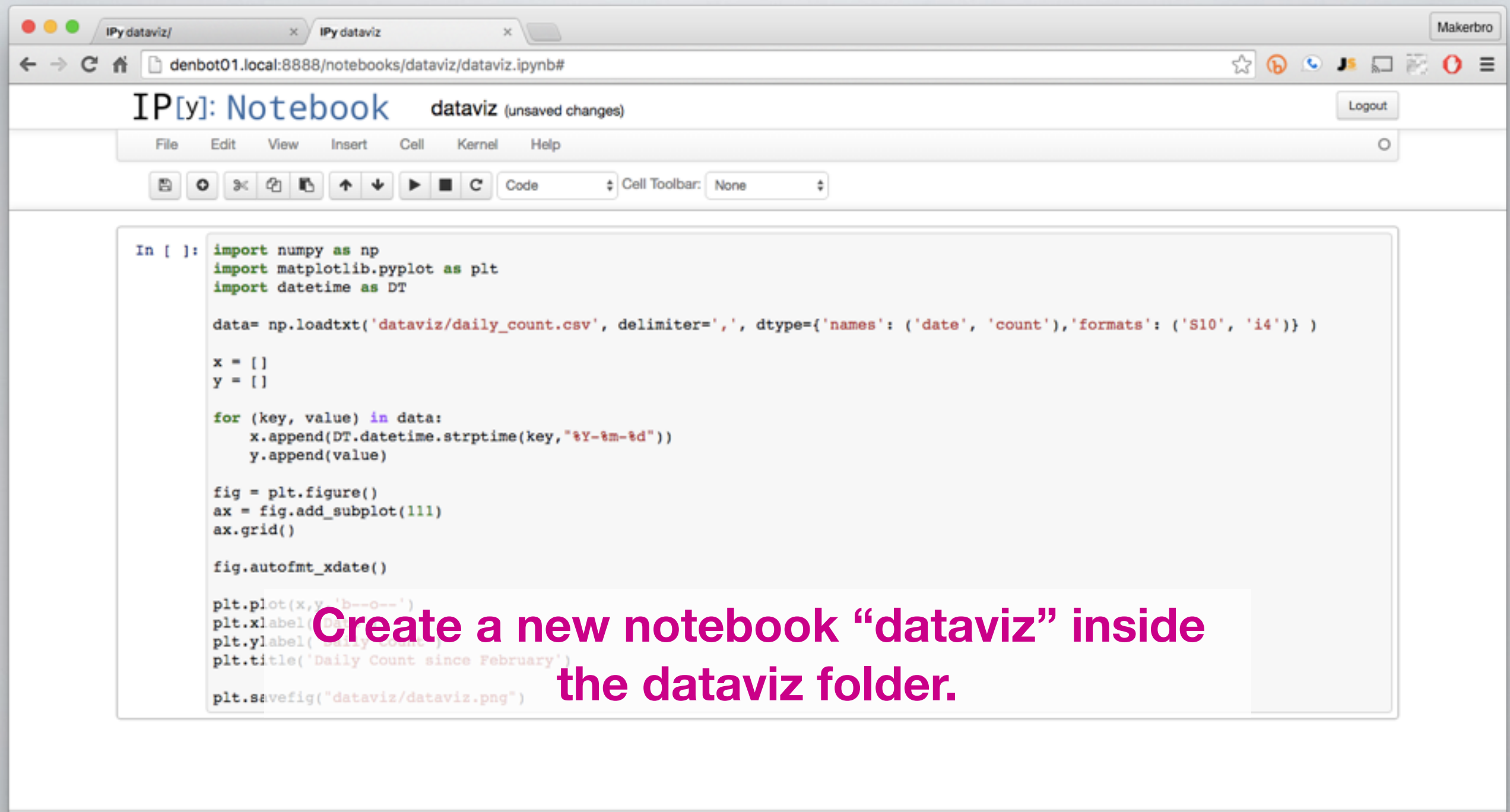


To access the web application navigate to:  
<http://denbot01.local:5000/>



# Python Dataviz Application Using Matplotlib

- With the necessary modules installed, let's use them to **plot data** in a **.csv** file!



The screenshot shows a web-based IPython Notebook interface. The browser address bar indicates the file path: `denbot01.local:8888/notebooks/dataviz/dataviz.ipynb#`. The notebook title is "IP[y]: Notebook" with a subtitle "dataviz (unsaved changes)" and a "Logout" button. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for file operations and code execution. The code cell contains the following Python code:

```
In [ ]: import numpy as np
import matplotlib.pyplot as plt
import datetime as DT

data= np.loadtxt('dataviz/daily_count.csv', delimiter=',', dtype={'names': ('date', 'count'),'formats': ('%S10', '%i4')})

x = []
y = []

for (key, value) in data:
    x.append(DT.datetime.strptime(key, "%Y-%m-%d"))
    y.append(value)

fig = plt.figure()
ax = fig.add_subplot(111)
ax.grid()

fig.autofmt_xdate()

plt.plot(x,y, 'b--o--')
plt.xlabel('Date')
plt.ylabel('Daily Count')
plt.title('Daily Count since February')

plt.savefig("dataviz/dataviz.png")
```

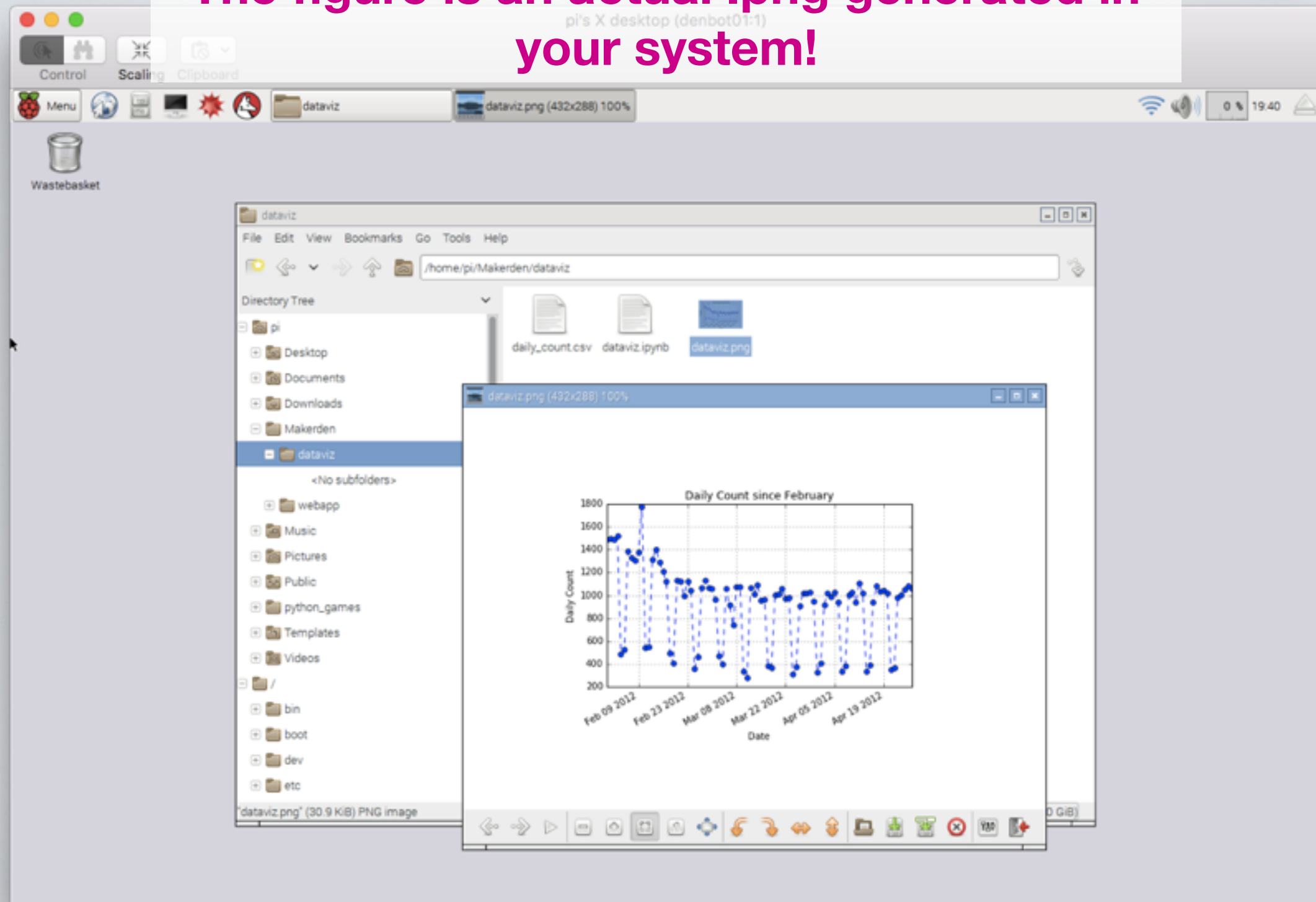
Overlaid on the bottom right of the code cell is a pink text box with the instruction: "Create a new notebook 'dataviz' inside the dataviz folder."



# Python Dataviz Application Using Matplotlib

- With the necessary modules installed, let's use them to **plot data** in a **.csv** file!

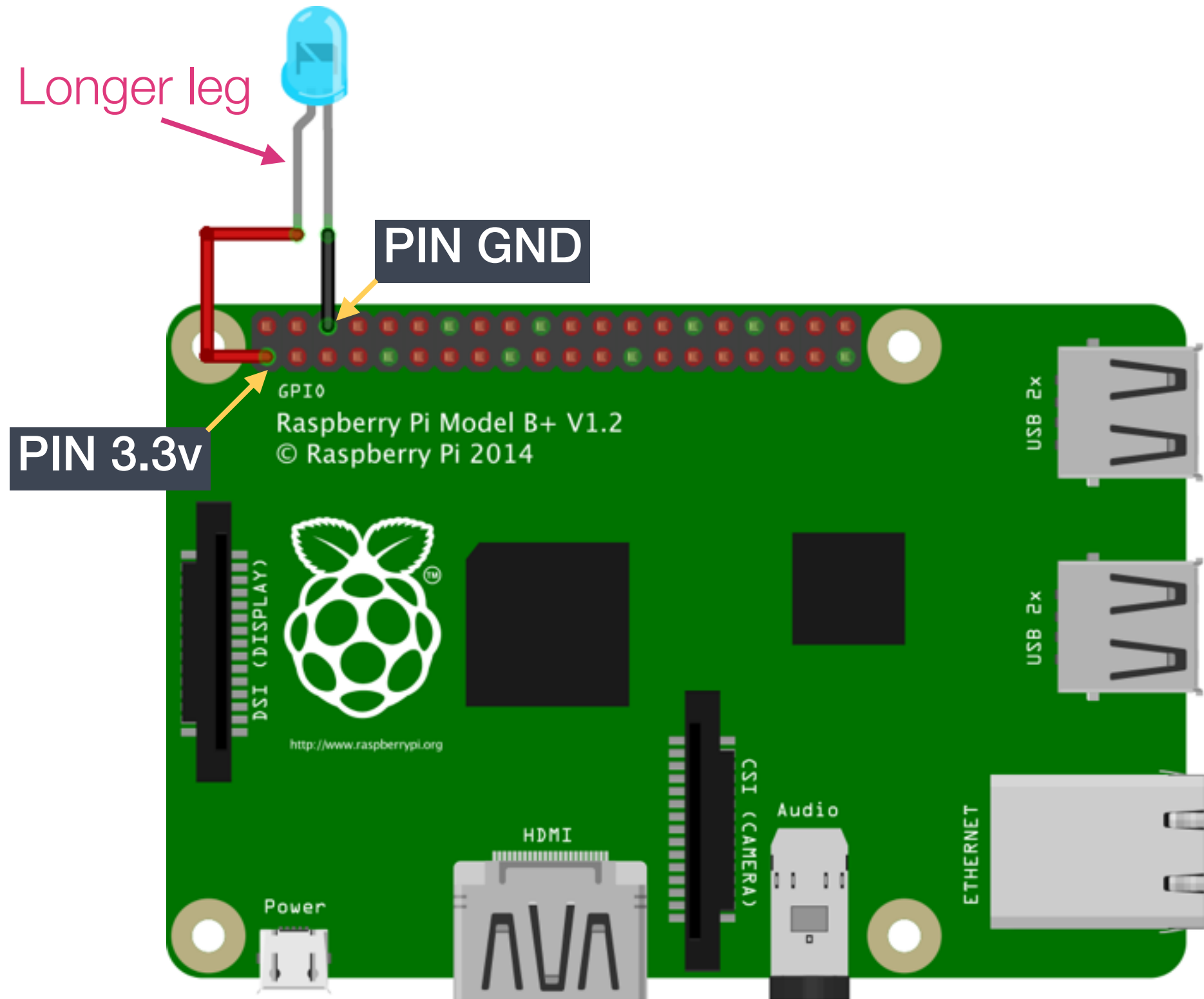
The figure is an actual .png generated in your system!























# **Writing Python Programs To Control Low-Level Hardware**

# Wiring Your First Circuit

- Let's jump right into the deep-end and wire our first circuit!



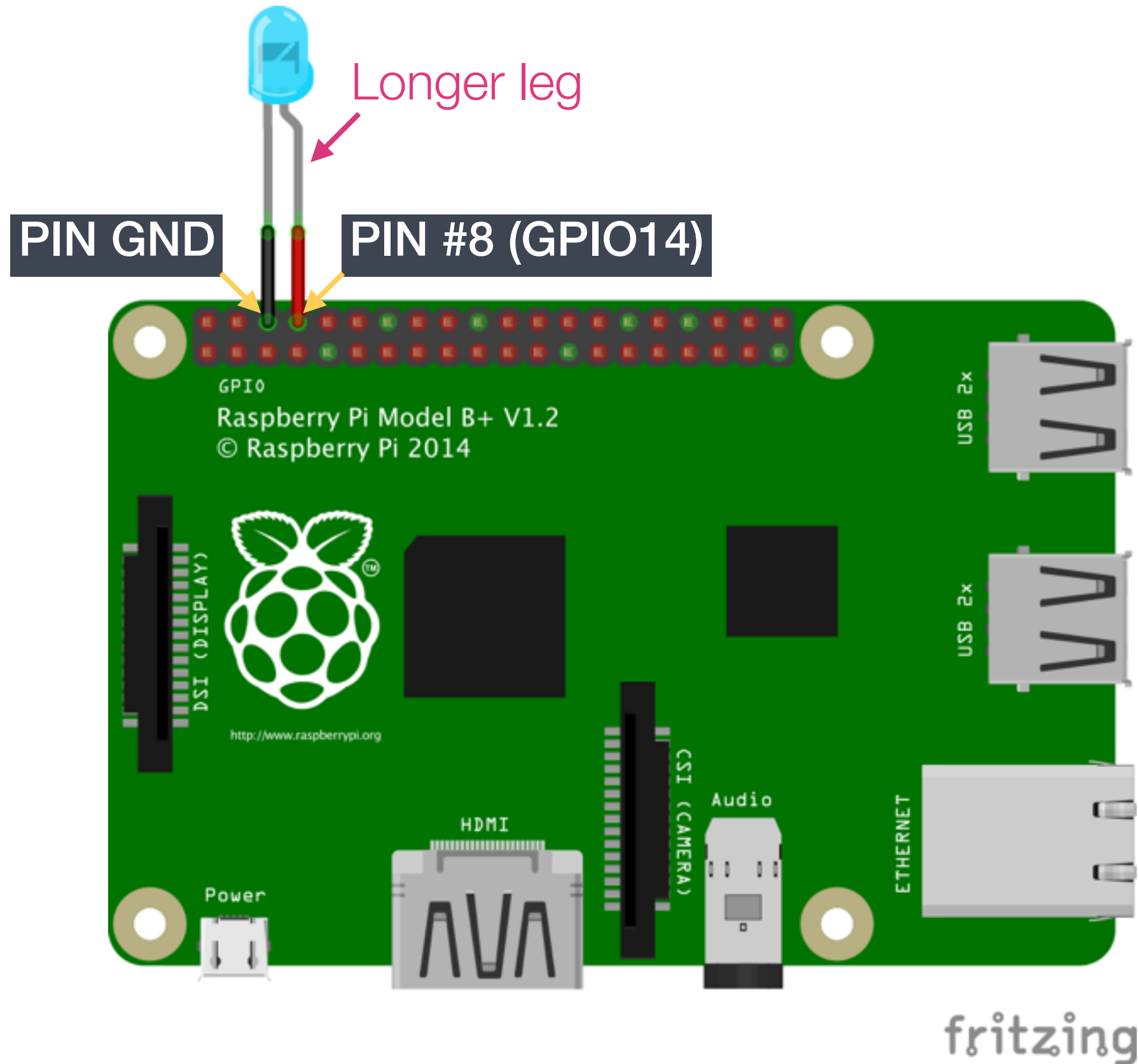
# Raspberry Pi2 GPIO Header

<i>Pin#</i>	<i>NAME</i>		<i>NAME</i>	<i>Pin#</i>
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I <sup>2</sup> C)		DC Power 5v	04
05	GPIO03 (SCL1 , I <sup>2</sup> C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I <sup>2</sup> C ID EEPROM)		(I <sup>2</sup> C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40



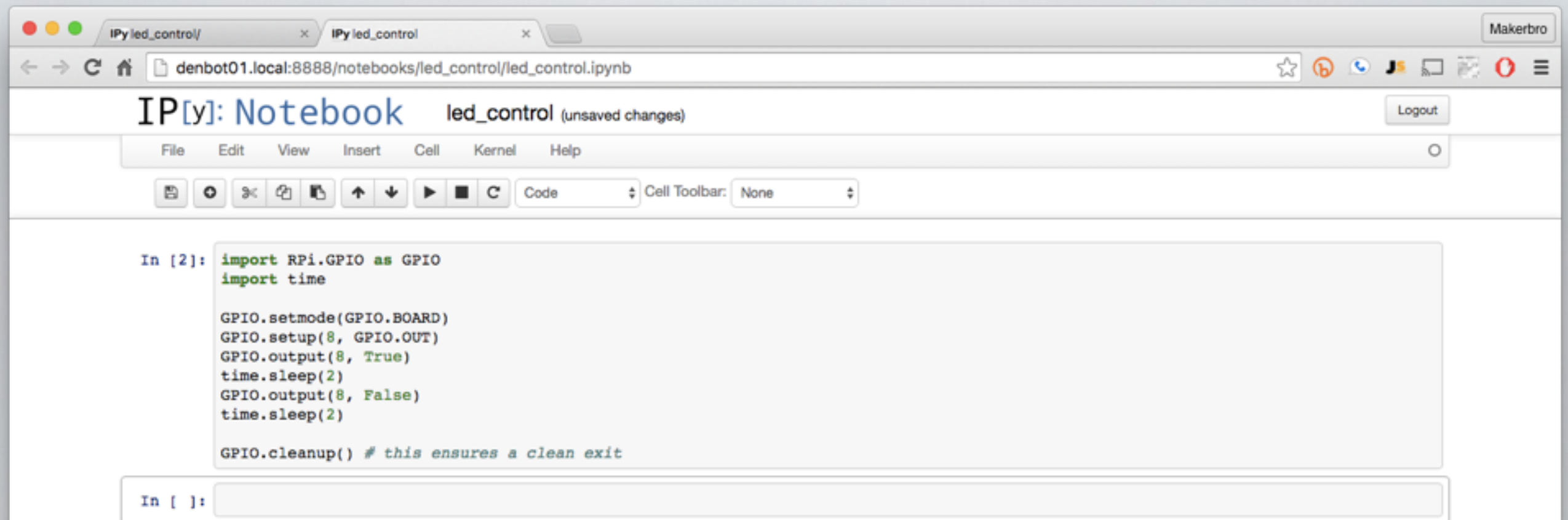
# Wiring Your First Circuit

- Let's jump right into the deep-end and wire our first circuit!



# Python LED Control Using RPi.GPIO

- With the necessary modules installed, let's use them to **blink** an LED!



The screenshot shows a web browser window with a Jupyter Notebook interface. The browser tabs are labeled 'IPy led\_control/'. The address bar shows 'denbot01.local:8888/notebooks/led\_control/led\_control.ipynb'. The notebook title is 'IP[y]: Notebook led\_control (unsaved changes)' with a 'Logout' button. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Help'. The toolbar contains icons for file operations and a 'Code' dropdown. The main area displays a Python code cell with the following code:

```
In [2]: import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setup(8, GPIO.OUT)
GPIO.output(8, True)
time.sleep(2)
GPIO.output(8, False)
time.sleep(2)

GPIO.cleanup() # this ensures a clean exit
```

Below the code cell is an empty input prompt 'In [ ]:'.

Create a new notebook “led\_control”  
inside the led\_control folder.