

IT-566

Computer Scripting

Techniques

```
start_date = self.get_monday(start_date)
date_span = end_date - start_date
week_color = start_color
day_color = start_color
market_color = start_color.value

for i in range(1, date_span.days):
    if start_date.weekday() == 0:
        week_color = day_color
    item = self.get_data(start_date, day_name=day_color.name, start_date=start_date, market_color=week_color, week_color_name=week_color.name, day_color_name=day_color.name, market_event_name=MarketWeekEvent(self.week_of_month(start_date), name, week_color_name=week_color.name, day_color_name=day_color.name))

    if self._insert == True:
        self._market_data_items.append(item)

    if self._debug == True:
        print(json.dumps(item))

    if day_color.value < 4:
        day_color = MarketColor(day_color.value + 1)
    else:
        day_color = MarketColor.RED
        start_date += timedelta(days=1)

def generate_historic_market_colors(self, start_date, end_date, start_color, debug=False, insert=False):
    items = self.generate_historic_dates_and_day_colors(start_date=start_date, end_date=end_date, start_color=start_color)
    market_data_items = self.set_historical_week_colors(items)
    if insert == True:
        self.insert_items(market_data_items)
    return market_data_items
```

92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118

```
--query "Exports [?contains(Name, 'text')]"
region ${_dep}
[?contains(Name, 'text')]
e: "${_dynamoc}"
"${_sns_top}"
ment_region} c
dler modules f
its and all su
ector.zip from sam
te.zip
```

Week 2

**Talking
Points**

Development Environment
Setup and Configuration
(Loose Ends)

Project Organization
& Layout

Bash Build Script

Git Development Workflow

Development Environment

Setup and Configuration

(Loose Ends)

Package Managers

tree Command

Configure ssh for GitHub

pipenv

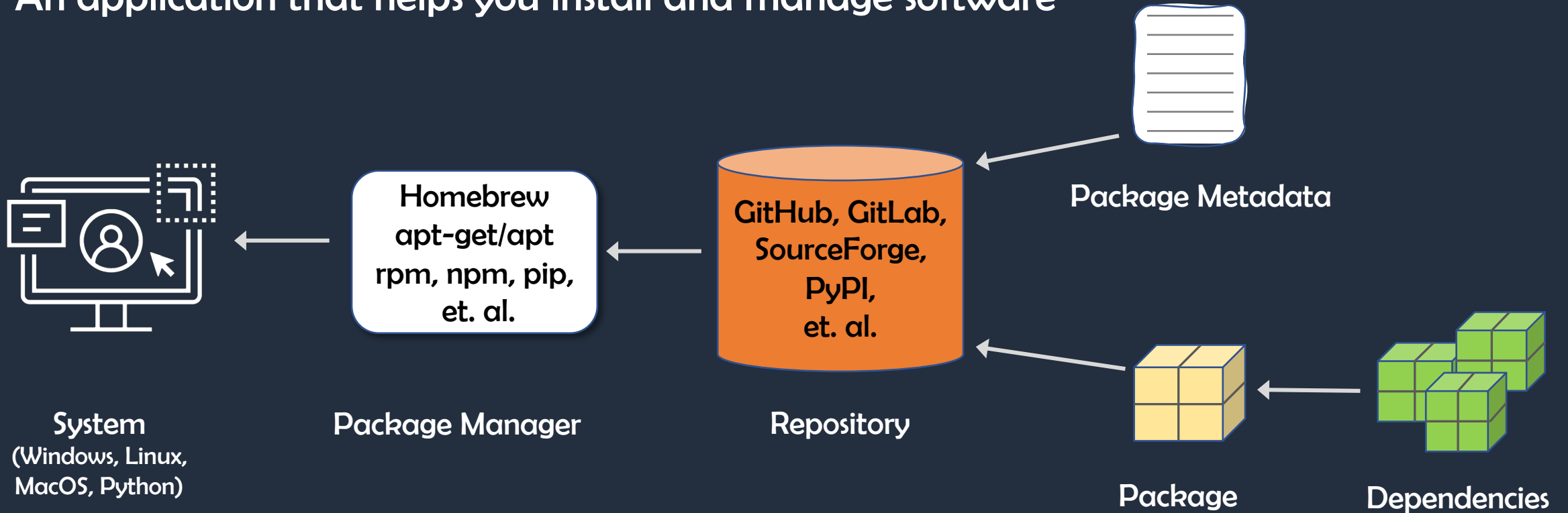
Setup, Usage & Why You
Need It

.bash_profile
Helpful Settings

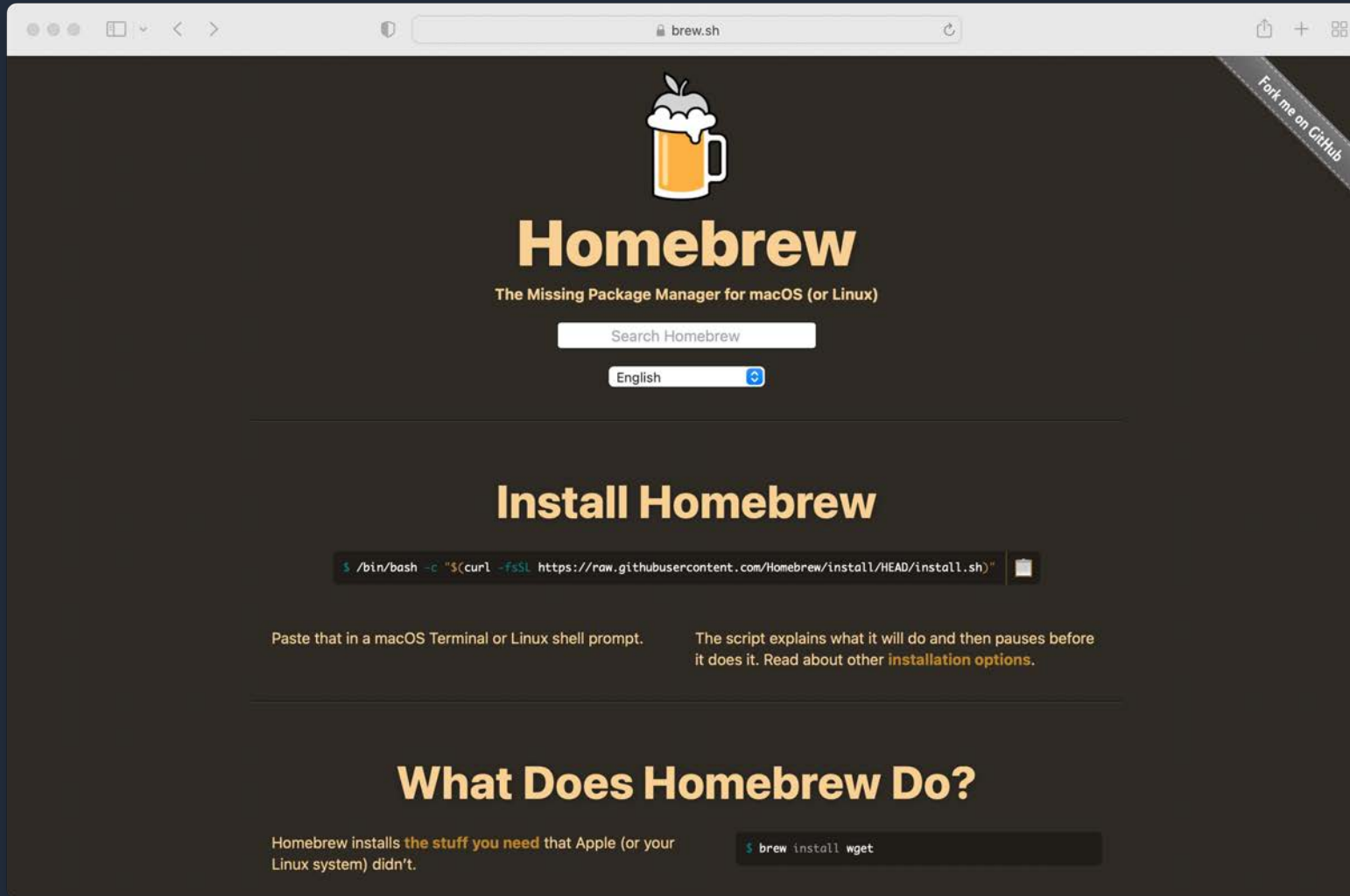
Package Managers

What Is A Package Manager?

- An application that helps you install and manage software



Homebrew MacOS (And Linux) Package Manager



The screenshot shows the Homebrew website homepage. At the top, there's a navigation bar with the URL 'brew.sh'. Below that is the Homebrew logo, a beer mug with a slice of apple on top. The main heading is 'Homebrew' in a large, bold, orange font, followed by the tagline 'The Missing Package Manager for macOS (or Linux)'. There's a search bar and a language selector set to 'English'. The main content area features a large heading 'Install Homebrew' and a code block containing the installation command: `curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh`. Below the code block, there's a note: 'Paste that in a macOS Terminal or Linux shell prompt. The script explains what it will do and then pauses before it does it. Read about other installation options.' Further down, there's a section titled 'What Does Homebrew Do?' with a sub-heading 'Homebrew installs the stuff you need that Apple (or your Linux system) didn't.' and a code block showing the command: `brew install wget`.

- Package Manager for MacOS
 - x86 & arm64
- Simplifies Software Installation
 - Especially Unix utils
 - Developer tools
- Linux, too
 - x86 (no arm64 binaries)

Linux Package Managers



Depends on Distribution

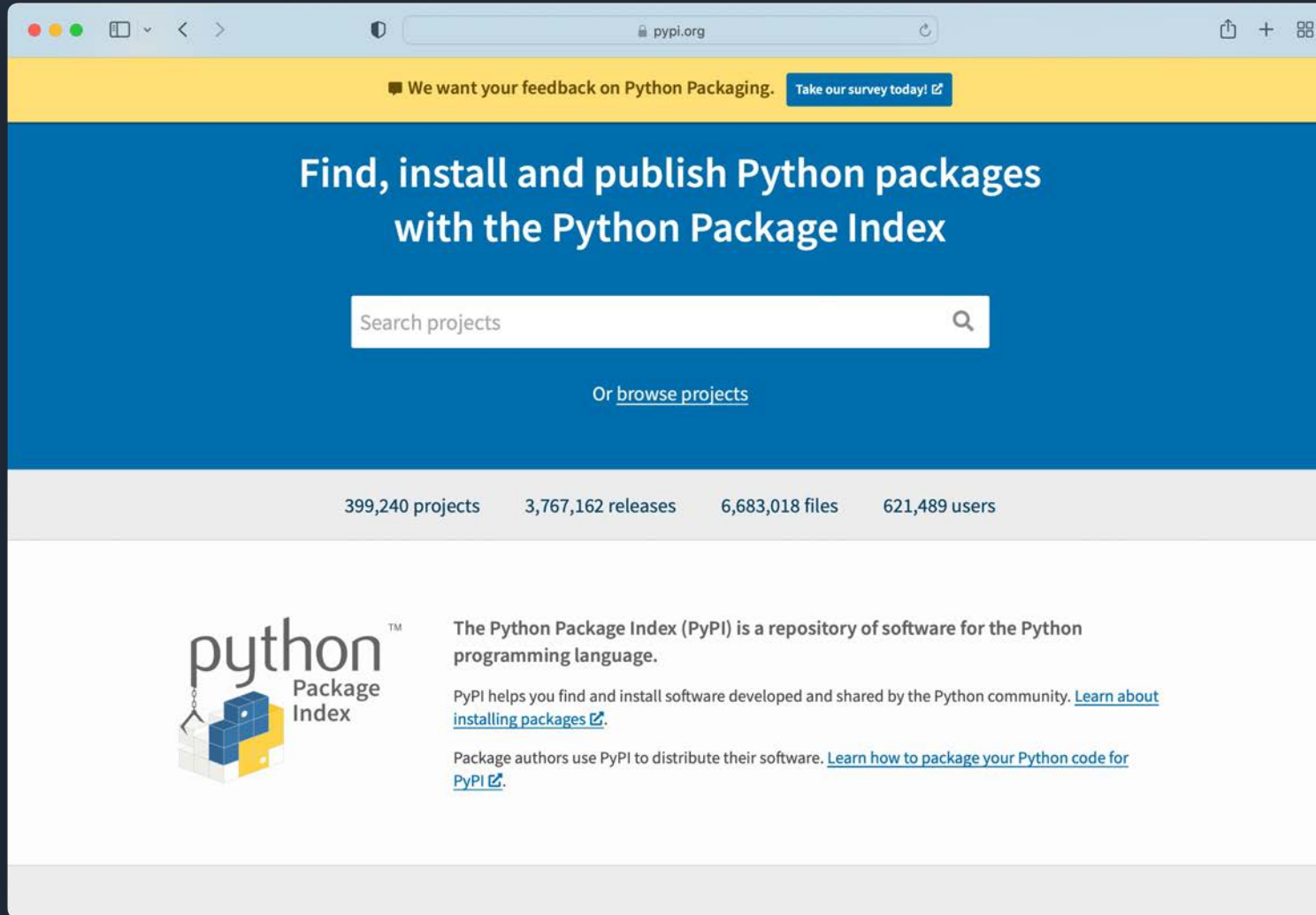


apt-get / apt



rpm

Python Package Manager — pip

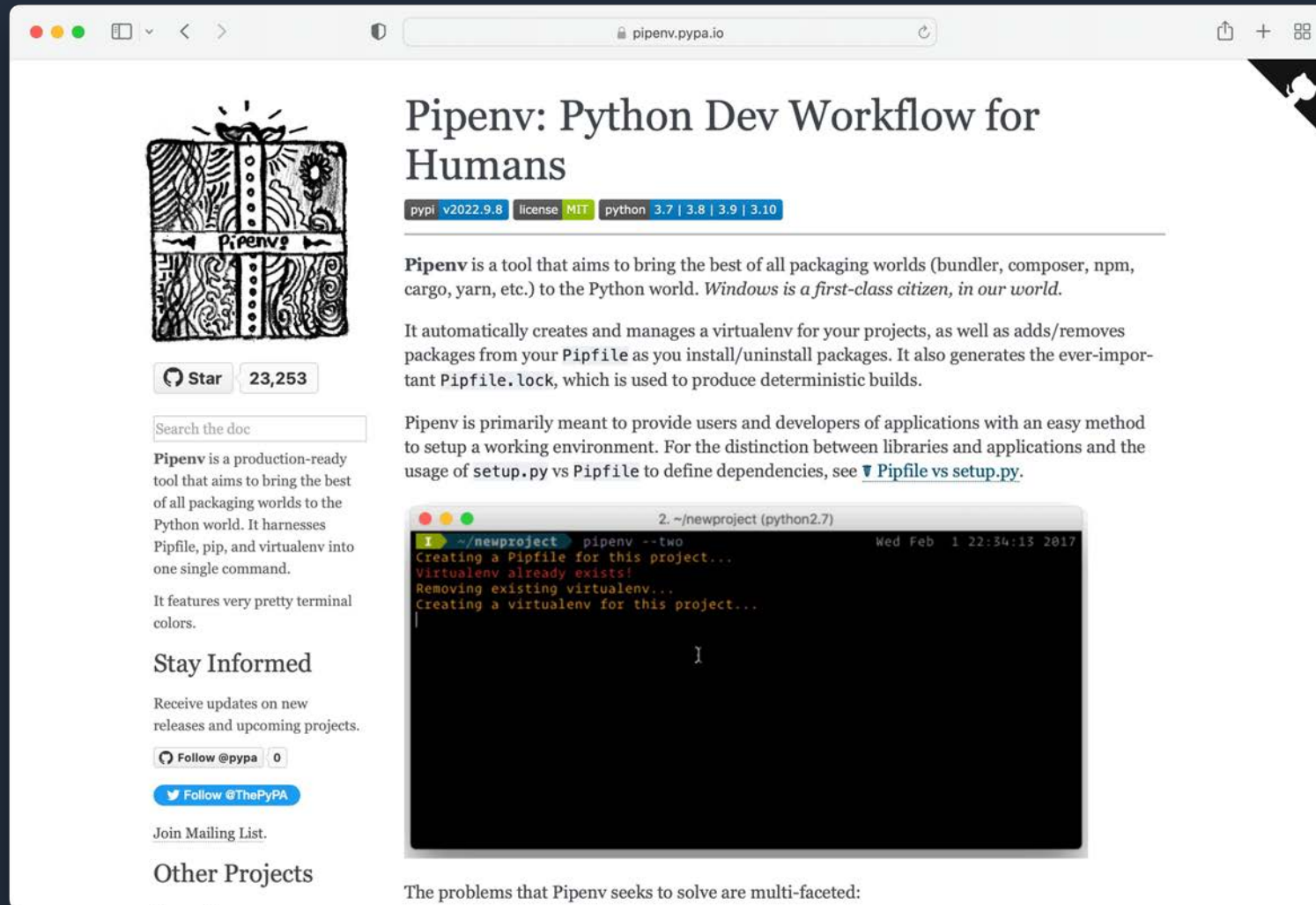


The screenshot shows the PyPI website homepage. At the top, there is a yellow banner with the text "We want your feedback on Python Packaging. Take our survey today!". Below this is a blue header with the text "Find, install and publish Python packages with the Python Package Index". A search bar with the placeholder "Search projects" and a magnifying glass icon is centered. Below the search bar is a link "Or [browse projects](#)". A statistics bar shows "399,240 projects", "3,767,162 releases", "6,683,018 files", and "621,489 users". At the bottom, there is a logo for the Python Package Index and a paragraph of text: "The Python Package Index (PyPI) is a repository of software for the Python programming language. PyPI helps you find and install software developed and shared by the Python community. [Learn about installing packages](#). Package authors use PyPI to distribute their software. [Learn how to package your Python code for PyPI](#)."

- pip or pip3
 - pip for Python 2
 - pip3 for Python 3
 - Generally...
- PyPI
 - Python Package Index

pipenv Setup, Usage, & Why You Need It

Pipenv — Why You Need It



The screenshot shows the Pipenv website homepage. At the top, there's a navigation bar with the URL 'pipenv.pypa.io'. The main heading is 'Pipenv: Python Dev Workflow for Humans'. Below the heading, there are badges for 'pypi v2022.9.8', 'license MIT', and 'python 3.7 | 3.8 | 3.9 | 3.10'. A decorative illustration of a gift box with 'Pipenv?' written on it is on the left. The main text describes Pipenv as a tool that brings the best of all packaging worlds (bundler, composer, npm, cargo, yarn, etc.) to the Python world. It mentions that Pipenv automatically creates and manages a virtualenv for projects and adds/removes packages from a Pipfile. A terminal screenshot shows the command 'pipenv --two' being executed, which creates a virtualenv for Python 2.7. The website also features a 'Star' button with 23,253 stars, a search bar, and social media links for GitHub, Twitter, and a mailing list.

Pipenv: Python Dev Workflow for Humans

pypi v2022.9.8 license MIT python 3.7 | 3.8 | 3.9 | 3.10

Pipenv is a tool that aims to bring the best of all packaging worlds (bundler, composer, npm, cargo, yarn, etc.) to the Python world. *Windows is a first-class citizen, in our world.*

It automatically creates and manages a virtualenv for your projects, as well as adds/removes packages from your Pipfile as you install/uninstall packages. It also generates the ever-important Pipfile.lock, which is used to produce deterministic builds.

Pipenv is primarily meant to provide users and developers of applications with an easy method to setup a working environment. For the distinction between libraries and applications and the usage of setup.py vs Pipfile to define dependencies, see [Pipfile vs setup.py](#).

```
~/newproject (python2.7)
└─$ pipenv --two
Creating a Pipfile for this project...
Virtualenv already exists!
Removing existing virtualenv...
Creating a virtualenv for this project...
```

The problems that Pipenv seeks to solve are multi-faceted:

- Create and Manage Python Virtual Environments
- Why?
 - You may have multiple python projects based on different Python versions
 - Helps manage and avoid python package dependency conflicts
 - Enables deterministic builds
- With very few exceptions, **avoid installing Python packages globally**
- Create a new venv with pipenv and install there

pipenv – Configuration and Usage

```
1 .bash_profile
# aliases
alias dir='ls -al'
alias cls='clear'
alias first-vpc='cd ~/dev/dev.aws/projects/first-vpc'
alias it590='cd ~/dev/dev.classes/it-590-aws'
alias it566='cd ~/dev/dev.classes/it-566-computer-scripting'
alias gasson='cd ~/dev/dev.gasson/marketpredictor'

test -e "${HOME}/.iterm2_shell_integration.bash" && source "${HOME}/.iterm2_shell_integration.bash"

# prompt
PS1="\n\[\033[35m\]\$(/bin/date)\n\[\033[32m\]\w \[\033[1;33m\]\$(__git_ps1
'(%s)')\n\[\$(iterm2_prompt_mark)\]\[\033[1;32m\][\!:\#\]\[\033[1;33m\] \u@h $"

test -e "${HOME}/.iterm2_shell_integration.bash" && source "${HOME}/.iterm2_shell_integration.bash"

# setup AWS cli environment
. ~/bin/aws/set-aws.sh
export PIPENV_VENV_IN_PROJECT="true"
export PATH
eval "$(/opt/homebrew/bin/brew shellenv)"
```

- export the environment variable: **PIPENV_VENV_IN_PROJECT="true"**
- Restart shell

Usage

- pipenv -- python 3
- or be more specific
 - pipenv --python 3.6
 - pipenv --python 3.10
 - etc....
- Install a package
 - pipenv install requests
- This will create a Pipfile
- ...and a .venv directory



**Oh Yeah?
Show me!**

tree Command

tree Command (MacOS)

```
~ /dev/dev.classes/it-590-aws/vpc (dev)
[613:116] swodog@RicksMacPro $ tree
.
├── README.md
├── build.sh
├── cloudformation
│   └── vpc.yaml
├── diagrams
│   └── vpc
│       ├── CustomVPC.png
│       ├── DefaultVPC.png
│       └── VPC.pdf
└── 3 directories, 6 files

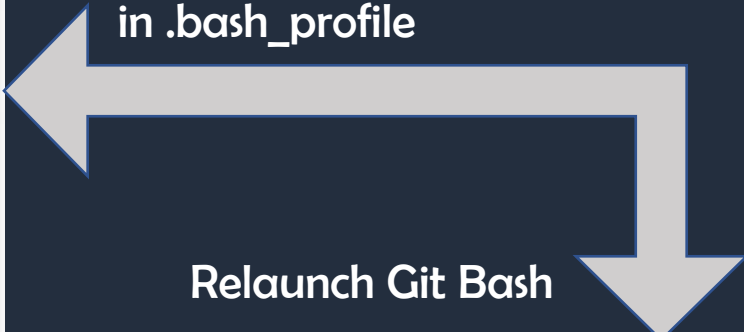
Sat Sep 10 10:41:35 EDT 2022
~/dev/dev.classes/it-590-aws/vpc (dev)
[614:117] swodog@RicksMacPro $
```

- Tree view of directory structure
- Install with brew

tree Command (Windows)

- Comes with Windows
 - Must configure to use in git bash

Add alias to tree command
in .bash_profile



Relaunch Git Bash

```
MINGW64:/c/Users/swodog
alias tree='cmd //c tree //a'
```

```
.bash_profile -- INSERT --
MINGW64:/c/Users/swodog/Projects/it-590-aws/vpc
swodog@RICKMILLERA0C8 MINGW64 ~/Projects/it-590-aws/vpc (main)
$ tree
Folder PATH listing
Volume serial number is 8445-6825
C:
+---cloudformation
 \---diagrams
      \---vpc

swodog@RICKMILLERA0C8 MINGW64 ~/Projects/it-590-aws/vpc (main)
$
```


tree Command (Show Hidden Files)

```
Mon Sep 12 11:36:05 EDT 2022
~/dev/dev.classes/it-566-computer-scripting/python/project_template (main)
[538:41] swodog@RicksMacPro $ tree -a -L 2
.
├── .gitignore
├── .venv
│   ├── .gitignore
│   ├── .project
│   ├── bin
│   ├── lib
│   ├── pyvenv.cfg
│   └── src
├── Pipfile
├── Pipfile.lock
├── README.md
├── build.sh
├── docs
│   └── sample_document.txt
├── src
│   └── example.py
└── tests
    ├── context.py
    └── test_example.py

7 directories, 12 files

Mon Sep 12 11:36:10 EDT 2022
~/dev/dev.classes/it-566-computer-scripting/python/project_template (main)
[539:42] swodog@RicksMacPro $
```

tree -a -L 2



**Really?
Show me!**

`.bash_profile` Helpful Settings

.bash_profile Helpful Settings

```
1 .bash_profile

# aliases
alias dir='ls -al'
alias cls='clear'
alias first-vpc='cd ~/dev/dev.aws/projects/first-vpc'
alias it590='cd ~/dev/dev.classes/it-590-aws'
alias it566='cd ~/dev/dev.classes/it-566-computer-scripting'
alias gasson='cd ~/dev/dev.gasson/marketpredictor'

test -e "${HOME}/.iterm2_shell_integration.bash" && source "${HOME}/.iterm2_shell_integration.bash"

# prompt
PS1="\n\[\033[35m\]\$(/bin/date)\n\[\033[32m\]\w \[\033[1;33m\]\$(__git_ps1
'(%s)')\n\[\033[32m\]\[\033[1;32m\][\!: \#\]\[\033[1;33m\] \u@\h \$ "

test -e "${HOME}/.iterm2_shell_integration.bash" && source "${HOME}/.iterm2_shell_integration.bash"

# setup AWS cli environment
. ~/bin/aws/set-aws.sh
[]
export PIPENV_VENV_IN_PROJECT="true"
export PATH
eval "$(/opt/homebrew/bin/brew shellenv)"

NORMAL .bash_profile 87% 21:1
```

- Command Aliases
- Environment Variables
- Fancy Prompt
- Links to other shell scripts
- Much, much, more...

Configure SSH for GitHub

Configure SSH for GitHub

The screenshot shows the GitHub Docs website. The browser address bar displays <https://docs.github.com/en/authentication/connecting-to-github-with-ssh>. The page title is "Authentication / Connect with SSH". The main heading is "Connecting to GitHub with SSH". Below the heading, the text reads: "You can connect to GitHub using the Secure Shell Protocol (SSH), which provides a secure channel over an unsecured network." A list of links is provided: [About SSH](#), [Checking for existing SSH keys](#), [Generating a new SSH key and adding it to the ssh-agent](#), [Adding a new SSH key to your GitHub account](#), [Testing your SSH connection](#), and [Working with SSH key passphrases](#). At the bottom, there are two sections: "Did this doc help you?" with thumbs up/down icons and a "Privacy policy" link, and "Help us make these docs great!" with a "Make a contribution" button and a link to "Or, learn how to contribute."

- Secure Repository Connection
- SSH Key Generation and Installation
 - Can be confusing
 - Let's walk through



GitHub SSH Key Generation and Configuration

Project Organization & Layout

Directory Structure & Project Artifacts

Python Modules & Packages

There's Bill, I'll ask him...



Hey Bill, me and Beth are having relationship issues. What do you think I should do?



Fix your Python project structure!



Project Organization

- Project Organization Refers to Several Issues
 - Directory Structure and Layout
 - Source Code Organization
 - Module and Package Organization
 - Application Architecture
- Today — Focus on Directory Structure and Project Artifacts
- Later Today — Focus on Application Architecture
 - ...and how to organize source code into modules and packages

Directory Structure & Project Artifacts

Directory Structure and Project Artifacts

```
Mon Sep 12 10:14:04 EDT 2022
~/dev/dev.classes/it-566-computer-scripting/python/project_template (main)
[526:29] swodog@RicksMacPro $ tree -a -L 1
.
├── .gitignore
├── .venv
├── Pipfile
├── Pipfile.lock
├── README.md
├── build.sh
├── docs
├── src
└── tests

4 directories, 5 files

Mon Sep 12 10:14:22 EDT 2022
~/dev/dev.classes/it-566-computer-scripting/python/project_template (main)
[527:30] swodog@RicksMacPro $
```

- `.gitignore` — List of files not to add to repo
- `.venv` — Virtual Environment created by pipenv
- `Pipfile` — List of required project packages
- `Pipfile.lock` — List of specific package versions
- `README.md` — Markdown document displayed by GitHub
- `build.sh` — Bash build file
- `docs` — Documentation directory
- `src` — Python source code directory
- `tests` — Unit and integration tests directory

Pipfile

```
Vim
~/d/d/i/p/p 0% 14 GB main + • build 9/12, 11:07 AM
1 Pipfile X
[[source]]
url = "https://pypi.org/simple"
verify_ssl = true
name = "pypi"

[packages]
pytest = "*"

[dev-packages]

[requires]
python_version = "3.10"

~
~
~
NORMAL main Pipfile 100% 14:1
"Pipfile" 14L, 154B written
```

- Lists repository source
- Installed packages
 - Production
 - Development
- I need to fix pytest
 - Should be in [dev-packages] section
- And required Python version

README.md

Markdown File

<https://www.markdownguide.org>

The Markdown Guide
how to use Markdown

Getting Started

An overview of Markdown, how it works, and what you can do with it.

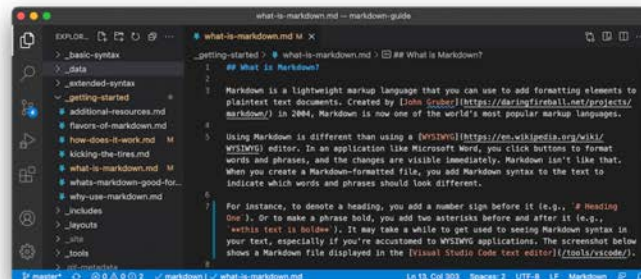
What is Markdown?

Markdown is a lightweight markup language that you can use to add formatting elements to plaintext text documents. Created by [John Gruber](#) in 2004, Markdown is now one of the world's most popular markup languages.

Using Markdown is different than using a [WYSIWYG](#) editor. In an application like Microsoft Word, you click buttons to format words and phrases, and the changes are visible immediately. Markdown isn't like that. When you create a Markdown-formatted file, you add Markdown syntax to the text to indicate which words and phrases should look different.

For example, to denote a heading, you add a number sign before it (e.g., `# Heading One`). Or to make a phrase bold, you add two asterisks before and after it (e.g., `**this text is bold**`). It may take a while to get used to seeing Markdown syntax in your text, especially if you're accustomed to WYSIWYG applications. The screenshot below shows a Markdown file displayed in the [Visual Studio Code](#) text editor.

- What is Markdown?
- Why Use Markdown?
- Kicking the Tires
- How Does it Work?
- What's Markdown Good For?
- Flavors of Markdown
- Additional Resources



```
1 # What is Markdown?
2
3 Markdown is a lightweight markup language that you can use to add formatting elements to
4 plaintext text documents. Created by John Gruber
5 in 2004, Markdown is now one of the world's most popular markup languages.
6
7 Using Markdown is different than using a WYSIWYG
8 editor, in an application like Microsoft Word, you click buttons to format
9 words and phrases, and the changes are visible immediately. Markdown isn't like that.
10 When you create a Markdown-formatted file, you add Markdown syntax to the text to
11 indicate which words and phrases should look different.
12
13 For instance, to denote a heading, you add a number sign before it (e.g., # Heading
14 One). Or to make a phrase bold, you add two asterisks before and after it (e.g.,
15 **this text is bold**). It may take a while to get used to seeing Markdown syntax in
16 your text, especially if you're accustomed to WYSIWYG applications. The screenshot below
17 shows a Markdown file displayed in the Visual Studio Code text editor.
```



Online flexible Web Developer courses. Try us for 3 weeks risk-free & transition to a well-paying job!

Directory Structure and Project Artifacts

```
~/dev/dev.classes/it-566-computer-scripting/python/project_template (main)
[527:30] swodog@RicksMacPro $ tree
.
├── Pipfile
├── Pipfile.lock
├── README.md
├── build.sh
├── docs
│   └── sample_document.txt
├── src
│   └── example.py
└── tests
    ├── context.py
    └── test_example.py


3 directories, 8 files

Mon Sep 12 10:50:31 EDT 2022
~/dev/dev.classes/it-566-computer-scripting/python/project_template (main)
[528:31] swodog@RicksMacPro $
```

Python source files go in src directory

Python unit and integration tests go in tests folder

context.py simplifies Python module and package location for unit tests



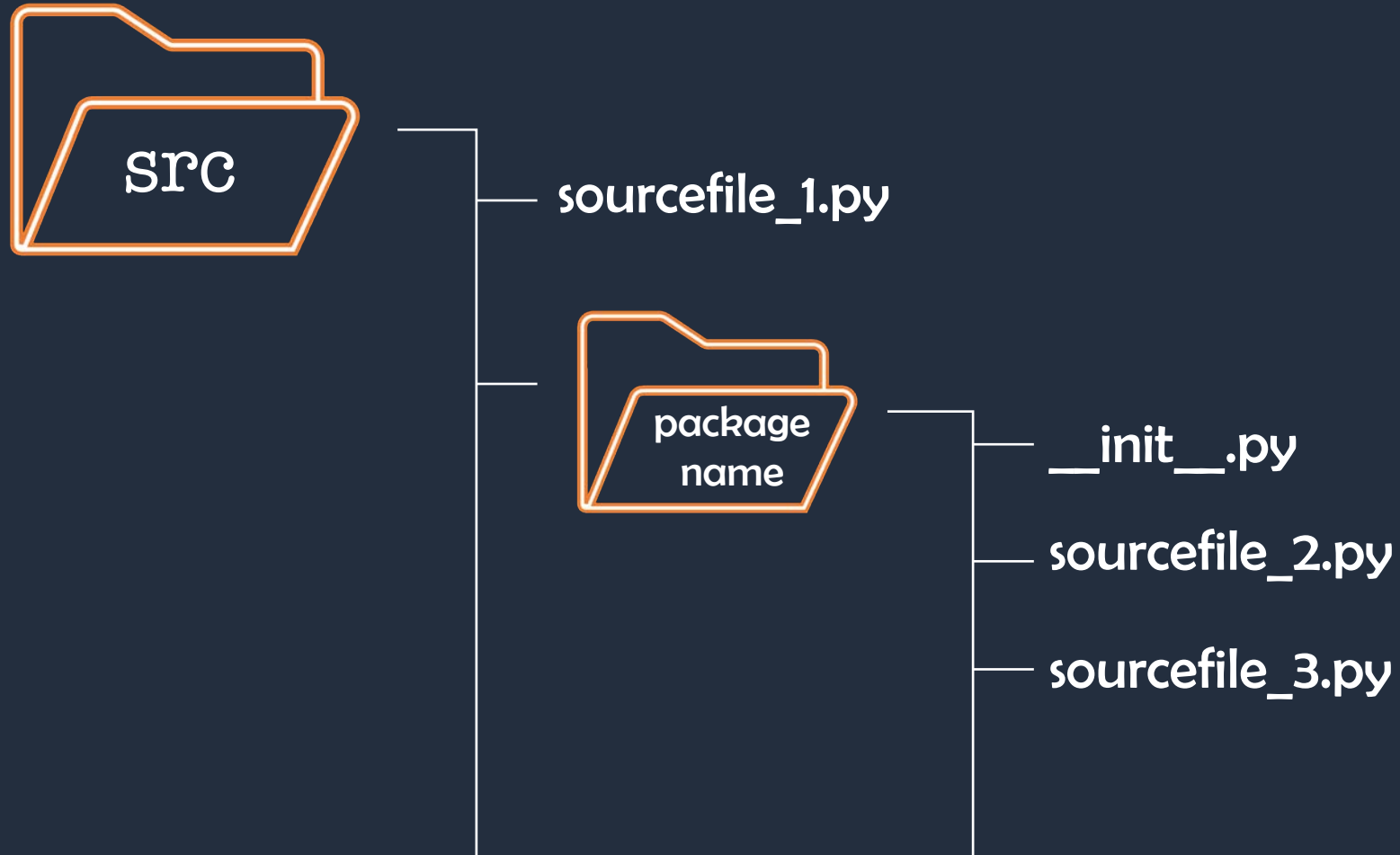
Maybe they won't notice...

Hey, not so fast.
What about the
Pipfile.lock?

Python Modules and Packages

Python Modules and Packages

<https://docs.python.org/3/reference/import.html#namespace-packages>



Bash Build Script

Bash Build Script (build.sh)

- Simplifies Project Management
- Consolidate Complex Commands
- Let's take a look...

Git Development Workflow

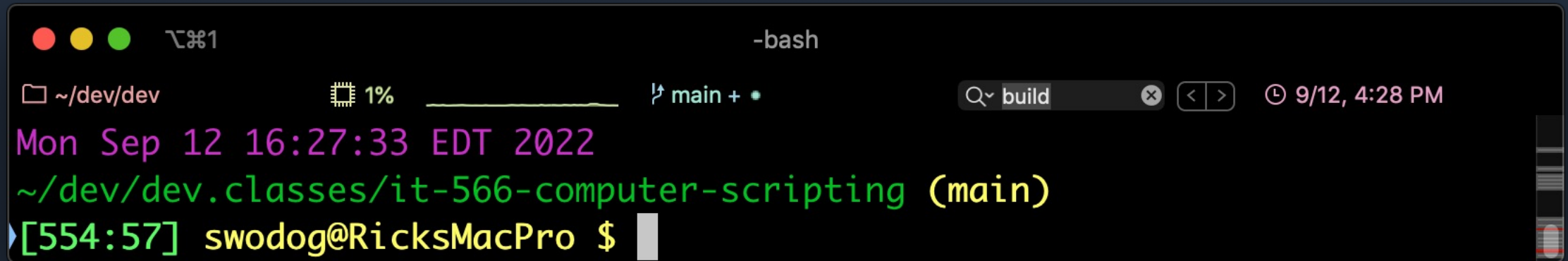
Git Development Workflow

- Initial Activities

- Create GitHub Repository
- Add README.md and .gitignore
- You will either start with a **main** or a **master** branch

- Clone The Repository Locally

- `git clone git@github.com:repository_name.git`
- Example: `git clone git@github.com:pulpfreepress/it-566-computer-scripting.git`



```
~ -bash
~/dev/dev 1% main + • build 9/12, 4:28 PM
Mon Sep 12 16:27:33 EDT 2022
~/dev/dev.classes/it-566-computer-scripting (main)
[554:57] swodog@RicksMacPro $
```

Git Development Workflow

```
Mon Sep 12 16:32:41 EDT 2022
~/dev/dev.classes/it-566-computer-scripting/python/project_template (main)
[562:65] swodog@RicksMacPro $ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

       modified:   ../../.DS_Store
       modified:   ../../Presentations/Week_2.pptx
       modified:   .gitignore
       modified:   Pipfile

Untracked files:
  (use "git add <file>..." to include in what will be committed)

       ../../Presentations/~$Week_2.pptx
       docs/


no changes added to commit (use "git add" and/or "git commit -a")

Mon Sep 12 16:32:48 EDT 2022
~/dev/dev.classes/it-566-computer-scripting/python/project_template (main)
[563:66] swodog@RicksMacPro $
```

- Add, Modify, Delete Files and Directories
- Check Status with `git status`

Followed by:

```
git add .
git commit -m "commit message"
git push
```

Would you like to see a demo?

Yes darling!