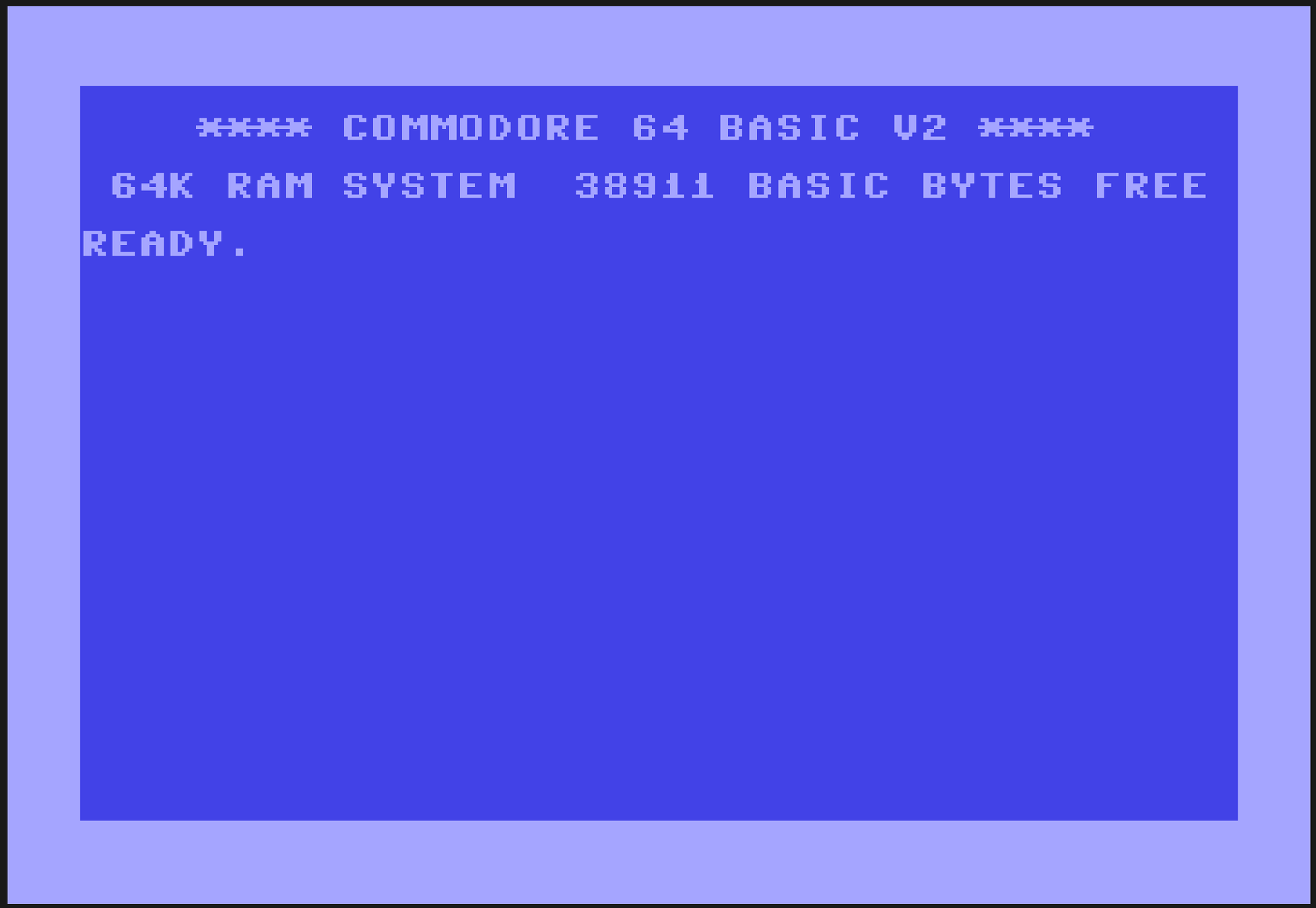
Python: the Multi-tool for System Management

olf22.bowmanjd.com

Learn, customize, tear down, build



olf22.bowmanjd.com

ε

olf22.bowmanjd.com

Hindered by tasks that are

- taking too much time
- hard to remember
- prone to human error

Different tools for system automation

- Bash (or ash/dash, zsh, fish, Elvish, xonsh, Nushell...)
- Python
- Ansible (or Salt, Chef, Puppet...)

Use Python when:

- Bash scripts are too large, complicated
- Processing complex string, array, or data
- Target machine lacks required commands
- Ansible is too heavy, slow, simple, or unavailable
- Cross-platform is desirable
- Already know or want to learn Python

An informative debate about the merits of Bash vs. Python on Stackoverflow

olf22.bowmanjd.com

Why Python (and not other general-purpose languages)?

It isn't a competition; collect them all!

"The Python space is beautiful, and great, and big"

Lorena Mesa, Engineer at GitHub, former chair of Python Software Foundation

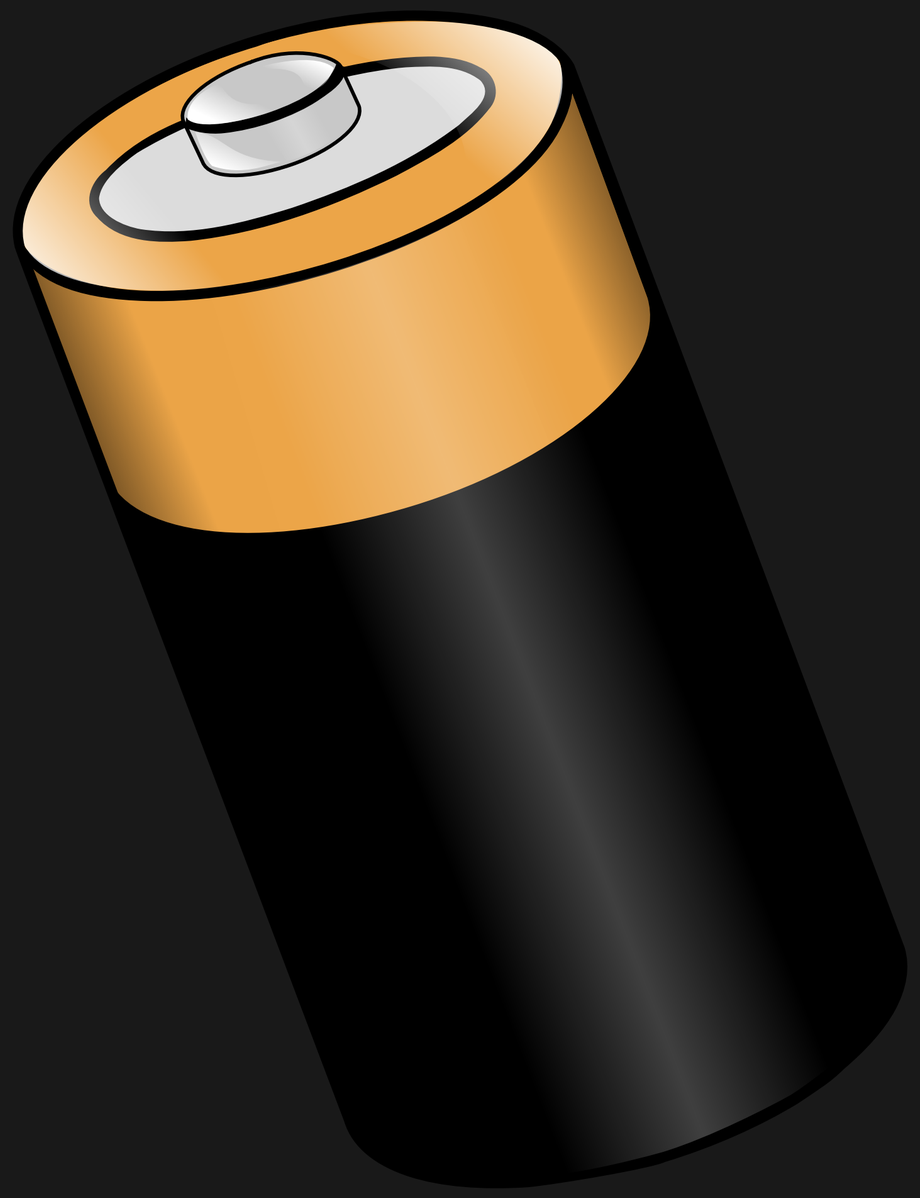
olf22.bowmanjd.com

"I came for the language, I stay for the community."

Brett Cannon, Dev lead on the Python extension for Visual Studio Code

olf22.bowmanjd.com

"batteries included"

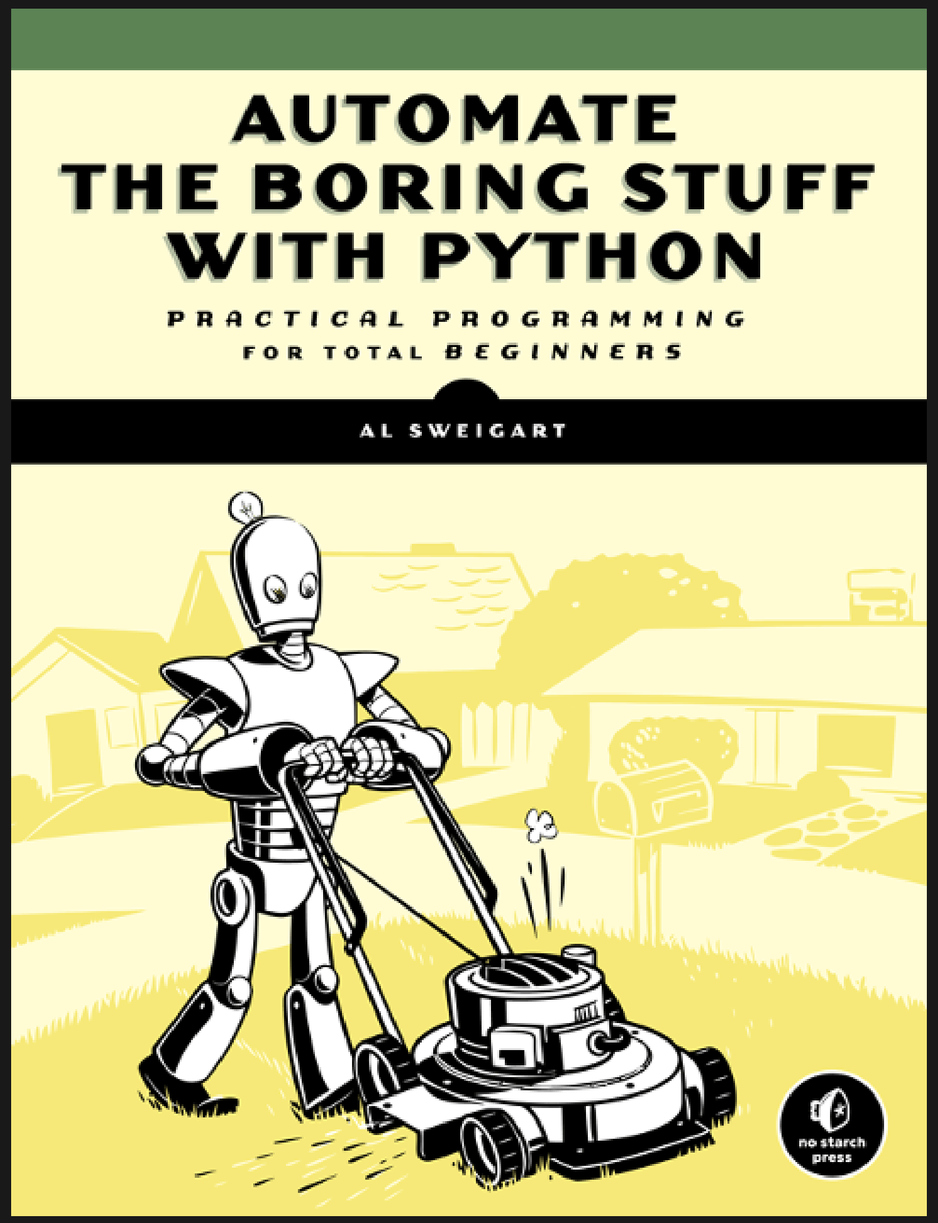


olf22.bowmanjd.com

Getting started with Python

Lots of good documentation. Here are a few options...

olf22.bowmanjd.com



olf22.bowmanjd.com

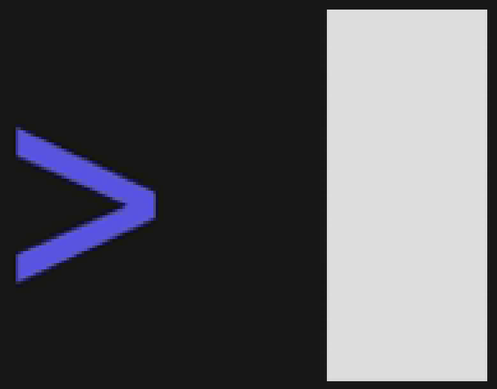
olf22.bowmanjd.com

olf22.bowmanjd.com

Awesome Python

List

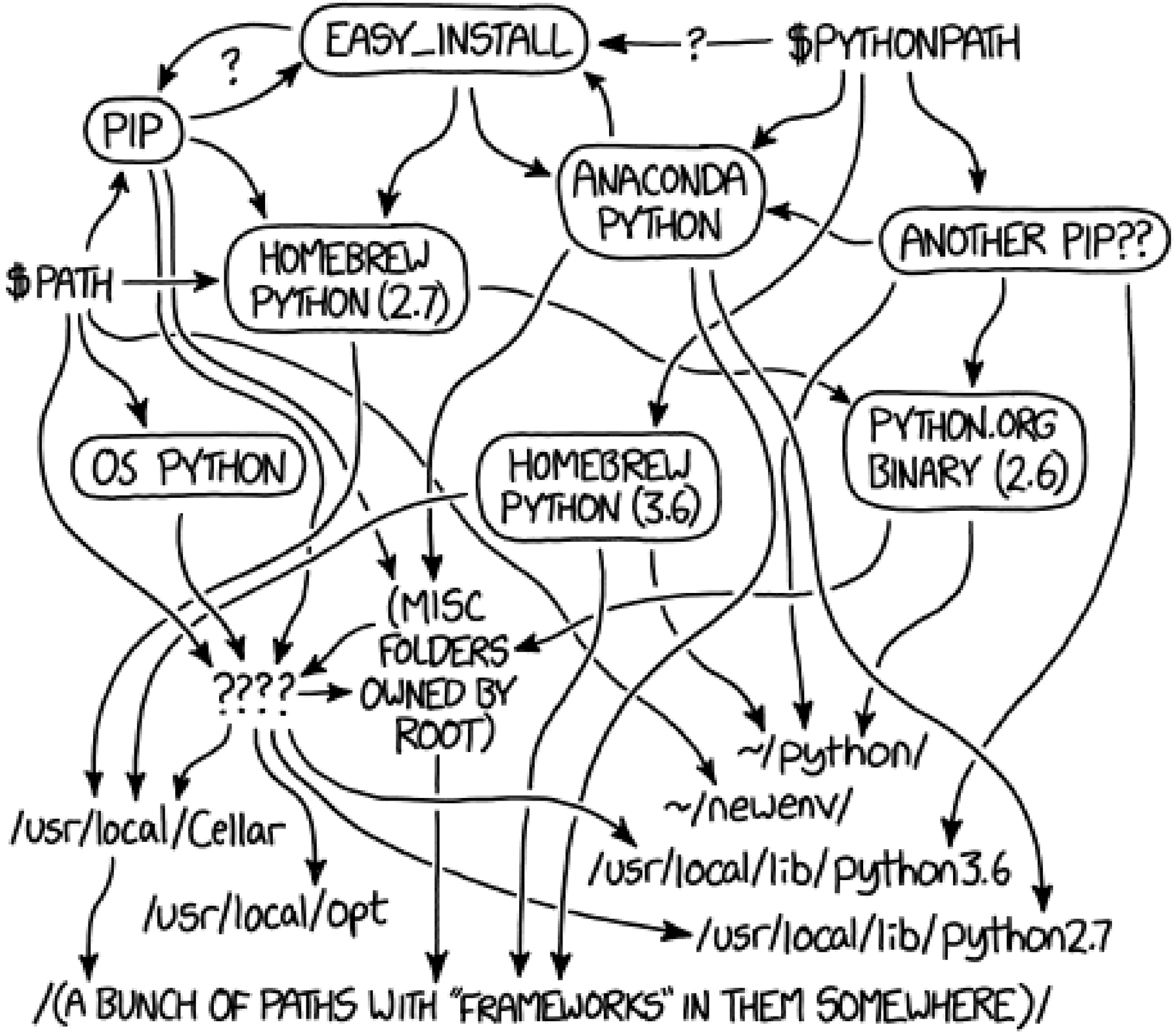
olf22.bowmanjd.com



olf22.bowmanjd.com

Installing Python

olf22.bowmanjd.com



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

- python.org/downloads/
- Linux:
  - Debian/Ubuntu: `apt install python3`
  - Fedora: `dnf install python3`
- Docker: `docker run -it python:alpine`
  - Phones: Termux (Android) or Pythonista (iOS)
  - Mac: see downloads or `brew install python`
  - Windows: see downloads or Store or scoop/chocolatey/winget

Making `python` launch `python3`?

- python-is-python3 (Ubuntu/Debian)
- pythonispython3 (Alpine)
- python-unversioned-command (Fedora)
- `alias python=python3`

Python REPL on the web:

- pythonanywhere.com
- python.org/shell
- github.com/codespaces
- repl.it/languages/python3
- paiza.cloud
- c9.io

Installing packages

```
> pip install requests
> pip install plumbum httpx sqlalchemy
```

Browse at pypi.org

Virtual environments

```
> python3 -m venv virtual_directory_name
> python3 -m venv .venv
> . .venv/bin/activate
```

Python editors

- VSCode
- PyCharm
- Vim/Neovim, Emacs, Sublime Text, Helix, Zed...
- For beginners: Thonny, Mu

Lightning intro to writing Python

github.com/bowmanjd/olf22python

olf22.bowmanjd.com

Functions and their arguments

olf22.bowmanjd.com

```python
def greet(greeting):
    return greeting + ", World!"

greet("Hello")
```

olf22.bowmanjd.com

Default arguments

```
1 def greet(greeting="Hello"):
2     return greeting + ", World!"
3
4
5 greet()
```

Named arguments and f-strings

```
1 def greet(greeting="Hello", audience="World"):
2     return f"{greeting}, {audience}!"
3
4
5 greet()
6 greet("Salutations", "Galaxy")
7 greet(audience="Galaxy")
```

Some types

```
a_number = 12
another_number = 7.1
a_string = "Some text"
another_string = "Some more text"
a_range = range(10)
some_bytes = b"\x00\x01\x02\x03Hi"
a_list = ["Some text", 14, another_number, "你好", 1]
a_lonely_number = a_list[4]
a_boolean = True
```

The dict

```
1 a_dict = {"a_key": "a_value",
2           "first_name": "Sheila",
3           "pi": 3.14159}
4
5 archimedes_constant = a_dict["pi"]
```

Comparisons and conditions

```
def compare(a, b):
    if a == b:
        print("equality")
    if not a == b:
        print("inequality")
    if a != b:
        print("also inequality")
    if a > b:
        print("greater than")
    else:
        print("less than or equal")
    if a <= b:
        print("also less than or equal")
```

Loops

```
1  import os
2
3  for variable in os.environ.keys():
4      print(variable)
5
6  for path in os.getenv("PATH").split(":"):
7      print(path)
8
9  for num in range(10):
10     print(num)
11
12 for _ in range(10):
13     print("Doing this thing 10 times.")
```

Shell scripting in the standard library

- `subprocess` for command execution
- `pathlib` for filesystem reading/manipulation
- `shutil` for copying/deleting directories
- `shlex` for parsing arguments
- `re` for regular expressions
- `fnmatch` for shell-like file matching
- `argparse` for parsing command-line arguments
- `os`
- `sys`

Data wrangling in the standard library

- `csv`
- `json`
- `xml.etree.ElementTree` (consider defusedxml)
- `sqlite`
- `urllib.request` for HTTP calls

The heart of shell scripting in Python: command execution with `subprocess`

olf22.bowmanjd.com

`subprocess.run()` with text output capture

```
import subprocess

result = subprocess.run(["ip", "addr"],
                        capture_output=True,
                        text=True)

if result.returncode == 0 and result.stdout:
    print(result.stdout)
```

The same, but with shell mode (be careful)

```
import subprocess

result = subprocess.run("ip addr",
                        capture_output=True,
                        shell=True,
                        text=True)

if result.returncode == 0 and result.stdout:
  print(result.stdout)
```

A shortcut with `subprocess.check_output`

```
import subprocess

output = subprocess.check_output(["ip", "addr"],
                                 text=True)

print(output)
```

A shortcut with `subprocess.check_call` if output capture is not needed

```
import subprocess

subprocess.check_call(["ip","addr"])
```

- `subprocess.run` for anything and everything
- `subprocess.check_output` for convenience, capturing output
- `subprocess.check_call` if execution is all that is needed
- `shell=True` parameter will pass the whole command as string to `sh`, and expand variables (security alert); otherwise pass the command as a `["list", "of", "command", "and", "parameters"]`

SSH

with Paramiko

olf22.bowmanjd.com

```python
import paramiko

client = paramiko.SSHClient()
client.load_system_host_keys()
client.connect("server", username="user")

stdin, stdout, stderr = client.exec_command('ls /')
root_directories = stdout.read()

sftp = client.open_sftp()
with sftp.file("/etc/os-release") as f:
  for line in f:
    print(line)
```

Have you tried plumbum?

```
from plumbum.cmd import ip
ip('addr')
```

And *so* much more...

olf22.bowmanjd.com

Such as SSH:

```
from plumbum import SshMachine

server = SshMachine("server", user="admin")
server["uname"](['-a'])
```

Other SSH options

- AsyncSSH
- fabric
- `ssh server python3 < script.py`

File reading, writing, and manipulation with

`pathlib`

```python
from pathlib import Path

filepath = Path("/etc/os-release")
os_info = filepath.read_text()
if "Fedora" in os_info:
    print("I still remember Redhat 5.1")
elif "Alpine" in os_info:
  print("Compiled with love and musl")
elif "archlinux" in os_info:
  print("A wiki and the AUR; it "
        "doesn't get better than this!")
```

olf22.bowmanjd.com

Reading line by line

```
1 from pathlib import Path
2
3 filepath = Path("/etc/os-release")
4 with filepath.open() as f:
5   for line in f:
6     if line.startswith("PRETTY_NAME"):
7       print(line.strip())
```

```
import pathlib
import shlex

os_release_file = pathlib.Path("/etc/os-release")
os_release = os_release_file.read_text()
lexer = shlex.shlex(os_release, posix=True)
lexer.whitespace_split = True
os_info = dict(i.split('=') for i in lexer if "=" in i)
print(os_info["PRETTY_NAME"])
```

olf22.bowmanjd.com

Writing text to a file

```
from pathlib import Path

filepath = Path("/etc/motd")
weather = "There will be temps today with a chance of weather
filepath.write_text(weather)
```

Processing command-line arguments

olf22.bowmanjd.com

Easy but inflexible with `sys.argv`

Save the following to something like `motder.py`

```python
import sys
from pathlib import Path

def motder(text):
  filepath = Path("/etc/motd")
  filepath.write_text(text)

if __name__ == "__main__":
    motder(sys.argv[1])
```

Execute with:

```
> sudo python3 motder.py "Good morning it is Friday!"
```

Add a shebang and make it executable

```python
#!/usr/bin/env python3
import sys
from pathlib import Path

def motder(text):
    filepath = Path("/etc/motd")
    filepath.write_text(text)

if __name__ == "__main__":
    motder(sys.argv[1])
```

```
> chmod a+x motder.py
> sudo ./motder.py "Good morning it is a lazy Friday!"
```

A more flexible way: `argparse`

```
#!/usr/bin/env python3
import argparse
from pathlib import Path

def motder(text):
  filepath = Path("/etc/motd")
  filepath.write_text(text)

if __name__ == "__main__":
  parser = argparse.ArgumentParser()
  parser.add_argument("message", help="The message of the day
  args = parser.parse_args()
    motder(args.message)
```

Some excellent third-party CLI libraries

- `click`
- `typer`
- `fire`
- `plumbum`

Create truly glamorous TUIs

- `rich`
- `textual`

Temporary files

```
import pathlib
import shutil
import subprocess
import tempfile

tempdir = pathlib.Path(tempfile.mkdtemp())
temp_motd = tempdir / "motd"
temp_motd.write_text("Welcome, user!")
motd = pathlib.Path("/etc/motd")
subprocess.check_call(["sudo", "cp", temp_motd, motd])
shutil.rmtree(tempdir, ignore_errors=True)
```

Data

olf22.bowmanjd.com

Writing JSON

```python
import json

data = {"name": "OLF conference", "age": 20}
json_data = json.dumps(data, indent=4)
print(json_data)
```

Reading JSON

```
1 import json
2 from pathlib import Path
3
4 filepath = Path("/etc/docker/daemon.conf")
5 filetext = filepath.read_text()
6 dockerd_conf = json.loads(filetext)
```

Reading CSV

sample.csv

```
Name,Age
Michael Palin,79
John Cleese,83
OLF Conference,20
```

```python
import csv
from pathlib import Path

inpath = Path("sample.csv")

with inpath.open(newline="", encoding="utf-8") as f:
    reader = csv.reader(f)
    next(reader) # Skip the header
    for row in reader:
        name = row[0]
        age = row[1]
        print(f"{name} is {age} years old.")
```

```python
import csv
from pathlib import Path

inpath = Path("sample.csv")

with inpath.open(newline="", encoding="utf-8") as f:
    reader = csv.DictReader(f)
    for row in reader:
        name = row["Name"]
        age = row["Age"]
        print(f"{name} is {age} years old.")
```

Writing CSV

```python
import csv
from pathlib import Path

out = Path("output.csv")

with out.open("w", newline="", encoding="utf-8-sig") as f:
    writer = csv.writer(f)
    writer.writerow(["Name", "Age"])
    writer.writerow(["John Cleese", 83])
```

Web requests

olf22.bowmanjd.com

GET some data

```
from urllib.request import urlopen

# Avoid unsanitized user inputs, because:
# url = "file:///etc/passwd"
url = "https://wttr.in/Columbus,OH?A1nF"
with urlopen(url) as response:
  print(response.read().decode())
```

POST some JSON

```python
import json
from urllib.request import Request, urlopen

url = "https://jsonplaceholder.typicode.com/posts"
data = {
    "userid": "1001",
    "title": "POSTing JSON for Fun and Profit",
    "body": "JSON! Don't forget the content type.",
}
postdata = json.dumps(data).encode()
headers = {"Content-Type": "application/json"}
httprequest = Request(url, method="POST",
                      data=postdata, headers=headers)

with urlopen(httprequest) as response:
```

Third-party libraries for Web

- `requests`
- `httpx`
- `BeautifulSoup`
- `scrapy`

Other data avenues to explore

- built-in `sqlite`
- `xml.etree.ElementTree` or `defusedxml` or `lxml`
- `SQLAlchemy`
- `pandas`
- `numpy`

For network engineers

- Netmiko: switch management
- Napalm: network automation
- Nornir: automate everything
- Free Python Network Automation Course from Twin Bridges Technology

For virtualization

- proxmoxer
- libvirt
- AWS SDK for Python
- Azure SDKs for Python
- Google Cloud Client Libraries
- vSphere Automation SDK
- VMware ESXi/vSphere API Python Bindings
- For Hyper-V: pywinrm WinRM client
- For Hyper-V: pypsrp PowerShell Remoting

For container management

- docker
- podman (if using socket)
- kubernetes

Thank you

olf22.bowmanjd.com