
BY COMMUNITY • FOR COMMUNITY •
#GLOBALAZURE
2024
MELBOURNE EDITION

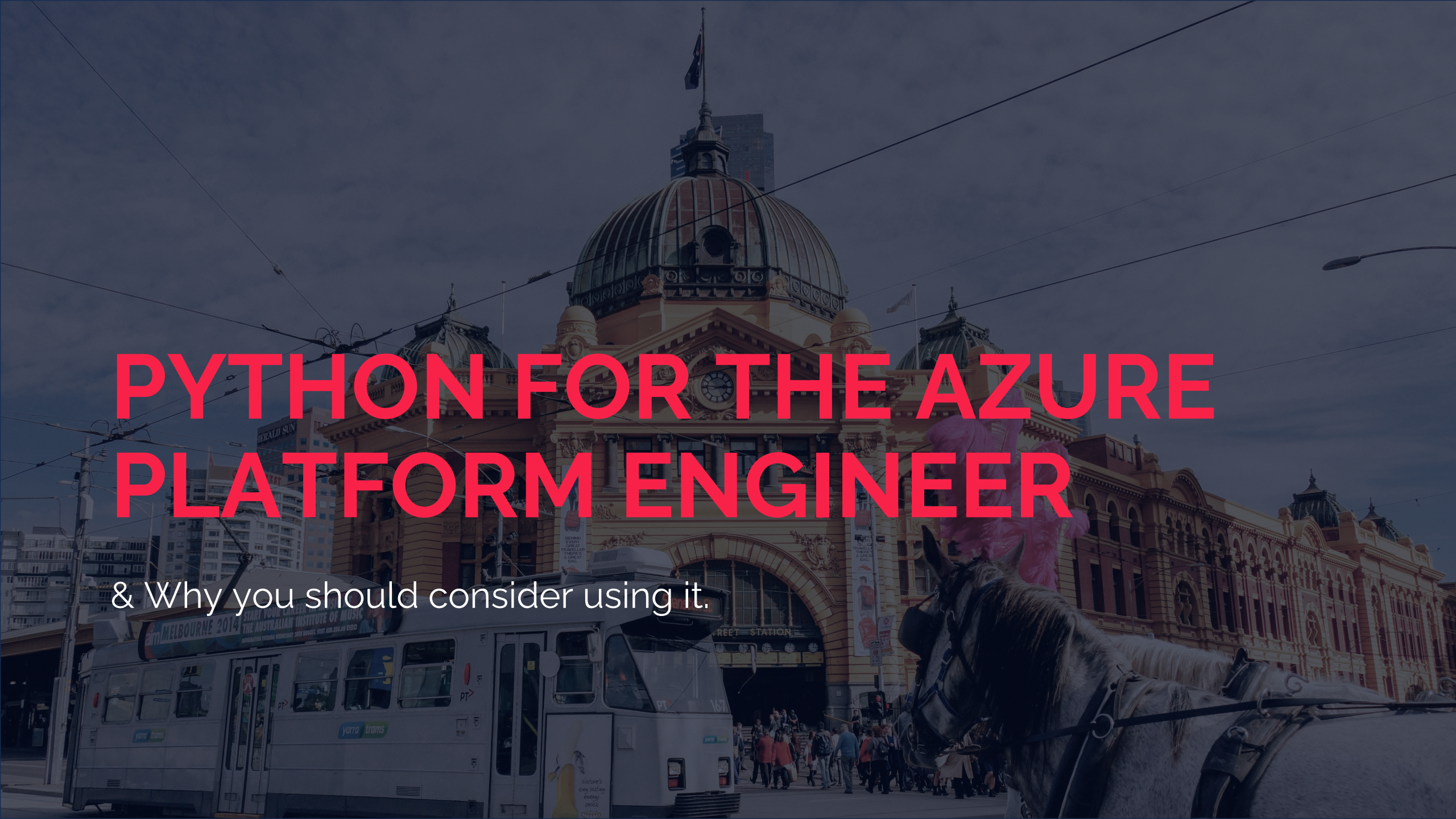
2024 SPONSORS

ARINCO

Mantel group

vNEXT

#GLOBALAZURE 2024



PYTHON FOR THE AZURE PLATFORM ENGINEER

& Why you should consider using it.

WHO AM I?

NAME Aaron Saikovski

WORK Principal Cloud Engineer at ING Australia
Former Microsoft CSA-E & Azure Tech Trainer

SOCIALS @melbourneazurebootcamp
@ruskyduck72
https://github.com/aaronsaikovski
https://dev.to/aaronsaikovski/

PASSIONS Coffee
Code – C#, Golang and Python
Retro Gaming
Music….the heavy kind
Azure (Of course)

AGENDA

- Python – a very brief history
- Why/Why not Python?
- Basic code samples
- Getting started with Python
- Tooling and tips
- Introducing the AzureSDK for Python
- Demos
- Q & A

PYTHON….A BRIEF HISTORY

- Released 20 Feb 2021 – 33 years ago
- Created by Guido Van Rossum (Now at Microsoft)
- Named after Monty Python and not the reptile
- Open sourced and maintained by many people globally
- Multiplatform and runs everywhere – inc. RaspberryPI
- Designed to be easy to learn, read and write
- Allows for shorter development time
- Used everywhere even at NASA!

#GLOBALAZURE
BY COMMUNITY • FOR COMMUNITY •
2024

WHY PYTHON?

- Very easy to learn
- Batteries included
- Very active community support
- #1 Programming language in the world right now
- Very rich and mature package ecosystem
- It's the default Data science and ML language
- Call native 'C' code (Cython) and even Rust (PyO3)
- Taught in primary schools, bootcamps etc
- Great linters and code editing tools

WHY NOT PYTHON?

- Large projects can become unwieldy
- Not high performance – interpreted (For loops etc)
- GIL (Global Interpreter Lock) – single threaded Mutex
- Package management can be very painful
- Not strongly or statically typed
- Virtual environments can be confusing
- Other options such as PowerShell etc…
- Might be overkill for simple tasks
- Not an energy efficient language

	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

PYTHON CODE SAMPLES

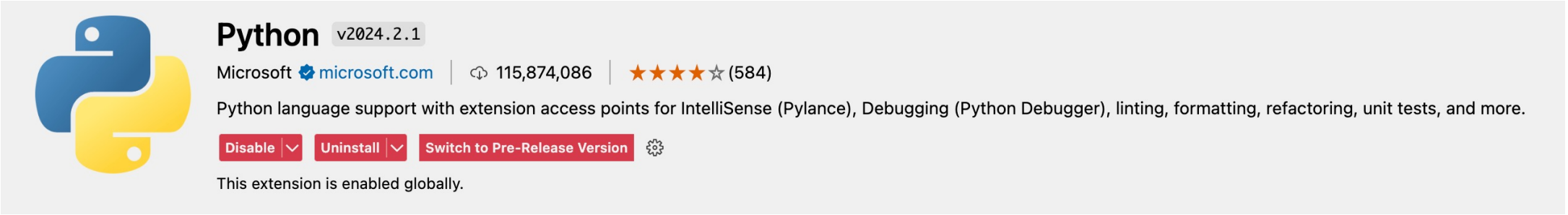
```
def main():
    print("Hello, World!")

if __name__ == "__main__":
    main()
```

```
def main():
    for _ in range(5):
        print("Hello, World!")

if __name__ == "__main__":
    main()
```

```
# Simple pygame program

# Import and initialize the pygame library
import pygame
pygame.init()

# Set up the drawing window
screen = pygame.display.set_mode([500, 500])

# Run until the user asks to quit
running = True
while running:

    # Did the user click the window close button?
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Fill the background with white
    screen.fill((255, 255, 255))

    # Draw a solid blue circle in the center
    pygame.draw.circle(screen, (0, 0, 255), (250, 250), 75)

    # Flip the display
    pygame.display.flip()

# Done! Time to quit.
pygame.quit()
```

GETTING STARTED WITH PYTHON..THE BASICS

- Install Python - https://www.python.org/
- Install VSCode
- VSCode Python extension from Microsoft



- Get coding!

PYTHON CODE BASIC MAIN STRUCTURE

```
def main():
    print("Hello World!")


if __name__ == "__main__":
    main()
```

- Note the identing – VS code tools help with this
- `main()` – denotes the 'main' entry point to the program
- `__name__` - name of the current module
- `__main__` - name of the main module
- `if __name__ == "__main__":` - checks that the current module is in fact being run as the main entry point of the program

TOOLS/TIPS AND TRICKS

- Recommend Python3.12.3 + (latest)
- Makefiles (*yes they are still around*) but look at Taskfile
- Ruff and Black for formatting/linting
- Annotate your methods with @decorators
- Use type hints - these improve readability
- Use doc hints for comments in methods and classes
- Keep project structure simple and flat
- Use Poetry instead of PIP for virtual environments
- Follow PEP8 style guides
- Use PyTest!!

A QUICK WORD ON PYTHON VIRTUAL ENVIRONMENTS

- Independent isolated environment
- Tracks packages and Python versions
- Isolated folder structure for your projects
- Can use PIP
  - **`python -m venv venv`**
- Or Poetry
  - **`poetry new poetry-project`**

- ***Remember to Activate your virtual environment before use!!***
  - **`source env/bin/activate`**

AZURE SDK FOR PYTHON

- Allows for communication to Azure services
- 180+ Python libraries that map to Azure services
- Authenticate to Azure to run code locally
- Azure SDK uses PIP or Conda – *recommend Poetry*
- Libraries - Management plane and data plane client libraries
- Azure CLI itself is written in Python (*did you know that?)*
- All Azure SDK libraries -> Azure REST APIs.
- Updated and deprecated Azure SDK libraries

source: https://learn.microsoft.com/en-us/azure/developer/python/sdk/azure-sdk-overview

GETTING STARTED WITH THE AZURE PYTHON SDK

- Install the Azure management API packages

```
pip install azure-mgmt-resource
pip install azure-identity
```

Or

```
poetry add azure-mgmt-resource
poetry add azure-identity
```

GETTING STARTED WITH THE AZURE PYTHON SDK

- Create a new resource group in the Azure Australia East region

```
from azure.identity import DefaultAzureCredential
from azure.mgmt.resource import ResourceManagementClient
import os

LOCATION = ‘australiaeast’
GROUP_NAME ='sample_resource_group'

sub_id = os.getenv("AZURE_SUBSCRIPTION_ID")
client = ResourceManagementClient(credential=DefaultAzureCredential(), subscription_id=sub_id)
client.resource_groups.create_or_update(GROUP_NAME, {'location': LOCATION})
```

DEMOS

USEFUL LINKS AND RESOURCES

- Azure libraries (SDK) for Python - https://learn.microsoft.com/en-us/azure/developer/python/sdk/azure-sdk-overview
- Google Python Style Guide - https://google.github.io/styleguide/pyguide.html
- Python.org - https://www.python.org/
- Poetry - https://python-poetry.org/docs/
- Black -Python code formatter - https://github.com/psf/black
- Awesome Python - https://github.com/vinta/awesome-python
- Ruff - Python linter and formatter - https://github.com/astral-sh/ruff
- Azure for Python Developers - https://learn.microsoft.com/en-us/azure/developer/python/
- Bandit – code security analyser - https://bandit.readthedocs.io/en/latest/index.html
- PyInstaller - single package bundler - https://github.com/pyinstaller/pyinstaller
- FastAPI - https://fastapi.tiangolo.com/
- Python Cheatsheet - https://www.pythoncheatsheet.org/
- Python Package Index (PyPI) - https://pypi.org/
- PEP8 style guides - https://peps.python.org/pep-0008/

TOP PYTHON LIBRARIES

QUESTIONS?

THANK YOU