



When You are Ready
to **GO**

Beyond PYTHON



Frank Seesink, UNC Chapel Hill



First, a message from
our sponsor...



What I picture in my head...



What it ends up looking like...



Actually that's not quite right. The guy who made this is **clearly** more talented.

Who am I?

Frank Seesink

- Senior Network Engineer, UNC Chapel Hill
- Part of network DevOps group
- Involved in network automation for years
- Love languages, both human & computer
- Programming since I was 12 years old
- Formally B.S. in Computer Science with all coursework for an M.S. in same
- JOAT - databases, OSes, networking,...



Story time...

django



Red Hat
OpenShift

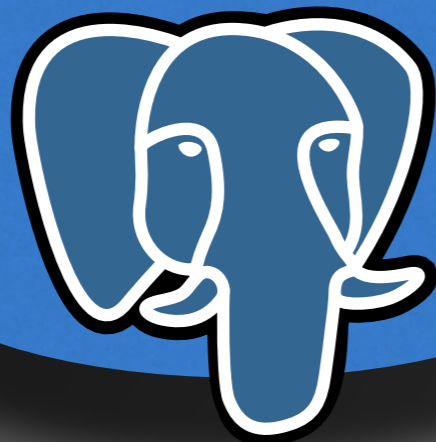
NETM^{KO}

SQLite



Nornir

db



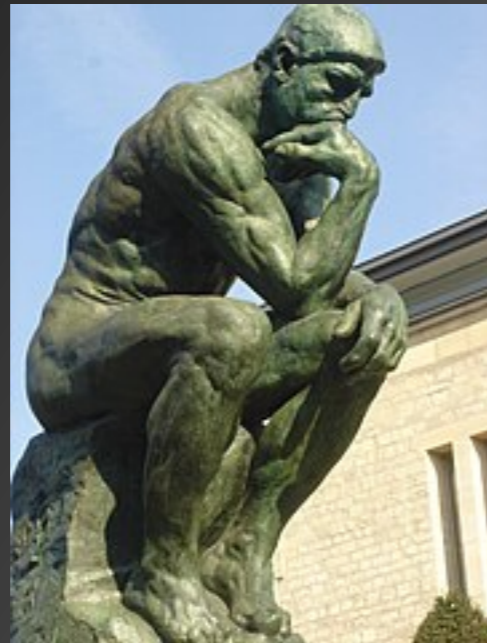
gunicorn

Work environment



Story time...

In January 2022, I was in a rut...



Why Go?

- Python's creator, Guido van Rossum, worked at Google from 2005-2012.
- For years Google heavily used Python internally and even offered Python classes to its employees.
 - <https://developers.google.com/edu/python>
- Google had also hired Rob Pike and Ken Thompson of Bell Labs (UNIX, C) fame. They, along with Robert Griesemer, created Go.
- In 2013 Guido van Rossum went to work at Dropbox. (Dropbox was known to use Python.) That seemed odd.
- In 2014 Google publicly released Kubernetes, which is written in Go.

The writing was on the wall?



Why Go?



“Language of the cloud”



HashiCorp
Terraform



Go (Golang)

<https://go.dev>

GO Why Go Learn Docs Packages Community

Build simple, secure, scalable systems with Go

- ✓ An open-source programming language supported by Google
- ✓ Easy to learn and great for teams
- ✓ Built-in concurrency and a robust standard library
- ✓ Large ecosystem of partners, communities, and tools

[Get Started](#) [Download](#)

Download packages for [Windows 64-bit](#), [macOS](#), [Linux](#), and [more](#)

The go command by default downloads and authenticates modules using the Go module mirror and Go checksum database run by Google. [Learn more.](#)

Companies using Go

Organizations in every industry use Go to power their software and services [View all stories](#)

go.dev uses cookies from Google to deliver and enhance the quality of its services and to analyze traffic. [Learn more.](#) [Okay](#)

<https://go.dev/>



Go (Golang)



- **Learning Go**

<https://www.linkedin.com/learning/learning-go>

- **Go for Python Developers**

<https://www.linkedin.com/learning/go-for-python-developers>

- <https://learnxinyminutes.com/docs/go/>



Fyne



A screenshot of the fyne.io website. The browser address bar shows 'fyne.io'. The website has a dark blue header with the 'fyne' logo and navigation links: CONFERENCE, DOCS, APPS, ADD-ONS, BLOG, EVENTS, SUPPORT. The main content area features the headline 'EASILY BUILD NATIVE APPS THAT WORK EVERYWHERE' in large, light blue, all-caps text. Below this is a dark blue section with white text: 'An easy to learn toolkit for creating graphical apps for desktop, mobile and web. Our free and open source libraries combine the simplicity of the Go programming language with a carefully crafted library of widgets to simplify coding any app. But also, Fyne apps can be built for all platforms and stores!'. At the bottom, there is a 'Gallery' section with a white background, showing several small screenshots of applications built with Fyne, including a widget catalog, a form with input fields and checkboxes, and a travel app for Edinburgh.



<https://fyne.io/>

To Learn a Programming Language...

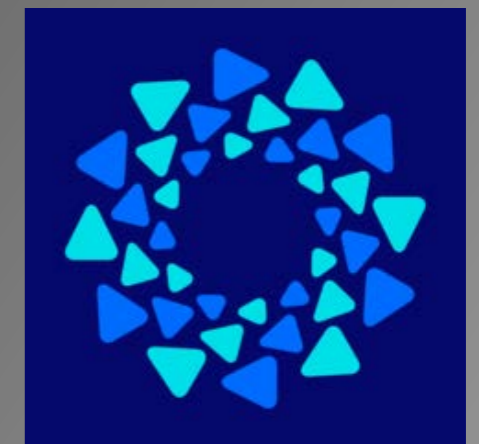
1. You need to program in it
2. You need to program in it
3. You need to program in it
4. You need to have a project/goal



Initial Go Test Project



A screenshot of a GitHub repository page for "fseesink/MySetup". The page shows a list of files and folders, including "img", ".DS_Store", ".gitignore", "BUILD.md", "icon.png", "LICENSE", "Mockup.drawio", "MySetup.go", "README.md", "buildapp.sh.example", "go.mod", "go.sum", and "settings.go.example". The "README.md" file is selected and its content is displayed below. The README describes "MySetup" as a simple network diagnostic utility for collecting information about a host, written in Go using the Fyne.io GUI toolkit. It includes a screenshot of the application's interface, which shows a window titled "MySetup" with tabs for "OS", "Host Interfaces", "Routing", "Public IP", "Command Output", and "Full Output". The "OS" tab is active, showing the hostname "fseesink@mysetup" and the operating system "Apple macOS [darwin]". Below this, there is a question: "May we collect the following information?" followed by a list of checks: "Check outbound path to 1.1.1.1", "Check outbound path to 8.8.8.8", "Check source IP as seen from https://icanhazip.com/", "Check source IP as seen from https://ifconfig.me/ip", and "Check source IP as seen from https://pinf.io/ip". The interface also shows the command "netstat -tn" and the output "Run command: ifconfig". At the bottom, there are "No" and "Yes" buttons.



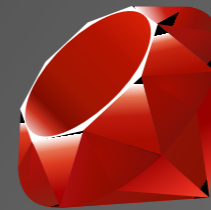
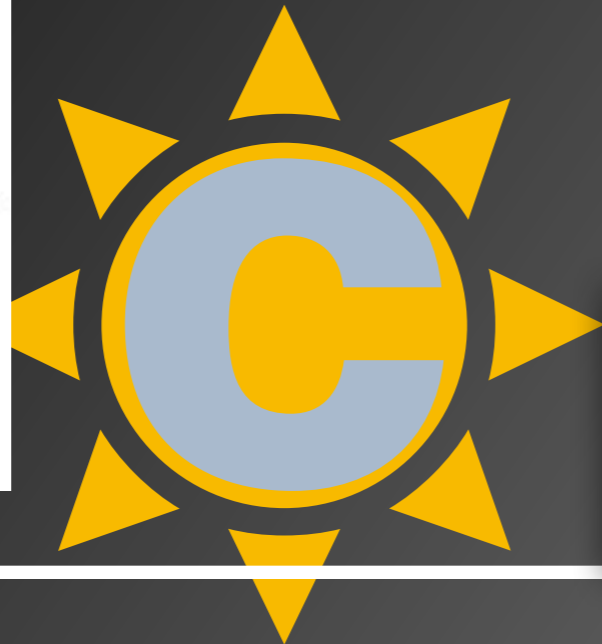
fyne.io

<https://github.com/fseesink/mysetup>

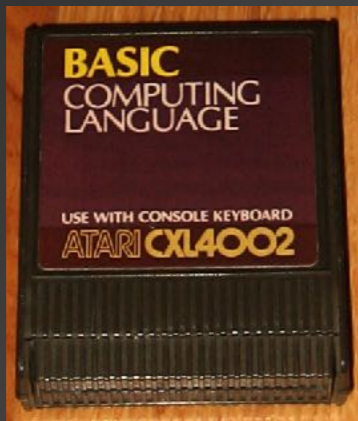


History of Programming Languages

0/1



1940s

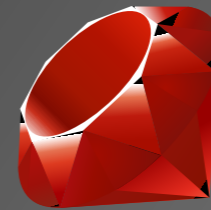
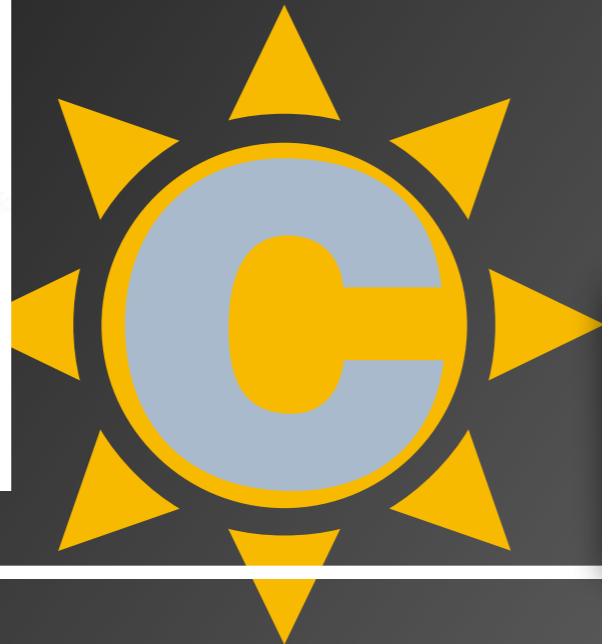


Present

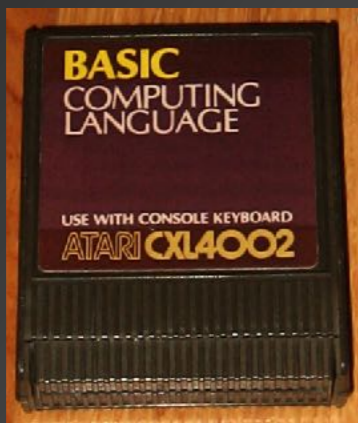


History of Programming Languages

0/1



1940s



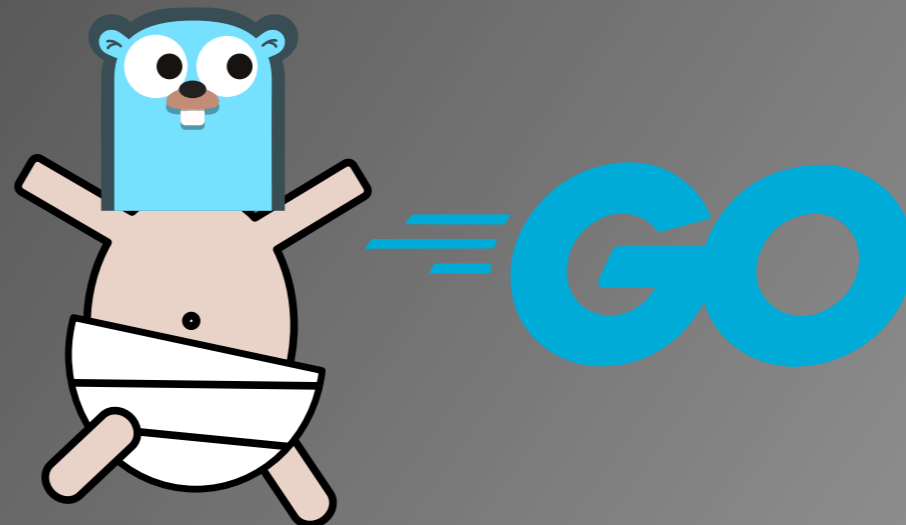
&



Present



History of Programming Languages





History/Comparison

	C	Python	Go
First appeared	1972	1992	2009
Designed by	Dennis Ritchie	Guido van Rossum	Robert Griesemer Rob Pike Ken Thompson
Typing	Static, weak, manifest, nominal	Duck, dynamic, strong typing	Inferred, static, strong, structural, nominal
Keywords	32	35	25



Features

	C	Python	Go
Built-in concurrency	N/A	N/A	Go routines
Concurrency via libraries	<code>fork()</code> *provides access to underlying OS concurrency features	<code>multiprocessing</code> <code>concurrent.futures</code> <code>asyncio</code>	
Native multi-core support	N/A	N/A due to GIL	
Memory Management	<code>malloc()/free()</code> *developer responsible for all memory mgmt	Garbage Collection	Garbage Collection



Libraries/Modules

	C	Python	Go
Standard Library	✓	✓	✓
Package ecosystem	N/A	PyPI.org	via VCS such as Git
Example package import	<code>#include <stdio.h></code>	<code>import netmiko</code>	<code>import ("github.com/ nornir-automation/ gornir/pkg/gornir")</code>
Largest library (e.g., AI/ML, data analysis)		✓	



Workflow

Python

```
#!/usr/local/bin/python3
```

```
print("Hello world")
```

```
$ python3 helloworld.py
```

or if permissions set, simply

```
$ helloworld.py
```

Go

```
package main
```

```
import "fmt"
```

```
func main() {  
    fmt.Println("Hello world")  
}
```

```
$ go run helloworld.go
```

or

```
$ go run .
```

to run interactively.

Compile and run executable
with

```
$ go build .
```

```
$ helloworld
```



Workflow Performance

Python

```
#!/usr/local/bin/python3
```

```
print("Hello world")
```

```
$ time python3 helloworld.py
Hello world
python3 helloworld.py 0.02s
user 0.02s system 36% cpu
0.111 total
```

Time to run
executable
binary

Go

```
package main
```

```
import "fmt"
```

```
func main() {
    fmt.Println("Hello world")
}
```

Time to compile
AND run the
program (when
developing)

```
$ time go run helloworld.go
Hello world
go run helloworld.go 0.14s
user 0.29s system 49% cpu
0.860 total
```

```
$ go build helloworld.go
```

```
$ time ./helloworld
```

```
Hello world
./helloworld 0.00s user 0.00s
system 2% cpu 0.135 total
```



Final Program Size

Python

```
#!/usr/local/bin/python3
```

```
print("Hello world")
```

C version
TOTAL == 32 KB
or 0.032 MB

46 bytes: helloworld.py
310 MB: Python install (*)

To run a Python script, you need Python installed.

TOTAL == ~310 MB

(*) v3.11.5 macOS installation on disk

Go

```
package main
```

```
import "fmt"
```

```
func main() {  
    fmt.Println("Hello world")  
}
```

72 bytes: helloworld.go
238 MB: Go install (*)
1.8 MB: helloworld binary

To run a Go compiled app, you just need the binary.

TOTAL == 1.8 MB

(*) v1.21.0 macOS installation on disk

Language Similarities

Python

```
import os
```

```
def itsvalid():  
    print("Valid day of the month")  
    cwd = os.getcwd()  
    print(cwd)
```

```
def main():  
    # Variable assignment  
    name = "Frank"  
    day = 19
```

```
    if day >= 1 and day < 31:  
        itsvalid()
```

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

Go

```
package main
```

```
import (  
    "fmt"  
    "os"  
)
```

```
func itsvalid() {  
    fmt.Println("Valid day of the month")  
    cwd, _ := os.Getwd()  
    fmt.Println(cwd)  
}
```

```
func main() {  
    // Variable assignment  
    name := "Frank"  
    day := 19
```

```
    if day >= 1 && day < 31 {  
        itsvalid()  
    }
```

```
    fmt.Println(name)  
}
```



Global Interpreter Lock (GIL)



GIL

“In CPython, the global interpreter lock, or GIL, is a mutex that protects access to Python objects, preventing multiple threads from executing Python bytecodes at once. The GIL prevents race conditions and ensures thread safety. A nice explanation of how the Python GIL helps in these areas can be found here. In short, this mutex is necessary mainly because CPython's memory management is not thread-safe.”

- <https://wiki.python.org/moin/GlobalInterpreterLock>



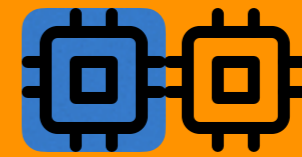
Python's Lack of Concurrency

1990s

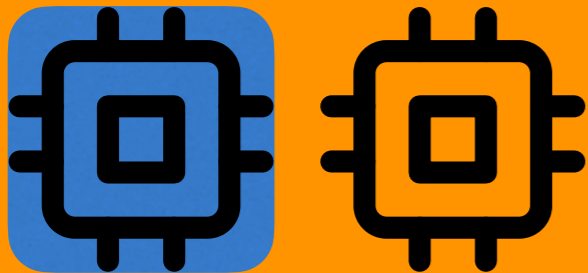
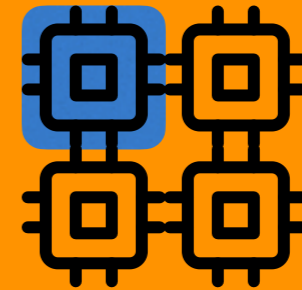


100%

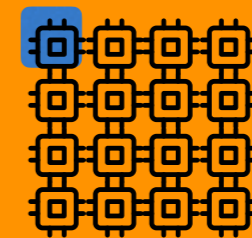
2000s



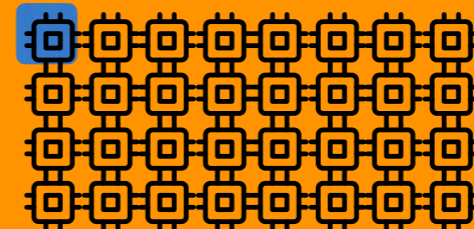
25%



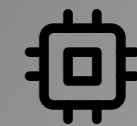
50%



6.25%



3.125%



== CPU core



== Python



To bypass the GIL

To use multiple threads/cores, you must take action. This requires extra effort.

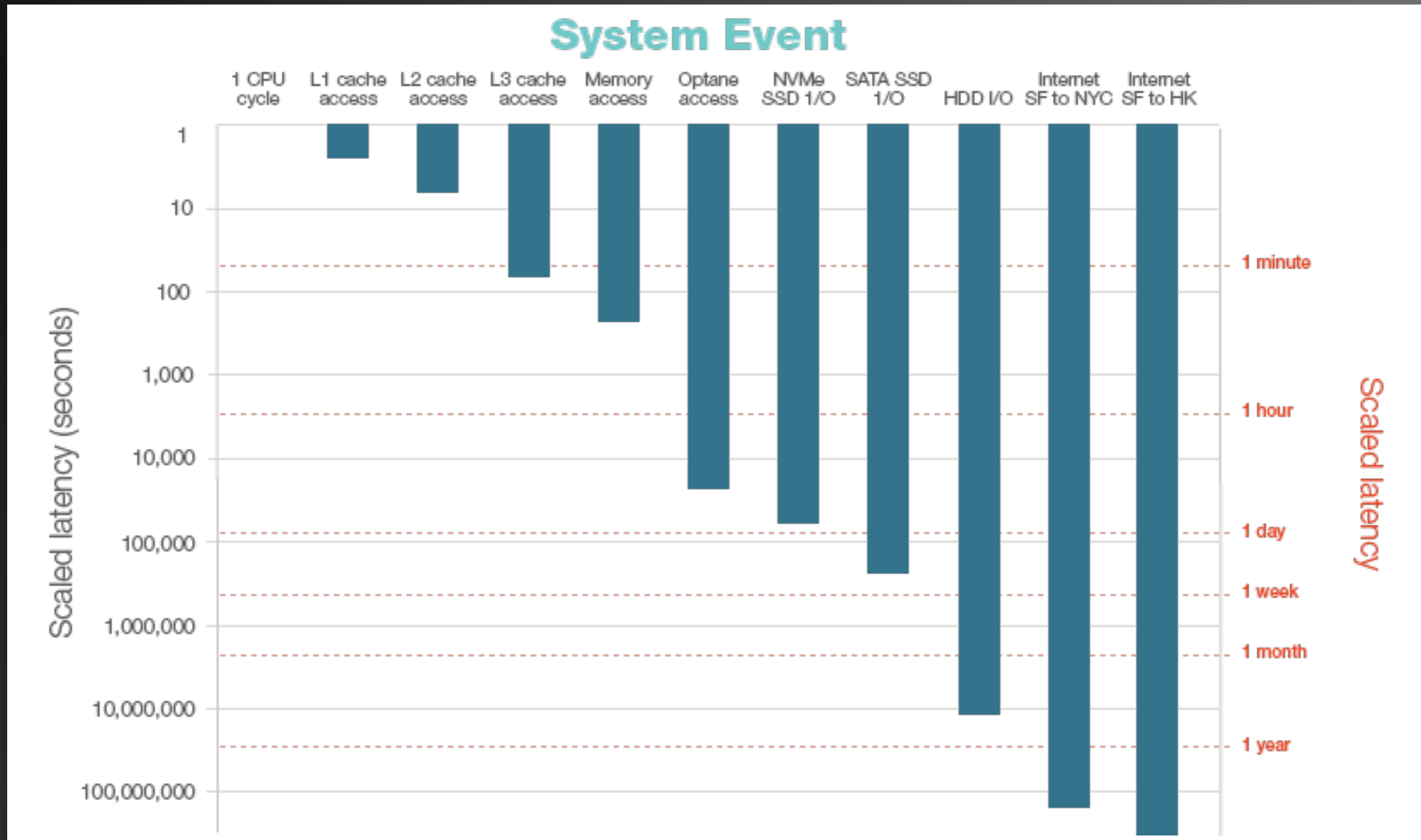
For example,

- multiprocessing or concurrent.futures module in Python standard library. (You must use processes and not threads in latter. Otherwise it stays within a single core.)
- Use modules like Nornir (which use concurrent futures)

`asyncio` does NOT help here. That is cooperative multi-threading. Again, single core.



Disclaimer



Network Automation tends to be
I/O-bound vs. **CPU-bound**



Possible Python Future

PEP 703 – Making the Global Interpreter Lock Optional in CPython

CPython's global interpreter lock ("GIL") prevents multiple threads from executing Python code at the same time. The GIL is an obstacle to using multi-core CPUs from Python efficiently. This PEP proposes adding a build configuration (`--disable-gil`) to CPython to let it run Python code without the global interpreter lock and with the necessary changes needed to make the interpreter thread-safe.

<https://peps.python.org/pep-0703/>



Go routines

1. Put 'go' in front of a function call.
2. ...
3. Profit!

Main routine waits for function

```
func main() {  
    // Variable assignment  
    ...  
    dosomething()  
    ...  
}
```

Main routine keeps going

```
func main() {  
    // Variable assignment  
    ...  
    go dosomething()  
    ...  
}
```



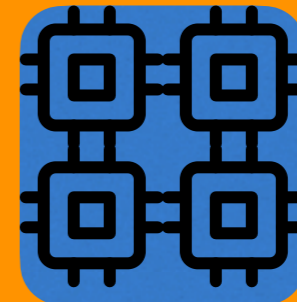
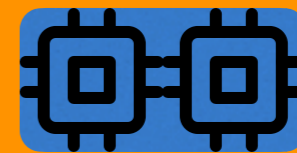
Go routines

1990s

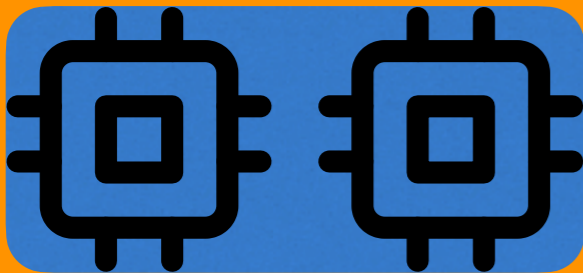


100%

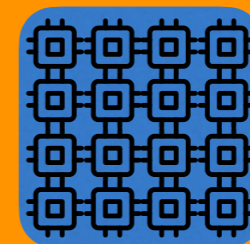
2000s



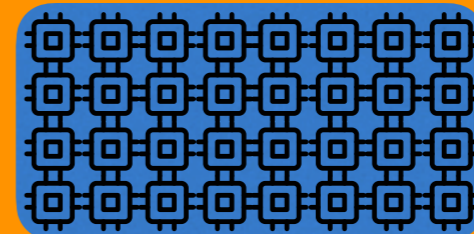
100%




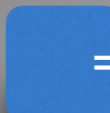
100%



100%



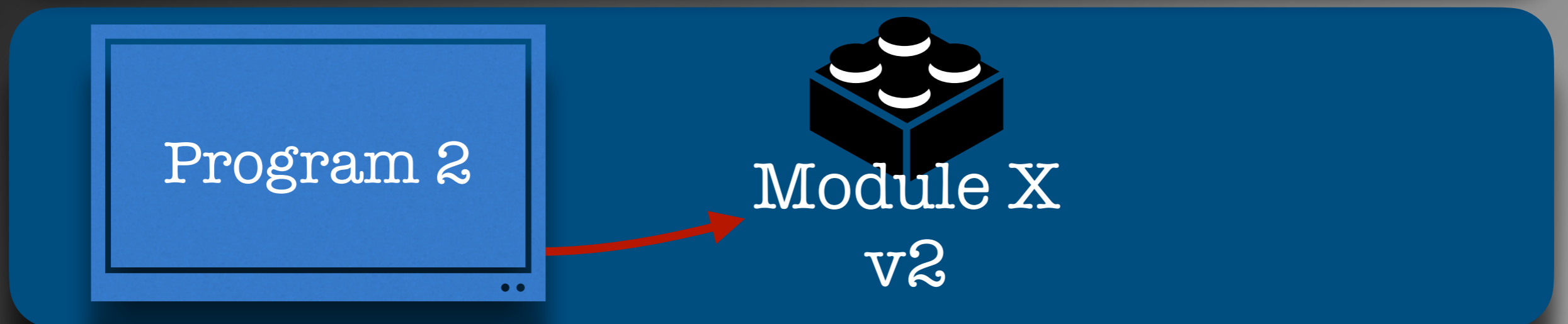
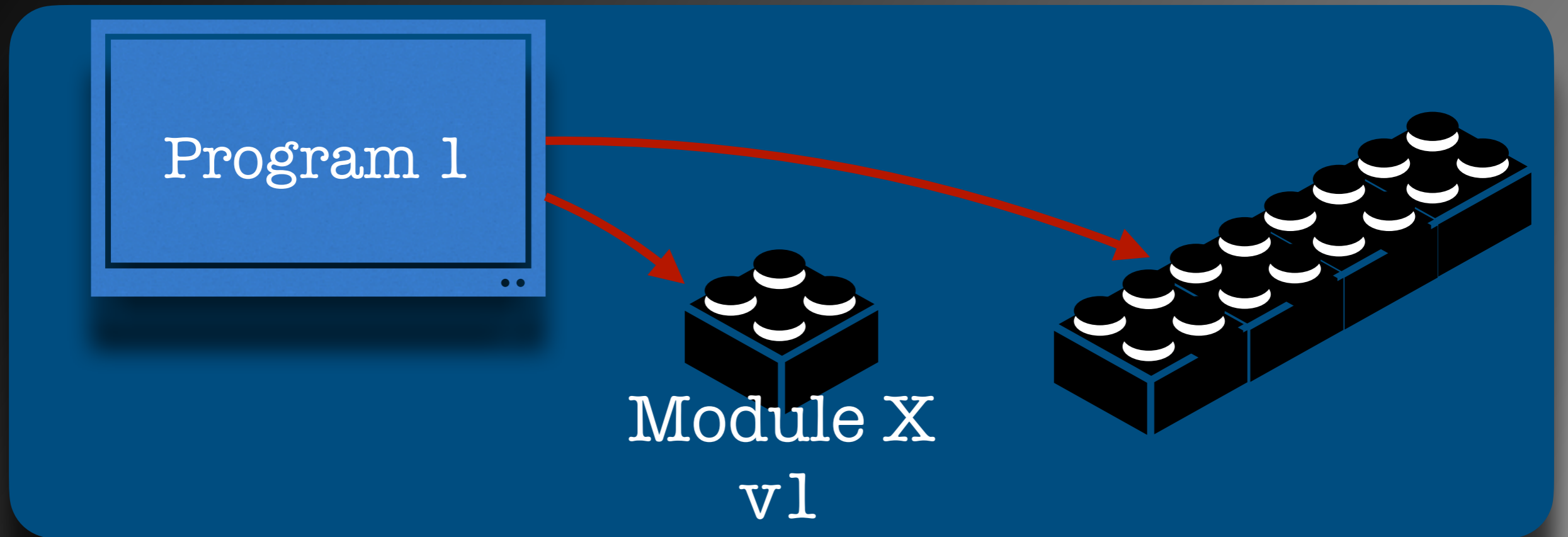
100%

 == CPU core
 == Go routines

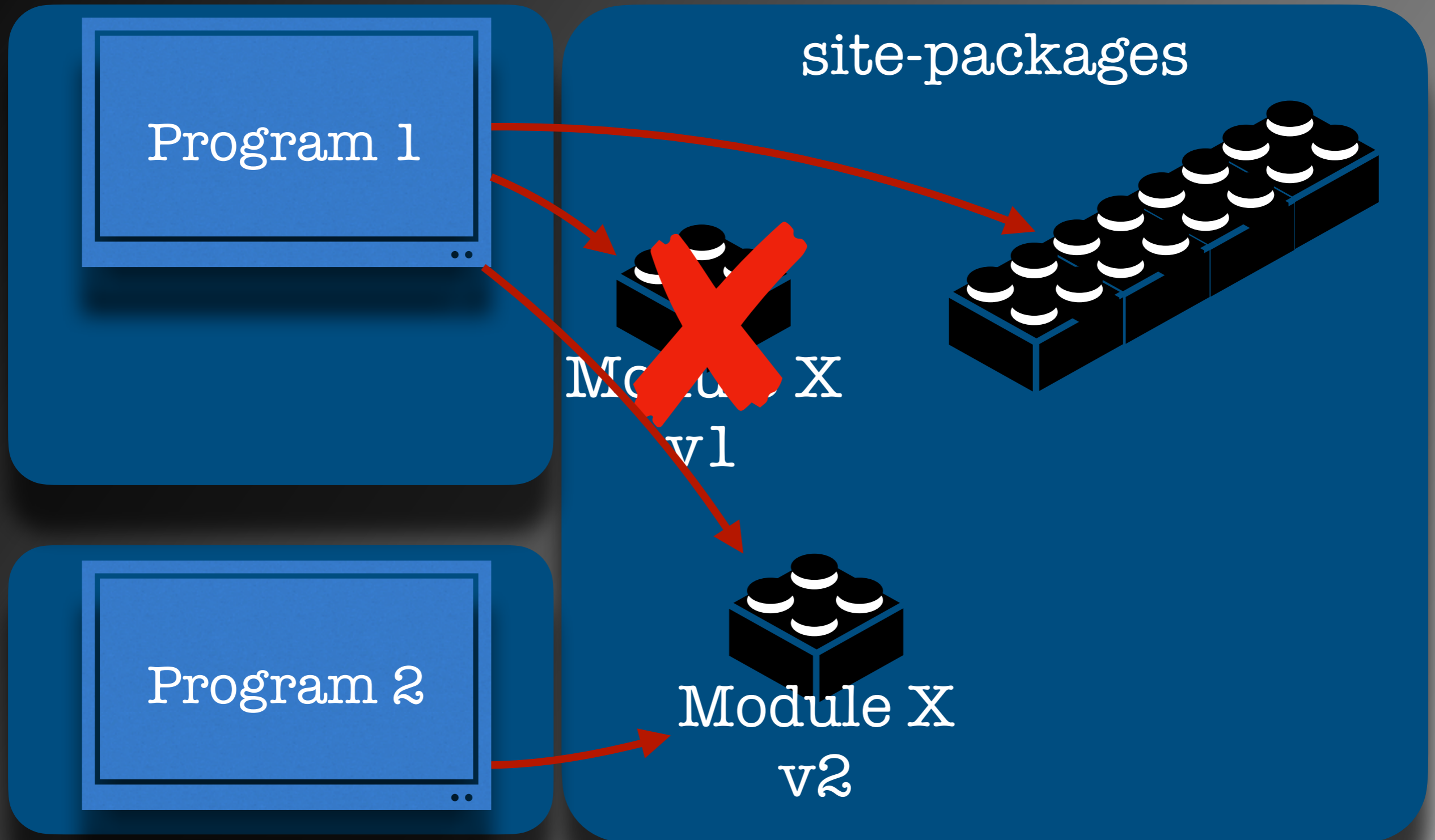
Dependency Hell



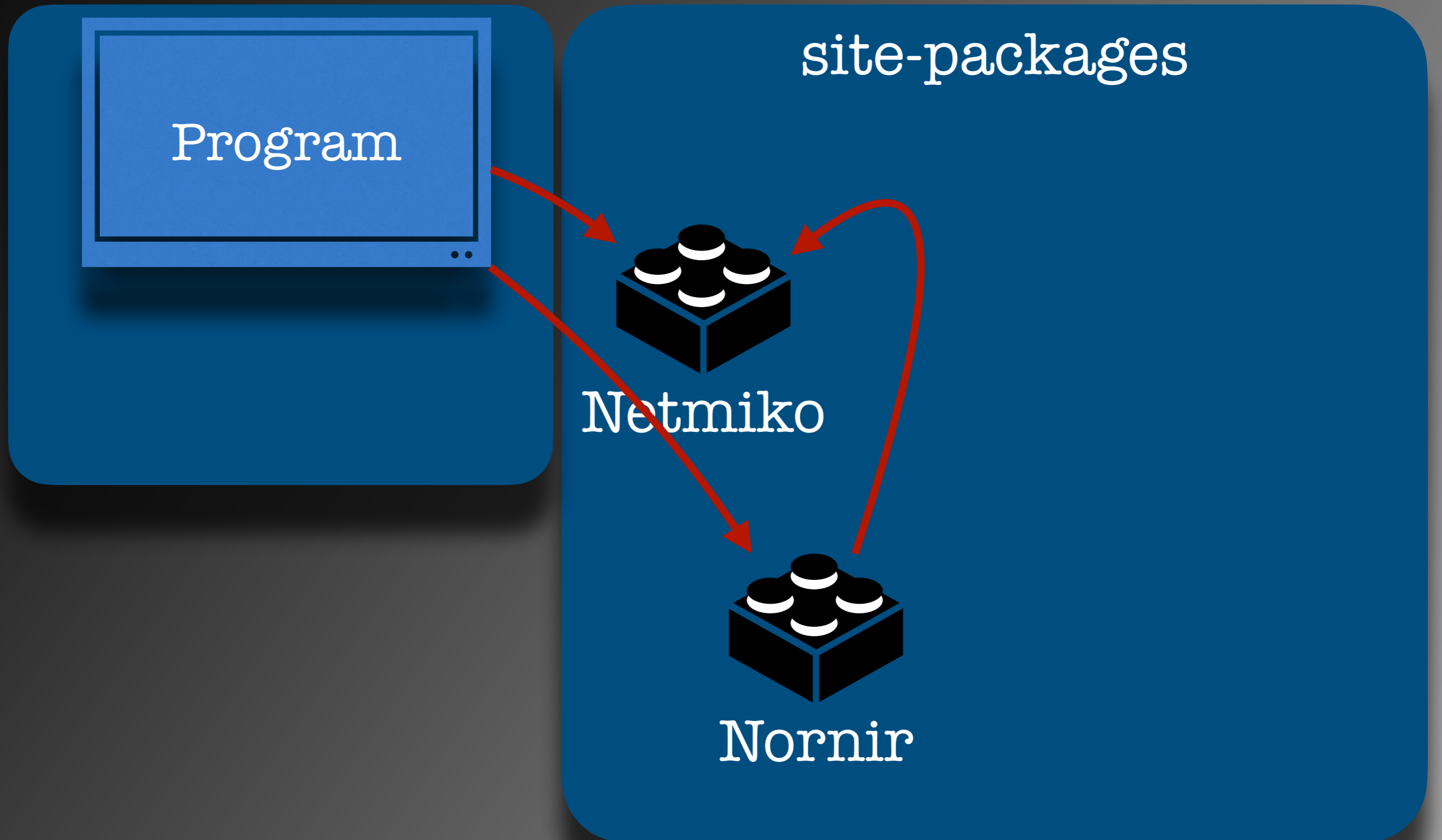
Dependency Hell



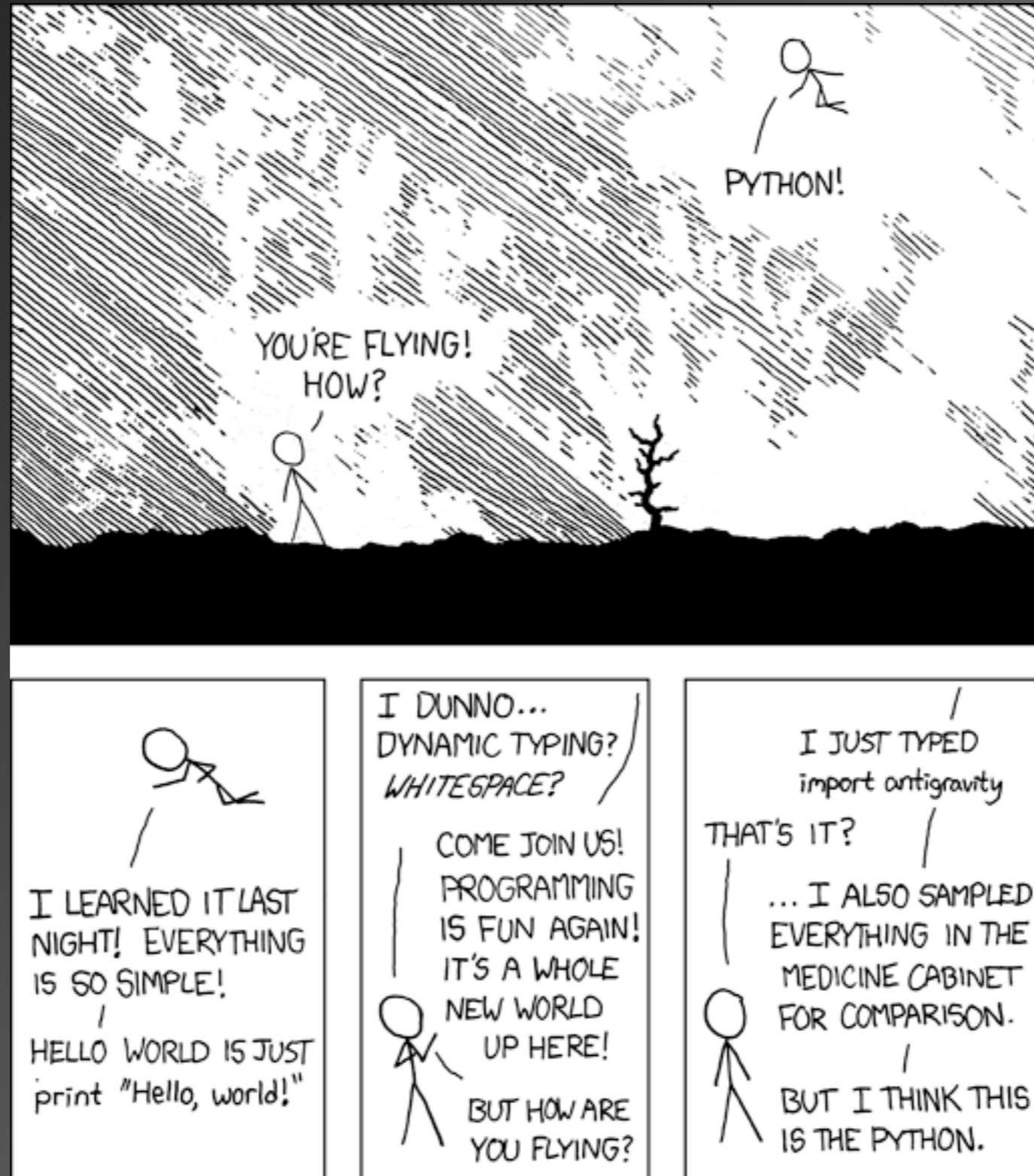
Dependency Hell



Dependency Hell (cont.)



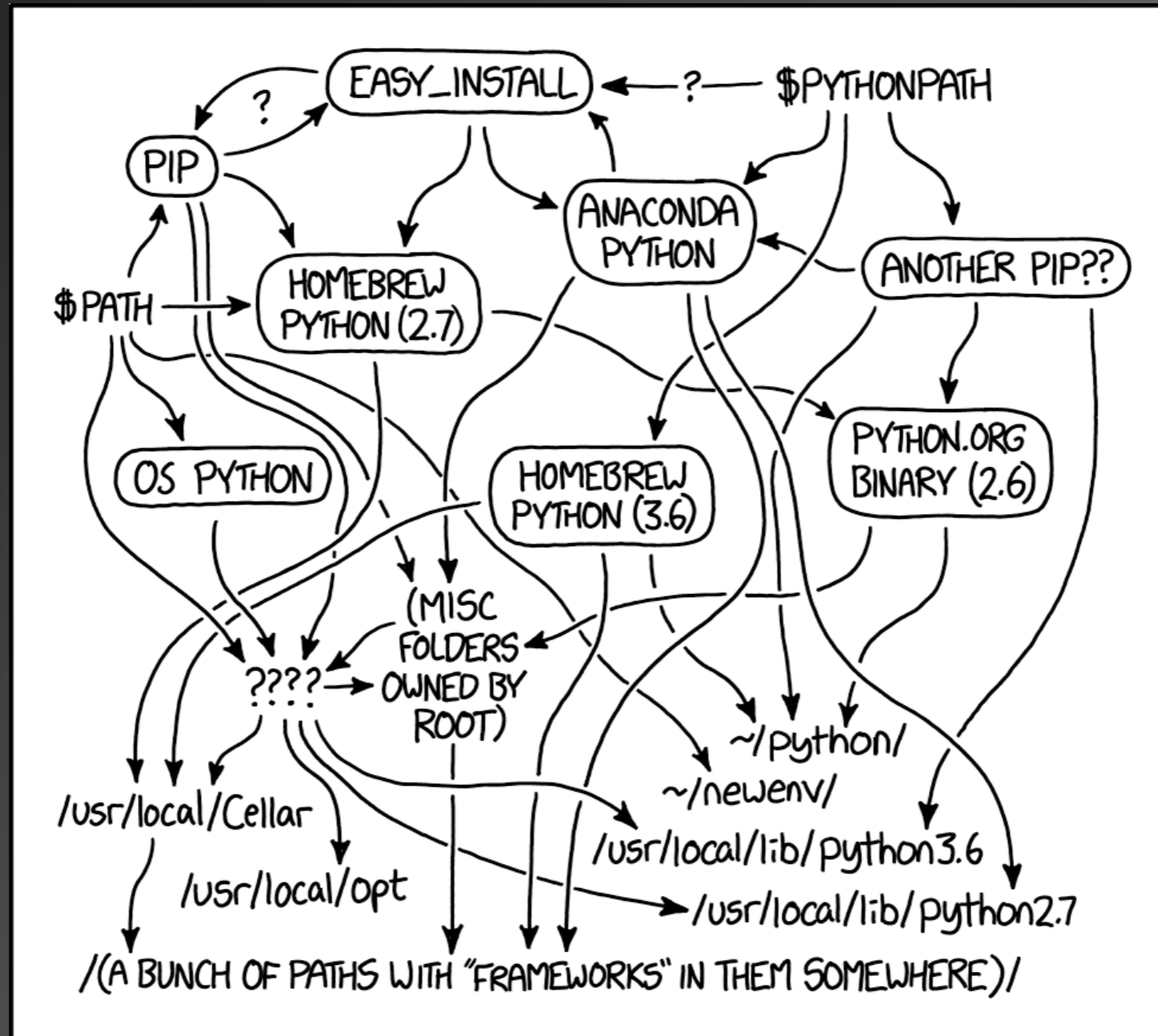
When you first learn Python, it's like this



<https://xkcd.com/353/>



Eventually, it becomes this...



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

<https://xkcd.com/1987/>



What about Python's
ability to run on
different platforms?



Go Cross-Compilation



Go Cross-Compilation

- Go creates binary executables specific to an OS/architecture (e.g., x64 Windows, ARM64 Linux)
- Go can cross-compile to ANY supported OS/architecture combination FROM any supported OS/architecture. Simply set GOOS and GOARCH environment variables.

```
$ GOOS=linux GOARCH=arm64 go build .
```



So when should you
use Go?



Depend



UNDERPADS



12
COUNT

NIGHT DEFENSE[®]
SOFT, TRIPLE LAYER **OVERNIGHT PROTECTION**



It Depends.



What makes Go worth considering

- Pythonic code (relatively easy transition)
- Go routines / native multi-core support
- Single binary executable with NO EXTERNAL DEPENDENCIES
- Can compile to any supported architecture/ OS from a single platform
- Performant: best balance between coding speed and execution speed
- BONUS: Fyne is a nice, cross-platform GUI framework



Thank You



[https://frank.seesink.com/presentations/
Internet2TechEx-Fall2023/](https://frank.seesink.com/presentations/Internet2TechEx-Fall2023/)

Frank Seesink
frank@seesink.com
frank@unc.edu

