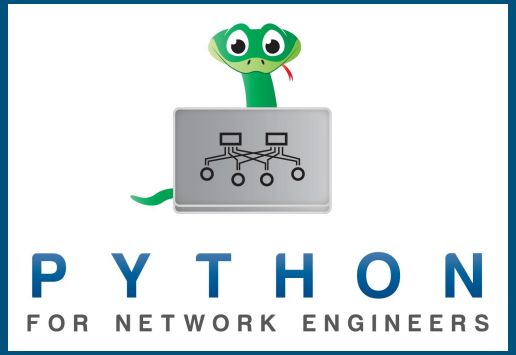
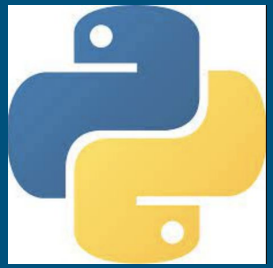
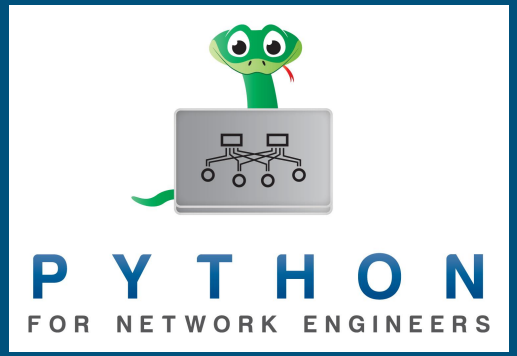




Live Training Session
December 2021

Day3 Schedule

pdbr and the Python debugger

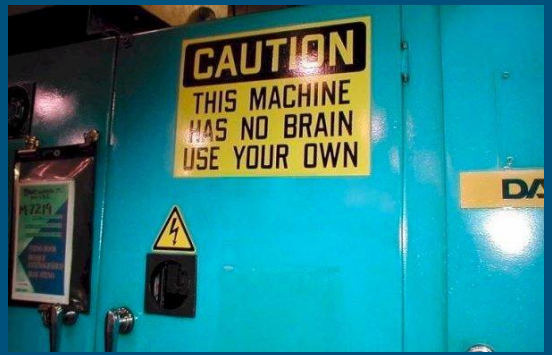


Juniper PyEZ Configurations

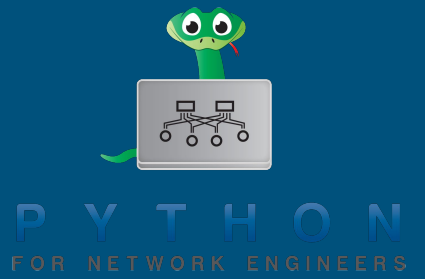
Juniper PyEZ Direct RPC (optional)

Arista and eAPI

SSH and Concurrency



Pdbr - The Python Debugger (+rich)




```
python -m pdbr my_script.py

import pdbr
pdbr.set_trace()

Pdb Commands
help (h)
list (l)
ll              # Long List

next (n)        # Step one line at a time; don't descend
step (s)        # Step one line at a time descend into callables
break 16 (b 16) # Set a breakpoint at line 16
continue (c)    # Continue execution
```

Pdb - The Python Debugger

./day3/pdb/pdb_ex1.txt

Pdb Commands

```
down (d)        # Move down the stack
up (u)          # Move up the stack

p foo           # Print out variable foo
pp foo          # Pretty print out variable foo

!print("hello")  # Exclamation point can prefix generic Python code

quit (q)        # Abort the current Pdb session and program
```

PYTHON
FOR NETWORK ENGINEERS

PyEZ config operations

Exercises:
./day3/jnpr/ex3.txt

```
a_device = Device(host="vmx2.lasthop.io", user="pyclass", password=getpass())
a_device.open()
a_device.timeout = 60

cfg = Config(a_device)
cfg.lock()

pdbr.set_trace()
cfg.load("set system host-name test123", format="set", merge=True)
cfg.rollback(0)

cfg.load("set system host-name test123", format="set", merge=True)
print(cfg.diff())

cfg.commit()
# cfg.commit(comment="Testing from pyez")

cfg.load(path="test_config.conf", format="text", merge=True)
print(cfg.diff())
cfg.rollback(0)
cfg.unlock()
```

PyEZ direct RPC

```
pyclass@vmx2> show version | display xml rpc
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/18.4R1/junos">
    <rpc>
        <get-software-information>
        </get-software-information>
    </rpc>
    <cli>
        <banner></banner>
    </cli>
</rpc-reply>
```

PyEZ direct RPC


```
a_device = Device(**device)
a_device.open()

# show version | display xml rpc
# <get-software-information>
xml_out = a_device.rpc.get_software_information()
print(etree.tostring(xml_out, encoding="unicode", pretty_print=True))
```

PyEZ direct RPC (XML)

Exercises:
./day3/jnpr/ex4.txt

```
<software-information>
  <host-name>vmx1</host-name>
  <product-model>vmx</product-model>
  <product-name>vmx</product-name>
  <junos-version>18.4R1.8</junos-version>
  <package-information>
    <name>os-kernel</name>
    <comment>JUNOS OS Kernel 64-bit  [20181207.6c2f68b_2_builder_stable_11]</comment>
  </package-information>
  <package-information>
    <name>os-libs</name>
    <comment>JUNOS OS libs [20181207.6c2f68b_2_builder_stable_11]</comment>
  </package-information>
  <package-information>
    <name>os-runtime</name>
    <comment>JUNOS OS runtime [20181207.6c2f68b_2_builder_stable_11]</comment>
```

Arista eAPI

{{ github_repo }}/eapi_examples

```
import pyeapi
from getpass import getpass
from rich import print

connection = pyeapi.client.connect(
    transport="https",
    host="arista1.lasthop.io",
    username="pyclass",
    password=getpass(),
    port="443",
)

device = pyeapi.client.Node(connection)
output = device.enable(["show version", "show ip arp"])
print(output)
```

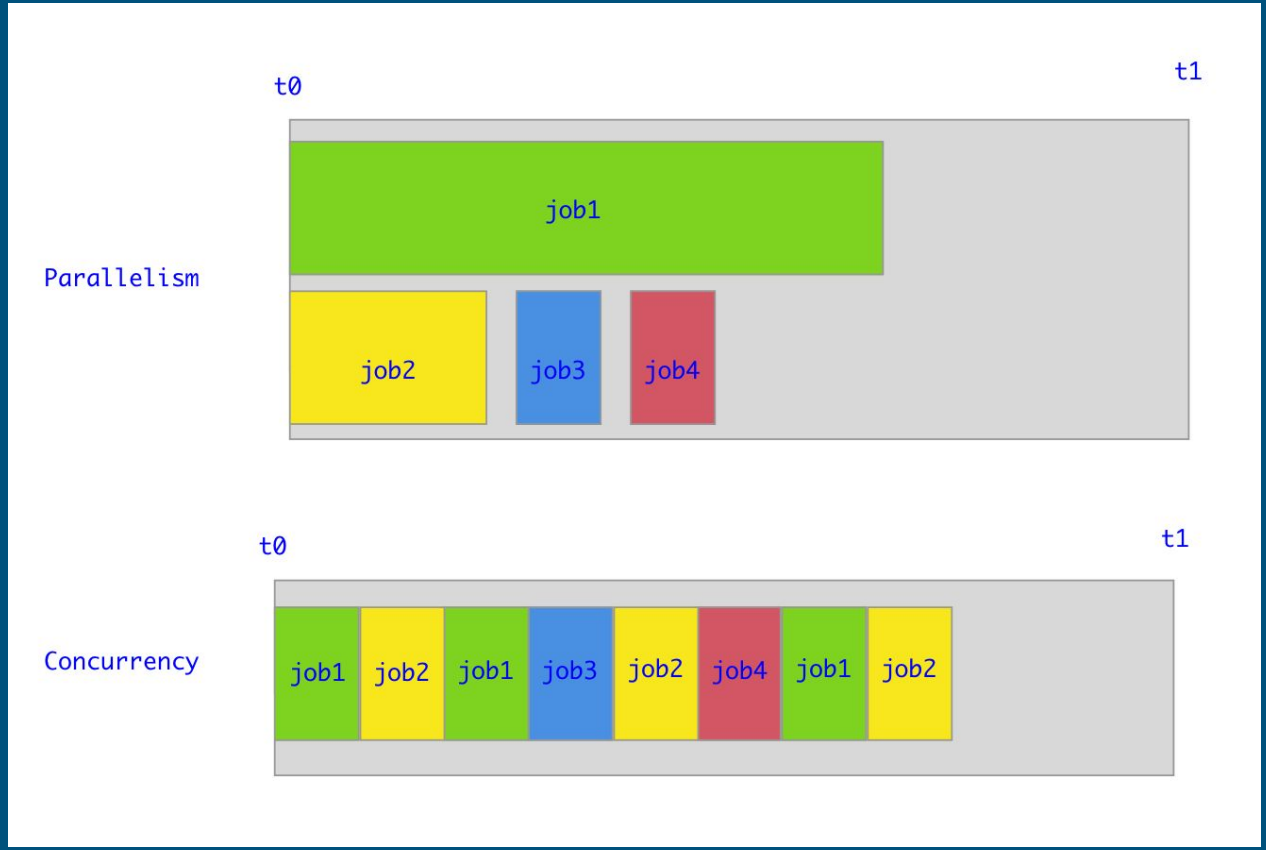
Arista eAPI (Config)

Exercises:
./day3/arista/ex1.txt
./day3/arista/ex2.txt

```
connection = pyeapi.client.connect(
    transport="https",
    host="arista1.lasthop.io",
    username="pyclass",
    password=getpass(),
    port="443",
)

cfg = ["vlan 225", "name green", "vlan 226", "name red"]
device = pyeapi.client.Node(connection)
output = device.config(cfg)
print(output)
```

Concurrency/Parallelism

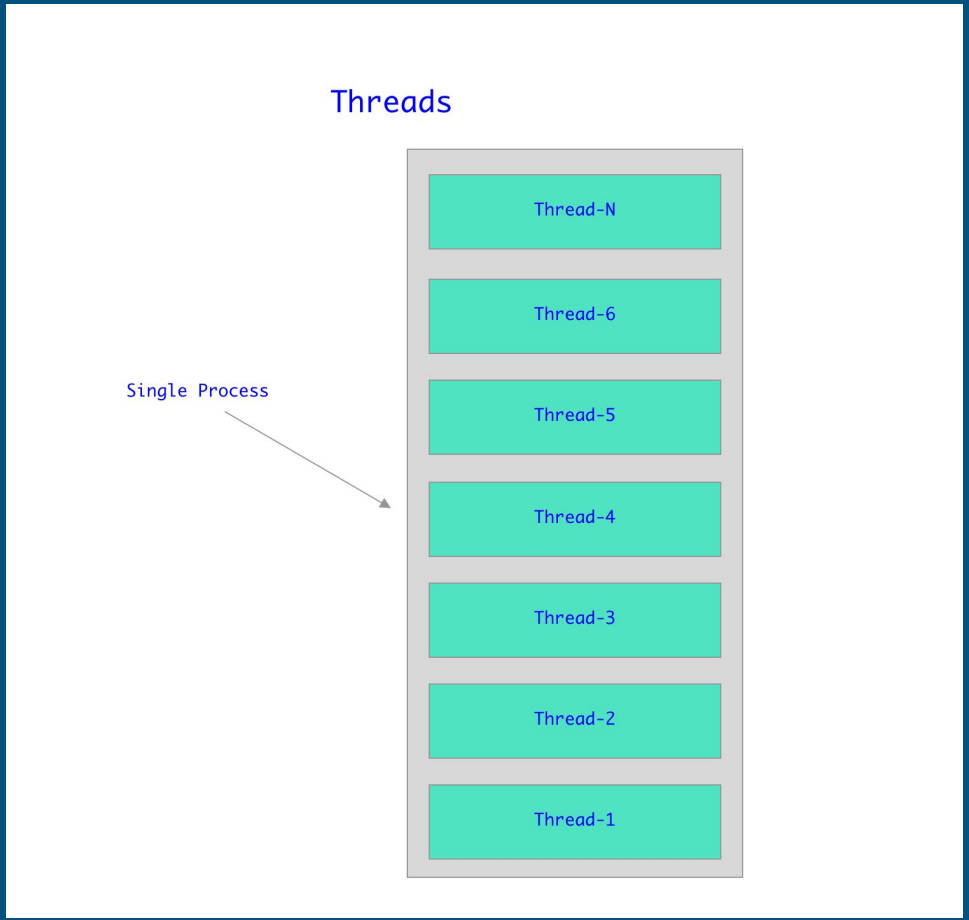


- Concurrency? Parallelism?
- Python and the GIL
- Concurrent Futures

Concurrent Futures

- Wrapper around Threading/Processes
- Provides consistent interface using either Threads or Processes -- meaning very easy to switch concurrency method
- Threads: for I/O bound things (waiting for stuff in the network)
- Processes: for CPU bound things (crunch lots and lots of numbers)

Concurrent Futures - ThreadPool



Concurrent Futures - ThreadPool

Reference Material in:
{{ github_repo }}/concurrency_example

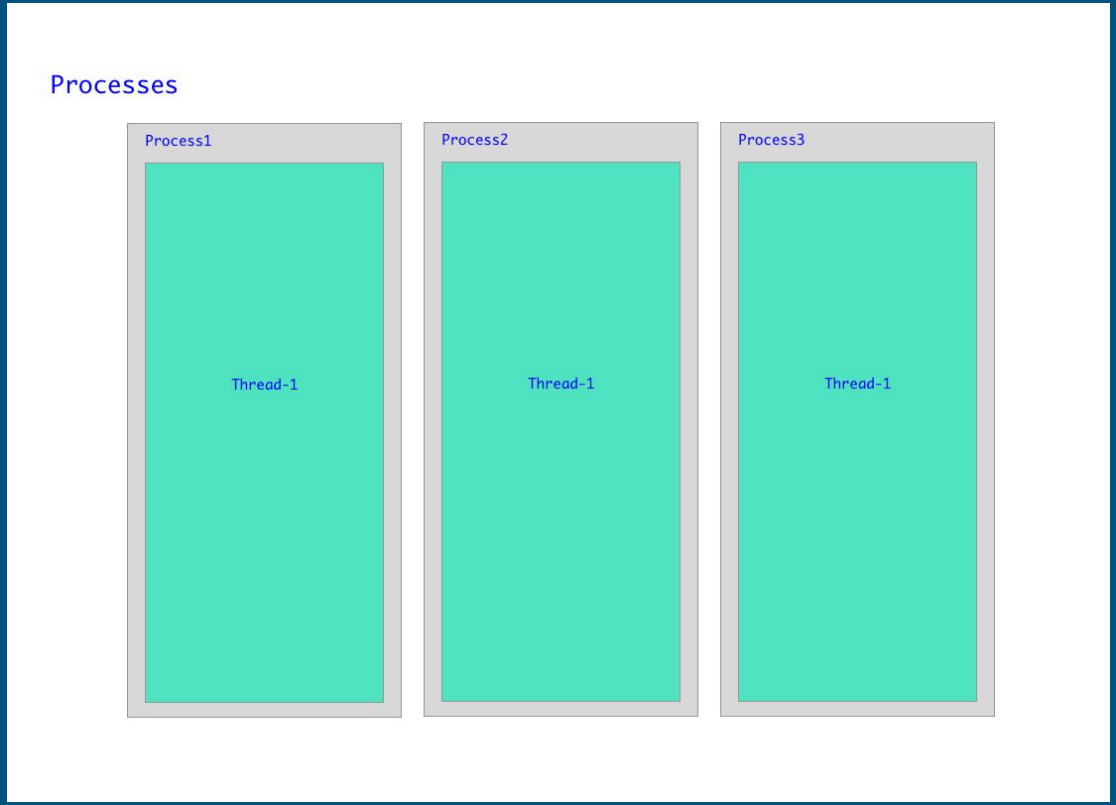
Exercises:
./day3/concurrency/ex1.txt

```
def main():
    # Create your thread pool
    pool = ThreadPoolExecutor(max_workers=WORKERS)
    futures = []

    # Submit the work to the thread pool
    for _ in range(TASKS):
        futures.append(pool.submit(math_calculation))

    # 'wait' will block until all of the tasks are complete
    wait(futures)
    for task_result in futures:
        print(task_result.result())
```

Concurrent Futures - ProcessPool



Concurrent Futures - ProcessPool

Exercises:
./day3/concurrency/ex2.txt

```
def main():
    # Create process pool
    pool = ProcessPoolExecutor(max_workers=PROC_POOL)
    futures = []

    # Submit work to process pool
    for _ in range(TASKS):
        futures.append(pool.submit(math_calculation))

    # Block waiting for tasks to complete
    wait(futures)
    for task_result in futures:
        print(task_result.result())
```

Concurrent Futures - As Completed

```
def main():
    # Create process pool
    pool = ProcessPoolExecutor(max_workers=PROC_POOL)
    futures = []

    # Submit work to process pool
    for _ in range(TASKS):
        futures.append(pool.submit(math_calculation))

    # Show results as the work completes
    for task_result in as_completed(futures):
        print(task_result.result())
```