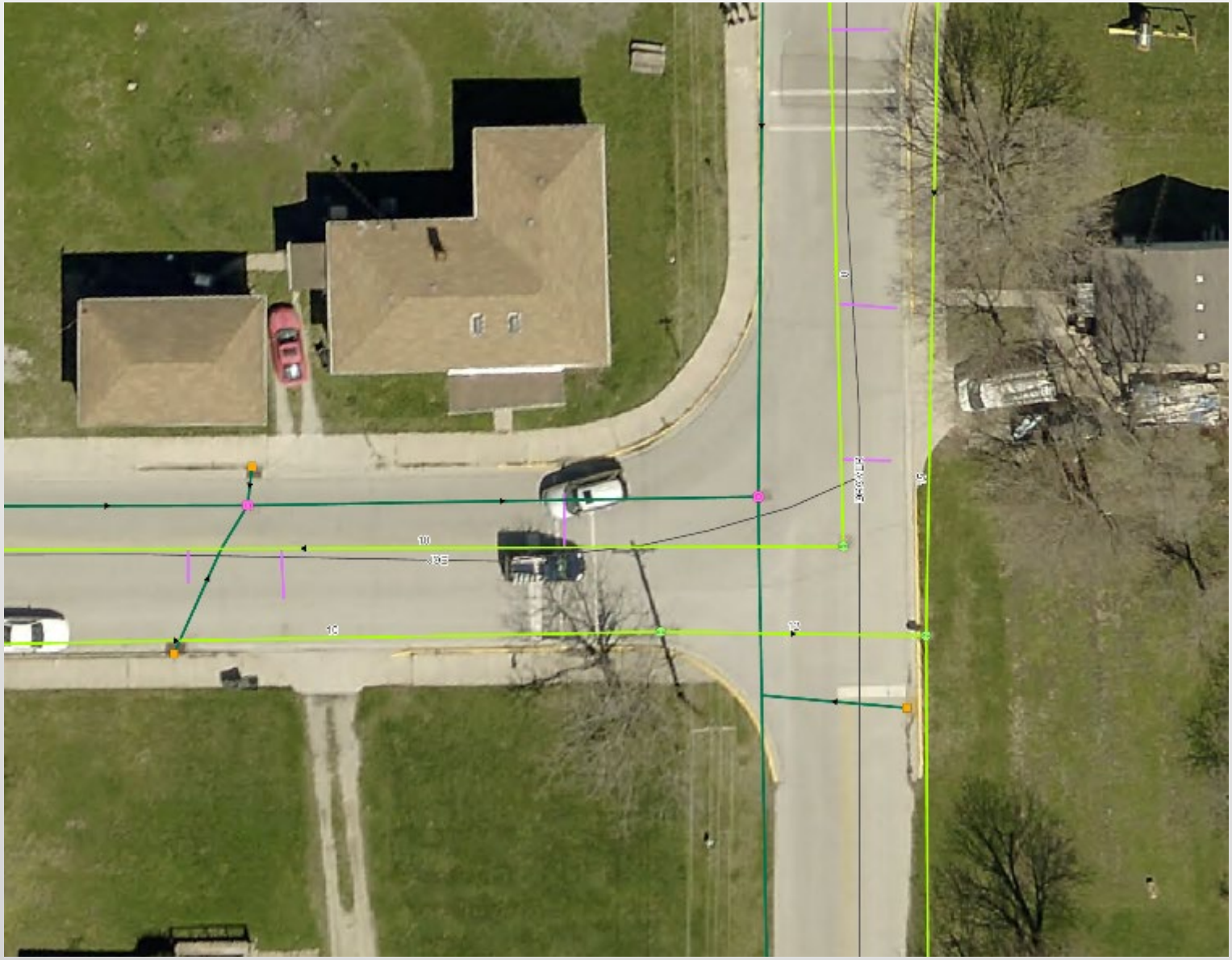
Python Multithreading

Speed up your Scripts 1,000%


PREDATOR


EXAMPLE




WORLD'S
STRONGEST
MAN
MAN

1

You can use map-reduce for this task.

Pseudo code:

```
map1(list): #runs on first file
  for each (i,x,y) in list:
     emit ((x,y),(1,i))
map2(list): #runs on 2nd file
  for each (x,y,temp) in list:
     emit ((x,y),(2,temp))
reduce((x,y),list): #runs on output of both mappers
  for each (aux, val) in list:
       if aux == 1:
            i = val
       else:
            temp = val
  if both i and temp initialized:
       emit(i,temp)
```

The idea is first to map each of the files into some kind of hash table (this is done internally by the framework), and you have two hash tables:

1. key=(x,y) value = id
2. key=(x,y) value = temprature

Once you have both hash tables, it is easy to find which id is connected to which temprature in a single pass, and once a connection is made -output it.

Complexity of this code is `O(n)` average case.

From SO, user amit link

Debugger Console

```
"C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\python.exe"
pydev debugger: process 4688 is connecting

Connected to pydev debugger (build 191.7141.48)
import sys; print('Python %s on %s' % (sys.version, sys.platform))
Python 3.6.6 |Anaconda, Inc.| (default, Jun 28 2018, 11:27:44) [MSC v.

Python 3.6.6 |Anaconda, Inc.| (default, Jun 28 2018, 11:27:44) [MSC v.
In[2]: %timeit geoLineDict.get('dp70whkzhvjqvxw9h9ku')
131 ns ± 2.34 ns per loop (mean ± std. dev. of 7 runs, 10000000 loops
```

The Great Trash Cart Debacle of 2018

Geocoder

```
# Single Line Geocoder Indiana Composite Locator
# Author: R. Stevens; B. Bond
import pandas
import requests
import time
from tqdm import tqdm


# Variables
inputTable = input("InputTable> ")
outputTable = input("Output Locaation> ")
addressColumn = "Address"  # from input table
idColumn = "ID"  # from input table
```

```python
def load_dataframe(input_table, address_column, id_column):
    result = [] # to hold results
    # Load CSV to Pandas Dataframe
    df = pandas.read_csv(input_table)

    if addressColumn not in df.columns:
        raise ValueError("Missing Address Column in input data")

    tqdm.write("DataFrame Created") # tqdm print
    # loop through pandas dataframe, zip fastest
    for uniqueID, address in tqdm(zip(df[id_column],
       df[address_column]), total=len(df.index),
       desc="Geocoding"):
        try:
            geocoded_result = geocode(address, uniqueID)
        except Exception as e:
            raise e
        else:
            result.append(geocoded_result)
    return result
```

```python
def geocode(address, id_num):
    geocode_url =
       "https://gis.in.gov/arcgis/rest/services/Indiana_Compos
       ite_Locator/GeocodeServer/findAddressCandidates?SingleL
       ine={}&f=json&outSR=4326&maxLocation=1&outFields=*".for
       mat(address)
    results = requests.get(geocode_url)
    results_dict = results.json()
    candidates = results_dict['candidates'][0]
    # Gather attributes and Sub Dicts
    location, attributes, score = candidates['location'],
       candidates['attributes'], candidates['score']
    # build output
    output = {
        "UniqueID": id_num,
        "input_string": address,
        "lat": location.get('y'),
        "lon": location.get('x'),
        "match_addr": attributes.get('Match_addr'),
        "match_type": attributes.get('Addr_type;'),
        "score": score
    }
    return output
```

```
def write_output(output_data, output_table):
    pandas.DataFrame(output_data).to_csv(output_table,
                    encoding='utf-8')
    print("CSV Created")

def main(input_table, address_column, id_column,
        output_table):
    t0 = time.time()
    results = load_dataframe(input_table,
       address_column, id_column)
    write_output(results, output_table)
    t1 = time.time() - t0
    print(t1)

if __name__ == '__main__':
    main(inputTable, addressColumn, idColumn,
      outputTable)
```

GdalTranslateEsri.py
geoHash.py
geoHash.spec
GPSAggregate.py
HRRetireeFormGenerator.py
Indiana_Geocoder_RyanOrig.py
IndianaGeocoder.py
IndianaGeocoderAsync.py
IndianaGeocoderConcurrent.py
IndianaGeocoderConcurrent2.py

```
    if addressColumn not in df.columns:
        raise ValueError("Missing Address Column in input data")

    print("DataFrame Created")  # use tqdm.write if progress bar messed up, its 5
    # for progress bar limits and worker count
    number_of_items = len(df.index)
    # Calculate Number of Workers any more than number of records to geocode is a
    workers = min(maxWorkers, number_of_items)
```

Run: IndianaGeocoder

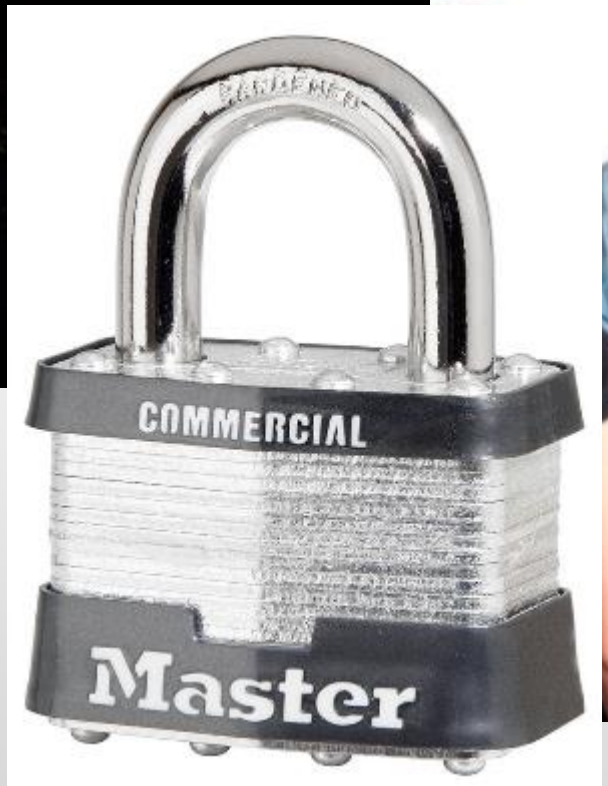
```
"C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\python.exe" L:/Scripts/Python3/IndianaGeocoder.py
```

4: Run | 6: TODO | 9: Version Control | Terminal | Python Console

4:16 CRLF


SLOTH
RUNNING TEAM
WE'LL GET THERE
WHEN WE GET THERE

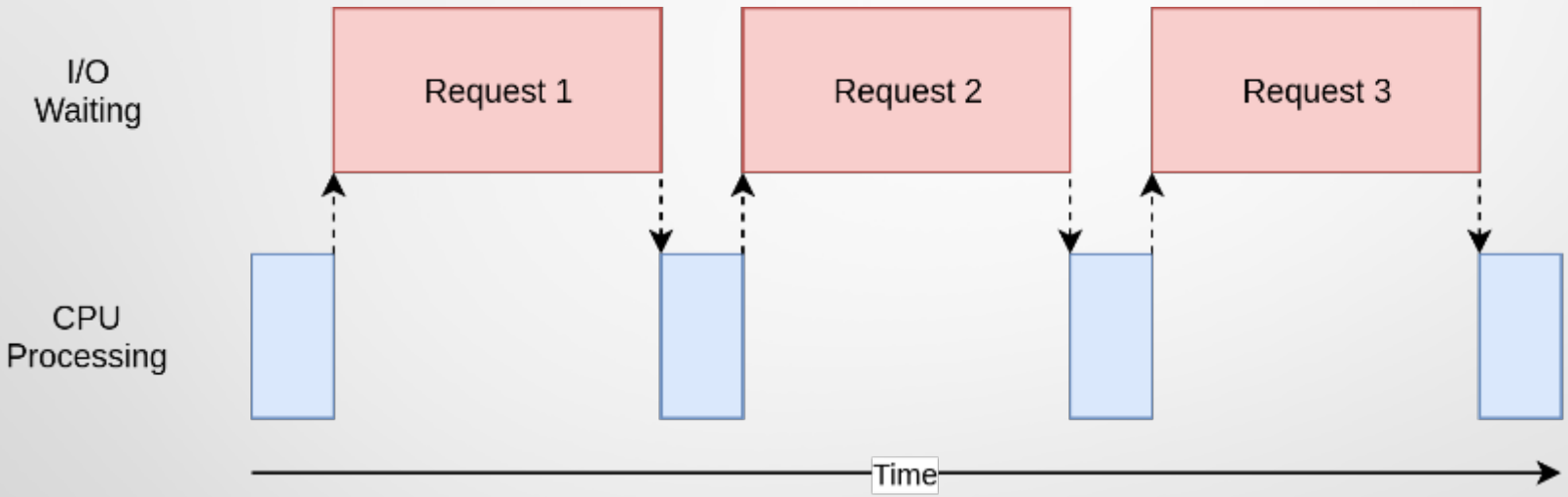

DEBUNKED


COMMERCIAL
Master

2 Types of Problems

- I/O bound
- CPU bound

I/O Bound


I/O
Waiting
CPU
Processing
Request 1
Request 2
Request 3
Time

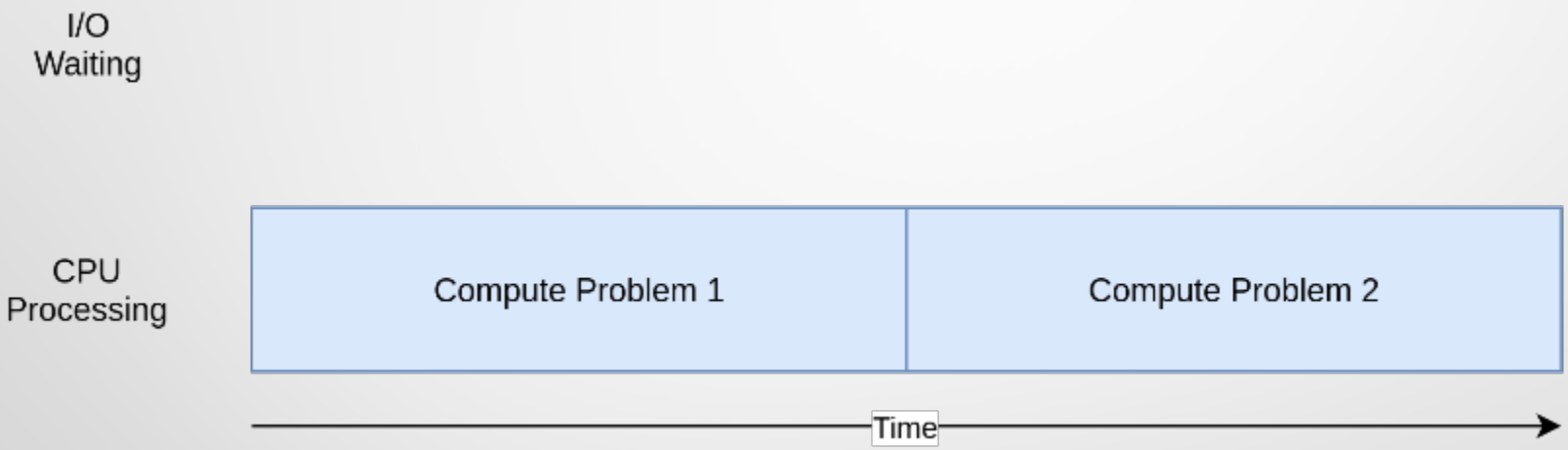
I/O Speeds

Device	CPU cycles	Proportional "human" scale
L1 cache	3	3 seconds
L2 cache	14	14 seconds
RAM	250	250 seconds
disk	41,000,000	1.3 years
network	240,000,000	7.6 years


Checklist

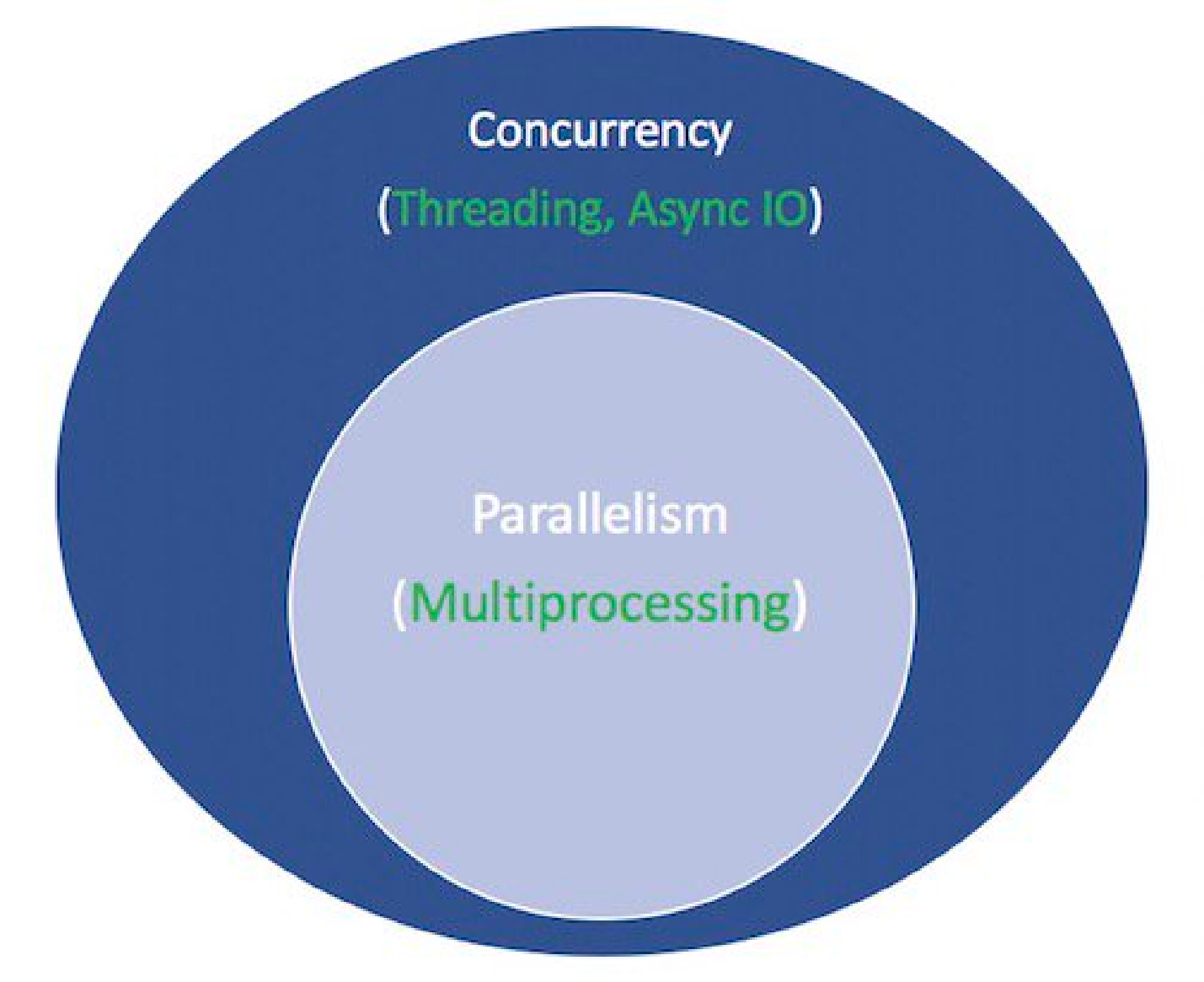

SHE SAID SHE'D BE
5 MINUTES

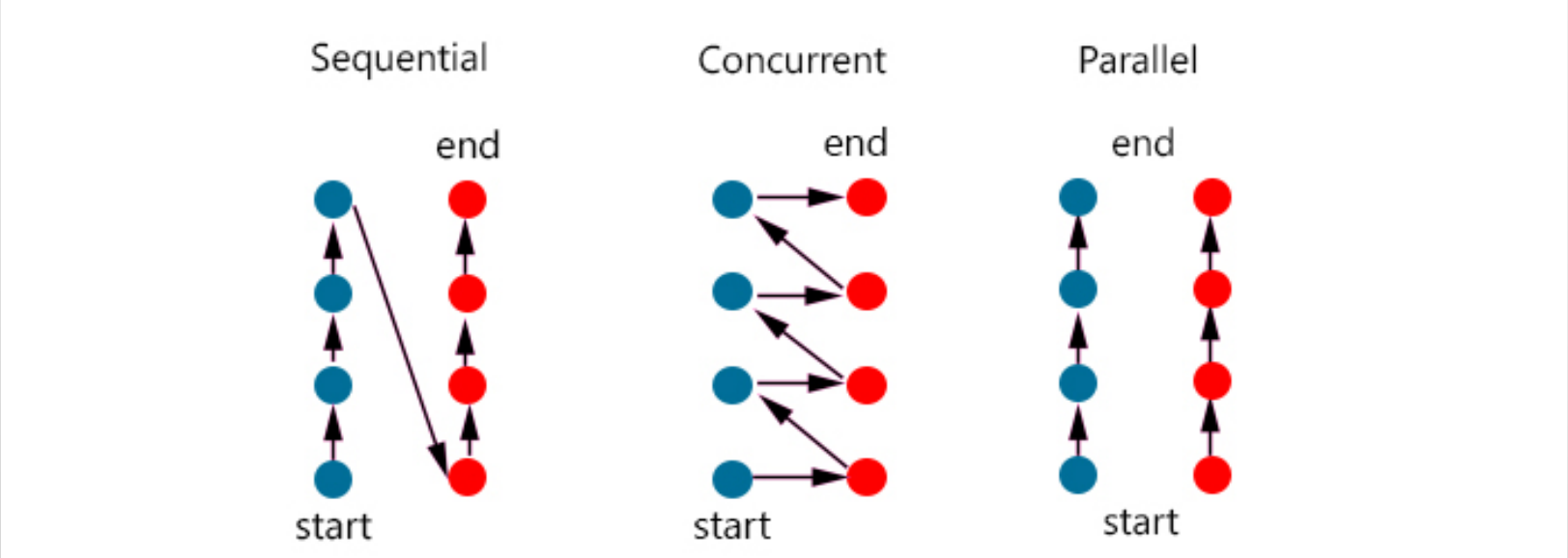
CPU Bound


I/O
Waiting
CPU
Processing
Compute Problem 1
Compute Problem 2
Time

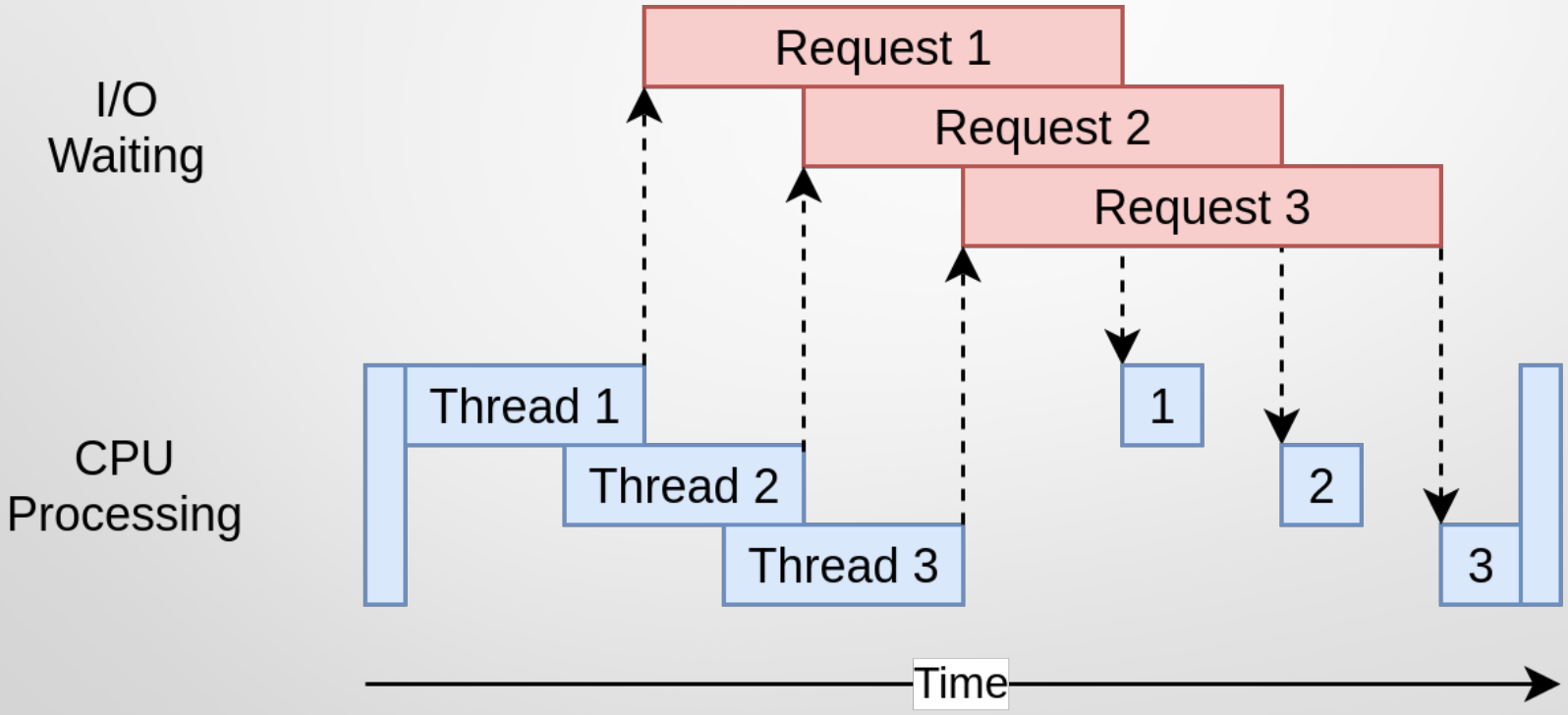
DID YOU KNOW?

To make your car go faster you can just shift into "race" gear after fifth.


Concurrency
(Threading, Async IO)
Parallelism
(Multiprocessing)

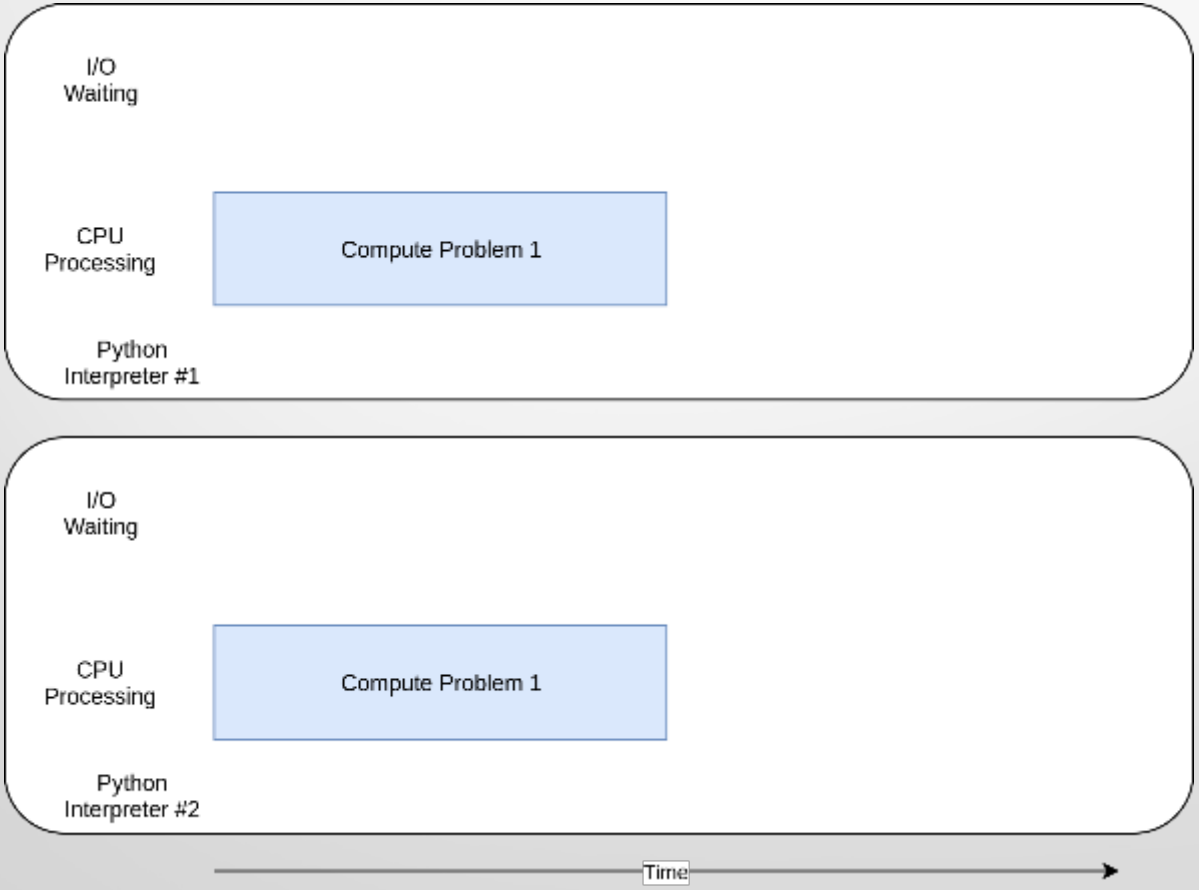

Sequential
end
start
Concurrent
end
start
Parallel
end
start

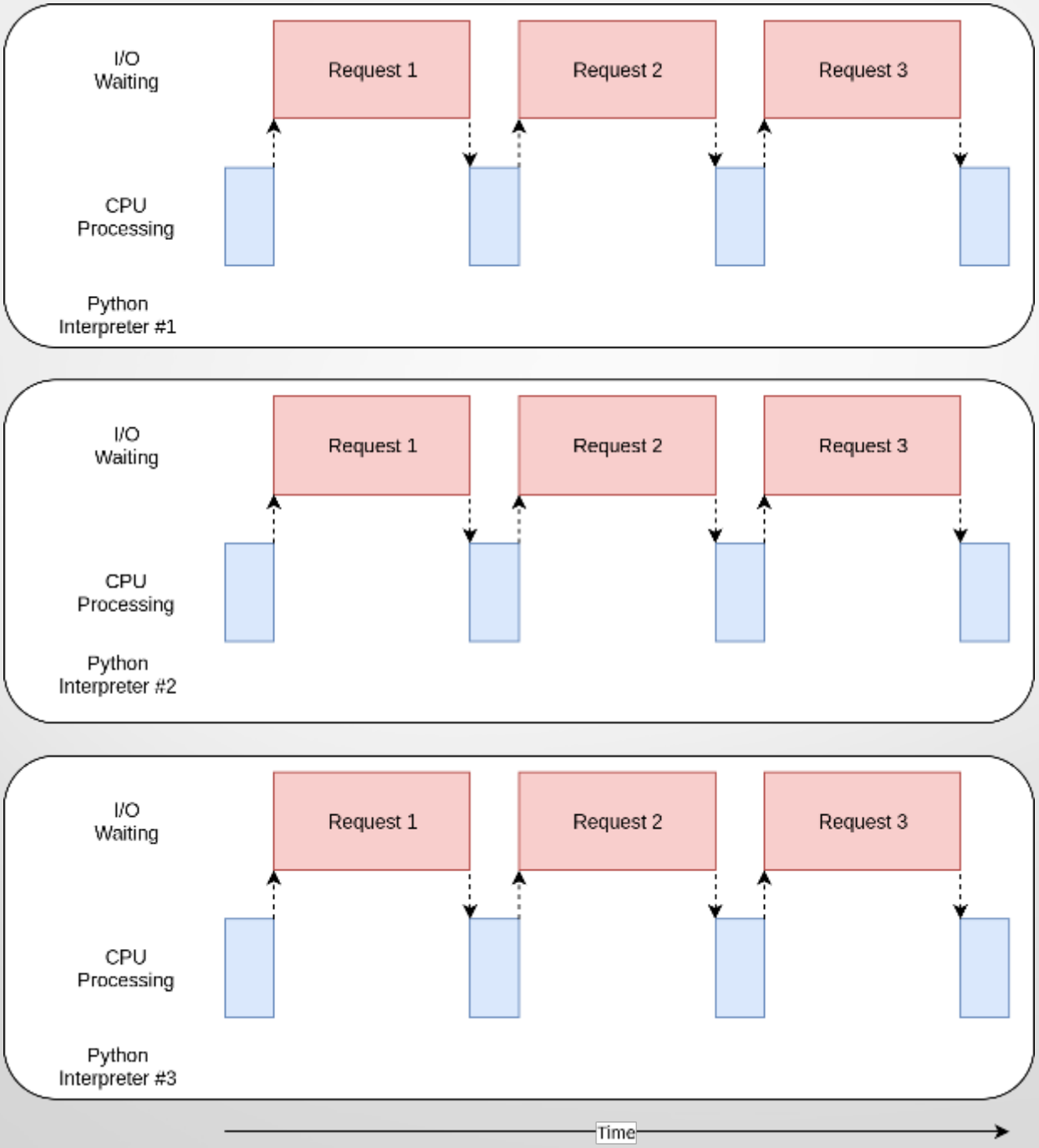
Multithreading




308

Multiprocessing




I/O
Waiting
Request 1
Request 2
Request 3
CPU
Processing
Python
Interpreter #1
I/O
Waiting
Request 1
Request 2
Request 3
CPU
Processing
Python
Interpreter #2
I/O
Waiting
Request 1
Request 2
Request 3
CPU
Processing
Python
Interpreter #3
Time

Quiz: How can we make the geocoder faster

- Buy it a Ferrari!
- Buy a bigger computer
- Multithreading (Concurrent)
- Asynchronous (Concurrent)
- Multiprocessing (Parallel)

Concurrent Package

- Used to be Hard, threading, multiprocessing
- Concurrent
  - Multihreading
    - Concurrent.futures.ThreadPoolExecutor(workers)
  - Multiprocessing
    - Concurrent.futures.ProcessPoolExecutor()
- 2 functions
  - Futures.map()
  - Executor.submit()

Multithreading (Map)

```
from concurrent import futures

maxWorkers = 20

def func():
    # do something to a single file or response

def concurrent_threading(process_list):
    workers = min(maxWorkers, len(process_list))
    with futures.ThreadPoolExecutor(workers) as executor:
        response = executor.map(func, process_list)
```

Multiprocessing (Map)

```
from concurrent import futures

def func():
    # do something to a single file or response

def concurrent_processing(process_list):
    with futures.ProcessPoolExecutor() as executor:
        response = executor.map(func, process_list)
```

Multithreading (submit)

```
from concurrent import futures

maxWorkers = 20

def func():
    # do something to a single file or response
    pass

def concurrent_threading(process_list):
    workers = min(maxWorkers, len(process_list))
    futures_list = []
    with futures.ThreadPoolExecutor(workers) as executor:
        for item_to_process in process_list:
            future = executor.submit(func, item_to_process)
            futures_list.append(future)

        done_iter = futures.as_completed(futures_list)
        for itm in done_iter:
            # Do stuff
```


You got a problem with
my driving, punk?

```
def geocode(address, id_num):
    # Indiana Composite Geocoder URL
    geocode_url = "https://gis.in.gov/arcgis/rest/services/
        Indiana_Composite_Locator/GeocodeServer/findAddressCandidates?S
        ingleLine={}&f=json&outSR=4326&maxLocation=1&outFields=*".format(
        address)
    results = requests.get(geocode_url)
    results_dict = results.json()
    # Return ID num, address, and results as tuple
    res = (id_num, address, results_dict)
    return res

def process_results(id_num, address, results_dict):
    # Process reusults to dict
    candidates = results_dict['candidates'][0]
    location, attributes, score = candidates['location'],
        candidates['attributes'], candidates['score'] # Get sub attr

    output = {
        "UniqueID": id_num,
        "input_string": address,
        "lat": location.get('y'),
        "lon": location.get('x'),
        "match_addr": attributes.get('Match_addr'),
        "match_type": attributes.get('Addr_type'),
        "score": score
    }
    return output
```

```python
def load_dataframe(input_table, address_column, id_column):
... # Code folded, same as sequential

    # Calculate Number of Workers, waste for workers > addresses
    workers = min(maxWorkers, number_of_items)
    # Start the Thread pool
    with futures.ThreadPoolExecutor(workers) as executor:
        to_do_map = {} # Dictionary to hold futures
        for uniqueID, address in
             zip(df[id_column], df[address_column]):
            # Submit tasks to thread and store futures
            future = executor.submit(geocode, address, uniqueID)
            to_do_map[future] = uniqueID
        # Move to completed list as they are finished
        done_iter = futures.as_completed(to_do_map)
        # Progress Bar (wraps an iterator)
        done_iter = tqdm(done_iter, total=number_of_items,
                                 desc="Geocoding")
        # Process responses from geocoder
        for future in done_iter:
            # variable unpacking frm geocode(id, addr, result)
            output = process_results(*future.result())
            result.append(output)
    return result
```

GdalTranslateEsri.py
geoHash.py
geoHash.spec
GPSAggregate.py
HRRetireeFormGenerator.py
Indiana_Geocoder_RyanOrig.py
IndianaGeocoder.py
IndianaGeocoderAsync.py
IndianaGeocoderConcurrent.py
IndianaGeocoderConcurrent2.py

```
23      if addressColumn not in df.columns:
24          raise ValueError("Missing Address Column in input data")
25
26      print("DataFrame Created")  # use tqdm.write if progress bar messed up, its 5
27      # for progress bar limits and worker count
28      number_of_items = len(df.index)
29      # Calculate Number of Workers any more than number of records to geocode is a
30      workers = min(maxWorkers, number_of_items)
31
```

Run: IndianaGeocoderConcurrent2

```
"C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\python.exe" L:/Scripts/Python3/IndianaGeocoderConcurrent2.p
```

4: Run | 6: TODO | 9: Version Control | Terminal | Python Console

14:20 CRLF

CONTINUE?

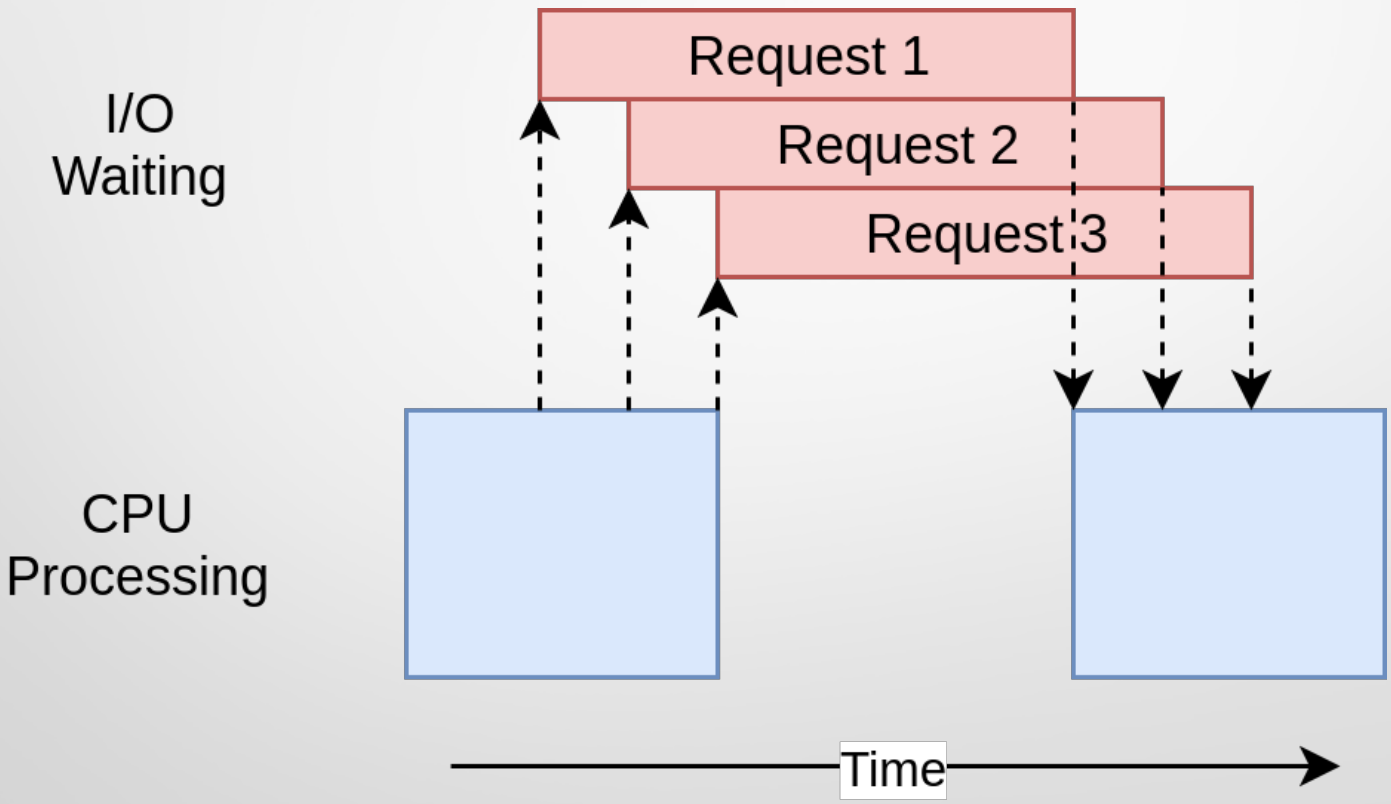
YES/NO

CREDIT 0


time
machine
Doritos

ASYNCIO

Asynchronous



GdalTranslateEsri.py
geoHash.py
geoHash.spec
GPSAggregate.py
HRRetireeFormGenerator.py
Indiana_Geocoder_RyanOrig.py
IndianaGeocoder.py
IndianaGeocoderAsync.py
IndianaGeocoderConcurrent.py
IndianaGeocoderConcurrent2.py

```
26  # for progress bar limits and worker count
27  number_of_items = len(df.index)
28
29  loop = asyncio.get_event_loop()
30  # loop.set_debug(True)
31  geocoro = async_geocode(df, number_of_items)
32  # result = loop.run_until_complete(async_geocode(df, number_of_items))
33  geocoded = loop.run_until_complete(geocoro)
34  loop.close()
```

Run: IndianaGeocoderAsync

```
"C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\python.exe" L:/Scripts/Python3/IndianaGeocoderAsync.py
```

7: Structure
2: Favorites

4: Run | 5: Debug | 6: TODO | 9: Version Control | Terminal | Python Console

Push successful: Pushed 1 commit to origin/master (41 minutes ago)

16:1 CRLF

```python
def load_dataframe(input_table, address_column, id_column):
    result = []
    # Load CSV to Pandas Dataframe
    df = pandas.read_csv(input_table)
    # Check Columns
    if addressColumn not in df.columns:
        raise ValueError("Missing Address Column in input data")

    print("DataFrame Created")  # use tqdm.write if bar is funky
    # for progress bar limits and worker count
    number_of_items = len(df.index)

    # Get reference to event loop
    loop = asyncio.get_event_loop()
    # loop.set_debug(True)  # for debugging assistance with Ayncio
    geocoro = async_geocode(df, number_of_items)
    # run async_geocode till its complete
    geocoded = loop.run_until_complete(geocoro)
    # close the loop
    loop.close()

    # Process the responses from geocoder, same way as before
    for resp in geocoded:
        # variable unpacking from geocode(id_num, address, result_dict)
        output = process_results(*resp)
        result.append(output)
    return result
```

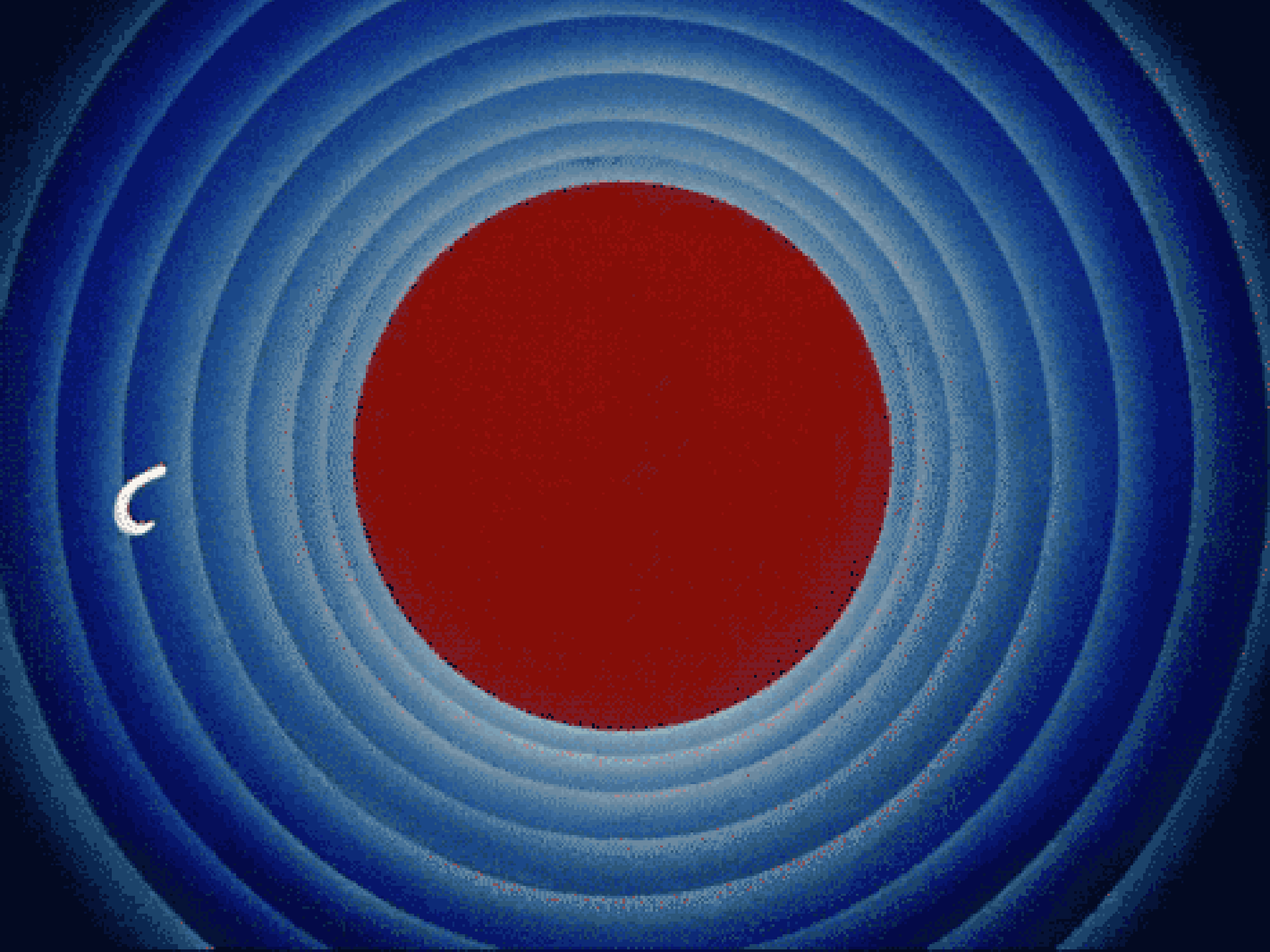
```python
async def async_geocode(df, number_of_items):
    result = []
    async with aiohttp.ClientSession() as session:

        to_do = [geocode(address, unique_id, session)
                for unique_id, address,
                in zip(df[idColumn], df[addressColumn])]
        to_do_iter = asyncio.as_completed(to_do)
        done_iter = tqdm(to_do_iter,
                         total=number_of_items, desc="Geocoding")
        for future in done_iter:
            res = await future
            result.append(res)
    return result

async def geocode(address, id_num, session):
    # Indiana Composite Geocoder URL
    geocode_url = "https://...”
    async with session.get(geocode_url) as resp:
        # Content returned from server is wrong type, override below.
        result = await resp.json(content_type="text/plain;charset=utf-8")
        results_dict = result
        res = (id_num, address, results_dict)
        return res
```

Food for Thought

- Know your problem
- Make sure you need to speedup your script
- Write sequentially before going concurrent or parallel



Questions?

Ben.bond@huntington.in.us