

Lab 1: Bash, Git and Python Warm-Up

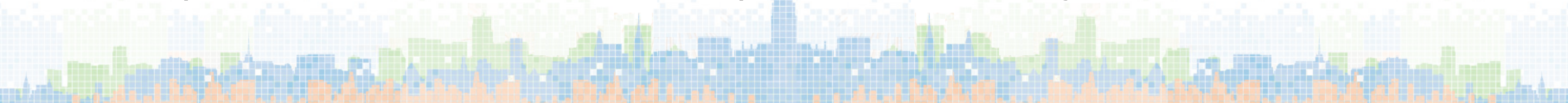


Objectives

Given one or more text files, where each contains a list of numbers line by line, compute the median value of all the lists combined.

Familiarization with online data portals and data representation

Hands-on experiences with static feed, URL request and JSON manipulation in Python



Setting Up Your Workspace

- Make sure you have Bash, Git and Python setup correctly

```
bash$ which python  
/usr/bin/python
```

```
bash$ which git  
/usr/bin/git
```

```
bash$ which curl  
/usr/bin/curl
```



Task 1: generate test input files

- Generate two test files containing continuous numbers from 1 to 99 and 100 to 199: select any method (including what Google tells you)

```
bash$ echo {1..99}
```

```
1 2 ...
```

```
bash$ echo {1..99} | tr " " "\n"
```

```
1
```

```
2
```

```
...
```

```
bash$ echo {1..99} | tr " " "\n" > input1.txt
```


Task 2: find the median of each file

- Use any methods
 - Excel?
 - Python script:

```
bash$ python find_median.py input1.txt  
50
```

```
bash$ python find_median.py input2.txt  
150
```




Task 3: find the median of both files

```
bash$ python find_median.py input1.txt  
50
```

```
bash$ python find_median.py input2.txt  
150
```

```
bash$ python find_median.py input1.txt input2.txt  
100
```

A decorative pixelated city skyline with various colored buildings (blue, green, orange) along the bottom edge of the slide.

Task 4: checkout the data and repo
<https://github.com/hvo/puilab2.git>

branch: median

Test with the data files in Task 3!



Task 5: Run on the new data sets

```
bash$ python find_median.py data/*/numbers.txt  
17089
```



Task 6: Write a program to find the average value of the lists

```
bash$ python find_average.py data/*/numbers.txt  
17074.4747475
```



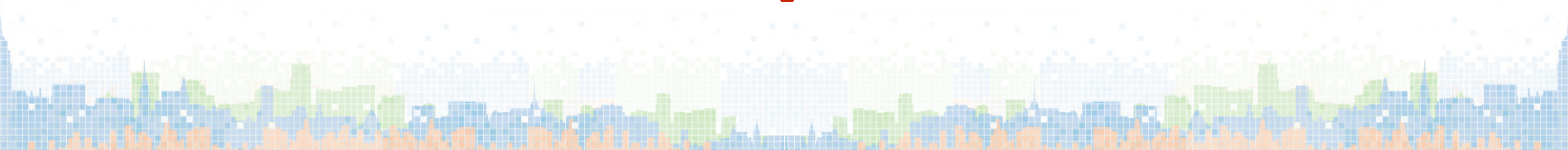
Task 7: Bash pipes

- Write a single piped Bash's command chain that computes the min or the max of the list in the data files!



Task 8: access metadata from Web UI

- Introducing NYC OpenData : <https://nycopendata.socrata.com>
 - A comprehensive data dump (more than needed) of open data in NYC
 - Each data has a unique handle
 - Metadata is included with each data
 - Data can be exported into CSV, XLS, etc.
- Given a data set handle **f9bf-2cp4** search for its creation date



Task 9: access metadata through API

- Find out the creation date of another data set **h9gi-nx95**
 - Tedious task to search, then several more clicks
- Let's do this through the API
 - metadata are available at:
<https://nycopendata.socrata.com/views/h9gi-nx95>
 - and are in JSON format! (enter the URL into your browser and notice the *createdAt* field)

Task 9 — continued

- Fetch the metadata to a file:

```
curl https://nycopendata.socrata.com/views/h9gi-nx95 > metadata.json
```

- Write a Python script, *task9.py*, to output the “*created at*” time as a human-readable string:

```
bash$ python task9.py metadata.json  
2014-04-28 12:41:44
```



Task 10

- Instead of using curl to download the metadata first, add to the Python script the ability to download this data given a data handle as well

```
bash$ python task10.py h9gi-nx95  
2014-04-28 12:41:44
```

```
bash$ python task10.py f9bf-2cp4  
2013-02-20 22:28:53
```



Task 11

- Find the list of CitiBike stations that are not in service (any station that has the status key of 3):

http://www.citibikenyc.com/stations/status_json

- Fetch the current feed of the stations and save to stations.json:

- <https://www.citibikenyc.com/stations/json>

- Write a Python script to list list all the station names and locations:

```
bash$ python task11.py stations.json  
S 4 St & Rodney St : 40.70934,-73.95608
```

...

Task 12

- Make the script in Task 11 output to a CSV file instead

```
bash$ python task12.py stations.json notinservice.csv
```

- And load `notinservice.csv` onto CartoDB
- Make sure to output a proper header for CartoDB to auto-recognize the geo-referencing columns

