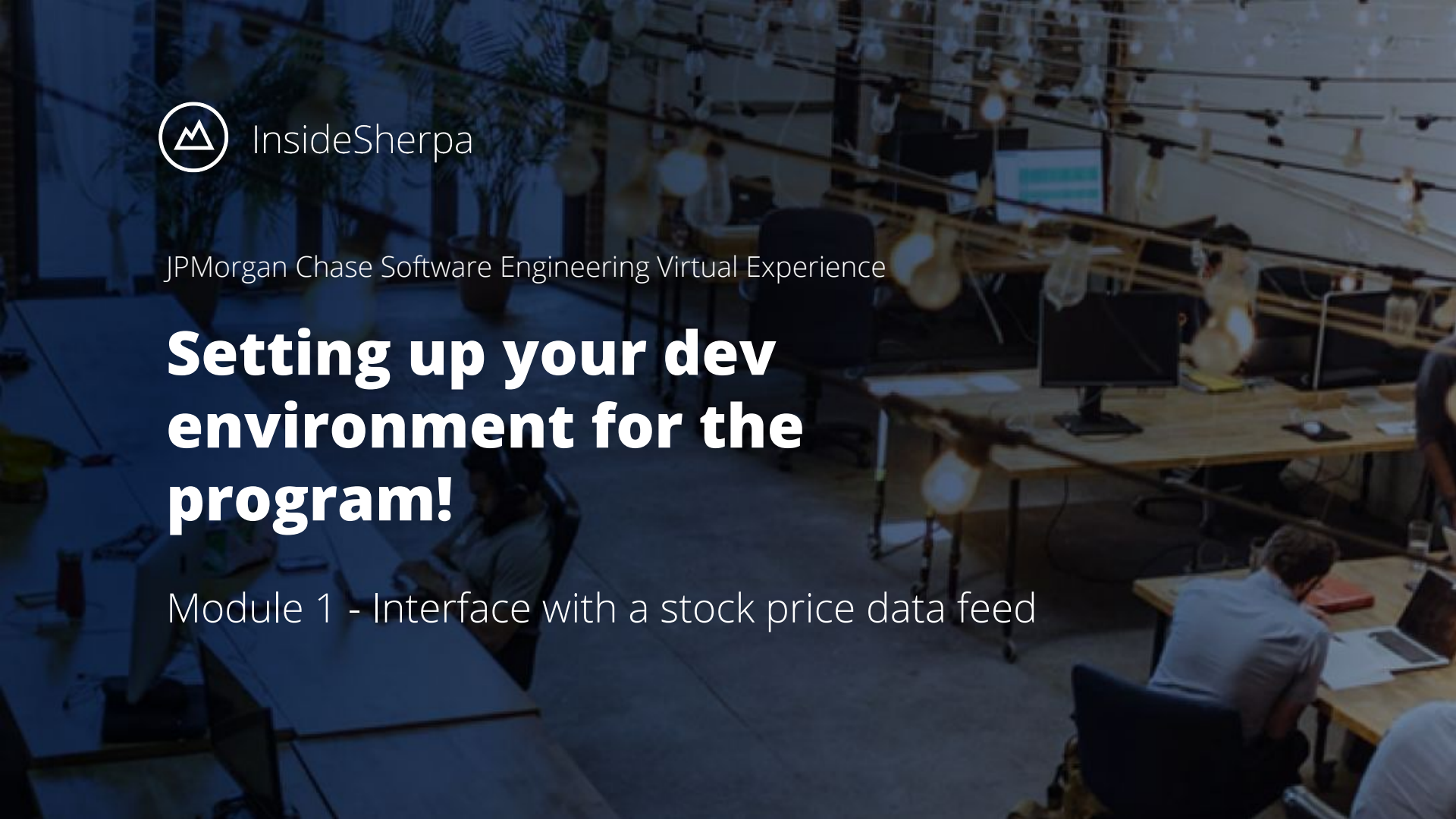


InsideSherpa

JPMorgan Chase Software Engineering Virtual Experience

Setting up your dev environment for the program!

Module 1 - Interface with a stock price data feed

We know your first time using Python, or setting up a web development environment at work, might be daunting.

Or feel like it uses technologies you haven't used before, or might feel like it takes too long.

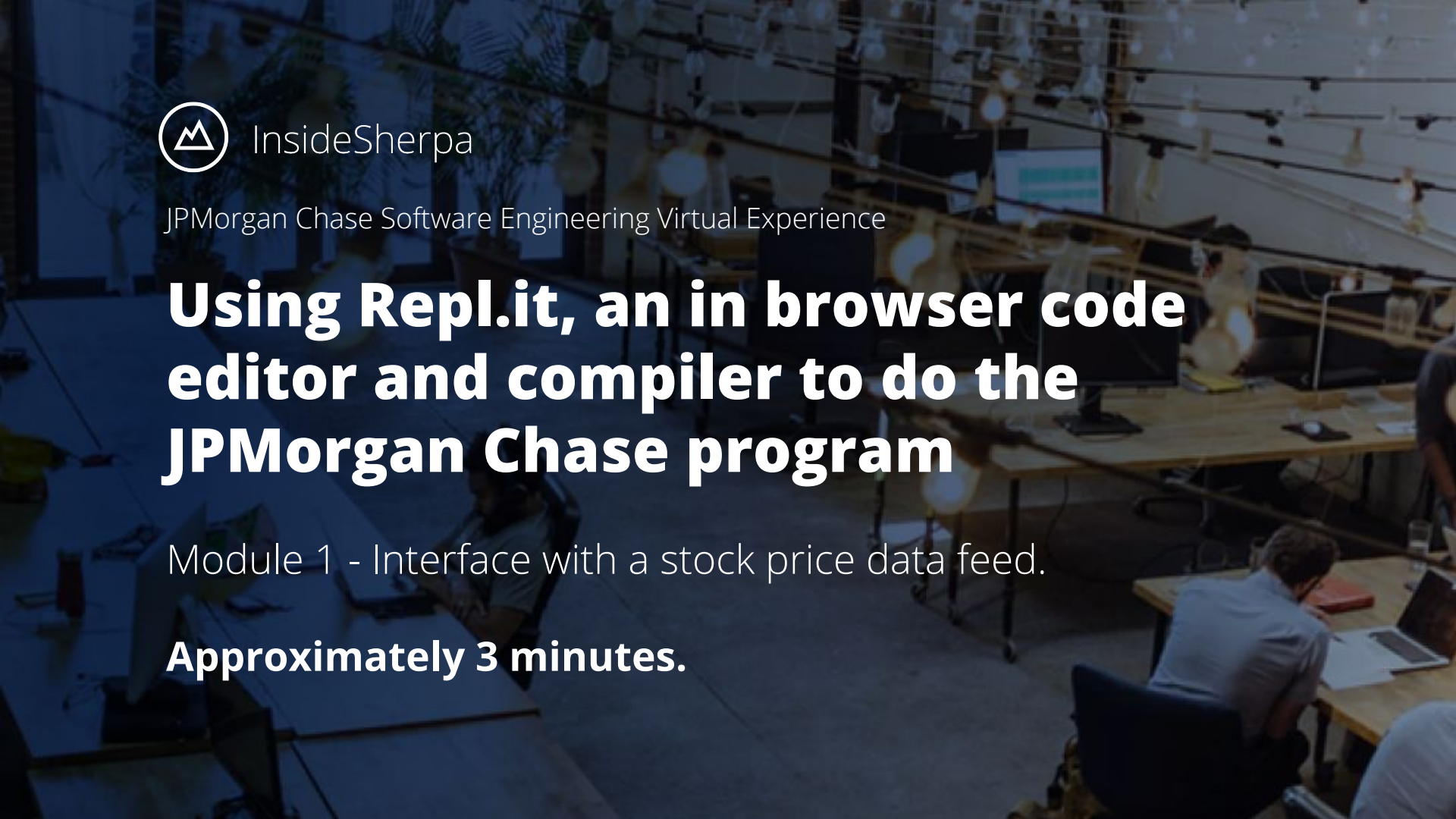
So to help you out we've created this step-by-step guide to setting up your computer for this task.

A lot of the things you do here, you will also do when you set yourself up at an in-office internship too. Look like an amazing hire when you breeze through dev environment setup!

With this guide, the approximate time to get a development environment working for you is **10 minutes.**

To start, choose the application environment based on your device & current skill level

REPL (best if you have not set up a dev environment before)	(Mac) Setting up your dev environment for task 1
(Windows) Setting up your dev environment for task 1	(Linux) Setting up your dev environment for task 1



InsideSherpa

JPMorgan Chase Software Engineering Virtual Experience

Using Repl.it, an in browser code editor and compiler to do the JPMorgan Chase program

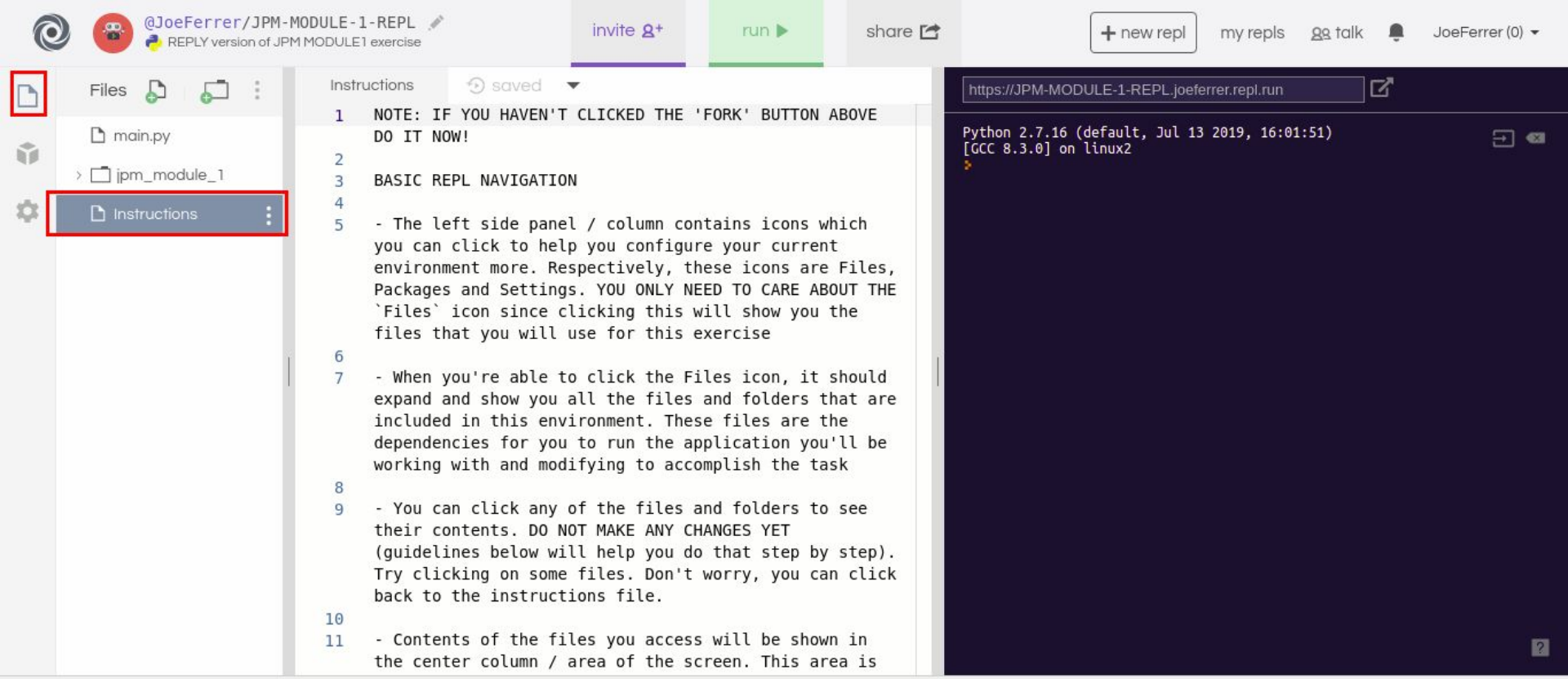
Module 1 - Interface with a stock price data feed.

Approximately 3 minutes.

Using REPL

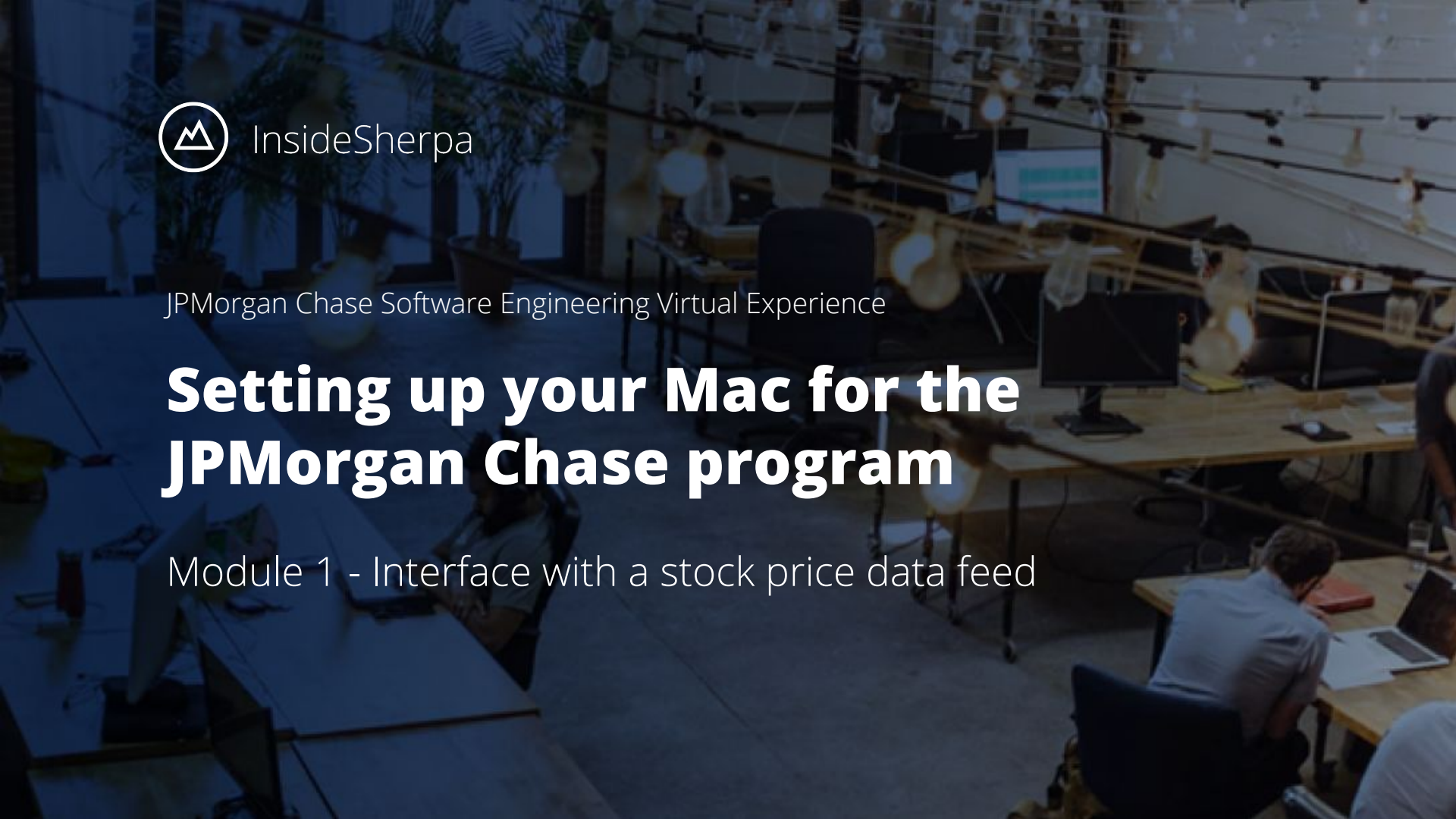
- REPL is an online coding platform that developers can use to run simulated applications / tests without having to worry about installing dependencies on their local machines

- For “Module 1 - Interface with a stock price data feed”, we’ve set up two REPL environments: Python2 Env and Python3 Env (*click on the link of the environment you want to use and you should end up on a page like the one shown in the next slide)*



- To get started, read thru the “Instructions” file in the REPL by clicking on the “Instructions” file on the left hand side of the screen

- To make the files show, you must click the “File” icon. (*also boxed in red*)



InsideSherpa

JPMorgan Chase Software Engineering Virtual Experience

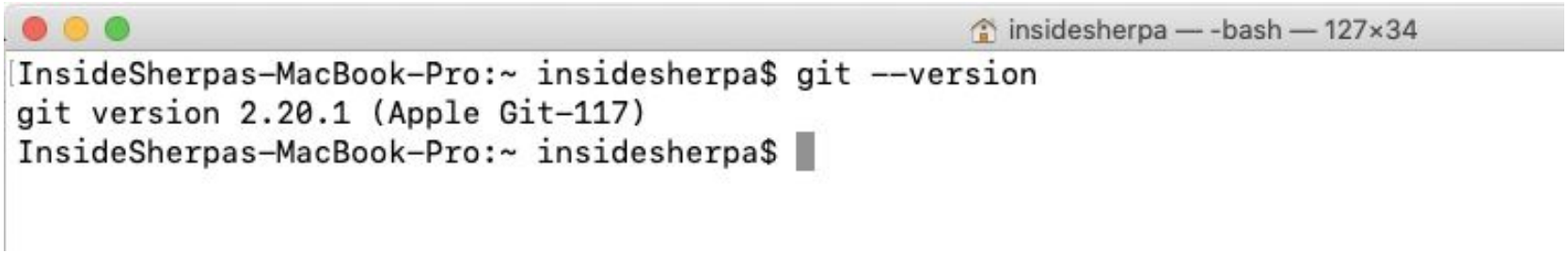
Setting up your Mac for the JPMorgan Chase program

Module 1 - Interface with a stock price data feed

Local Setup (Mac)

- Use this method if you chose not to use the REPL method.
  If your machine is running on Mac, follow this setup guide to get started.

- First you must have git installed in your system. Git is used by most programmers today to collaborate with code/software projects. To install git, follow this quick guide. You know you have installed successfully when you get a version output on your terminal by typing `git --version`:



Local Setup (Mac)

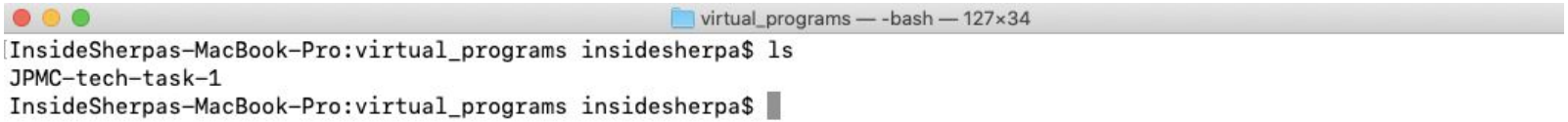
- Once you have git installed, you need a copy of the application code you'll be working with on your machine. To do this, you must execute the following commands on your terminal:

```
git clone https://github.com/insidesherpa/JPMC-tech-task-1.git
git clone https://github.com/insidesherpa/JPMC-tech-task-1-py3.git
```

- This command will download the code repositories from github to your machine in the current working directory of the terminal you executed the command in. Downloading the 2 repositories above will give you options later

Local Setup (Mac)

- You'll know you cloned successfully if you have the copy of the application code on your machine:



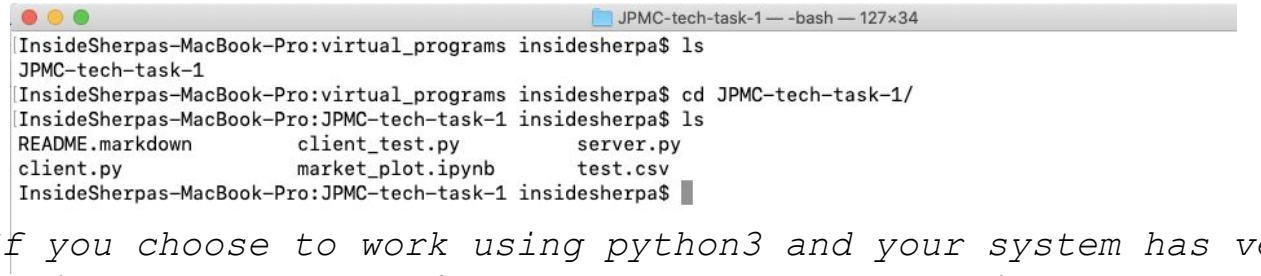
*note: the image above and in the next slide just does not contain the other repository but it should if you did the previous slides and execute the **ls** command. `ls` just lists the files/folders in the current directory*

Local Setup (Mac)

- To access the files inside from the terminal, just change directory by typing the following commands:

cd JPMC-tech-task-1

ls



note: If you choose to work using python3 and your system has version python3 or above instead of python2.7.x, then choose to go into the other repository you downloaded instead. (otherwise, use the other repo above)

cd JPMC-tech-task-1-py3

note: `cd` means change directory. `ls` lists contents in the current directory. *Check this for more info on* ***cd***

Local Setup (Mac)

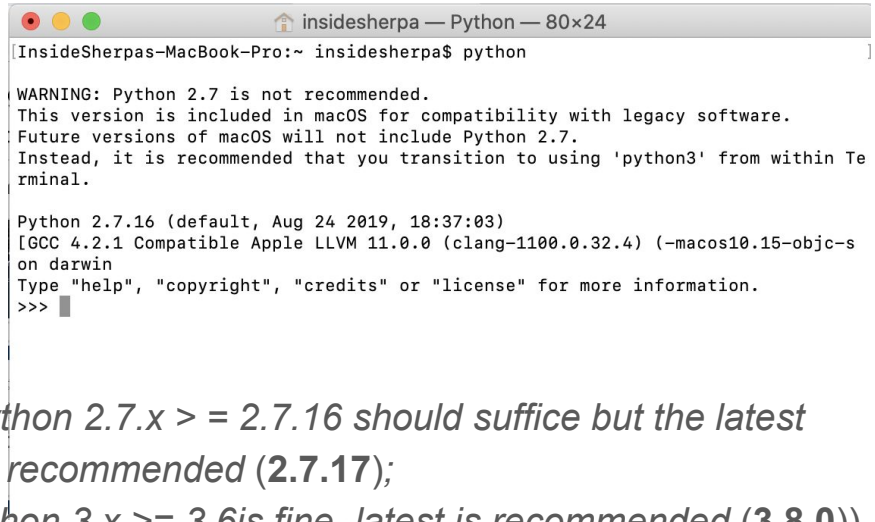
- To clarify, you're only supposed to work on one of the repositories you cloned / downloaded into your system. It all depends on what Python version you primarily use.

- Python is just a scripting / programming language we developers use quite often in the field. This application you'll be working on uses it.

- We'll discuss checking / installing Python in your system in the following slides

Local Setup (Mac)

- Next, you'll need to have Python 2.7 or Python 3 installed on your machine. Follow the instructions here(python 2.7) or here (python 3) You can verify this on your terminal if you get a result like:



(*any python 2.7.x > = 2.7.16 should suffice but the latest 2.7.x is recommended* (**2.7.17**)*;*
any python 3.x >= 3.6is fine, latest is recommended (**3.8.0**))

Execute the command below to verify what version you have:

```
python --version
```

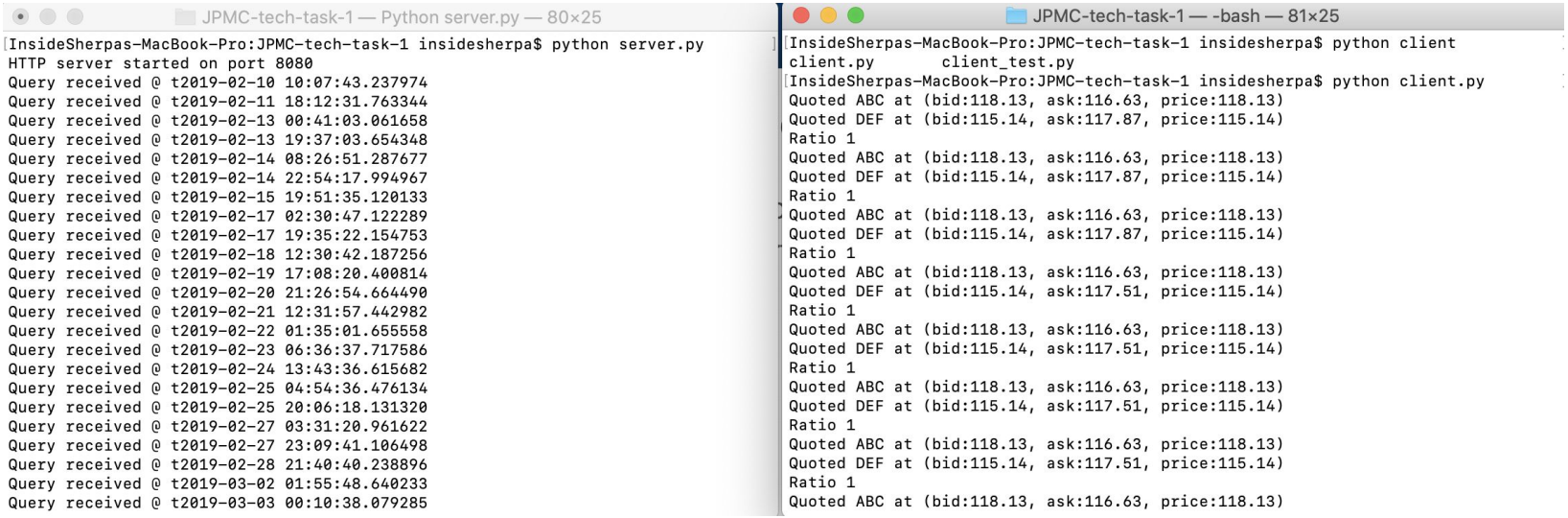
Note: the image here is only of 2.7 but it should be similar if you check for python3

Sometimes your system might have it as

```
python3 --version
```

Local Setup (Mac)

- Once you have Python 2.7 or Python3 installed, all you have to do get the application up and running is to start the server and client scripts in two separate terminals.



Local Setup (Mac)

- (*note: just choose to run one server and one client; either the python 2 or python 3 version of server and client applications*. Run the **commands** below on separate terminals, starting with the server and then the client. The commands will vary depending on your primary python version)

// If python --version = 2.7+, you must be in the JPMC-tech-task-1
// If python --version = 3+ , you must be in JPMC-tech-task-1-py3 directory

```
python server.py
python client.py
```

// If your system makes the distinction of python3 as `python3`,
// you must be in JPMC-tech-task-1-py3 directory

```
python3 server3.py
python3 client3.py
```

If ever you encounter an error when starting the server application, see troubleshooting in this slide

Local Setup: Troubleshooting (Mac)

- If you did not encounter any issues, your setup is finished. From here on, you can make changes to the code (see another guide in the module page for this) and eventually arrive at the desired output.

- If you did encounter issues, check if the commonly encountered issues listed in the next few slides will solve your problem:

    - dateutil_dependency
    - Socket unavailable

Local Setup: Troubleshooting (Mac)

- In some cases, dependency issues might arise like when you run `server.py`:

```
Traceback (most recent call last):
  File "server.py", line 26, in <module>
    import dateutil.parser
ImportError: No module named dateutil.parser
```

In this case, you must install pip first. pip is python's package manager for installing python dependencies. Make sure you install pip for the right Python version you're working with in this project. You can check your pip version by **pip --version** and it will tell which python version it maps too

Local Setup: Troubleshooting (Mac)

- Installing pip nowadays usually involves downloading the get-pip.py script. If you followed the instructions in the last slide, it usually involves using the command:

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
```

If you don't have curl, just install it in your system. For mac, it's this way
Then just run the script using python:

```
//if python --version = 2.7+ this will install pip for python2
//if python --version = 3+ this will install pip for python3
python get-pip.py

//if your system makes the distinction of python3 as `python3` then
//doing the command below will install pip for python3
python3 get-pip.py
```

Local Setup: Troubleshooting (Mac)

- Then afterwards, you can run the following command on your terminal to install the dependency:

```
pip install python-dateutil
```

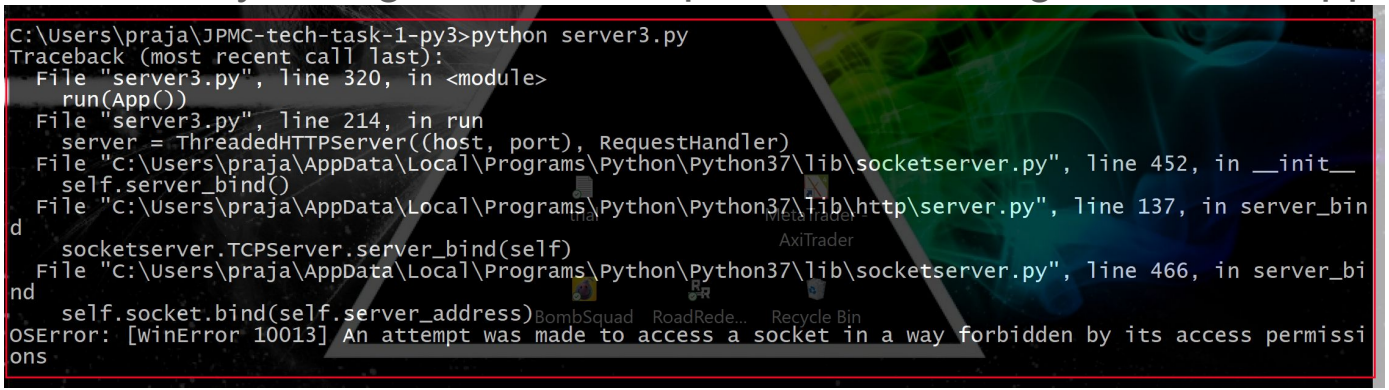
Afterwards, you can rerun the server and then rerun the client

Note: For the command above, whatever python version your pip corresponds to (i.e. the output of **pip --version**, that is the python version that will have the dependency installed). So if you're **pip** corresponds to python2.7.x then doing the command above will install python-dateutil for python2.7.x

Local Setup: Troubleshooting (Mac)

- In other cases, you might encounter problems running the server app

CASE A:



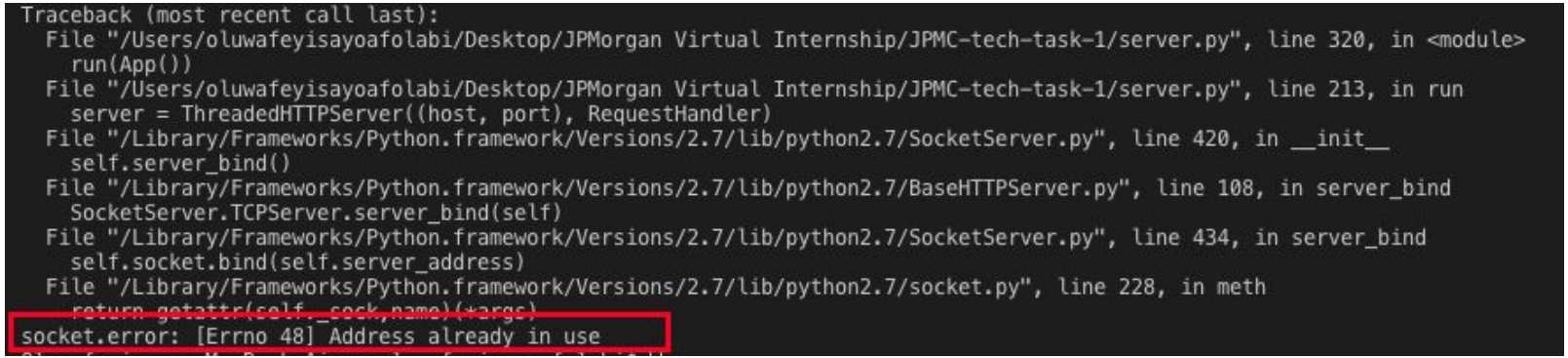
note: the example here is from windows but a similar error might appear for mac

This is most likely because you have a firewall open preventing you from accessing 8080. You can try the following workarounds:

- Temporarily turn off your firewall
- Using any text editor, open the server.py or server3.py in the repository using your code editor and look for the line where it says port = 8080. change that to port = 8085
- Similarly, open the client.py or client3.py and change the line where it has 8080 to 8085

Local Setup: Troubleshooting (Mac)

CASE B:

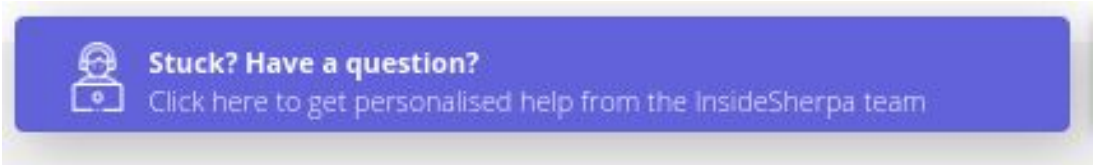


In this case make sure you're only running one instance of the server.py because it hooks itself to port 8080, and once that port is used nothing else can use it. If you want to free that up, terminate the old server.py you're running from one of your terminals by hitting cmd+c. Alternatively you can kill the process listening on a port (i.e. in this case 8080) by following this guide

Local Setup: Troubleshooting (Mac)

- If you did encounter any other issues, please post your issue/inquiry here: https://github.com/insidesherpa/JPMC-tech-task-1/issues or https://github.com/insidesherpa/JPMC-tech-task-1-py3/issues depending on what repository you chose to work in. When submitting a query, please don't forget to provide as much context as possible, i.e. your OS, what you've done, what your errors is/are, etc (screenshots would help too)

- You can also submit your query in the module page's support modal that pops out when you click the floating element on the page (see image below)

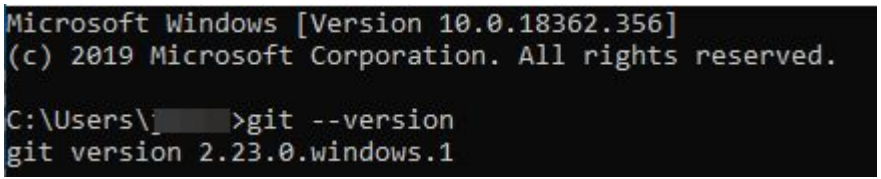



InsideSherpa
JPMorgan Chase Software Engineering Virtual Experience
Setting up your Windows for the JPMorgan Chase program
Module 1 - Interface with a stock price data feed

Local Setup (Windows)

- Use this method if you chose not to use the REPL method.
  If your machine is running on Windows, follow this setup guide to get started (*the examples here are on Windows X but it should be relatively similar for other versions*)

- First you must have git installed in your system. Git is used by most programmers today to collaborate with code/software projects.To install git, follow this quick guide. You know you have installed successfully when you get this output on your command line (cmd). (*any git version should suffice but the latest is recommended)*



Local Setup (Windows)

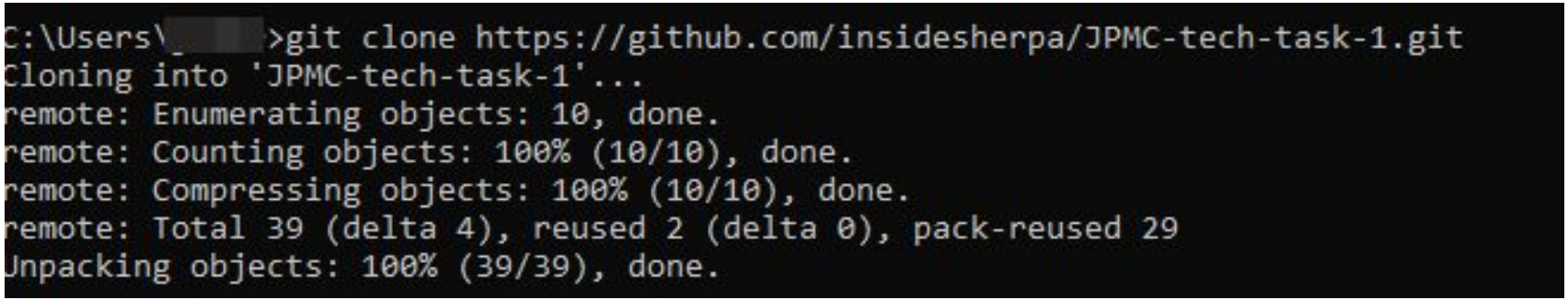
- Once you have git installed, you need a copy of the application code you'll be working with on your machine. To do this, you must execute the following command on your terminal:

```
git clone https://github.com/insidesherpa/JPMC-tech-task-1.git
git clone https://github.com/insidesherpa/JPMC-tech-task-1-py3.git
```

- This command will download the code repositories from github to your machine in the current working directory of the terminal you executed the command in. Downloading the 2 repositories above will give you options later

Local Setup (Windows)

- You'll know you cloned successfully if you have the copy of the application code on your machine:

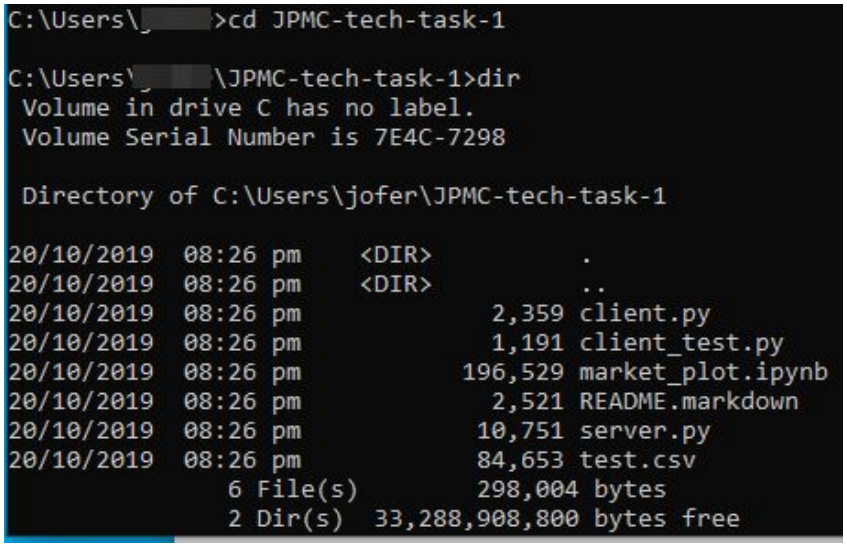


*note: the image above just does not contain the other repository but it should if you did the previous slides and execute the **dir** command. `dir` will list the contents of the current directory*

Local Setup (Windows)

- To access the files inside from the terminal, just change directory by typing:

cd JPMC-tech-task-1



note: `cd` means change directory (more on cd)

*note: The image on this slide just does not contain the other repository but it should if you did the previous slides and execute the **dir** command.*

note: If you choose to work using python3 and your system has version python3 or above instead of python2.7.x, then choose to go into the other repository you downloaded instead. (otherwise, use the other repo above)

cd JPMC-tech-task-1-py3

Local Setup (Windows)

- To clarify, you're only supposed to work on one of the repositories you cloned / downloaded into your system. It all depends on what Python version you primarily use.

- Python is just a scripting / programming language we developers use quite often in the field. This application you'll be working on uses it.

- We'll discuss checking / installing Python in your system in the following slides

Local Setup (Windows)

- Next, you'll need to have Python 2.7 or Python3 installed on your machine. Follow the instructions here (for python2), or here (for python3) You can verify this on your command line (cmd) if you get a result like:

(*any python 2.7.x > = 2.7.16 should suffice but the latest 2.7.x is recommended* (**2.7.17**)*;*
any python 3.x >= 3.6is fine, latest is recommended (**3.8.0**))

Execute the **command** below to verify what version you have:

```
python --version
```

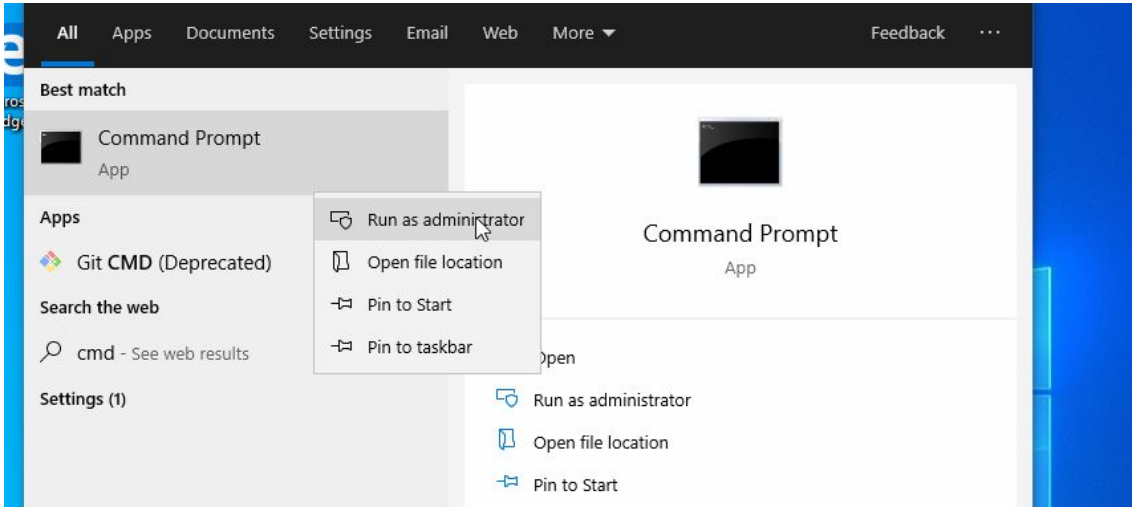
Note: the image here is only of 2.7 but it should be similar if you check for python3

Sometimes your system might have it as

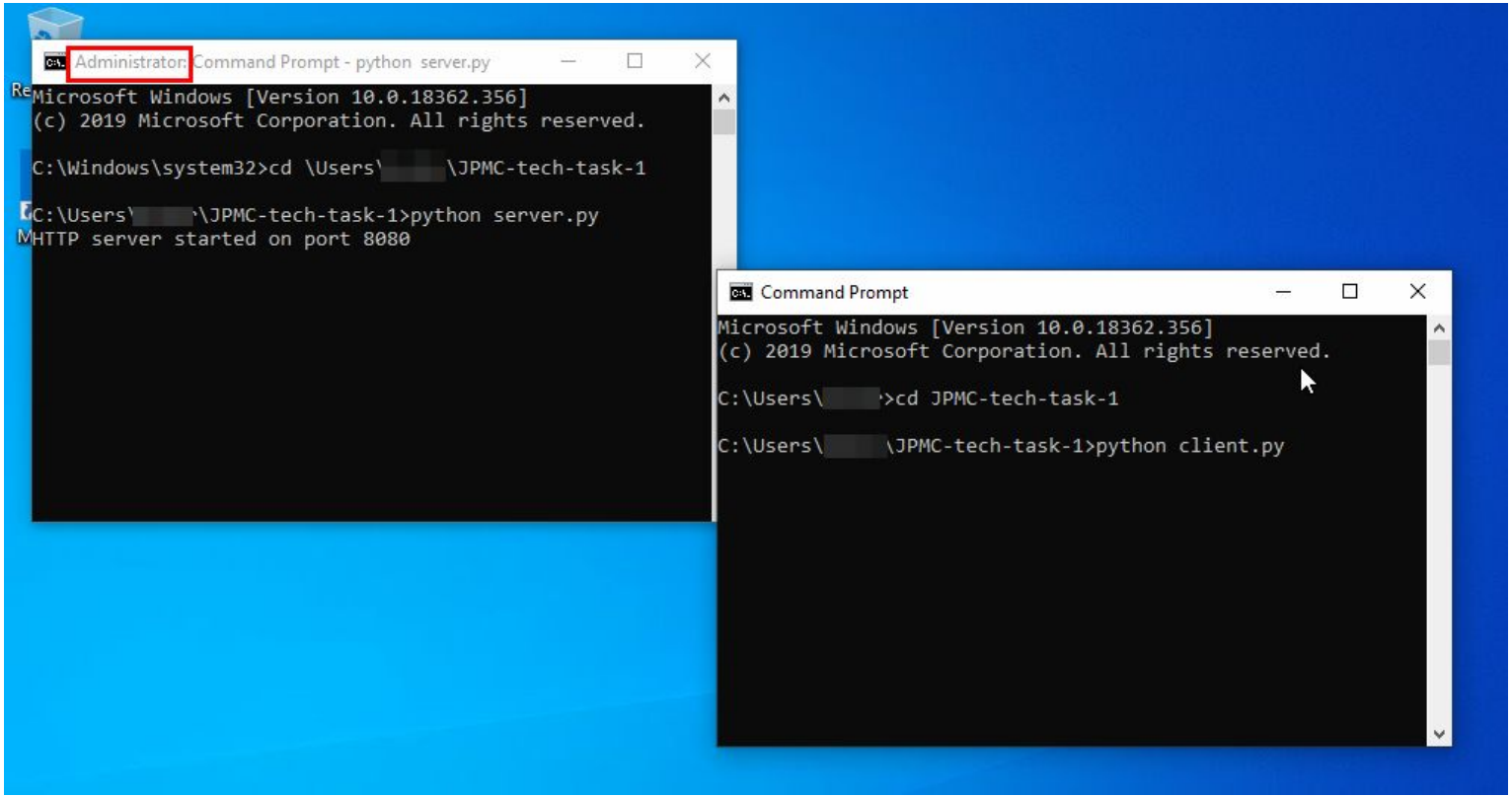
```
python3  --version
```

Local Setup (Windows)

- Once you have Python 2.7 or Python3 installed, all you have to do get the application up and running is to start the server and client scripts in two separate cmds (*see image in the next slide*). Ensure that the command line wherein you run `python server.py` is on Administrator mode:



Local Setup (Windows)



Local Setup (Windows)

- When you open the cmd in admin mode, you'll notice that its current directory starts from C:\Windows\System32. Assuming you cloned the repository in your home directory you need to use the **cd** command to get there. For instance if I cloned it in C:\Users\insidesherpa, then I should do something like:

  cd \Users\insidesherpa\JPMC-tech-task-1
  Or
  cd \Users\insidesherpa\JPMC-tech-task-1-py3

note: the example above isn't what you're going to type on your system. You have a different user account in \Users where you probably cloned the repo. Use that instead...

Local Setup (Windows)

- (*note: just choose to run one server and one client; either the python 2 or python 3 version of server and client applications*. Run the **commands** below on separate command lines, starting with the server and then the client. The commands will vary depending on your primary python version)

// If python --version = 2.7+, you must be in the JPMC-tech-task-1
// If python --version = 3+ , you must be in JPMC-tech-task-1-py3 directory

```
python server.py
python client.py
```

// If your system makes the distinction of python3 as `python3`,
// you must be in JPMC-tech-task-1-py3 directory

```
python3 server3.py
python3 client3.py
```

If ever you encounter an error when starting the server application, see troubleshooting in this slide

Local Setup: Troubleshooting (Windows)

- If you did not encounter any issues, your setup is finished. From here on, you can make changes to the code (see another guide in the module page for this) and eventually arrive at the desired output.

- If you did encounter issues, check if the commonly encountered issues listed in the next few slides will solve your problem:

    - dateutil dependency
    - python not recognized or not returning in your command line
    - Socket unavailable

Local Setup: Troubleshooting (Windows)

- In some cases, dependency issues might arise like when you run `server.py`:

```
Traceback (most recent call last):
  File "server.py", line 26, in <module>
    import dateutil.parser
ImportError: No module named dateutil.parser
```

In this case, you must install pip first. pip is python's package manager for installing python dependencies. Make sure you install pip for the right Python version you're working with in this project. You can check your pip version by **pip --version** and it will tell which python version it maps too

Local Setup: Troubleshooting (Windows)

- Installing pip nowadays usually involves downloading the get-pip.py script. If you followed the instructions in the last slide, it usually involves using the command:

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
```

If you don't have curl, just install it in your system. For mac, it's this way
Then just run the script using python:

```
//if python --version = 2.7+ this will install pip for python2
//if python --version = 3+ this will install pip for python3
python get-pip.py

//if your system makes the distinction of python3 as `python3` then
//doing the command below will install pip for python3
python3 get-pip.py
```

Local Setup: Troubleshooting (Windows)

- After that, for pip to be a recognizable command in your terminal/command line, you need to add it in your system's path environment variable
  - On Windows, for Python2.7 it's usually in C:\\Python27\Scripts. It would also be similar for Python3.x if you followed the installation guide for Python earlier (e.g. C:\\Python3X\Scripts)
  - Make sure you open a new command line too and use that instead after doing this
- To edit your system path environment variable it's similar to the slides here.
- Alternatively you can access it doing something like:
  - **C:\\Python27\Scripts\pip.exe <parameters>** (similar for python3x if it was installed in C:\\)
  - <parameters> could be something like **C:\\Python27\Scripts\pip.exe install python-dateutil**
  - Take note though, this assumes that you have your python installed in drive C:\\

Local Setup: Troubleshooting (Windows)

- Then afterwards, you can run the following command on your terminal to install the dependency:

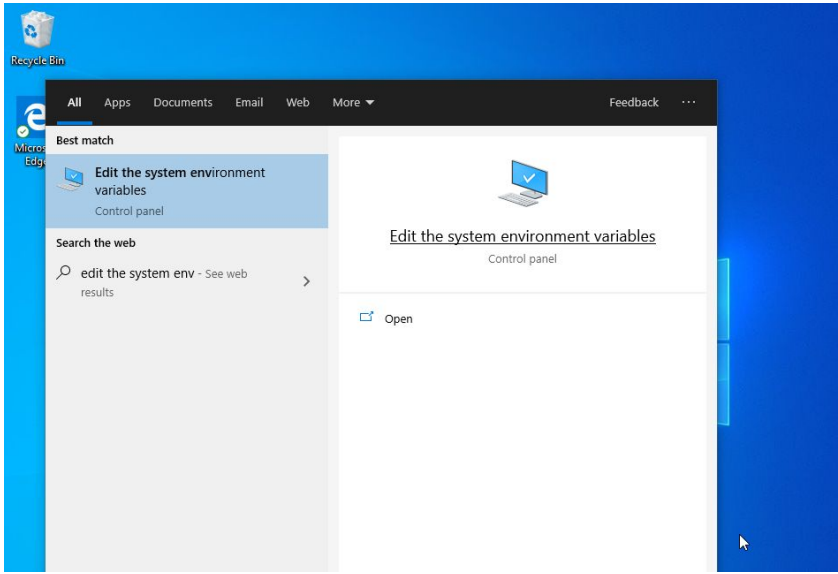
```
pip install python-dateutil
```

Afterwards, you can rerun the server and then rerun the client

Note: For the command above, whatever python version your pip corresponds to (i.e. the output of **pip --version**, that is the python version that will have the dependency installed). So if you're **pip** corresponds to python2.7.x then doing the command above will install python-dateutil for python2.7.x

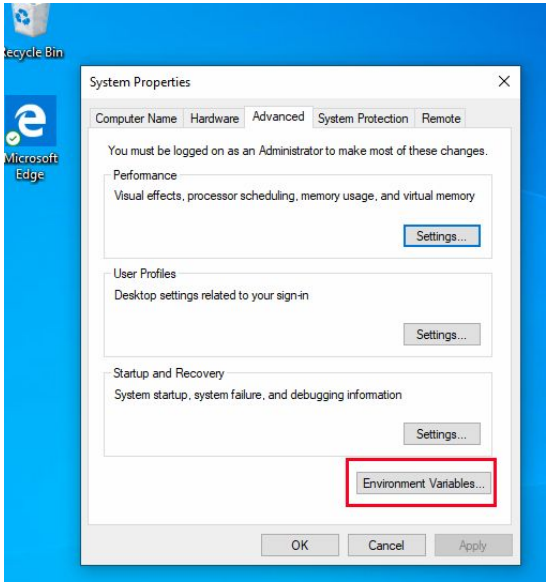
Local Setup: Troubleshooting (Windows)

- There are some cases when you tried installing python e.g. through the usual installation process or via other software like anaconda, and when you open your command line and try executing any python command like **python** or **python --version** nothing returns.The problem here is usually because python isn't properly set in your system environment's path variable properly. To do this go to your start menu,search "edit the system environment" and you should be able to see something like the image on the right

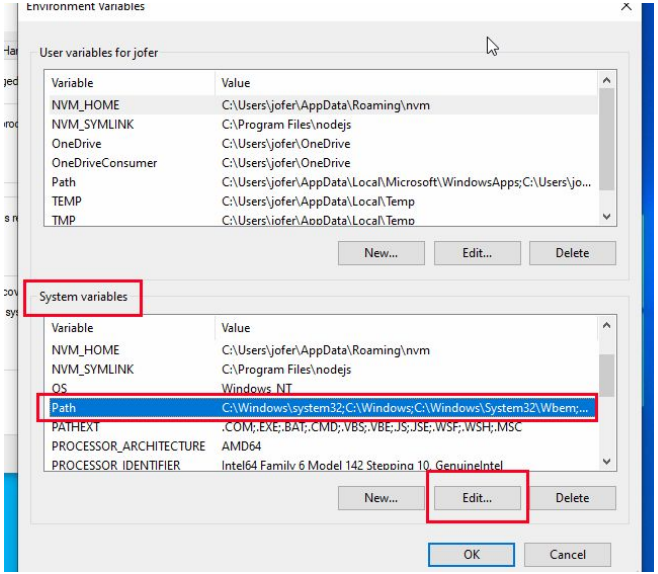


Local Setup: Troubleshooting (Windows)

- Click that and click “Environment Variables” in the next window (image on left side)



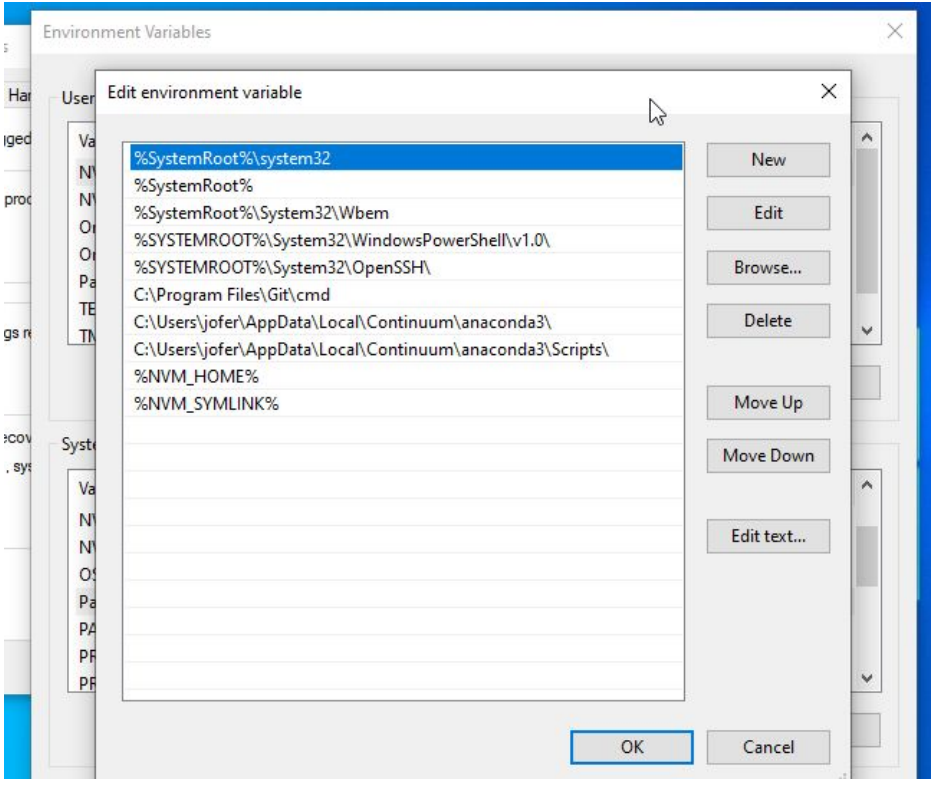
- You should then come up with the image below:



You want to edit the **Path** under **System variables**

Local Setup: Troubleshooting (Windows)

- You should end up with a window like this after the steps from the previous slide:



You want to make sure the directory where the **Python** executable/application you want to use is in is included in the list of paths.

If you followed how to install **Python2.7.x** in the earlier slides, you should have it in C:\\Python27**.** Make sure to include that in your path.

Same goes for **Python3.x** if you want to enable it. But make sure to remove your Python2.7.x path first .

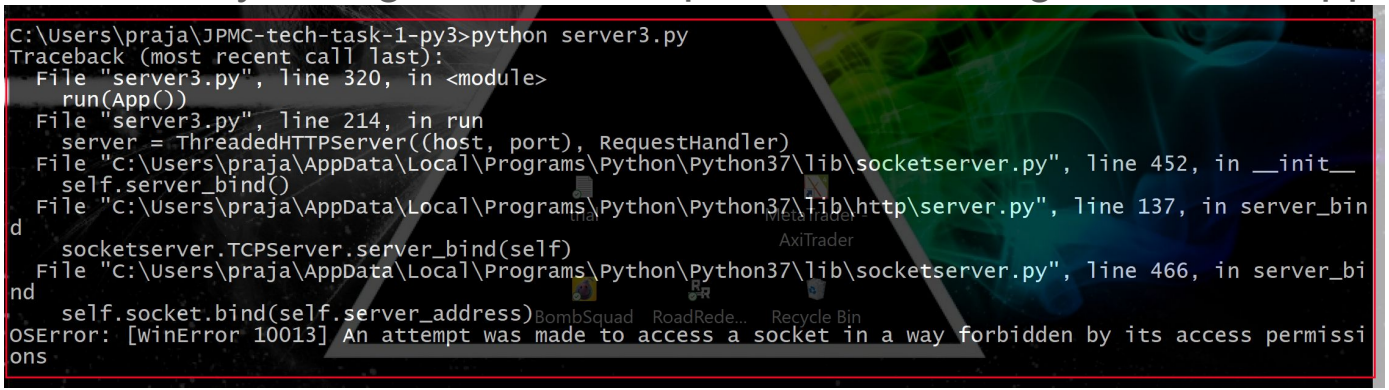
If you installed python using other means, e.g. anaconda, its python executable is located elsewhere but same method of putting the path applies

Don’t forget to restart your command line after setting a new path to reflect the changes...

Local Setup: Troubleshooting (Windows)

- In other cases, you might encounter problems running the server app

CASE A:



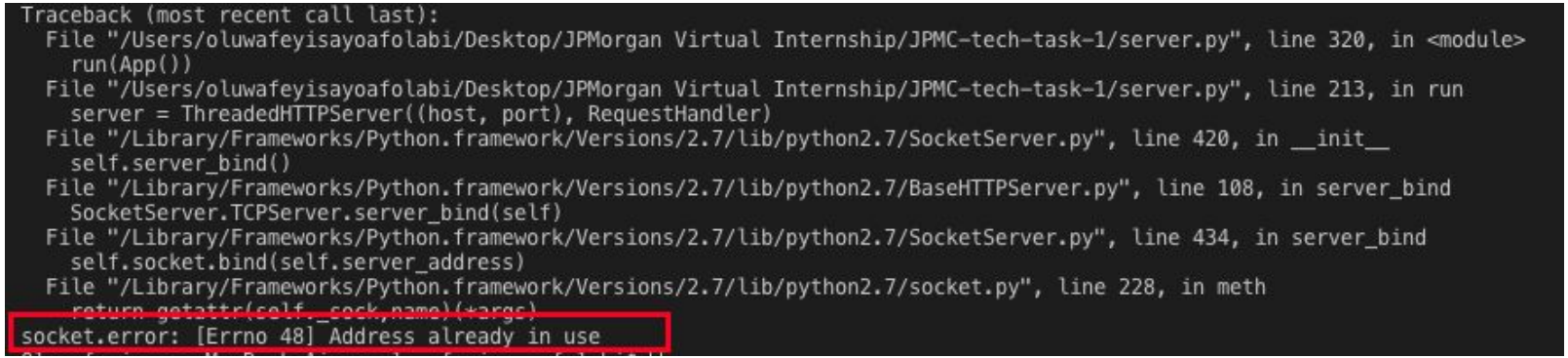
note: the example here is from windows but a similar error might appear for mac

This is most likely because you have a firewall open preventing you from accessing 8080. You can try the following workarounds:

- Temporarily turn off your firewall
- Using any text editor, open the server.py or server3.py in the repository using your code editor and look for the line where it says port = 8080. change that to port = 8085
- Similarly, open the client.py or client3.py and change the line where it has 8080 to 8085

Local Setup: Troubleshooting (Windows)

CASE B:

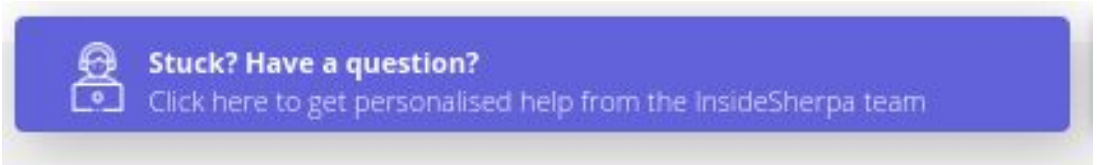


In this case make sure you're only running one instance of the server.py because it hooks itself to port 8080, and once that port is used nothing else can use it. If you want to free that up, terminate the old server.py you're running from one of your terminals by hitting cmd+c. Alternatively you can kill the process listening on a port (i.e. in this case 8080) by following this guide

Local Setup: Troubleshooting (Windows)

- If you did encounter any other issues, please post your issue/inquiry here: https://github.com/insidesherpa/JPMC-tech-task-1/issues or https://github.com/insidesherpa/JPMC-tech-task-1-py3/issues depending on what repository you chose to work in. When submitting a query, please don't forget to provide as much context as possible, i.e. your OS, what you've done, what your errors is/are, etc (screenshots would help too)

- You can also submit your query in the module page's support modal that pops out when you click the floating element on the page (see image below)



InsideSherpa

JPMorgan Chase Software Engineering Virtual Experience

Setting up your Linux for the JPMorgan Chase program

Module 1 - Interface with a stock price data feed

Local Setup (Linux)

- Use this method if you chose not to use the REPL method.
  If your machine is running on any flavor of linux, follow this setup guide to get started

- First you must have git installed in your system.Git is used by most programmers today to collaborate with code/software projects.You can install git by simply running the command below in your terminal (ctrl+alt+t):

```
~$ sudo apt-get install git
```

- You'll know you have git if you get a similar result on your terminal:

```
~$ git version
git version 2.17.1
```

Local Setup (Linux)

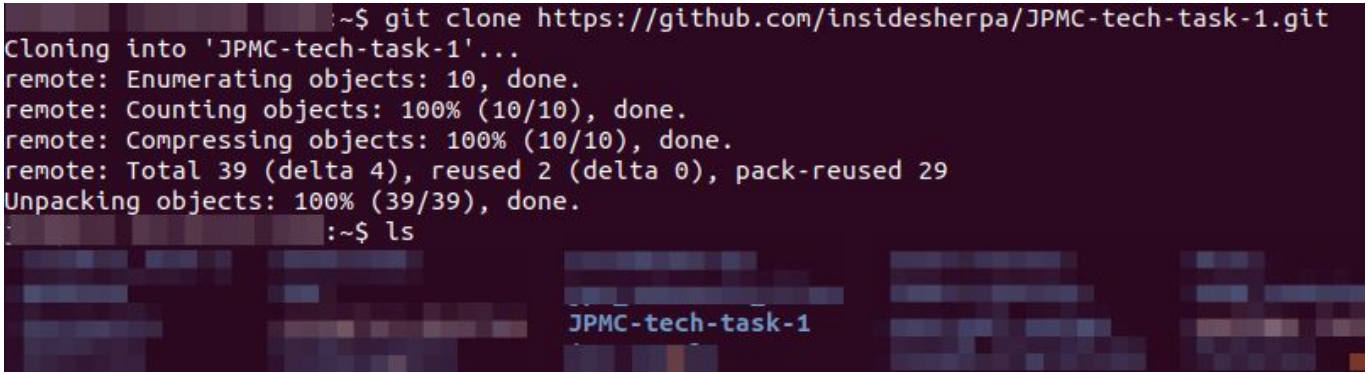
- Once you have git installed, you need a copy of the application code you'll be working with on your machine. To do this, you must execute the following command on your terminal:

```
git clone https://github.com/insidesherpa/JPMC-tech-task-1.git
git clone https://github.com/insidesherpa/JPMC-tech-task-1-py3.git
```

- This command will download the code repositories from github to your machine in the current working directory of the terminal you executed the command in. Downloading the 2 repositories above will give you options later

Local Setup (Linux)

- You'll know you cloned successfully if you have the copy of the application code on your machine:



note: the image above just does not contain the other repository but it should if you did the previous slides and execute the ***ls*** *command. `ls` just lists the contents in the current directory*

Local Setup (Linux)

- To access the files inside from the terminal, just change directory by typing:

cd JPMC-tech-task-1

note: If you choose to work using python3 and your system has version python3 or above instead of python2.7.x, then choose to go into the other repository you downloaded instead. (otherwise, use the other repo above).

cd JPMC-tech-task-1-py3

note: `cd` means change directory. *Check this for more info on how to use* ***cd***

Local Setup (Linux)

- To clarify, you're only supposed to work on one of the repositories you cloned / downloaded into your system. It all depends on what Python version you primarily use

- Python is just a scripting / programming language we developers use quite often in the field. This application you'll be working on uses it.

- We'll discuss checking / installing Python in your system in the following slides

Local Setup (Linux)

- Next, you'll need to have Python 2.7 or Python 3 installed on your machine. Follow the instructions here. (python2) or here (python3) For most cases, Linux environments already have Python 2.7 or Python3. You can verify this on your terminal if you get a result like:

(*any python 2.7.x > = 2.7.16 should suffice*
but the latest 2.7.x is recommended (**2.7.17**)*;*
any python 3.x >= 3.6is fine, latest is recommended (**3.8.0**))

Execute the **command** below to verify what version you have:

```
python --version
```

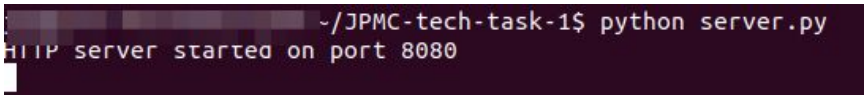
Note: the image here is only of 2.7 but it should be similar if you check for python3

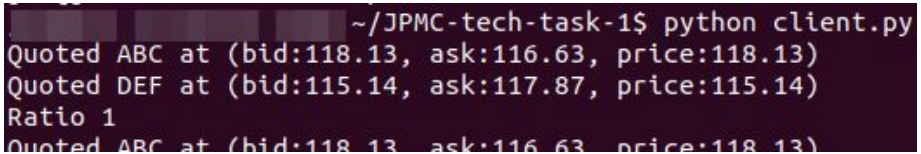
Sometimes your system might have it as

```
python3  --version
```

Local Setup (Linux)

- Once you have Python 2.7 or Python3 installed, all you have to do get the application up and running is to start the server and client scripts in two separate terminals.





Local Setup (Linux)

- (*note: just choose to run one server and one client; either the python 2 or python 3 version of server and client applications*. Run the **commands** below on separate terminals, starting with the server and then the client. The commands will vary depending on your primary python version)

// If python --version = 2.7+, you must be in the JPMC-tech-task-1
// If python --version = 3+ , you must be in JPMC-tech-task-1-py3 directory

```
python server.py
python client.py
```

// If your system makes the distinction of python3 as `python3`,
// you must be in JPMC-tech-task-1-py3 directory

```
python3 server3.py
python3 client3.py
```

If ever you encounter an error when starting the server application, see troubleshooting in this slide

Local Setup: Troubleshooting (Linux)

- If you did not encounter any issues, your setup is finished. From here on, you can make changes to the code (see another guide in the module page for this) and eventually arrive at the desired output.

- If you did encounter issues, check if the commonly encountered issues listed in the next few slides will solve your problem:

    - dateutil_dependency
    - Socket unavailable

Local Setup: Troubleshooting (Linux)

- In some cases, dependency issues might arise like when you run `server.py`:

```
Traceback (most recent call last):
  File "server.py", line 26, in <module>
    import dateutil.parser
ImportError: No module named dateutil.parser
```

In this case, you must install pip first. pip is python's package manager for installing python dependencies. Make sure you install pip for the right Python version you're working with in this project. You can check your pip version by **pip --version** and it will tell which python version it maps too

Local Setup: Troubleshooting (Linux)

- Installing pip nowadays usually involves downloading the get-pip.py script. If you followed the instructions in the last slide, it usually involves using the command:

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
```

If you don't have curl, just install it in your system. For mac, it's this way
Then just run the script using python:

```
//if python --version = 2.7+ this will install pip for python2
//if python --version = 3+ this will install pip for python3
python get-pip.py

//if your system makes the distinction of python3 as `python3` then
//doing the command below will install pip for python3
python3 get-pip.py
```

Local Setup: Troubleshooting (Linux)

- Then afterwards, you can run the following command on your terminal to install the dependency:

```
pip install python-dateutil
```

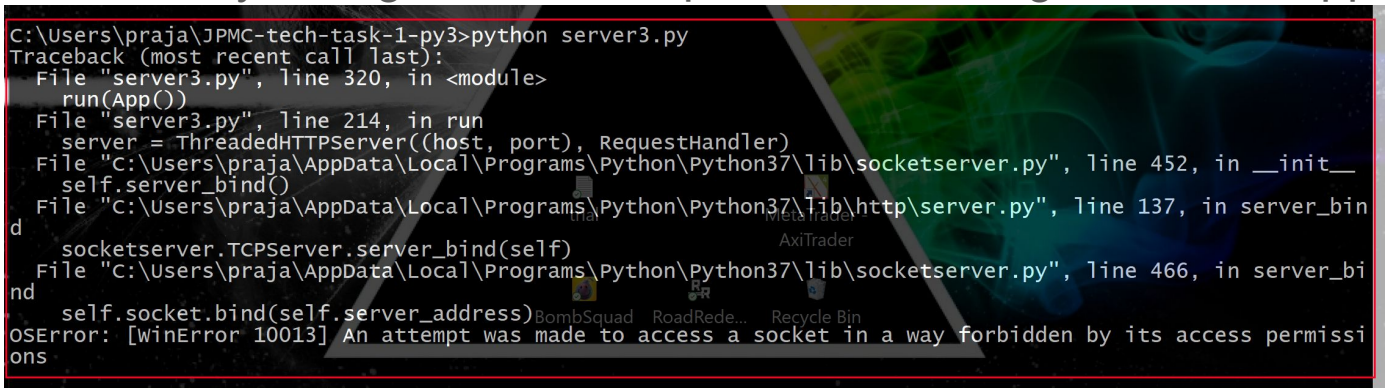
Afterwards, you can rerun the server and then rerun the client

Note: For the command above, whatever python version your pip corresponds to (i.e. the output of **pip --version**, that is the python version that will have the dependency installed). So if you're **pip** corresponds to python2.7.x then doing the command above will install python-dateutil for python2.7.x

Local Setup: Troubleshooting (Linux)

- In other cases, you might encounter problems running the server app

CASE A:



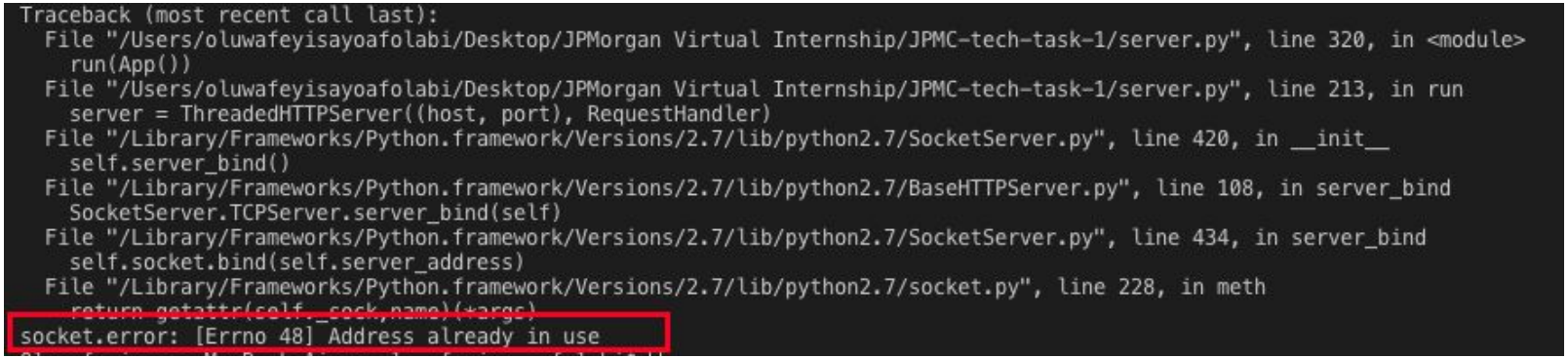
note: the example here is from windows but a similar error might appear for mac

This is most likely because you have a firewall open preventing you from accessing 8080. You can try the following workarounds:

- Temporarily turn off your firewall
- Using any text editor, open the server.py or server3.py in the repository using your code editor and look for the line where it says port = 8080. change that to port = 8085
- Similarly, open the client.py or client3.py and change the line where it has 8080 to 8085

Local Setup: Troubleshooting (Linux)

CASE B:



In this case make sure you're only running one instance of the server.py because it hooks itself to port 8080, and once that port is used nothing else can use it. If you want to free that up, terminate the old server.py you're running from one of your terminals by hitting cmd+c. Alternatively you can kill the process listening on a port (i.e. in this case 8080) by following this guide

Local Setup: Troubleshooting (Linux)

- If you did encounter any other issues, please post your issue/inquiry here: https://github.com/insidesherpa/JPMC-tech-task-1/issues or https://github.com/insidesherpa/JPMC-tech-task-1-py3/issues depending on what repository you chose to work in. When submitting a query, please don't forget to provide as much context as possible, i.e. your OS, what you've done, what your errors is/are, etc (screenshots would help too)

- You can also submit your query in the module page's support modal that pops out when you click the floating element on the page (see image below)