W04 –
SETTING UP MINIMAL SPARK ENVIRONMENT WITHOUT DOCKER INSTALLED

2024

The Hoang

Notes

The following slides guide you how to install the environment to practice Spark coding in this course.

For each software or dependency, there are multiple ways to install. Feel free to pick up one which is most familiar to you. You’re not required to fully comply with all the steps.

The ultimate acceptance: criteria is you can execute notebooks containing Pyspark codes with no errors.

Environment setup – Python & PIP

Install Python

- Preferred version: 3.11.9. Note that the higher versions (3.12.x) can cause error during executing Spark applications

Verification

- `python --version`

Upgrade pip

- `python -m pip install --upgrade pip`

Verification

- `pip --version`

Environment setup - Pyspark

Install latest Pyspark version:

```
pip install pyspark
```

Verification:

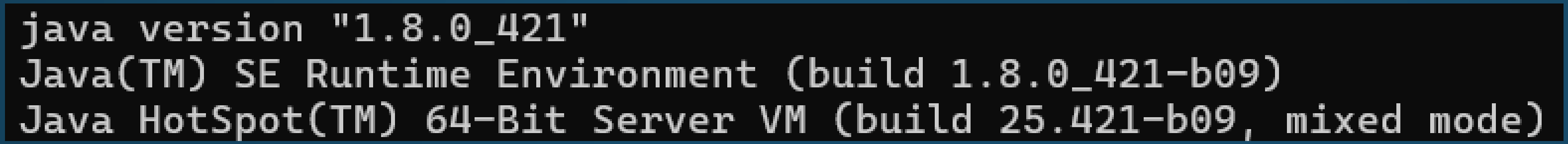
`pip list` => the pyspark version 3.5.x should be displayed

Environment setup - Java

Download & Install latest Java 8 from Oracle (8u421)

Verification:

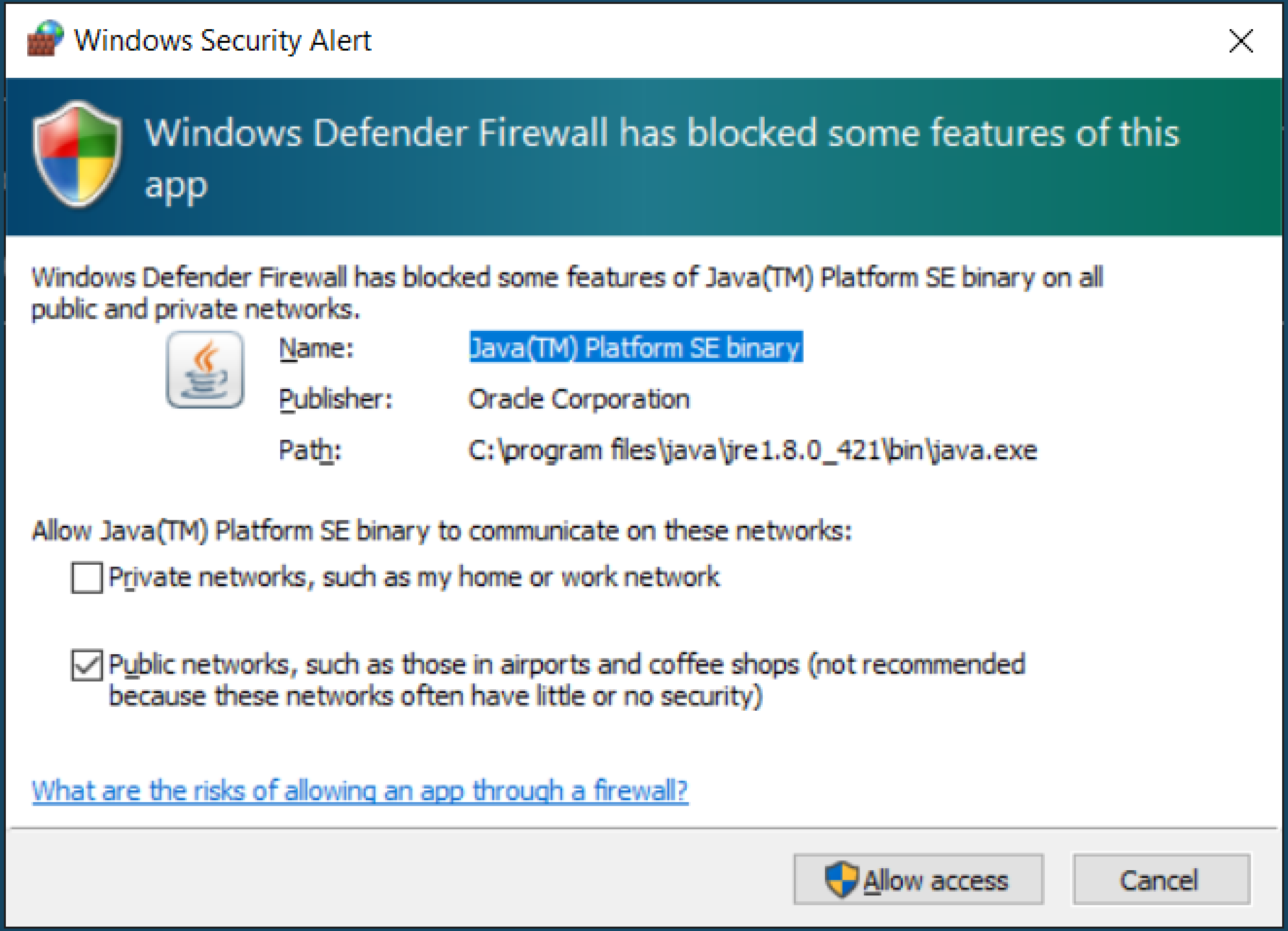
```
java -version
```



```
java version "1.8.0_421"
Java(TM) SE Runtime Environment (build 1.8.0_421-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.421-b09, mixed mode)
```

Environment setup - Java

- Download & Install latest Java 8 from Oracle (8u421)
- Allow network access if the below popup shows up


Windows Security Alert

Windows Defender Firewall has blocked some features of this app

Windows Defender Firewall has blocked some features of Java(TM) Platform SE binary on all public and private networks.

Name:	Java(TM) Platform SE binary
Publisher:	Oracle Corporation
Path:	C:\program files\java\jre1.8.0_421\bin\java.exe

Allow Java(TM) Platform SE binary to communicate on these networks:

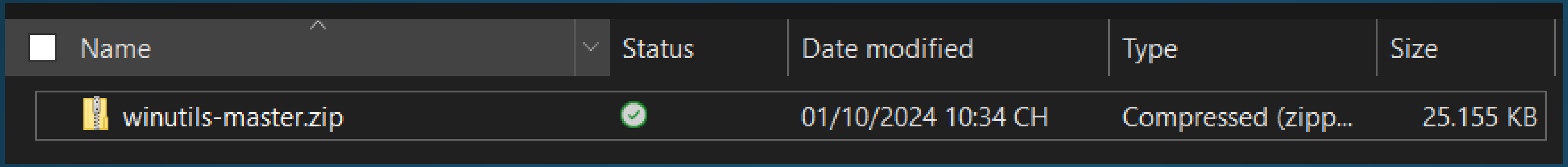
☐ Private networks, such as my home or work network

☑ Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

What are the risks of allowing an app through a firewall?

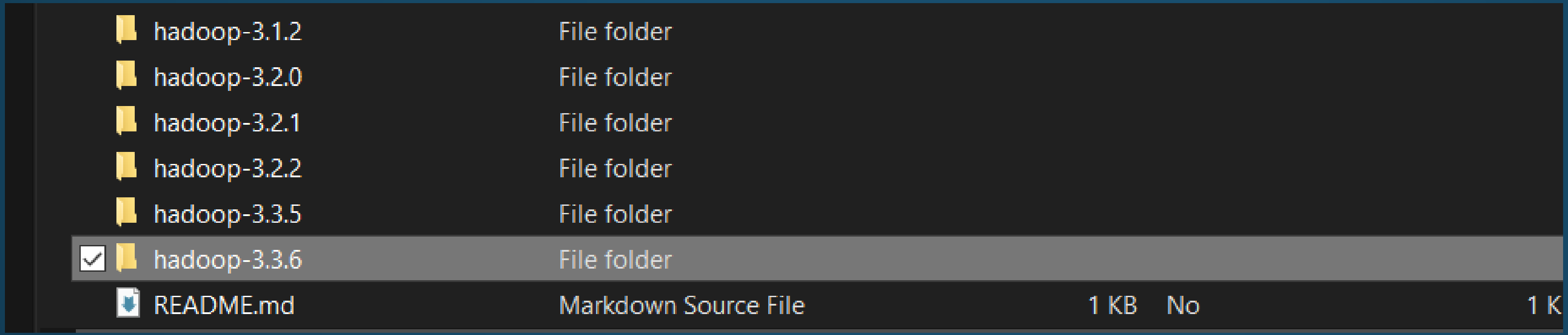
Environment setup – Hadoop binary

- Download winutils-master.zip from this Github link:



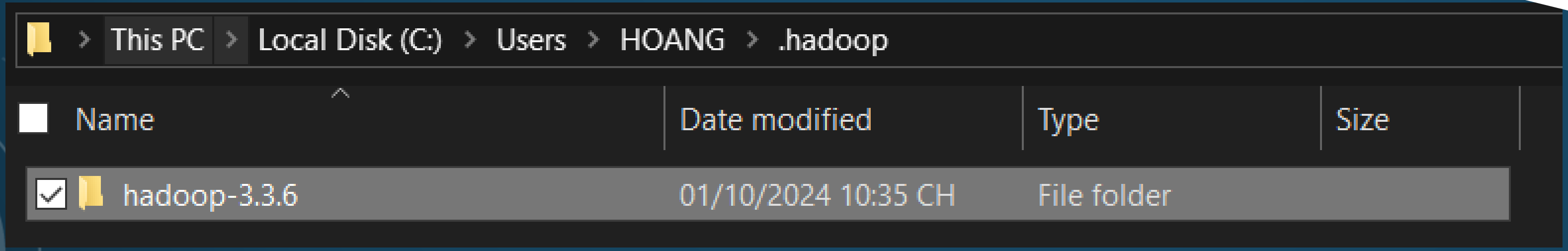
Name	Status	Date modified	Type	Size
winutils-master.zip	✓	01/10/2024 10:34 CH	Compressed (zipp...	25.155 KB

- Unzip the downloaded file, copy the Hadoop-3.3.6 folder:



hadoop-3.1.2	File folder			
hadoop-3.2.0	File folder			
hadoop-3.2.1	File folder			
hadoop-3.2.2	File folder			
hadoop-3.3.5	File folder			
✓ hadoop-3.3.6	File folder			
README.md	Markdown Source File	1 KB	No	1 K

- Paste the copied folder to a new folder. Example: C:\Users\HOANG\.hadoop\



This PC > Local Disk (C:) > Users > HOANG > .hadoop

Name	Date modified	Type	Size
✓ hadoop-3.3.6	01/10/2024 10:35 CH	File folder	

Change to your PC username

Environment setup – System variables

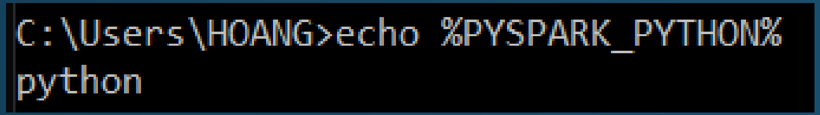
Pyspark Python:

Run this command in windows cmd as administrator:

```
setx PYSPARK_PYTHON “python” /M
```

Verification:

```
echo %PYSPARK_PYTHON%
```

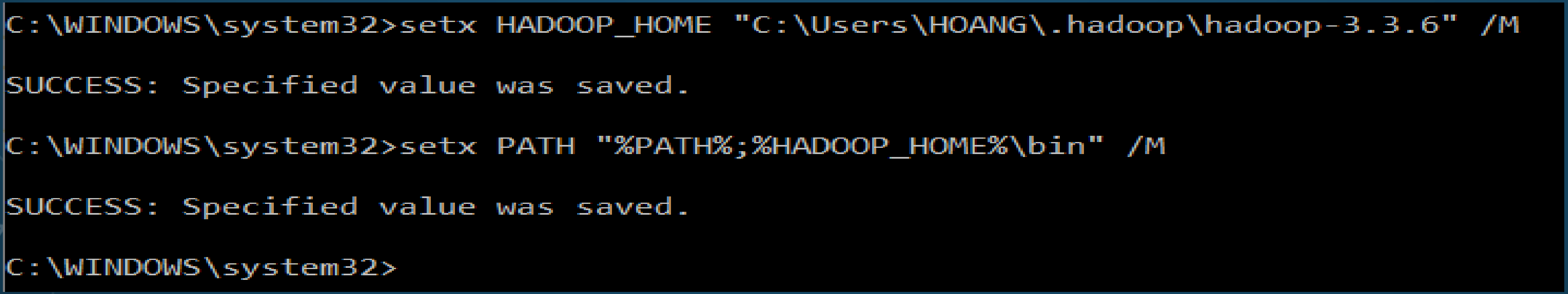


```
C:\Users\HOANG>echo %PYSPARK_PYTHON%
python
```

Hadoop home

Change to your PC username

```
setx HADOOP_HOME "C:\Users\HOANG\.hadoop\hadoop-3.3.6" /M
setx PATH "%PATH%;%HADOOP_HOME%\bin" /M
```



```
C:\WINDOWS\system32>setx HADOOP_HOME "C:\Users\HOANG\.hadoop\hadoop-3.3.6" /M

SUCCESS: Specified value was saved.

C:\WINDOWS\system32>setx PATH "%PATH%;%HADOOP_HOME%\bin" /M

SUCCESS: Specified value was saved.

C:\WINDOWS\system32>
```

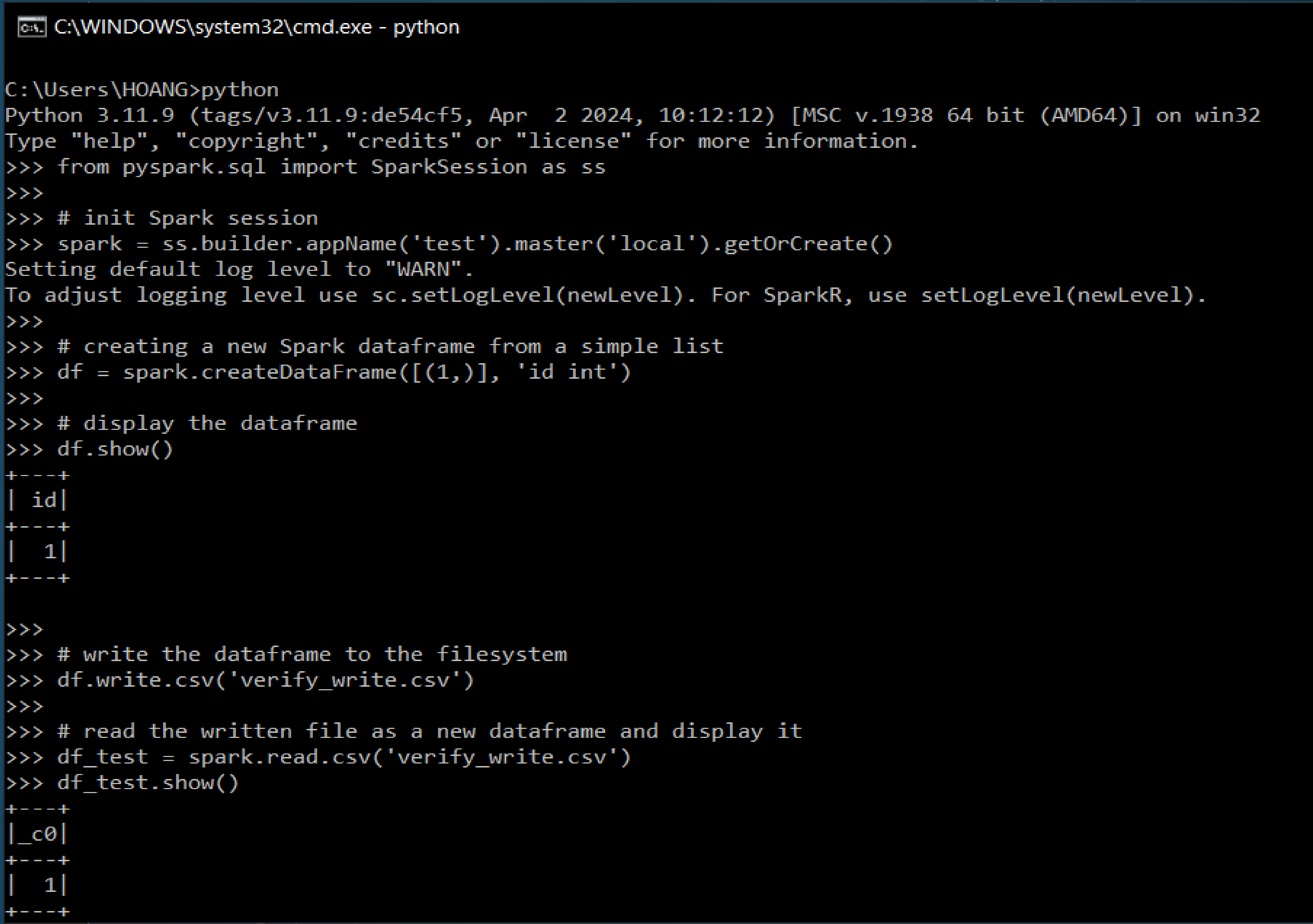
Try executing a simple Pyspark Application

- Launch Python shell
- Execute following codes

```
from pyspark.sql import SparkSession as ss

# init Spark session
spark = ss.builder.appName('test').master('local').getOrCreate()

# creating a new Spark dataframe from a simple list
df = spark.createDataFrame([(1,)], 'id int')

# display the dataframe
df.show()

# write the dataframe to the filesystem
df.write.csv('verify_write.csv')

# read the written file as a new dataframe and display it
df_test = spark.read.csv('verify_write.csv')
df_test.show()
```

Try executing a simple Pyspark Application

- Expected results



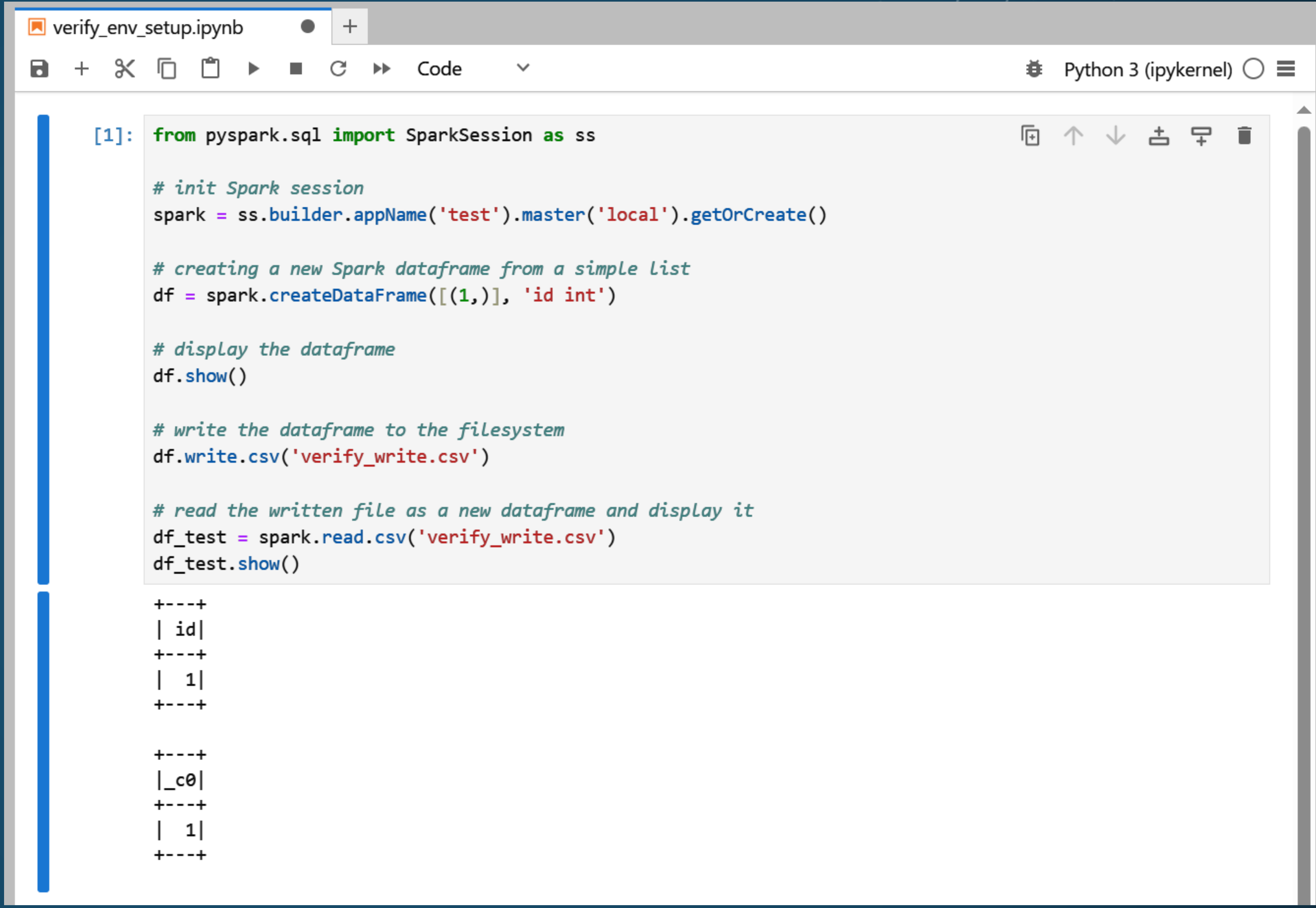
```
C:\Users\HOANG>python
Python 3.11.9 (tags/v3.11.9:de54cf5, Apr  2 2024, 10:12:12) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from pyspark.sql import SparkSession as ss
>>>
>>> # init Spark session
>>> spark = ss.builder.appName('test').master('local').getOrCreate()
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
>>>
>>> # creating a new Spark dataframe from a simple list
>>> df = spark.createDataFrame([(1,)], 'id int')
>>>
>>> # display the dataframe
>>> df.show()
+---+
| id|
+---+
|  1|
+---+

>>>
>>> # write the dataframe to the filesystem
>>> df.write.csv('verify_write.csv')
>>>
>>> # read the written file as a new dataframe and display it
>>> df_test = spark.read.csv('verify_write.csv')
>>> df_test.show()
+---+
|_c0|
+---+
|  1|
+---+
```

Install Jupyterlab

Execute following command to install and launch Jupyterlab

```
pip install jupyterlab
jupyter lab
```

Execute the codes copied from previous slide to verify the Pyspark work well from JupyterLab

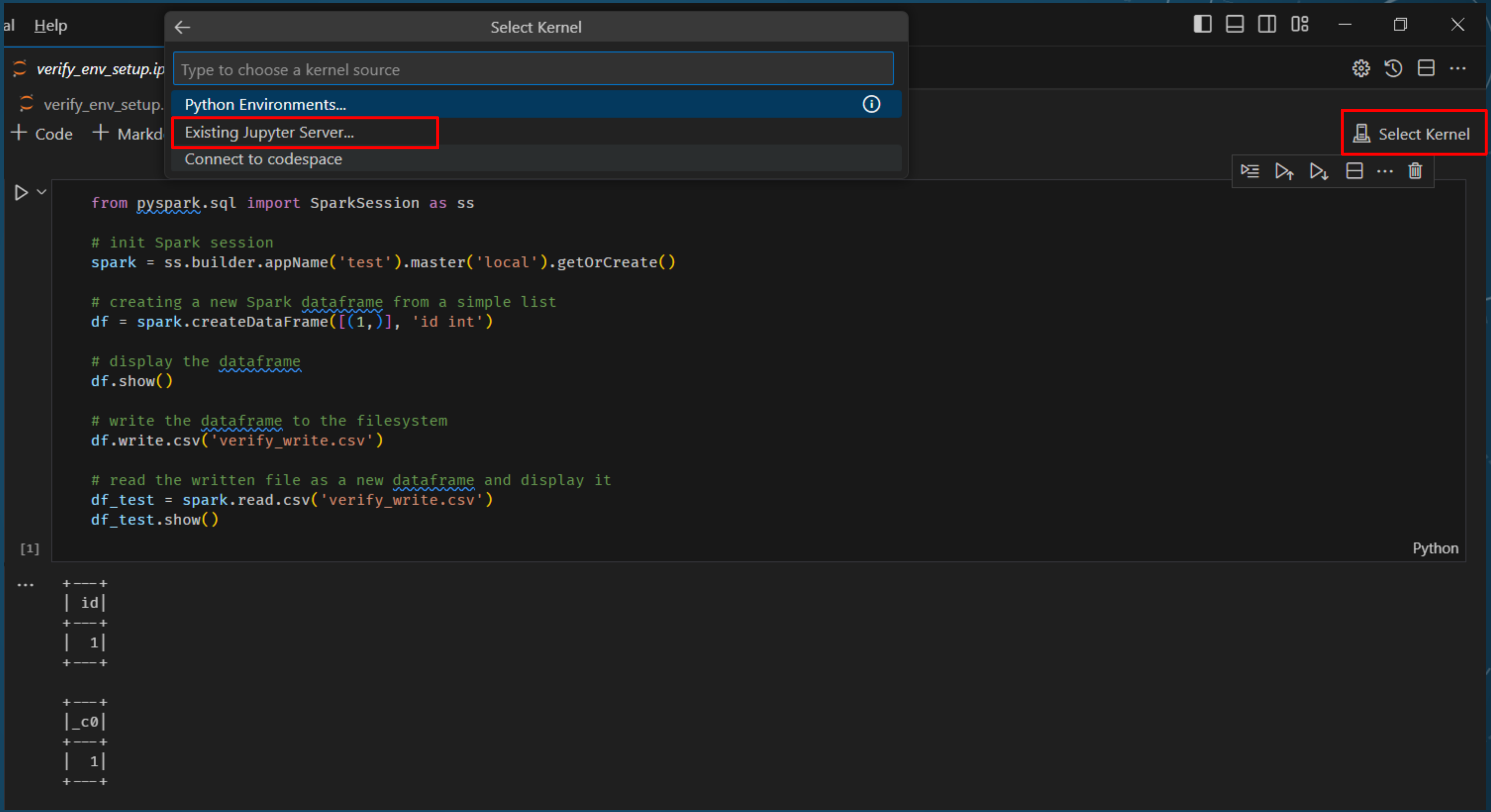


```
[1]: from pyspark.sql import SparkSession as ss

     # init Spark session
     spark = ss.builder.appName('test').master('local').getOrCreate()

     # creating a new Spark dataframe from a simple list
     df = spark.createDataFrame([(1,)], 'id int')

     # display the dataframe
     df.show()

     # write the dataframe to the filesystem
     df.write.csv('verify_write.csv')

     # read the written file as a new dataframe and display it
     df_test = spark.read.csv('verify_write.csv')
     df_test.show()
```

```
+---+
| id|
+---+
|  1|
+---+

+---+
|_c0|
+---+
|  1|
+---+
```

Integrate with Visual Code

1. Select Kernel
2. Existing Jupyter Server
3. Enter password
4. Save as “localhost”



```
from pyspark.sql import SparkSession as ss

# init Spark session
spark = ss.builder.appName('test').master('local').getOrCreate()

# creating a new Spark dataframe from a simple list
df = spark.createDataFrame([(1,)], 'id int')

# display the dataframe
df.show()

# write the dataframe to the filesystem
df.write.csv('verify_write.csv')

# read the written file as a new dataframe and display it
df_test = spark.read.csv('verify_write.csv')
df_test.show()
```

```
+---+
| id|
+---+
|  1|
+---+

+---+
|_c0|
+---+
|  1|
+---+
```

Optional

1. You can setup pyenv to easily switch between Python versions. Link
2. If you want to isolate this environment, create a virtual environment for it (venv). Link