Numpy and Pandas

Python for Economics

Morning, Aug 17, 2023

Zhenhua He

The need for better number-crunching

Lists don't have many built-in methods for interacting with data.

The base Python data types also take up a lot of space compared to other languages.

The **Numpy** and **Pandas** modules offer powerful tools for improving performance when you're using lots of data and doing lots of operations on them.

Arrays and Series: Arrays

Numpy Arrays supports common operations, such as arithmetic, on an element-by-element (or “vectorized”) basis.

Example:

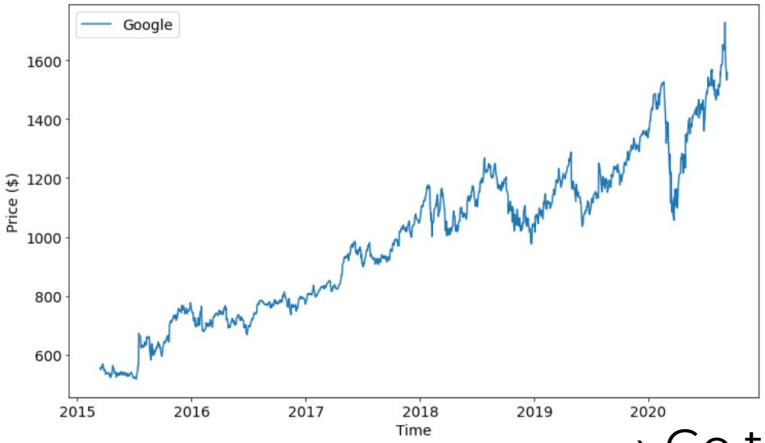
```
array C = array A + array B
```

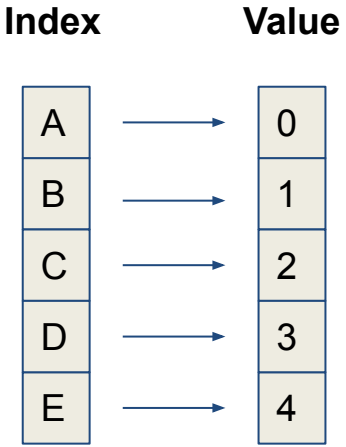
This adds the elements of A and B pair-wise (Instead of concatenating the elements as would happen with lists).

Pandas Series and DataFrames further expand on this.

Arrays and Series: Series

- One-dimensional labeled array
- Capable of holding any data type (integers, strings, floating point numbers, etc.)
- Example:





→ Go to notebook to practice

Break Time Reminder Slide

10 minutes break

Matplotlib

Python for Economics

Morning, Aug 17, 2023

Zhenhua He

This Module

1. Line Plots
2. Scatter plots

Matplotlib Setup

Matplotlib does lots of plotting for us–specifically using the "pyplot" submodule. Import a module with a nickname using `as`:

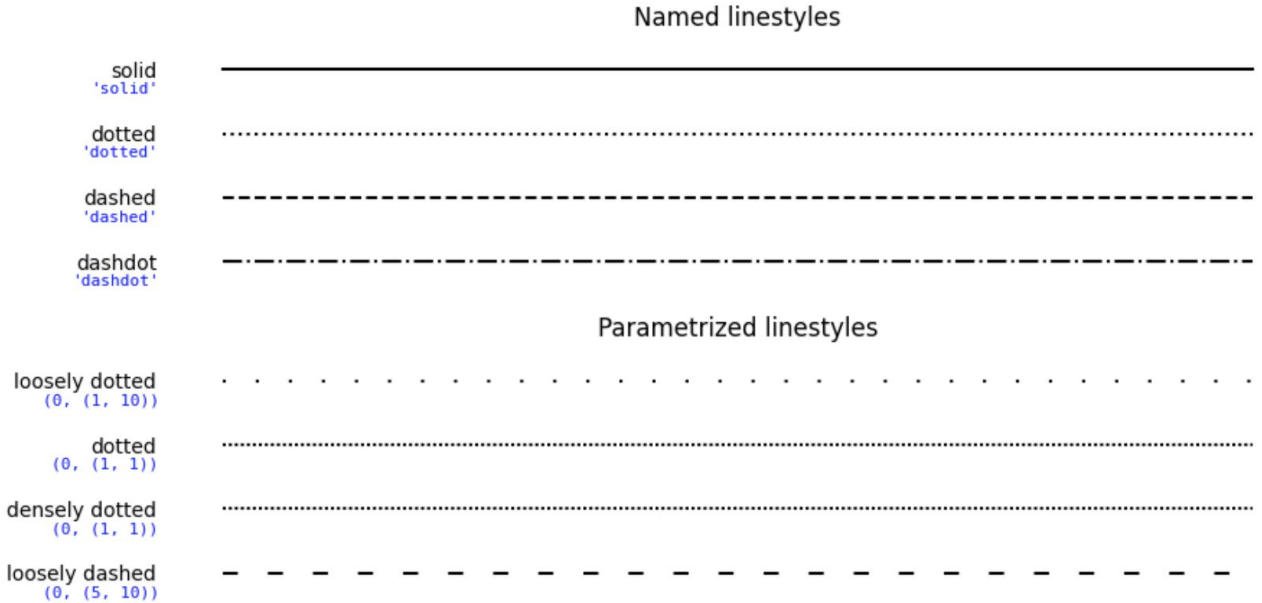
```
import matplotlib.pyplot as plt
import numpy as np
```

(The following slides show some plot style reference, but otherwise…)

→ Go to notebook to practice

Line Plots Styles

Simple line styles can be defined using the strings "solid", "dotted", "dashed" or "dashdot".



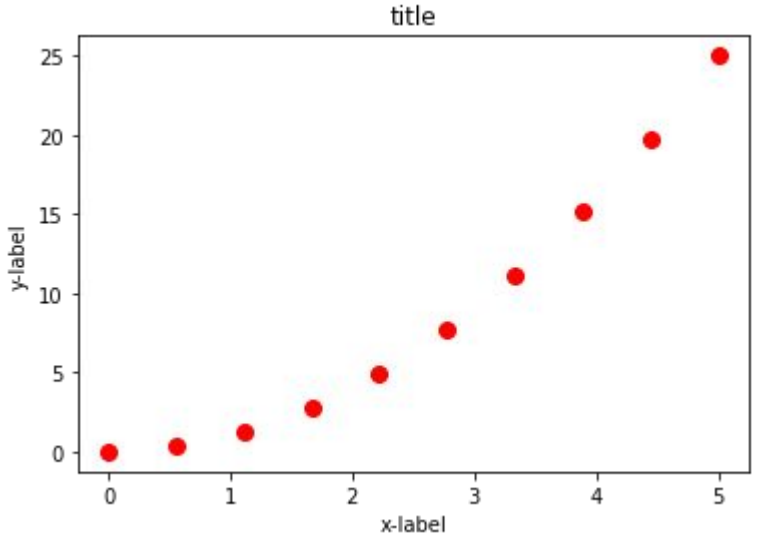
Anatomy of a Plot

Marker

- style
- size
- color

Figure

- title
- xlabel
- ylabel



Plot Marker symbols

marker	symbol	description
`"."`		point
`","`		pixel
`"o"`		circle
`"v"`		triangle_down
`"^"`		triangle_up
`"<"`		triangle_left
`">"`		triangle_right
`"1"`		tri_down
`"2"`		tri_up
`"3"`		tri_left
`"4"`		tri_right
`"8"`		octagon
`"s"`		square
`"p"`		pentagon
`"P"`		plus (filled)
`"*"`		star

Break Time Reminder Slide

10 minutes break

Data Manipulation

Python for Economics

Morning and Afternoon, Aug 17, 2023

Wesley Brashear, Josh Winchell

This Module

1. Array/Series data selection
2. DataFrames
3. Columns and Filtering
4. DataFrame Methods

Intro: Pandas VS NumPy

NumPy	Pandas
Faster mathematical operations ✅	Slower mathematical operations
Only supports integer index	Customized index ✅
must use structured arrays	Easily handles different data types ✅
better performance when number of rows is 500K or less	better performance when number of rows is 500K or more ✅
more complicated to read and write files	simpler to read and write more file formats ✅

Array/Series Data Selection

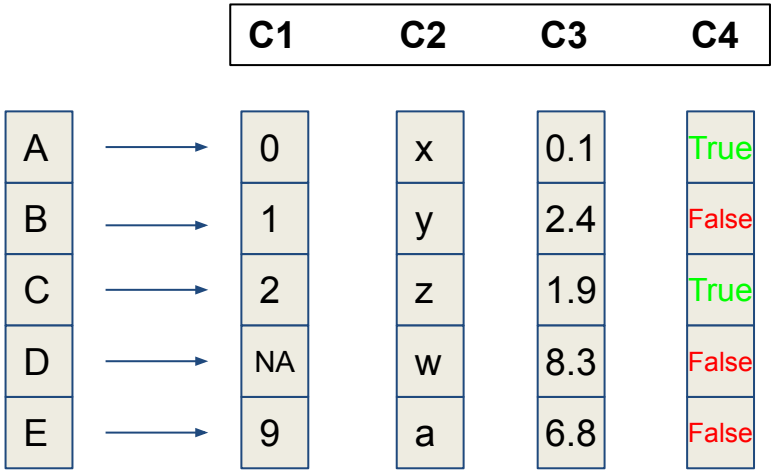
Say we have a lot of data–and now that we have matplotlib we want to plot it… but only some of it.

Arrays provide us with ways to select data that are more nuanced than the options provided by plain lists.

→ Go to notebook to practice

DataFrames

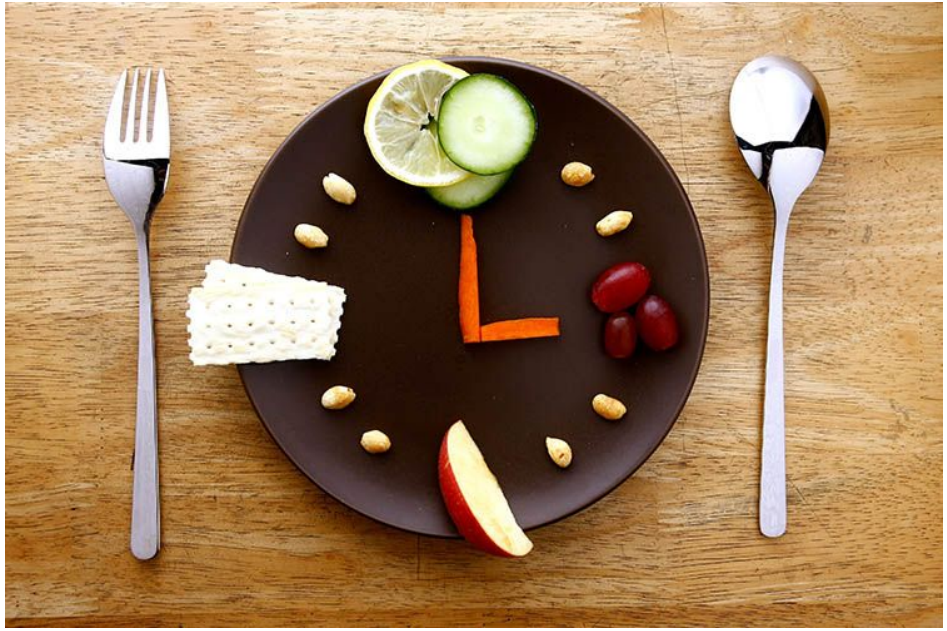
- Primary Pandas data structure
- A dict-like container for Series objects
- Two-dimensional size-mutable
- Heterogeneous tabular data structure



→ Go to notebook to practice

Lunch Break Reminder Slide

1 hour break

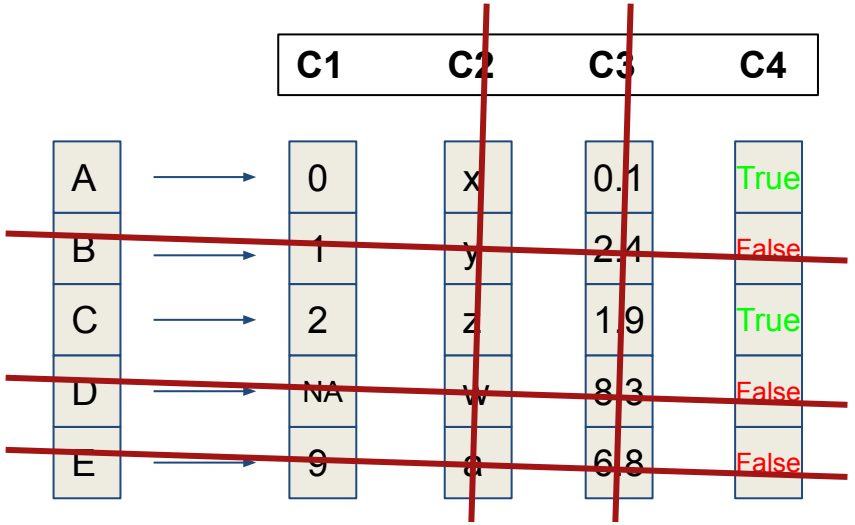


Columns and Filtering

Like arrays and series, DataFrames can be indexed, sliced, and filtered.

You can select specific rows and/or columns by name or based on some criteria.

Say we only want columns 1 and 4 when column 4 is "True"...



→ Go to notebook to practice

DataFrame Methods

There's a lot we can do via DataFrame Methods:

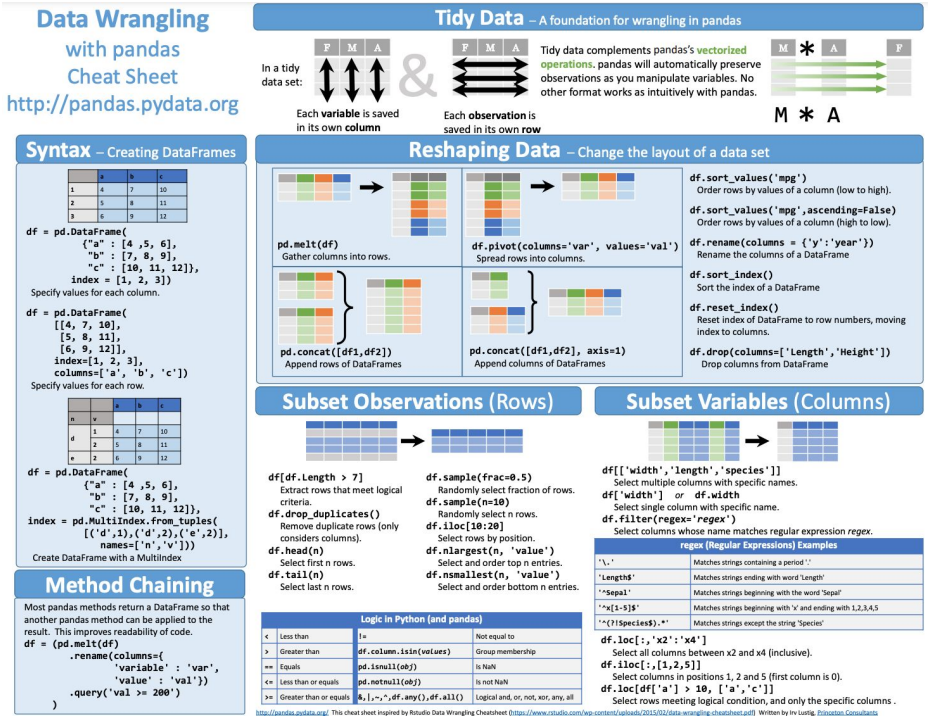
- Selecting/slicing and filtering
- Sorting or grouping by index or values
- Reading or writing to files
- Plotting
- Data cleanup
- Data merging

→ Go to notebook to practice

Break Time Reminder Slide

10 minutes break

Pandas Cheat Sheet (continued learning)



Data Wrangling with pandas Cheat Sheet
http://pandas.pydata.org

Tidy Data – A foundation for wrangling in pandas

In a tidy data set:

Each **variable** is saved in its own **column**

Each **observation** is saved in its own **row**

Tidy data complements pandas's **vectorized operations**. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.

M * A

Syntax – Creating DataFrames

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
        index = [1, 2, 3])
```
Specify values for each column.

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```
Specify values for each row.

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
index = pd.MultiIndex.from_tuples(
        [('d',1),('d',2),('e',2)],
         names=['n','v']))
```
Create DataFrame with a MultiIndex

Method Chaining

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = (pd.melt(df)
        .rename(columns={
                'variable' : 'var',
                'value' : 'val'})
        .query('val >= 200')
    )
```

Reshaping Data – Change the layout of a data set

`pd.melt(df)`
Gather columns into rows.

`df.pivot(columns='var', values='val')`
Spread rows into columns.

`pd.concat([df1,df2])`
Append rows of DataFrames

`pd.concat([df1,df2], axis=1)`
Append columns of DataFrames

`df.sort_values('mpg')`
Order rows by values of a column (low to high).

`df.sort_values('mpg',ascending=False)`
Order rows by values of a column (high to low).

`df.rename(columns = {'y':'year'})`
Rename the columns of a DataFrame

`df.sort_index()`
Sort the index of a DataFrame

`df.reset_index()`
Reset index of DataFrame to row numbers, moving index to columns.

`df.drop(columns=['Length','Height'])`
Drop columns from DataFrame

Subset Observations (Rows)

`df[df.Length > 7]`
Extract rows that meet logical criteria.

`df.drop_duplicates()`
Remove duplicate rows (only considers columns).

`df.head(n)`
Select first n rows.

`df.tail(n)`
Select last n rows.

`df.sample(frac=0.5)`
Randomly select fraction of rows.

`df.sample(n=10)`
Randomly select n rows.

`df.iloc[10:20]`
Select rows by position.

`df.nlargest(n, 'value')`
Select and order top n entries.

`df.nsmallest(n, 'value')`
Select and order bottom n entries.

Subset Variables (Columns)

`df[['width','length','species']]`
Select multiple columns with specific names.

`df['width']` or `df.width`
Select single column with specific name.

`df.filter(regex='regex')`
Select columns whose name matches regular expression *regex*.

regex (Regular Expressions) Examples

`'\.'`	Matches strings containing a period '.'
`'Length$'`	Matches strings ending with word 'Length'
`'^Sepal'`	Matches strings beginning with the word 'Sepal'
`'^x[1-5]$'`	Matches strings beginning with 'x' and ending with 1,2,3,4,5
`'^(?!Species$).*'`	Matches strings except the string 'Species'

`df.loc[:,'x2':'x4']`
Select all columns between x2 and x4 (inclusive).

`df.iloc[:,[1,2,5]]`
Select columns in positions 1, 2 and 5 (first column is 0).

`df.loc[df['a'] > 10, ['a','c']]`
Select rows meeting logical condition, and only the specific columns .

Logic in Python (and pandas)

<	Less than	!=	Not equal to
>	Greater than	`df.column.isin(values)`	Group membership
==	Equals	`pd.isnull(obj)`	Is NaN
<=	Less than or equals	`pd.notnull(obj)`	Is not NaN
>=	Greater than or equals	&,|,~,^,`df.any()`,`df.all()`	Logical and, or, not, xor, any, all

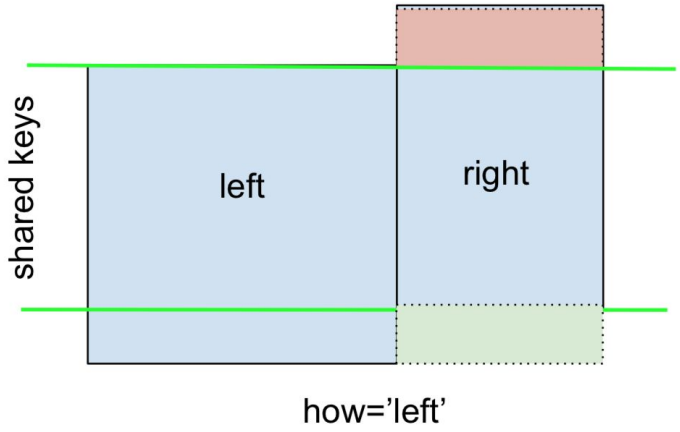
http://pandas.pydata.org/ This cheat sheet inspired by Rstudio Data Wrangling Cheatsheet (https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf) Written by Irv Lustig, Princeton Consultants

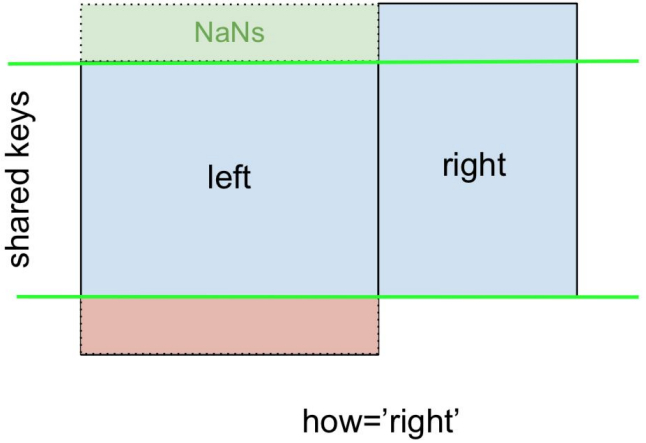
https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf

DataFrame - Merging Data

Merge DataFrame with a database-style join.

- left join
- right join

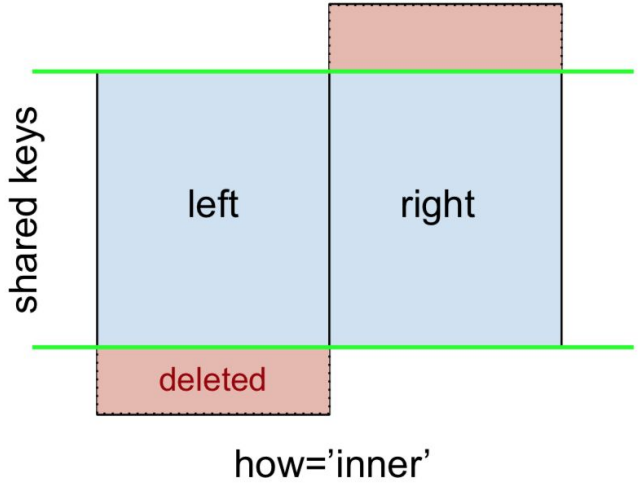


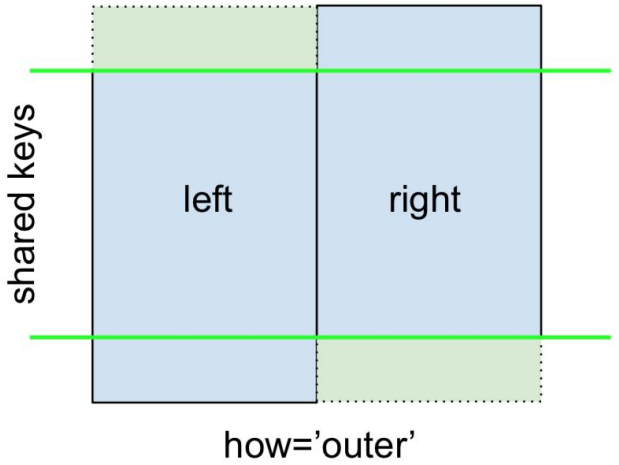


DataFrame - Merging Data

Merge DataFrame with a database-style join.

- inner join
- outer join





APIs

Python for Economics

Afternoon, Aug 17, 2023

Richard Lawrence

This Module

1. JSON Format
2. Requests
3. FRED API

4. Capstone Project
5. ACES

JSON - JavaScript Object Notation

- A text format for storing data
- language-independent (why they should know json)

JSON string examples:

```
'{"name":"Jack", "age":20, "major":"computer science"}'
```

```
'{  "args": {},   "data": "",   "files": {},   "form": {
"soup": "hot soup"  },  .... }'
```

loads() function -> Python dictionary

JSON module

- Python built-in module **json**
- `json.loads()`: converts JSON string to Python dictionary
- Example:

```
import json
text = "{ keys : values, ... }"
dict = json.loads(text)
```

Requests

Requests library for HTTP activities.

Replicate the experience of visiting a web page,
but in a Notebook instead of a Browser.

FRED API

Accessing Federal Reserve Economic Data

Web Scraping API Exercise

Fred API
Retrieve economic data from the FRED® and ALFRED® websites hosted by the Economic Research Division of the Federal Reserve Bank of St. Louis

Reference: https://fred.stlouisfed.org/docs/api/fred/

Get an API Key

Register and log into your fredaccount.stlouisfed.org user account and request your API Key.

Most web services require an API key to identify who owns a request.

Break Time Reminder Slide

10 minutes break

Capstone

Putting It All Together

ACES

Supercomputing Cluster at Texas A&M

ACES

Accelerating Computing for Emerging Sciences

Our Mission:

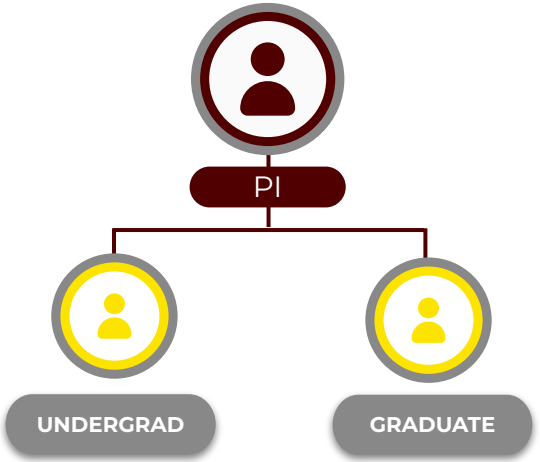
- Offer an accelerator testbed for numerical simulations and **AI/ML workloads**
- Provide consulting, technical guidance, and training to researchers
- Collaborate on computational and data-enabled research.

ACES

ACCELERATING COMPUTING FOR EMERGING SCIENCES

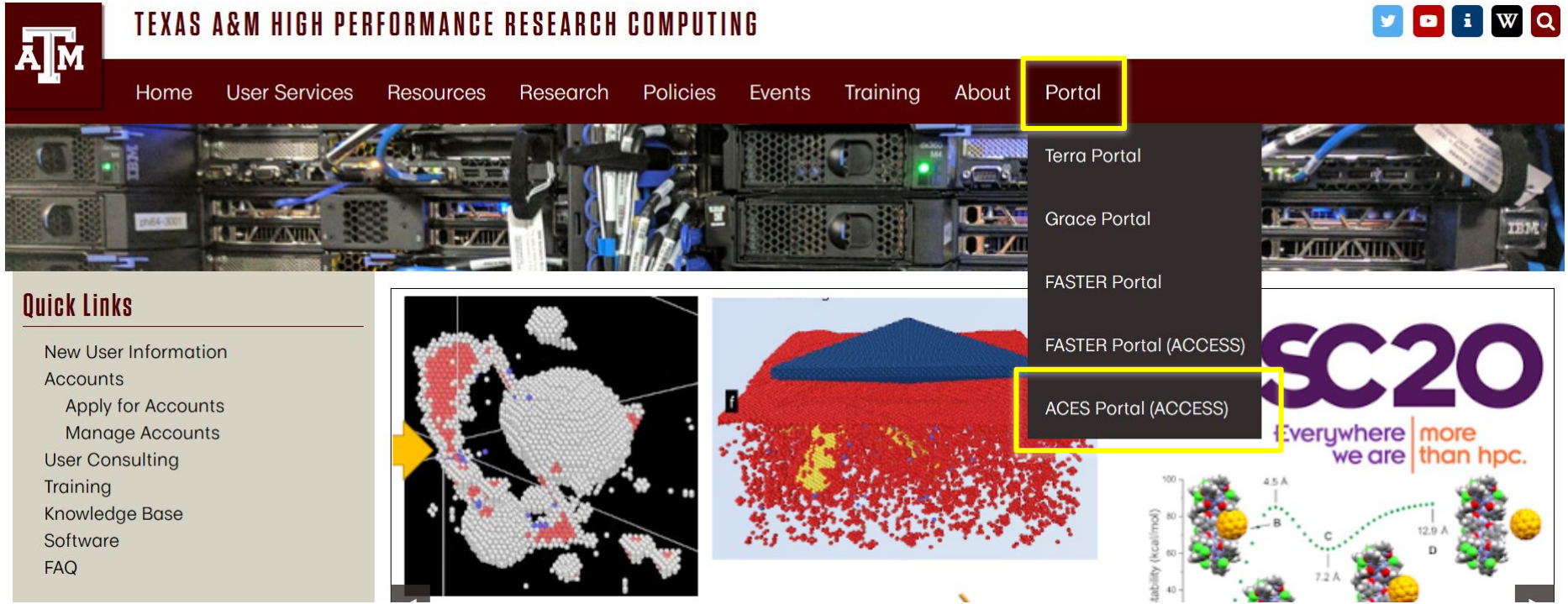
Getting on ACES

- You must have an ACCESS account (we did this yesterday)
- PI's can apply for allocations directly
- Students will use our training allocation today
- Email us at help@hprc.tamu.edu for questions, comments, and concerns.



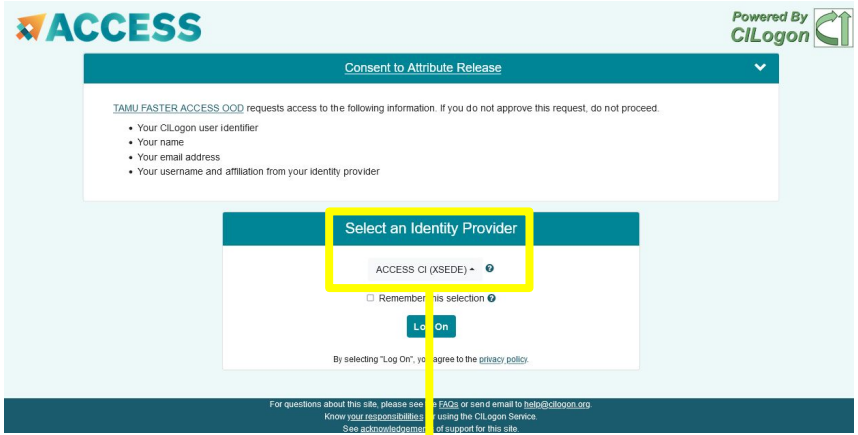
PIs can apply for an account and sponsor accounts for their researchers.

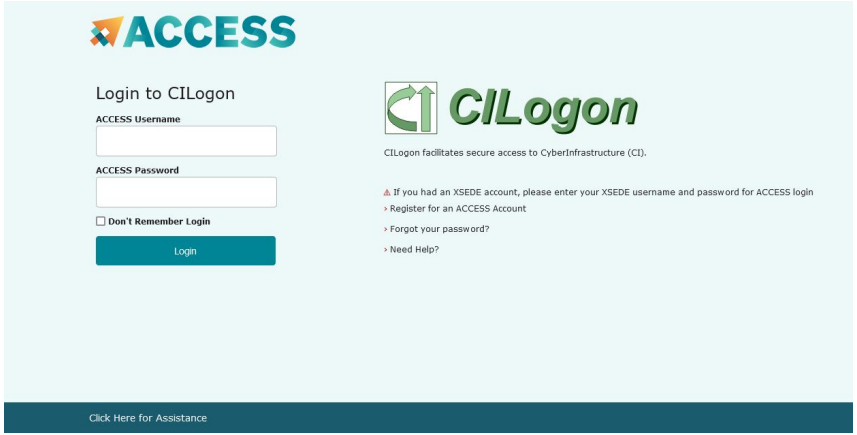
HPRC Portal

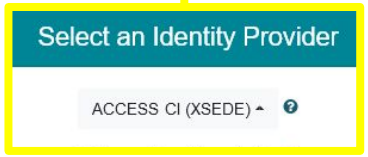

TEXAS A&M HIGH PERFORMANCE RESEARCH COMPUTING
Home
User Services
Resources
Research
Policies
Events
Training
About
Portal
Terra Portal
Grace Portal
FASTER Portal
FASTER Portal (ACCESS)
ACES Portal (ACCESS)
Quick Links
New User Information
Accounts
Apply for Accounts
Manage Accounts
User Consulting
Training
Knowledge Base
Software
FAQ
SC20
Everywhere we are | more than hpc.

Accessing ACES via the HPRC Portal (ACCESS)

Log-in using your ACCESS credentials.





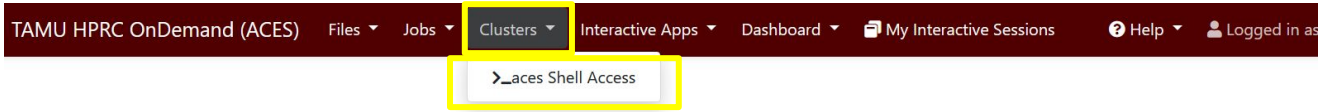


Select the Identity Provider appropriate for your account.
You need an ACCESS account, but can choose to log in with your TAMU NetID here.

Shell access via the HPRC Portal

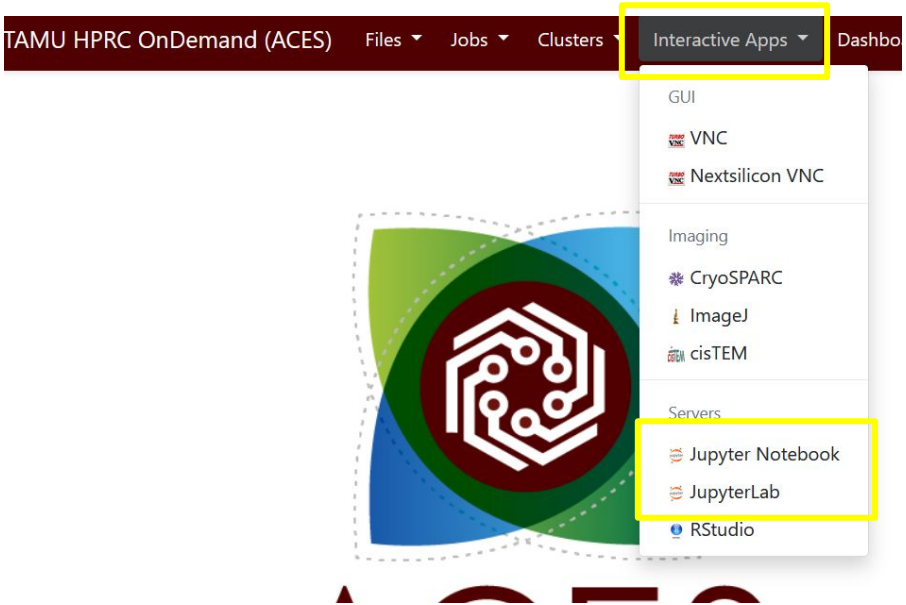
Access through (most) web browsers

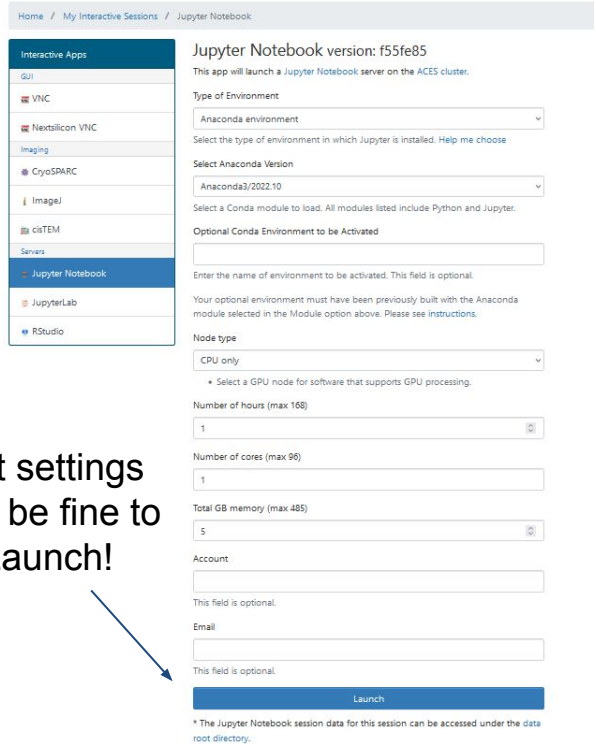
–Top Banner Menu “Clusters” -> “Shell Access”



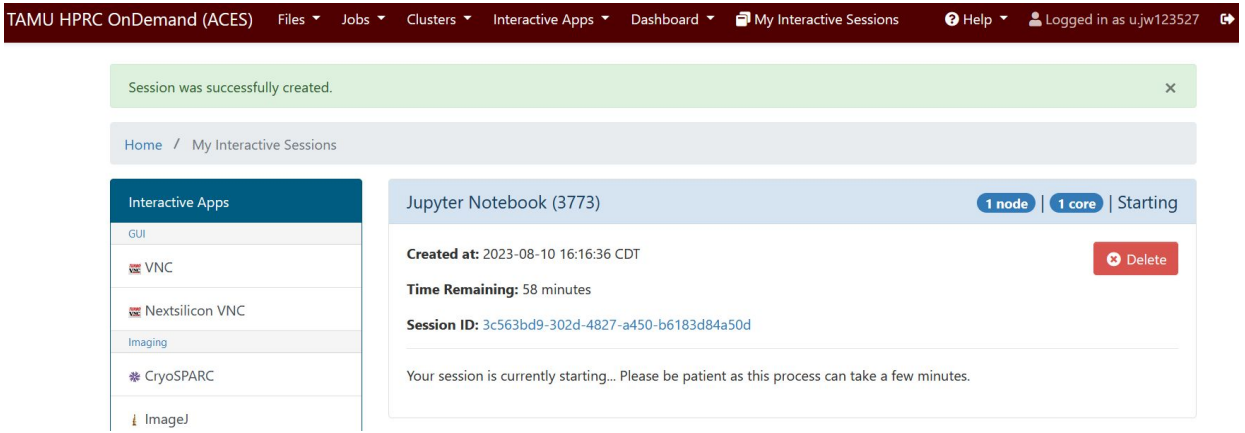
ACES

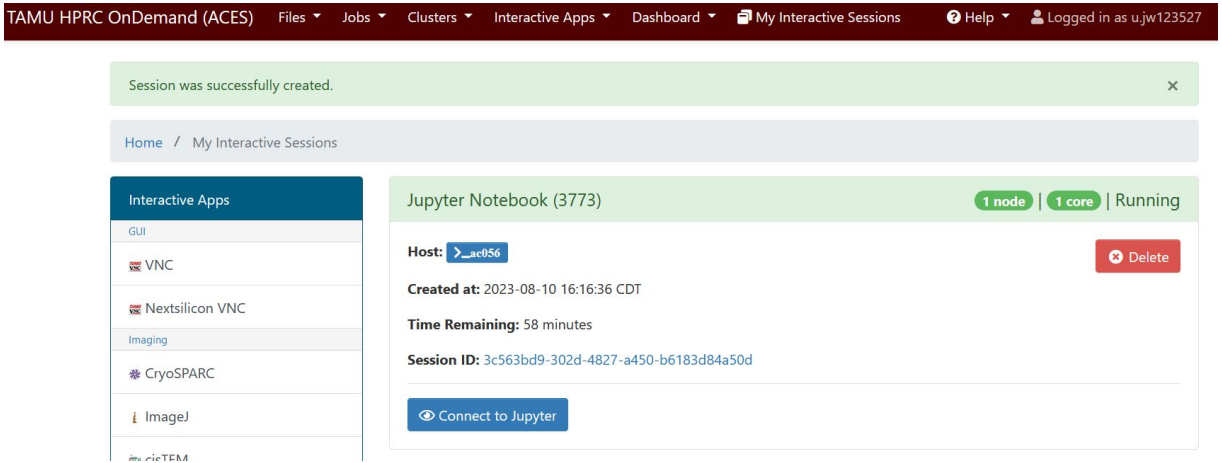
Accessing Jupyter Notebooks on ACES





Default settings should be fine to start. Launch!


TAMU HPRC OnDemand (ACES)
Files
Jobs
Clusters
Interactive Apps
Dashboard
My Interactive Sessions
Help
Logged in as u.jw123527
Session was successfully created.
Home / My Interactive Sessions
Interactive Apps
GUI
VNC
Nextsilicon VNC
Imaging
CryoSPARC
ImageJ
Jupyter Notebook (3773)
1 node | 1 core | Starting
Created at: 2023-08-10 16:16:36 CDT
Delete
Time Remaining: 58 minutes
Session ID: 3c563bd9-302d-4827-a450-b6183d84a50d
Your session is currently starting... Please be patient as this process can take a few minutes.


TAMU HPRC OnDemand (ACES)
Files
Jobs
Clusters
Interactive Apps
Dashboard
My Interactive Sessions
Help
Logged in as u.jw123527
Session was successfully created.
Home / My Interactive Sessions
Interactive Apps
GUI
VNC
Nextsilicon VNC
Imaging
CryoSPARC
ImageJ
Jupyter Notebook (3773)
1 node | 1 core | Running
Host: >_ac056
Delete
Created at: 2023-08-10 16:16:36 CDT
Time Remaining: 58 minutes
Session ID: 3c563bd9-302d-4827-a450-b6183d84a50d
Connect to Jupyter