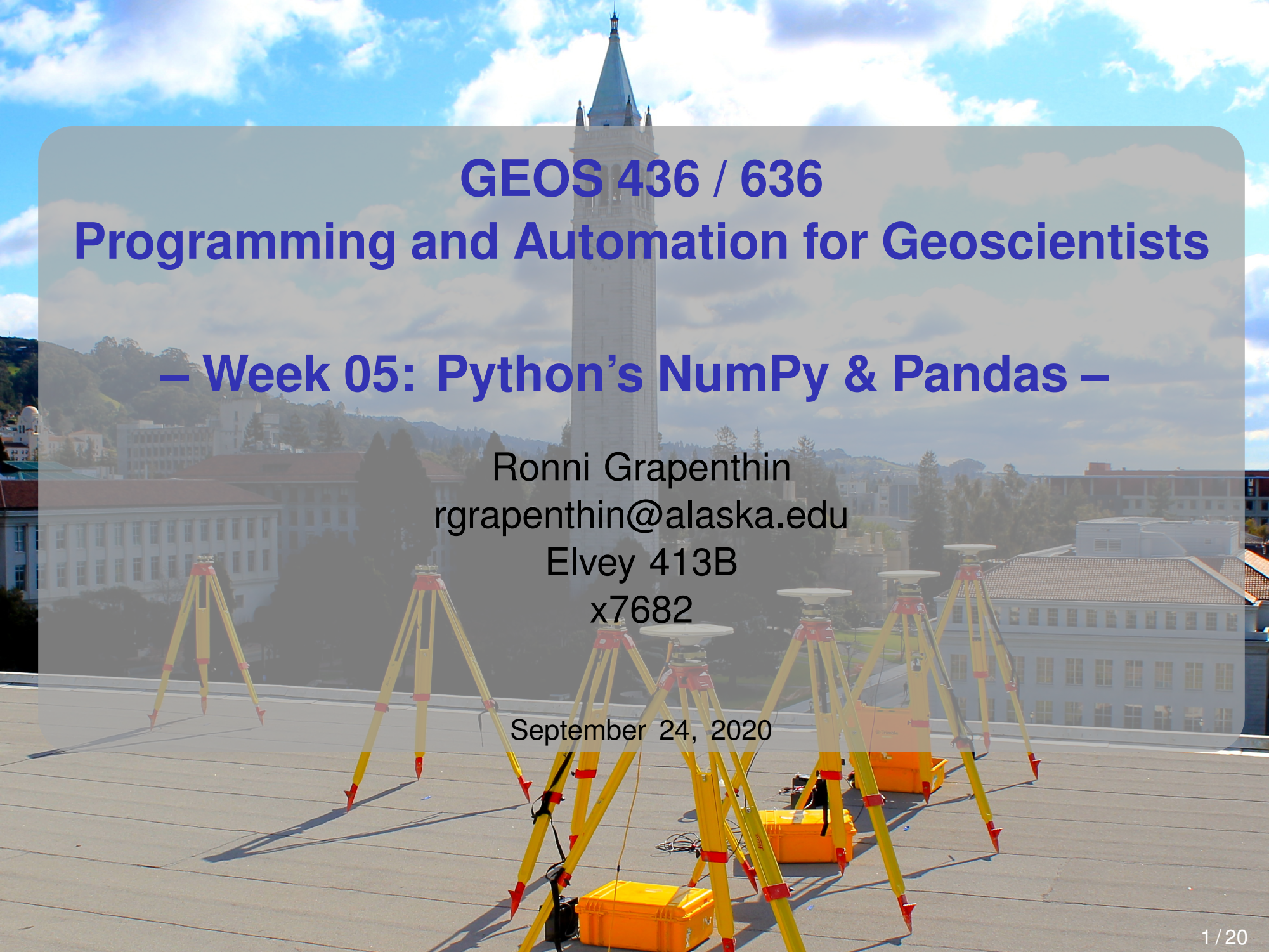


GEOS 436 / 636
Programming and Automation for Geoscientists

– Week 05: Python's NumPy & Pandas –

Ronni Grapenthin
rgrapenthin@alaska.edu
Elvey 413B
x7682

September 24, 2020

Key concept in programming and automation: Use existing tools!

Advanced Data Types

- We've dealt with a few basic data types so far: `int, boolean, float, ...`
- You know about more complex data types like lists (arrays), dictionaries in Python
- Most of your data are some form of table where measurements (float values) are connected to moments in time or locations in space (other float values)
- Even images fall in this category
- Have data types that provide efficient storage and operation on such data: Numpy array, Pandas Series & DataFrame

NumPy

- “The fundamental package for scientific computing with Python”
- It’s a library (or package) that expands the functionality of core-Python
- Provides multidimensional array (`ndarray`) object (and efficient storage)
- Provides functions that efficiently (read fast) operate on this data type
- As opposed to the Python list, dictionary types, data in `ndarray` must be homogeneous, i.e. of the same type
- Large infrastructure of Python-based packages use `ndarray` internally, being proficient in its use is necessary to leverage those packages

NumPy

NumPy = `ndarray` + functions

NumPy Overview

POWERFUL N-DIMENSIONAL ARRAYS

Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.

NUMERICAL COMPUTING TOOLS

NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

INTEROPERABLE

NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.

PERFORMANT

The core of NumPy is well-optimized C code. Enjoy the flexibility of Python with the speed of compiled code.

EASY TO USE

NumPy's high level syntax makes it accessible and productive for programmers from any background or experience level.

OPEN SOURCE

Distributed under a liberal BSD license, NumPy is developed and maintained publicly on GitHub by a vibrant, responsive, and diverse community.

numpy.org

NumPy Ecosystem

Quantum Computing	Statistical Computing	Signal Processing	Image Processing	3-D Visualization	Symbolic Computing	Astronomy Processes	Cognitive Psychology
							
QuTiP PyQuil Qiskit	Pandas statsmodels Seaborn	SciPy PyWavelets	Scikit-image OpenCV	Mayavi Napari	SymPy	AstroPy SunPy SpacePy	PsychoPy

Bioinformatics	Bayesian Inference	Mathematical Analysis	Simulation Modeling	Multi-variate Analysis	Geographic Processing	Interactive Computing
						
BioPython Scikit-Bio PyEnsembl	PyStan PyMC3	SciPy SymPy cvxpy FEniCS	PyDSTool	PyChem	Shapely GeoPandas Folium	Jupyter IPython Binder

numpy.org

NumPy Array Programming

- Array Programming: captures the operations on arrays / matrices in intuitive syntax (closely related to the math that's trying to be expressed)
- Instead of looping over arrays to perform operations, NumPy efficiently implements the fundamental operations internally
- Results in concise, easy to-read code
- Reduces bugs, enhances reproducibility

NumPy Array Programming

```
import numpy as np
import timeit

#generate random data (1Mx1 entries)
a = np.random.rand(1000000,1)
b = 3.0

#time the runtime of the loop
start1 = timeit.default_timer()
for i in range(len(a)):
    a[i]=a[i]*3
stop1 = timeit.default_timer()

#time the runtime of the numpy operation
start2 = timeit.default_timer()
a = a*b
stop2 = timeit.default_timer()

#compare the two
print ("Time loop: %.5f s" % (stop1-start1))
print ("Time np  : %.5f s" % (stop2-start2))
```

```
Time loop: 1.39619 s
Time np  : 0.00164 s
```

ORDERS OF MAGNITUDE DIFFERENCE!

NumPy Basics

```
import numpy as np

#create an ndarray
data = np.array([4,9,16,25])

#calculate the sum of the elements
#(call the function sum() on the OBJECT
#data)
print("sum=", data.sum())

#calculate sqrt, hand data to
#numpy function
print("sqrt=", np.sqrt(data))

#calculate vector norm: norm = sqrt(sum(x(i)^2))
print("norm=",np.linalg.norm(data))
```

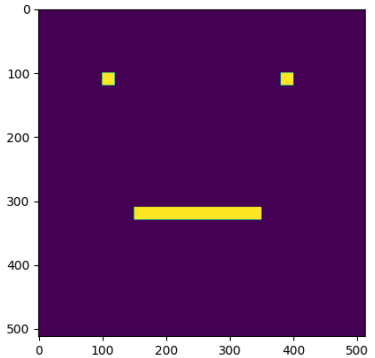
```
sum= 54
sqrt= [2. 3. 4. 5.]
norm= 31.272991542223778
```

NumPy Basic Matrix

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

width  = 512
height = 512

#create an empty 2D matrix of type int
data = np.zeros( (512,512), dtype=np.uint8)

#put some data in some places
data[100:120, 100:120] = 1
data[100:120, 380:400] = 1
data[310:330, 150:350] = 1

#display data as an image
plt.imshow(data)

#show on the screen
plt.show()
```

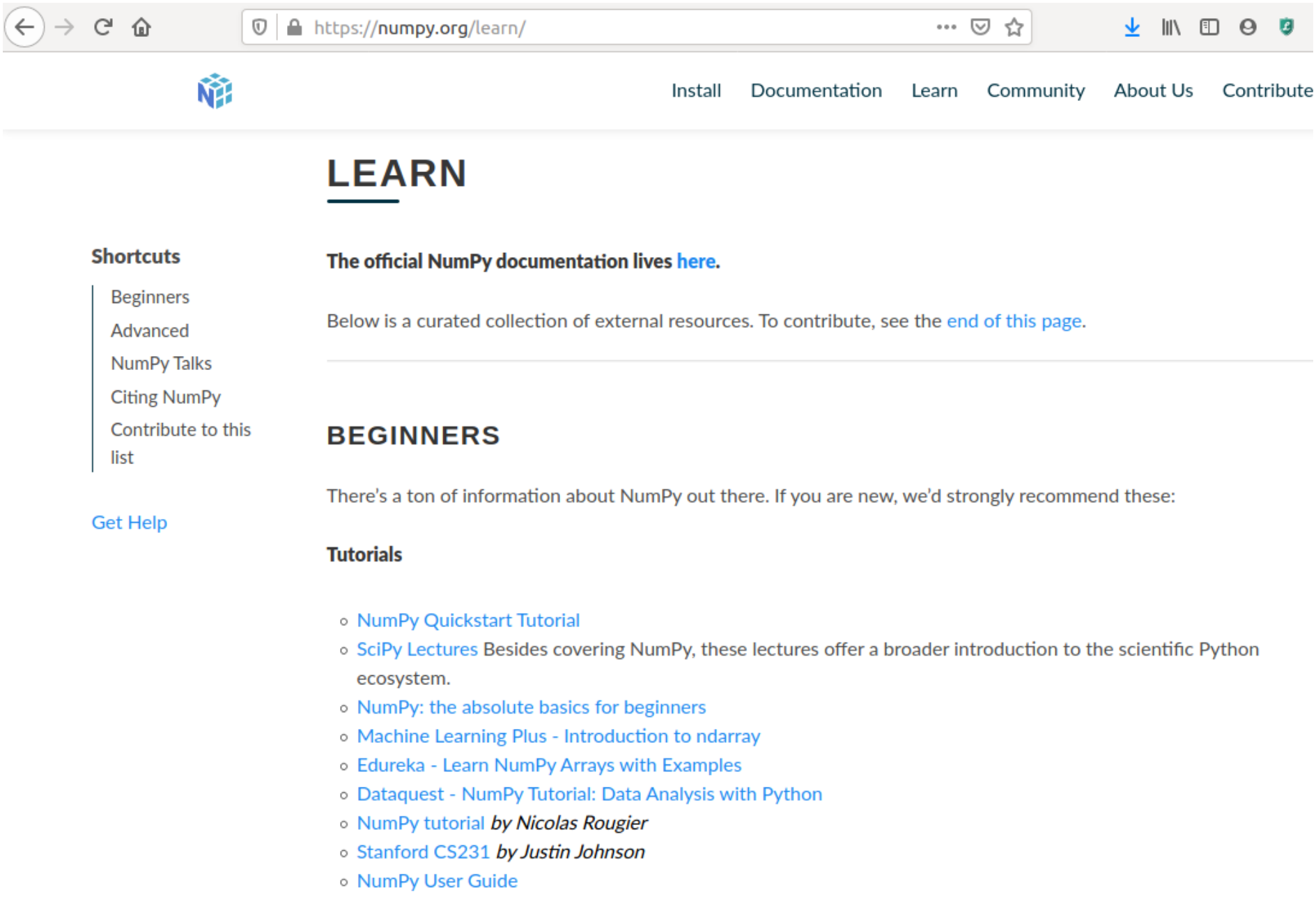
NumPy Basic Matrix

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

width  = 512
height = 512

#create an empty 2D matrix of type int
data = np.zeros( (512,512), dtype=np.uint8)

#put some data in some places
data[100:120, 100:120] = 1
data[100:120, 380:400] = 1
data[310:330, 150:350] = 1

#display data as an image
plt.imshow(data)

#show on the screen
plt.show()
```



Output

PLAY!


https://numpy.org/learn/
Install
Documentation
Learn
Community
About Us
Contribute
LEARN
Shortcuts
Beginners
Advanced
NumPy Talks
Citing NumPy
Contribute to this list
Get Help
The official NumPy documentation lives here.
Below is a curated collection of external resources. To contribute, see the end of this page.
BEGINNERS
There's a ton of information about NumPy out there. If you are new, we'd strongly recommend these:
Tutorials
NumPy Quickstart Tutorial
SciPy Lectures Besides covering NumPy, these lectures offer a broader introduction to the scientific Python ecosystem.
NumPy: the absolute basics for beginners
Machine Learning Plus - Introduction to ndarray
Edureka - Learn NumPy Arrays with Examples
Dataquest - NumPy Tutorial: Data Analysis with Python
NumPy tutorial by Nicolas Rougier
Stanford CS231 by Justin Johnson
NumPy User Guide

Pandas

Mission (from https://pandas.pydata.org)

pandas aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language.

Pandas Data Structures

- `Series` - one-dimensional labeled array holding any data type
- `DataFrame` - two-dimensional labeled data structure with columns of potentially different types (think spreadsheet), most commonly used pandas data type

Pandas Data Structures

```
import numpy as np
import pandas as pd

s = pd.Series([23, 3, 2.3, 1, 2.3332],
              index=['a', 'b', 'c', 'd', 'e'])

print("Series:")
print(s)

df = pd.DataFrame(np.random.randn(6, 4),
              columns=['A', 'B', 'C', 'D'])

print("\nDataFrame:")
print(df)
```

Pandas Data Structures

```
import numpy as np
import pandas as pd

s = pd.Series([23, 3, 2.3, 1, 2.3332],
              index=['a', 'b', 'c', 'd', 'e'])

print("Series:")
print(s)

df = pd.DataFrame(np.random.randn(6, 4),
              columns=['A', 'B', 'C', 'D'])

print("\nDataFrame:")
print(df)
```

```
Series:
a    23.0000
b     3.0000
c     2.3000
d     1.0000
e     2.3332
dtype: float64

DataFrame:
          A         B         C         D
0 -1.135908  0.391234  1.584341  0.963969
1  0.145715  1.121390  0.875102  0.373681
2 -0.933898  1.151053  1.749787  1.153207
3 -1.416136  0.436026  0.600013  0.863668
4 -0.463835  1.284227  3.251272  0.776569
5  0.202191  0.570245 -0.850101 -1.021765
```

Pandas Data Frame and Time

```
import pandas as pd
import numpy as np

#we want 10 data
num_data = 10

#create 10 days starting on the day
#this lecture is published
index = pd.date_range('09/24/2020',
                periods=num_data)

#make the (random) data frame of
#num_data rows and 3 colums labled
#north, east and vertical;
#indexed by the time series created above
df=pd.DataFrame(np.random.randn(num_data, 3),
        index=index,
        columns=('North', 'East', 'Vertical'))

print(df)

#print summary statistics
print(df.describe())
```

Pandas Data Frame and Time

```
import pandas as pd
import numpy as np

#we want 10 data
num_data = 10

#create 10 days starting on the day
#this lecture is published
index = pd.date_range('09/24/2020',
                periods=num_data)

#make the (random) data frame of
#num_data rows and 3 colums labled
#north, east and vertical;
#indexed by the time series created above
df=pd.DataFrame(np.random.randn(num_data, 3),
        index=index,
        columns=('North', 'East', 'Vertical'))

print(df)

#print summary statistics
print(df.describe())
```

```
                North      East  Vertical
2020-09-24 -2.329823 -1.709525 -0.071239
2020-09-25 -1.127769  0.390906  1.099989
2020-09-26 -0.486678 -0.123021  0.176733
2020-09-27 -0.676743  1.153277 -0.365258
2020-09-28  0.548767  0.040909  1.585026
2020-09-29  0.955241 -0.077114 -1.783357
2020-09-30 -0.481305 -0.985396 -0.782370
2020-10-01 -1.598926  0.625368  0.079308
2020-10-02 -2.165262 -1.692497  0.570225
2020-10-03 -0.982975  0.485549  0.219306
           North       East   Vertical
count  10.000000  10.000000  10.000000
mean   -0.834547  -0.189154   0.072836
std     1.058643   0.973692   0.943455
min    -2.329823  -1.709525  -1.783357
25%    -1.481137  -0.769802  -0.291753
50%    -0.829859  -0.018102   0.128020
75%    -0.482648   0.461888   0.482495
max     0.955241   1.153277   1.585026
```

PLAY!

https://pandas.pydata.org/docs/getting_started/intro_tutorials/09_timeseries.html

pandas

Getting started | User Guide | API reference | Development | Release notes

Search the docs ...

- Installation
- Package overview
- **Getting started tutorials**
  - What kind of data does pandas handle?
  - How do I read and write tabular data?
  - How do I select a subset of a DataFrame?
  - How to create plots in pandas?
  - How to create new columns derived from existing columns?
  - How to calculate summary statistics?
  - How to reshape the layout of tables?
  - How to combine data from multiple tables?
  - **How to handle time series data with ease?**
  - How to manipulate textual data?
- Comparison with other tools
- Community tutorials

```
In [1]: import pandas as pd

In [2]: import matplotlib.pyplot as plt
```

Data used for this tutorial:

Air quality data

```
In [3]: air_quality = pd.read_csv("data/air_quality_no2_long.csv")

In [4]: air_quality = air_quality.rename(columns={"date.utc": "datetime"})

In [5]: air_quality.head()
Out[5]:
    city country                  datetime location parameter  value   unit
0  Paris      FR  2019-06-21 00:00:00+00:00  FR04014       no2   20.0  µg/m³
1  Paris      FR  2019-06-20 23:00:00+00:00  FR04014       no2   21.8  µg/m³
2  Paris      FR  2019-06-20 22:00:00+00:00  FR04014       no2   26.5  µg/m³
3  Paris      FR  2019-06-20 21:00:00+00:00  FR04014       no2   24.9  µg/m³
4  Paris      FR  2019-06-20 20:00:00+00:00  FR04014       no2   21.4  µg/m³
```

```
In [6]: air_quality.city.unique()
Out[6]: array(['Paris', 'Antwerpen', 'London'], dtype=object)
```

How to handle time series data with ease?

Using pandas datetime properties