PREDICT 420

Atef Bader, PhD

Agenda

- Pandas Data Analysis Library
- A-MUST readings for DataFrame
- Exercise #2 Walkthrough & Deliverable

Pandas – Data Analysis Library

Pandas is a library of pakcgaes for Data Analysis in Python

File Edit View History Bookmarks Tools Help

Python Data Analysis Library ...

pandas.pydata.org

Search

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

Fork me on GitHub

overview // get pandas // documentation // community // talks

Python Data Analysis Library

pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

pandas is a NUMFocus sponsored project. This will help ensure the success of development of *pandas* as a world-class open-source project.

A Fiscally Sponsored Project of

NUMFOCUS

OPEN CODE = BETTER SCIENCE

VERSIONS

Release
0.19.2 - December 2016
download // docs // pdf

Development
0.20.0 - 2017
github // docs

Previous Releases
0.19.1 - download // docs // pdf
0.19.0 - download // docs // pdf
0.18.1 - download // docs // pdf
0.18.0 - download // docs // pdf

Pandas – Data Analysis Library

File Edit View History Bookmarks Tools Help

pandas 0.18.1 : Python Pac... | Python for Data Analysis -... | Python Data Analysis Library ...

pandas.pydata.org

What problem does *pandas* solve?

Python has long been great for data munging and preparation, but less so for data analysis and modeling. *pandas* helps fill this gap, enabling you to carry out your entire data analysis workflow in Python without having to switch to a more domain specific language like R.

Combined with the excellent IPython toolkit and other libraries, the environment for doing data analysis in Python excels in performance, productivity, and the ability to collaborate.

pandas does not implement significant modeling functionality outside of linear and panel regression; for this, look to statsmodels and scikit-learn. More work is still needed to make Python a first class statistical modeling environment, but we are well on our way toward that goal.

What do our users have to say?

Roni Israelov, PhD
Portfolio Manager

AOR CAPITAL MANAGEMENT

Here is the main theme behind pandas package.

That is, to do data munging, preparation,modeling, and analysis without the need to switch to R language

Pandas – Data Analysis Library

All you need to work on for a given dataset: slice/dice/index/groupby/CSV/SQL/NoSQL

File Edit View History Bookmarks Tools Help

Python Data Analysis Library ...

pandas.pydata.org

Search

Fork me on GitHub

Library Highlights

- A fast and efficient **DataFrame** object for data manipulation with integrated indexing;
- Tools for **reading and writing data** between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format;
- Intelligent **data alignment** and integrated handling of **missing data**: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form;
- Flexible **reshaping** and pivoting of data sets;
- Intelligent label-based **slicing**, **fancy indexing**, and **subsetting** of large data sets;
- Columns can be inserted and deleted from data structures for **size mutability**;
- Aggregating or transforming data with a powerful **group by** engine allowing split-apply-combine operations on data sets;

DataFrame – Required Reading

File Edit View History Bookmarks Tools Help

pandas 0.18.1 : Python Pac... | Python for Data Analysis -... | Python Data Analysis Library ...

Python Software Foundation (US) https://pypi.python.org/pypi/pandas

python™ search

» Package Index > pandas > 0.18.1

PACKAGE INDEX
- Browse packages
- Package submission
- List trove classifiers
- RSS (latest 40 updates)
- RSS (newest 40 packages)
- PyPI Tutorial
- PyPI Security
- PyPI Support
- PyPI Bug Reports
- PyPI Discussion
- PyPI Developer Info

ABOUT
NEWS
DOCUMENTATION
DOWNLOAD
COMMUNITY
FOUNDATION
CORE DEVELOPMENT

pandas 0.18.1

Powerful data structures for data analysis, time series,and statistics

Downloads ↓

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It 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**. It is already well on its way toward this goal.

Not Logged In
- Login
- Register
- Lost Login?
- Use OpenID
- Login with Google

Status
- Nothing to report

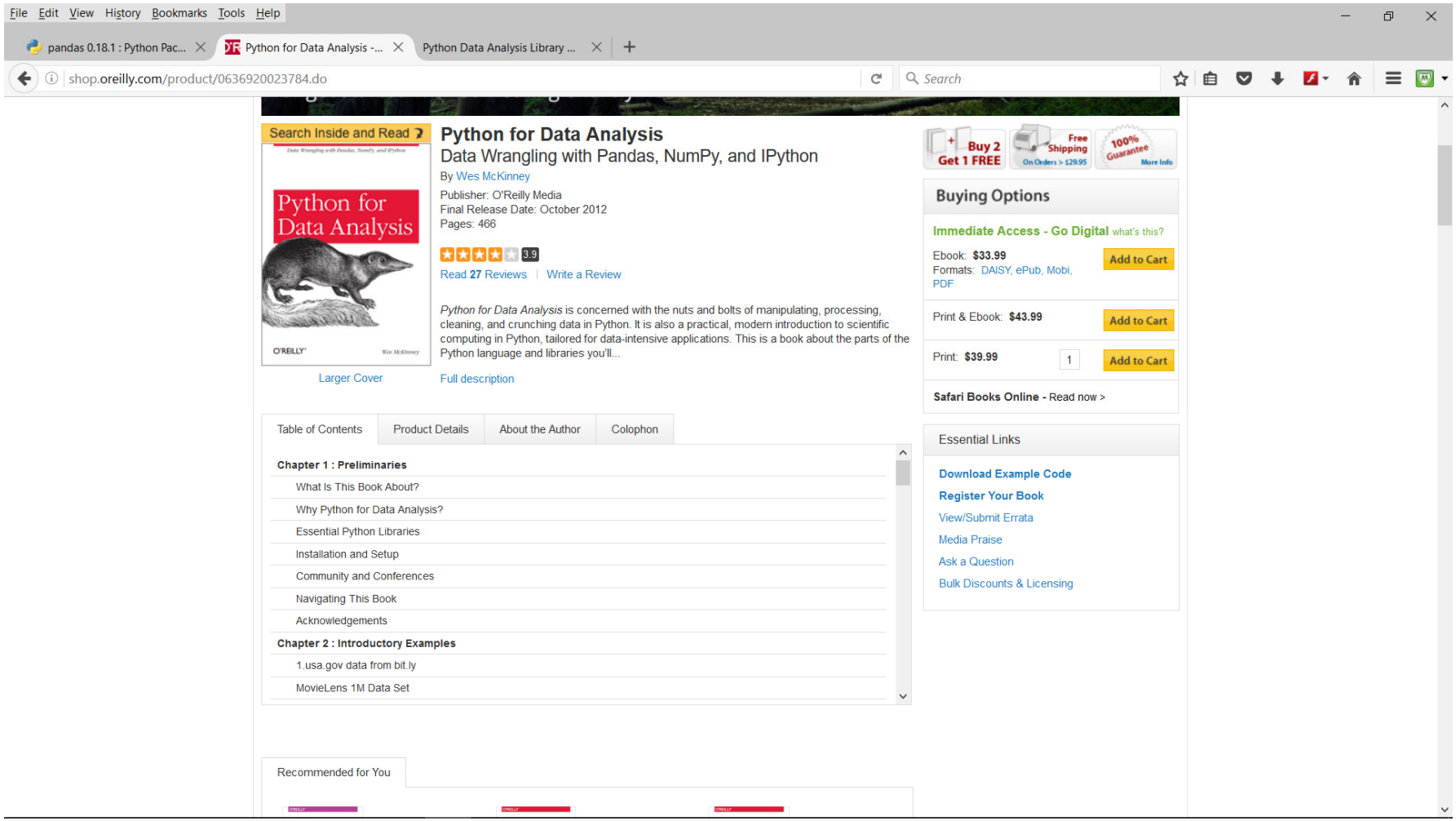
pandas is well suited for many different kinds of data:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
- Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure

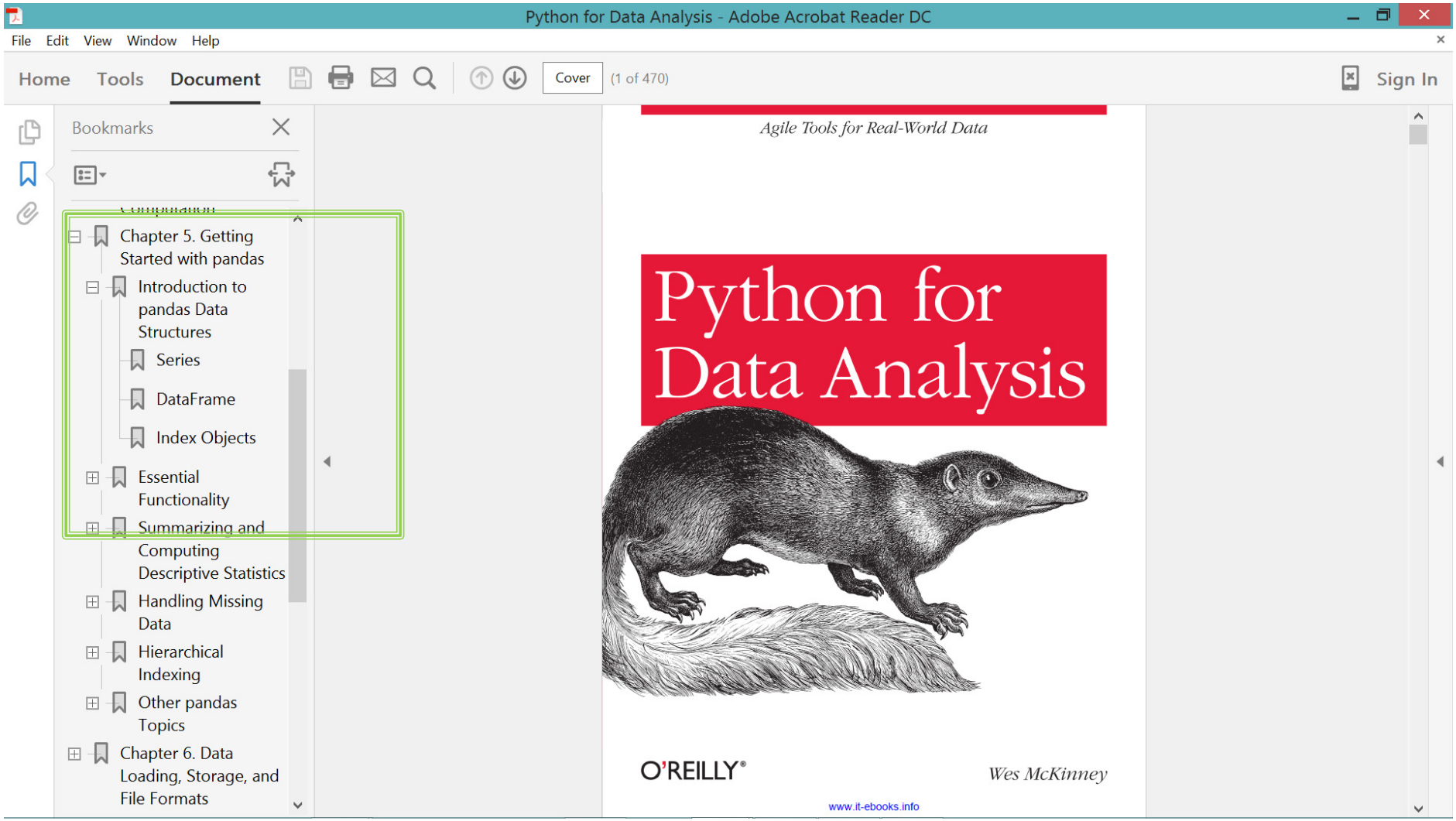
The two primary data structures of pandas, Series (1-dimensional) and DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, DataFrame provides everything that R's `data.frame` provides and much more. pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

Here are just a few of the things that pandas does well:

DataFrame – Required Reading



DataFrame – Required Reading



DataFrame – API

File Edit View History Bookmarks Tools Help

pandas.DataFrame — pandas 0...

pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html

Search

pandas 0.16.2 documentation » API Reference » previous | next | modules | index

Table Of Contents

What's New
Installation
Contributing to pandas
Frequently Asked Questions (FAQ)
Package overview
10 Minutes to pandas
Tutorials
Cookbook
Intro to Data Structures
Essential Basic Functionality
Working with Text Data
Options and Settings
Indexing and Selecting Data
MultiIndex / Advanced Indexing
Computational tools
Working with missing data
Group By: split-apply-combine
Merge, join, and concatenate
Reshaping and Pivot Tables

pandas.DataFrame

class pandas.**DataFrame**(*data=None*, *index=None*, *columns=None*, *dtype=None*, *copy=False*)

Two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). Arithmetic operations align on both row and column labels. Can be thought of as a dict-like container for Series objects. The primary pandas data structure

Parameters:

data : *numpy ndarray (structured or homogeneous), dict, or DataFrame*

Dict can contain Series, arrays, constants, or list-like objects

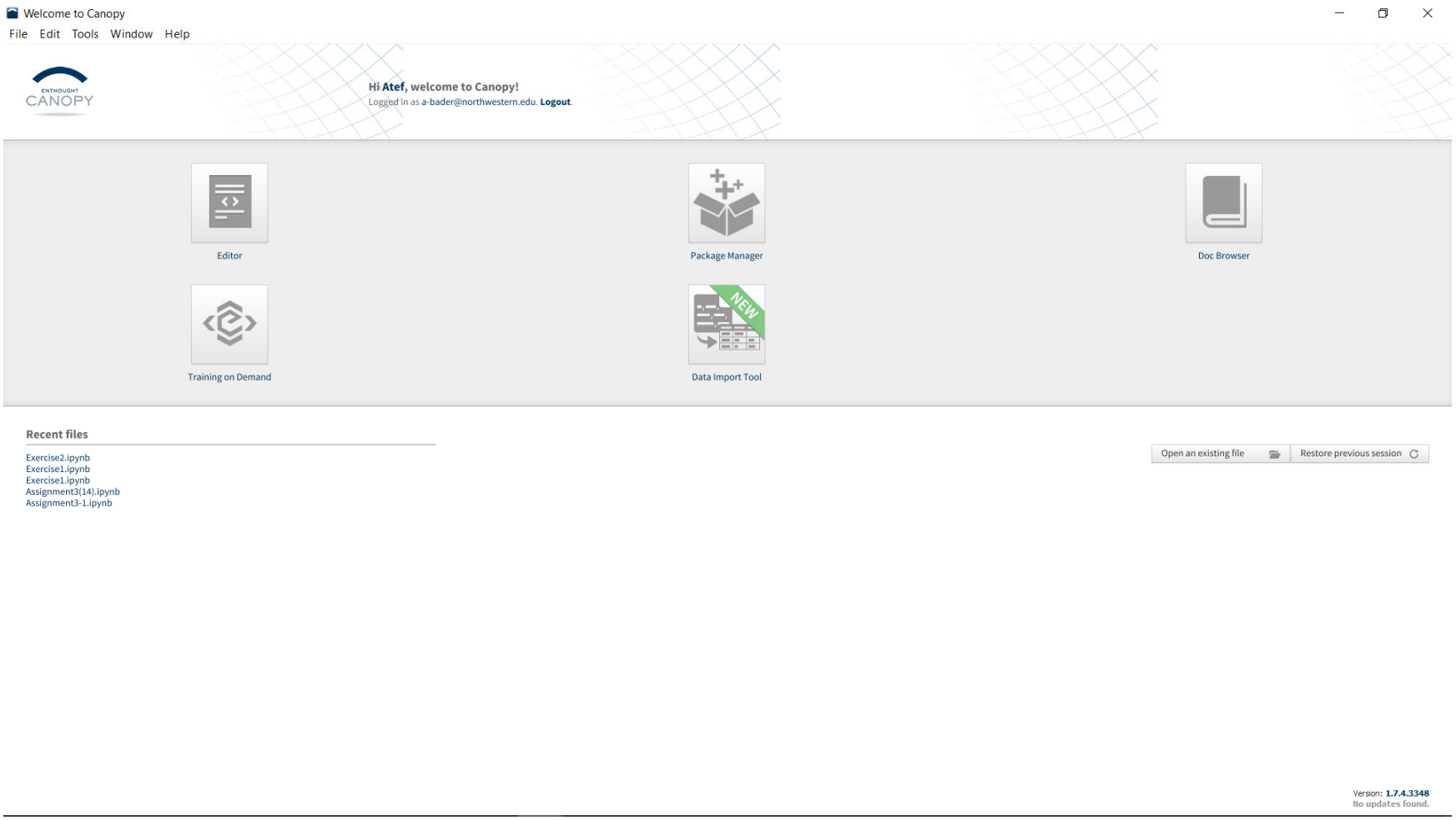
index : *Index or array-like*

Index to use for resulting frame. Will default to np.arange(n) if no indexing information part of input data and no index provided

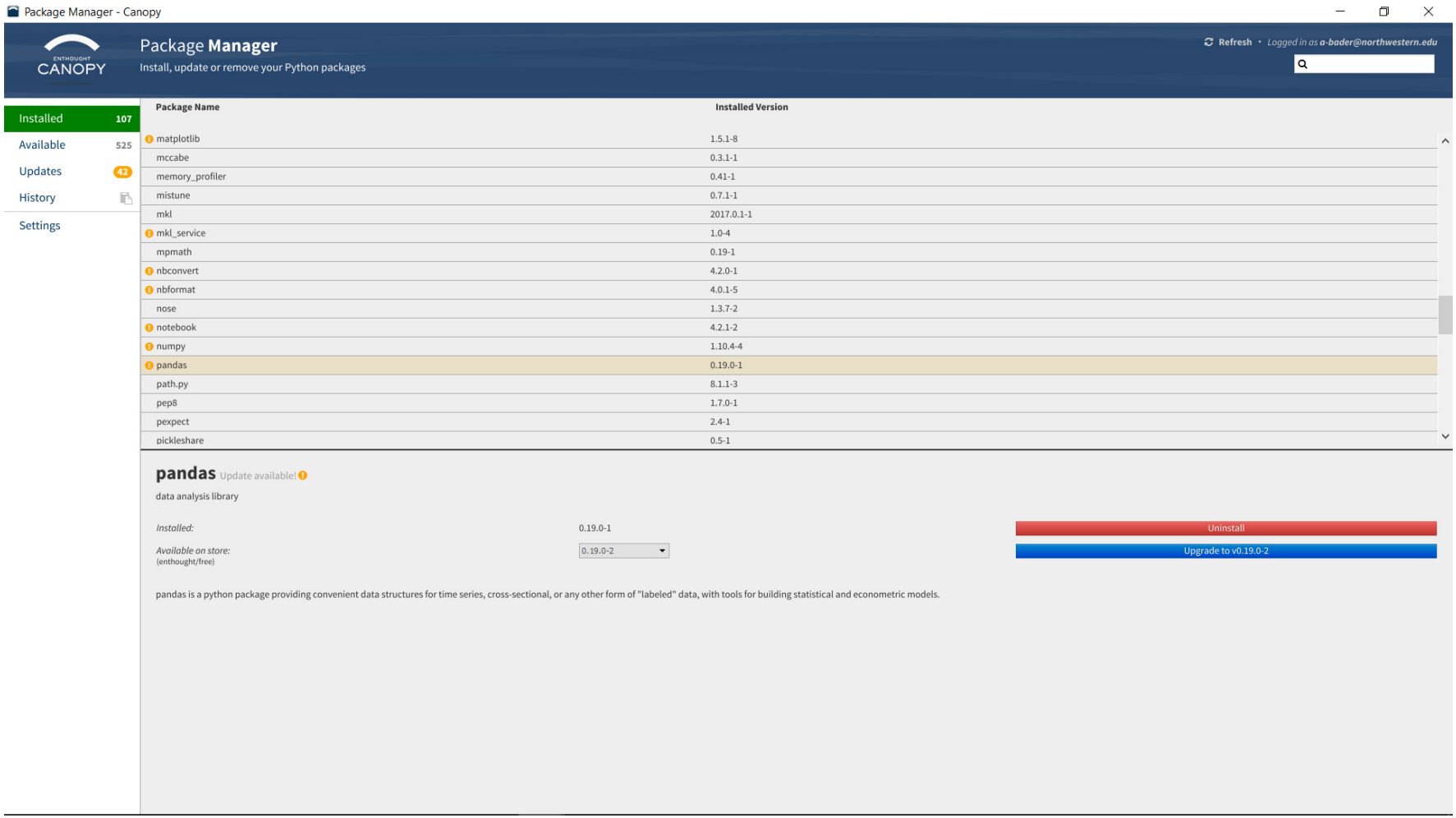
columns : *Index or array-like*

Column labels to use for resulting frame. Will default to np.arange(n) if no column labels are provided

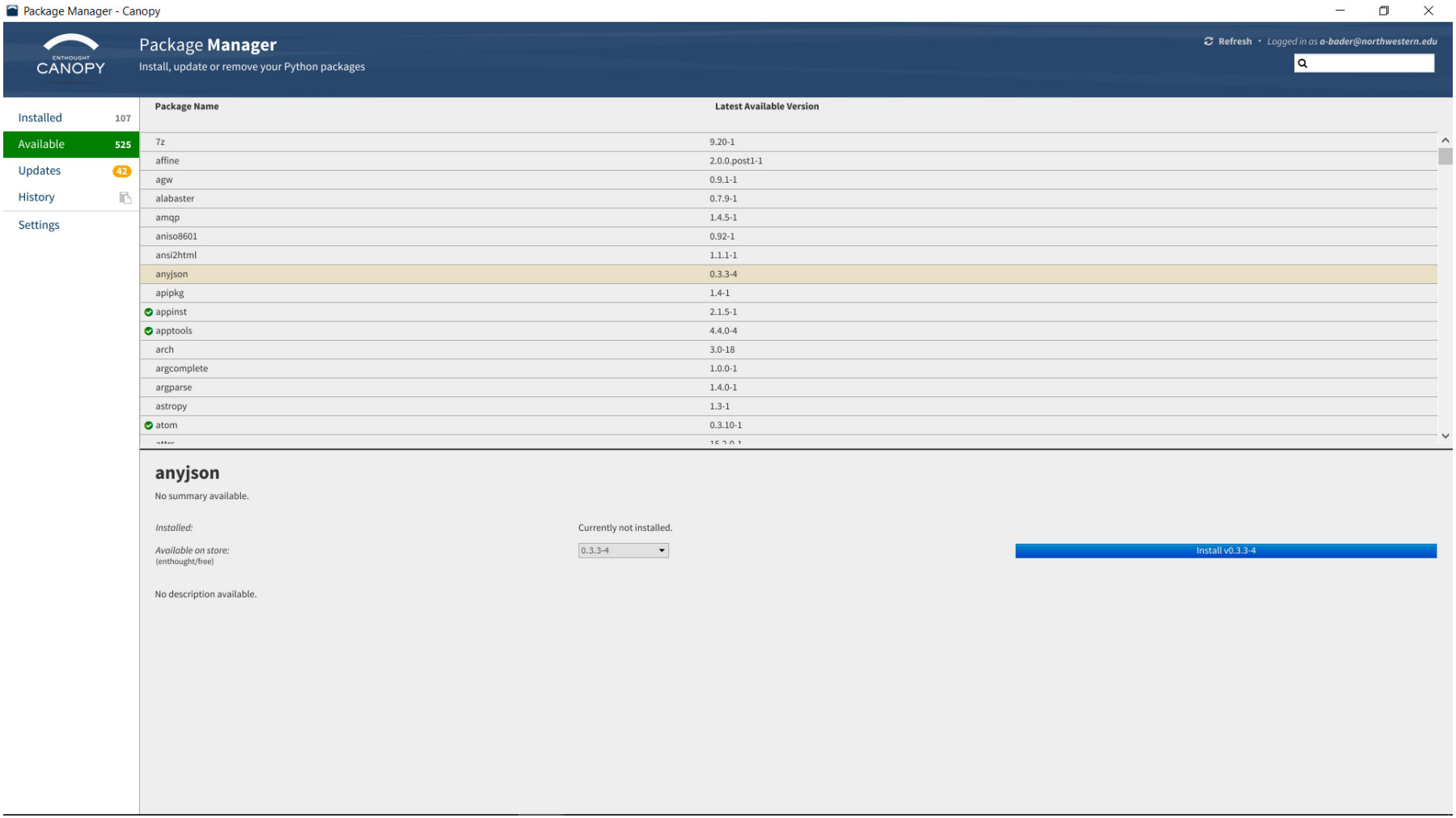
Package Manager



Package Manager – Installed Packages



Package Manager – Available Packages



Exercise #2

File Edit View History Bookmarks Tools Help

exercise2

localhost:8890/notebooks/bader/nu/420/spring 2016/sync sessions/sync session 2/exercisepractice2/exercise2

jupyter exercise2 Last Checkpoint: 04/05/2015 (autosaved)

File Edit View Insert Cell Kernel Help Python 2

Markdown Cell Toolbar: None

Week #2 - Python Practice

Some Ins and Outs with Python

There are many different kinds of data to be managed and analyzed today, and there are many ways to do it using Python. Being able to manage and modify data isn't useful unless you can also get data into Python, and save your results from it. Beginning with this session we're going to review techniques for input and output from Python starting with the simplest file formats and some basic Python tools. We'll also take a first look at the pandas package. Pandas has become very popular amongst "Pythonic" data scientists and is being used at the largest of the big data firms. In the sessions that follow we'll consider more complicated file types and data "munging" tools and techniques.

Time (and the term) is a'wasting, so lets crank up your Canopy environment and get to work with some examples. In what follows it's assumed that you are using the Canopy "Editor," which includes a version of IPython, and so the command prompts will look like In '[n]:' without the single quotes. What "n" is depends on where you are in the current session. As I'm sure you realize by now since you have most likely use Canopy or IPython before, for many "Ins" in IPython there is an "Out" of some kind which is a result of the "In" that precedes it.

So, let's start with flat files. A flat file is just a file that's, well, flat. It's typically a string of characters that may include end of line markers like a newline or carriage return code. Let's write a simple flat file out to disk by entering the following command:

In [32]:
```
outfile = open('myflatfile.txt','w')  # open to write to a text file
```

Note: "In [1]:" represents the command prompt in your IPython session. Depending on what you've been doing in a session, the digit or digits you see in it will vary. But you knew that, right?

Exercise #2

File Edit View History Bookmarks Tools Help

exercise2

localhost:8890/notebooks/bader/nu/420/spring 2016/sync sessions/sync session 2/exercisepractice2/exercise2

jupyter exercise2 Last Checkpoint: 04/05/2015 (autosaved)

File Edit View Insert Cell Kernel Help Python 2

Markdown Cell Toolbar: None

DataFrames and Series have many useful attributes and features, some of which we'll explore in upcoming exercises. But now let's try reading a less trivial csv file into a DataFrame. The file is xyzcust10.csv, and it should be available to you on Canvas. Take a look at it with your favorite text editor. Then, put it in a place you can find it from Canvas, and input it into a DataFrame:

In [59]:
```
xyzcust10=pd.read_csv('xyzcust10.csv')
```

The file has 10 variables in it. The rows, or records, are XYZ customers. How many records are in xyzcust10?

What types of variables are in the columns of xyzcust10? To find out:

In [60]:
```
xyzcust10.dtypes
```

Out[60]:
```
ACCTNO                    object
ZIP                        int64
ZIP4                       int64
LTD_SALES                float64
LTD_TRANSACTIONS           int64
YTD_SALES_2009           float64
YTD_TRANSACTIONS_2009      int64
CHANNEL_ACQUISITION       object
BUYER_STATUS              object
ZIP9_Supercode             int64
ZIP9_SUPERCODE             int64
dtype: object
```

Look at the first and last rows in xyzcust10:

In [61]:
```
xyzcust10.head()
```

Exercise #2

File Edit View History Bookmarks Tools Help

exercise2

localhost:8890/notebooks/bader/nu/420/spring 2016/sync sessions/sync session 2/exercisepractice2/exercise2

jupyter exercise2 Last Checkpoint: 04/05/2015 (autosaved)

File Edit View Insert Cell Kernel Help Python 2

Markdown Cell Toolbar: None

```
CHANNEL_ACQUISITION        object
BUYER_STATUS               object
ZIP9_Supercode              int64
ZIP9_SUPERCODE              int64
dtype: object
```

Look at the first and last rows in xyzcust10:

In [61]:
```
xyzcust10.head()
```

Out[61]:

	ACCTNO	ZIP	ZIP4	LTD_SALES	LTD_TRANSACTIONS	YTD_SALES_2009	YTD_TRANSACTIONS_2009	CHANNEL_ACQUISITION	BUYE
0	WDQQLLDQL	60084	5016	90	1	0	0	IB	INAC
1	WQWAYHYLA	60091	1750	4227	9	1263	3	RT	ACTI
2	GSHAPLHAW	60067	900	420	3	129	1	RT	ACTI
3	PGGYDYWAD	60068	3838	6552	6	0	0	RT	INAC
4	LWPSGPLLS	60090	3932	189	3	72	1	RT	ACTI

In [62]:
```
xyzcust10.tail()
```

Out[62]:

	ACCTNO	ZIP	ZIP4	LTD_SALES	LTD_TRANSACTIONS	YTD_SALES_2009	YTD_TRANSACTIONS_2009	CHANNEL_ACQUISITION
30466	SYDQYLSWH	60098	3951	2736	9	96	1	RT
30467	SAPDQHQLP	60098	9681	2412	8	108	1	RT
30468	SASYAPDSY	60098	0	429	1	0	0	RT
30469	PWQPDWHA	60098	7927	651	1	0	0	RT
30470	SOQHDYHWH	60098	4160	4527	16	672	2	RT

What you need to submit for Exercise #2 on Canvas?

1. Change the Ipython Notebook script to use the VacationHistory.csv file instead of xyzcust10.csv file

2. Run the IPython Notebook Script Again

3. Save your updated IPython Notebook Script along with the OUTPUT for the cells

4. Submit your updated IPython Notebook Script on Canvas

Exercise #2

Add your VacationHistory file

File Edit View History Bookmarks Tools Help

exercise2 | exercise2practice

localhost:8891/notebooks/bader/nu/420/exercisepractices/exercisepractice2/exercise2prac

jupyter exercise2practice Last Checkpoint: 04/05/2015 (autosaved)

File Edit View Insert Cell Kernel Help | Python 2

Markdown | Cell Toolbar: None

available to you on Canvas. Take a look at it with your favorite text editor. Then, put it in a place you can find it from Canvas, and input it into a DataFrame:

In [29]:
```
VacationHistory=pd.read_csv('VacationHistory.csv')
```

The file has 10 variables in it. The rows, or records, are XYZ customers. How many records are in xyzcust10?

What types of variables are in the columns of xyzcust10? To find out:

In [31]:
```
VacationHistory.dtypes
```

Out[31]:
```
Chicago    object
IL         object
5           int64
2008        int64
dtype: object
```

Look at the first and last rows in xyzcust10:

In [32]:
```
VacationHistory.head()
```

Out[32]:

	Chicago	IL	5	2008
0	NewYork	NY	8	2014
1	Detriot	MI	2	2007
2	SanFransico	CA	7	2003
3	Boston	MA	4	2010

Exercise #2

File Edit View History Bookmarks Tools Help

exercise2 | exercise2practice

localhost:8891/notebooks/bader/nu/420/exercisepractices/exercisepractice2/exercise2practice.ipynb

jupyter exercise2practice Last Checkpoint: 04/05/2015

File Edit View Insert Cell Kernel Help

Markdown Cell Toolbar: N

available to you on Canvas. Take a look at it with it from Canvas, and input it into a DataFrame:

Does this look right?
Any changes you might think of?
Are these objects the Header for the table?

In [29]:
```
VacationHistory=pd.read_csv('VacationHistory.csv')
```

The file has 10 variables in it. The rows, or records, are XYZ customers. How many records are in xyzcust10?

What types of variables are in the columns of xyzcust10? To find out:

In [31]:
```
VacationHistory.dtypes
```

Out[31]:
```
Chicago    object
IL         object
5           int64
2008        int64
dtype: object
```

Look at the first and last rows in xyzcust10:

In [32]:
```
VacationHistory.head()
```

Out[32]:

	Chicago	IL	5	2008
0	NewYork	NY	8	2014
1	Detriot	MI	2	2007
2	SanFransico	CA	7	2003
3	Boston	MA	4	2010

Exercise #2

File Edit View History Bookmarks Tools Help

exercise2 | exercise2practice

localhost:8891/notebooks/bader/nu/420/exercisepractices/exercisepractice2/exercise2practice.ipynb

jupyter exercise2practice Last Checkpoint: 04/05/2015 (autosaved)

File Edit View Insert Cell Kernel Help | Python 2

Markdown | Cell Toolbar: None

available to you on Canvas. Take a look at it with your favorite text editor. Then, put it in a place you can find it from Canvas, and input it into a DataFrame:

In [29]:
```
VacationHistory=pd.read_csv('VacationHisto
```

The file has 10 variables in it. The rows

What types of variables are in the columns

In [31]:
```
VacationHistory.dtypes
```

Out[31]:
```
Chicago    object
IL         object
5           int64
2008        int64
dtype: object
```

Look at the first and last rows in xyzcust10:

In [32]:
```
VacationHistory.head()
```

Out[32]:

	Chicago	IL	5	2008
0	NewYork	NY	8	2014
1	Detriot	MI	2	2007
2	SanFransico	CA	7	2003
3	Boston	MA	4	2010

Does the header of the table look right?

Any changes you might think of?

Is this the right table header?