**CS506 Programming for Computing**

**HOS04 – NumPy & pandas**

03/15/2025 Reviewed by Clark Ngo

04/24/2025 Reviewed by Naveena Moddu

04/24/2025 Reviewed by Ashwin Paturi

School of Technology & Computing (STC)

City University of Seattle (CityU)

A close up of a screen

Description automatically generated A close up of a sign

Description automatically generated

**Before You Start**

* **Screenshots may be different from your environment.**
* The directory path shown in screenshots may be different from yours.
* There might be subtle discrepancies along with the steps. Please use your best judgment while going through this cookbook-style tutorial to complete each step.
* Some steps may not be explained in detail. If you are not sure what to do:

1. Consult the resources from the course.
2. If you cannot solve the problem after a few tries (usually 15 -30 minutes), ask a TA for help.

**Resources**

* Data from Kaggle: <https://www.kaggle.com/datasets>
* NumPy Documentation: <https://numpy.org/doc/stable/>
* McIntire, G., Martin, B., & Washington, L. Pandas A Complete Introduction. Retrieved from <https://www.learndatasci.com/tutorials/python-pandas-tutorial-complete-introduction-for-beginners/>
* Pandas User Guide: <https://pandas.pydata.org/pandas-docs/stable/user_guide/index.html>
* Pandas Documentation: <https://pandas.pydata.org/docs/user_guide/io.html?highlight=json#io-json-writer>
* Pandas Documentation: <https://pandas.pydata.org/docs/user_guide/index.html>
* Pandas.qcut(): <https://pandas.pydata.org/docs/reference/api/pandas.qcut.html>
* Pandas.cut(): <https://pandas.pydata.org/docs/reference/api/pandas.cut.html>
* Pandas.tranform(): <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.transform.html?highlight=transform#pandas.DataFrame.transform>
* Stanford University. (2020). CS231n: Convolutional Neural Networks for Visual Recognition: NumPy.

**Learning Outcomes**

* **Section 1: Jupyter Notebook in Codespaces**
* **Section 2: NumPy**
* **Section 3: What is pandas?**
* **Section 4: Read and write with pandas**
* **Section 5: Data transformation**
* **Section 6: Data Discretization**

**Section 1: Jupyter Notebook in Codespaces**

Follow instructions on how to install Jupyter Notebook on GitHub Codespaces here: <https://cityuseattle.github.io/docs/environment/codespaces_jupyter/>

Learn and try Jupyter Notebook from JupyterLab: https://jupyter.org/try-jupyter/lab/

**Section 2: NumPy,** Which stands for Numerical Python, is an open-source and the core library for scientific computing in Python. It provides a high-performance multidimensional array object and tools for working with these arrays.

NumPy is memory efficient, meaning it can handle a vast amount of data more accessible than any other library. We’ll cover the following in this module:

* + 1. Datatypes
    2. Arrays

**Datatypes** - The object defined in NumPy is an N-dimensional array type called ndarray, describing the collection of items of the same type. An element in ndarray is an object of data-type object called dtype.

* + 1. A **dtype** described in the following aspects:
       - Type of data (integer, float, Python object, etc.)
       - Size of the data (how many bytes is in e.g., the integer)
       - Byte order of the data (little-endian or big-endian)
       - Structured data type (*e.g.*, describing an array item consisting of an integer and a float)
  1. Create a new file called datatypes.ipynb and simply click on the file to open notebook.
  2. Type the following to see how NumPy says about data types. Run selected cell to see result for that cell.

A screenshot of a cell phone

Description automatically generated

We also can define a structured data type. For example, we will define a data type named student with the 3 fields: name as string, age as integer, points as float.

A screenshot of a computer

AI-generated content may be incorrect.

Let’s apply to ndarray object.

A screenshot of a cell phone

Description automatically generated

As you can see here, “*student2withlongname”* has exceeded the string length, only 10 characters acceptance was defined.

**Arrays** - A numpy array is a grid of values like Python lists, but all the same type, and is indexed by a tuple of nonnegative integers. The rank and shape are the number of dimensions and a tuple of integers of the size of the array, respectively.

* + 1. Create a new file called arrays.ipynb and simply click on the file to open notebook.
    2. Type the following to create numpy arrays. Accessing items in array is similar to Python list.

A screenshot of a cell phone

Description automatically generated

* + 1. Stay in same file, type to following functions from numpy to generate arrays in a different way. You will see how it can make your life easier with array.

A screenshot of a cell phone

Description automatically generated

* + 1. Indexing and slicing in NumPy array is similar to Lists in Python. Try the following into a new cell.

Screen of a cell phone

Description automatically generated

* + 1. In NumPy, you can do integer indexing which helps in selecting any arbitrary item in an array using the data from another array like the example below. Add this to new cell.

Screen of a cell phone

Description automatically generated

**Section 3: What is pandas?**

Pandas (derived from the word Panel Data – an Econometrics from Multidimensional data – Tutorialspoint) is a powerful, open-source Python library providing high-performance, easy-to-use data structures for data analysis, manipulation, and visualization.

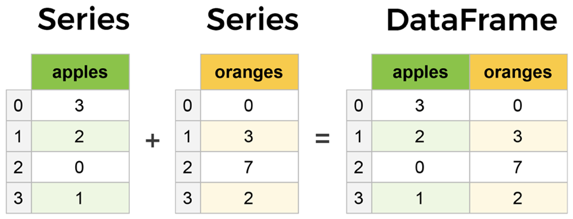
Features of Pandas

* + - * Fast and efficient DataFrame object
      * Tools for loading data into in-memory data objects from different file formats.
      * Label-based slicing, indexing and subsetting of large data sets.
      * Columns from a data structure can be deleted or inserted.
      * Group by data for aggregation and transformations.
      * High performance merging and joining of data.

**Core Components** - The two primary data structures of pandas, Series and DataFrame.

* + 1. **Series** – 1D labeled homogeneously-typed array
    2. **DataFrame** – General 2D labeled, size-mutable tabular structure with potentially heterogeneously-typed column

A Series is essentially a column, and a DataFrame is a multi-dimensional table made up of a collection of Series.



* 1. Create a new file called pandas\_object.ipynb and simply click on the file to open notebook.
  2. Type the following into the file just created. Run selected cell to see each result.

A screenshot of a cell phone

Description automatically generated

* 1. We just created Series object with pandas, next we will create DataFrame, which is a collection of Series.

A screenshot of a cell phone

Description automatically generatedEach (key, value) item in data corresponds to a column in the resulting DataFrame.

* 1. The previous DataFrame, index was given to the creation by default, but we can create our own labels as the following example.

A screenshot of a cell phone screen with text

Description automatically generated

* 1. We gave string names as index replacement, so it is more convenient to locate the data by name. For example:

buyer.loc['Tom']

Output:

A screenshot of a cell phone

Description automatically generated

* 1. That’s the basic pandas, there is a lot more from pandas that you can do. Learn more here:

<https://pandas.pydata.org/pandas-docs/stable/user_guide/index.html>

**Section 4: Read and write with pandas**

Pandas is a very powerful and popular framework for data analysis and manipulation. One of the most useful features of Pandas is the ability to read and write various types of files including CSV. The Pandas I/O API is a set of top-level reader functions accessed like pd.read\_csv() that generally return a Pandas object. More available readers and writers can be found here <https://pandas.pydata.org/docs/user_guide/io.html>

**Common functionalities:**

| **Format Type** | **Data Description** | **Reader** | **Writer** |
| --- | --- | --- | --- |
| text | CSV | read\_csv | to\_csv |
| text | Fixed-Width Text File | read\_fwf |  |
| text | JSON | read\_json | to\_json |
| text | HTML | read\_html | to\_html |
| text | Local clipboard | read\_clipboard | to\_clipboard |
|  | MS Excel | read\_excel | to\_excel |

* 1. Verify that you have the file sales\_data\_sample.csv
  2. Create a new file called reading\_writing.ipynb and simply click on the file to open notebook.
  3. Type the following into the file just created. Run selected cell to see each result.

**A screenshot of a computer

Description automatically generated**

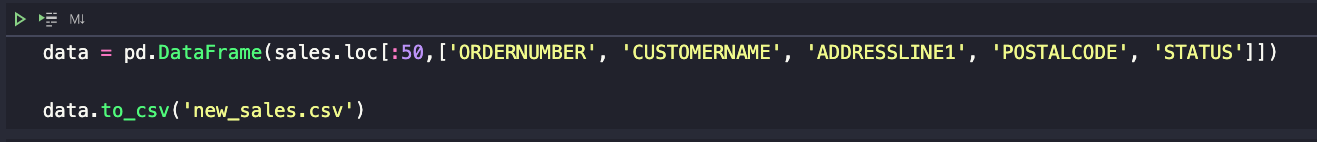
'latin-1' or 'iso-8859-1' is the simplest text encoding maps the code points 0–255 to the bytes 0x0–0xff, which means that a string object that contains code points above U+00FF can’t be encoded with this codec. Doing so will raise a **UnicodeEncodeError.**

* 1. This data has 25 columns, you might not see the whole thing, but you still can see what they are with the columns DataFrame like below.

A screenshot of a cell phone

Description automatically generated

* 1. Now we will write a new CSV file with multiple columns selected by the indexing operator by passing a list of column names.



* 1. The Series and DataFrame objects have an instance method to\_csv which allows storing the contents of the object as a comma-separated-values file.
     1. There are a couple common exceptions that arise when doing selections with just the indexing operator.
     2. If you misspell a word, you will get a KeyError
     3. If you forgot to use a list to contain multiple columns, you would also get a KeyError
     4. You just created a new CSV file named new\_sales.csv with to\_csv method of DataFrame.
     5. Check the contents of your new\_sales.csv

**Section 5: Data transformation**

The process of changing the format, structure, or values of data so that the result may be more efficient and easier to understand. This involves converting data from one structure to another so you can integrate it with different applications.

With Pandas, we can perform transformation. For this section, we will use the transform function.

1. Create a new file named data\_transformation.ipynb.
2. Create a DataFrame with the following rows and columns.

A screenshot of a cell phone

Description automatically generated

1. Now we will use transform() function to add 10 to each element of the data frame.
   * 1. DataFrame.transform call func on self-producing a DataFrame with transformed values. Produced DataFrame will have same axis length as self.

A screenshot of a computer screen

Description automatically generated

Notice that we used an anonymous function (lambda) to add 10 to each value. So, all non-empty values have been added successfully.

**Section 6: Data Discretization**

Defined as a process of converting continuous data attribute values into a finite set of intervals with minimal loss of information. For example, you have height data and want to discretize it to 0 and 1 interval depending on if the height is below or above a certain value of height.

* 1. Within the same file, create a numpy array with 10 random integers between 10 and 200.

A screenshot of a cell phone screen with text

Description automatically generated

* 1. Numpy provides some of the functions you can perform discretize with, one is using digitize() for doing so.
  2. In this step, we will discretize where 50 is the threshold to divide the data into two categories: one less than 50 and ones above 50.

A picture containing player, meter

Description automatically generated

* The bins argument is a list and so we can specify multiple binning or discretizing conditions.
* As you can see, there are only two numbers lower than 50 represented as 0. There are more than 50 represented as 1.
  1. Let's say if we want to have 3 categories, how many values should we specify in the bins array? You got it, 2 values.

A close up of a sign

Description automatically generated

We have what we wanted. There are 3 categories here: 0, 1, and 2. 0 category has a value of less than 50, the 1 category has a value of less than 100 and category 3 has a value of more than 100.

* 1. Now let’s use a function from Pandas called cut() to discretize our data.
  2. It does the same thing as what Numpy’s digitize() does, but the way it works is a bit different. Let’s first create a DataFrame with random number from previous.

A screenshot of a video game

Description automatically generated

* We created 10 values in height columns with index 0-9.
  1. Categorize the height variable into four categories using Pandas cut function.

A screenshot of a cell phone

Description automatically generated

* The height values between 0 and 25 are in one category, height between 25 and 50 are in the second category, 50-100 in third category, and 100-200 are in the fourth category.
  1. We also can label them as follows.

A screen shot of a computer

Description automatically generated

* As my random data does not have values below 25 so there is no category 1 in this example. Your data may be different.
* You can change labels to string instead of number.
  1. Pandas also have another function called qcut() which discretize variable into equal-sized buckets. You only need to tell the function the number of quantiles, pandas will figure out how to bind that data.

A screenshot of a cell phone

Description automatically generated

* Notice that qcut discretized the data into equal size but the interval is not equal. Binned into 5 groups with 2 members each. Unlike what cut function does which is no guarantee about the distribution of items in each bin, but the interval pretty much the same for all bin.

A screenshot of a cell phone

Description automatically generated

* 1. Let’s look at a familiar example below of how to use discretize with data.

A screenshot of a cell phone

Description automatically generated

* Line number 3, we sorted for easier to see in each category. We categorized what grades belong to which categories. Then, we put the data into DataFrame.
  1. If you are a visual person, you need to see the chart. You can do that with a basic plotting provided by pandas like the below image.
  2. We will do a bar plot based on the number of students in each group.

A screenshot of a graph

AI-generated content may be incorrect.

* 1. If you want to see what value\_counts() does, remove “.plot.bar()”

**Push your work to GitHub Codespaces**

Follow instructions here: <https://cityuseattle.github.io/docs/git/codespaces_submission/>