Programming in Python Lecture 7- Numpy, Pandas

Plan for today

- Numpy
- Pandas

Reminder

- Random
- Datetime
- OS & Shutil

Random

- The **Random** package provides to generate the pseudo-random variables. It can be used perform some action randomly such as to get a random number, selecting a random elements from a list, shuffle elements randomly, etc.

Random Methods

Function	Description
seed()	Initialize the random number generator
getstate()	Returns the current internal state of the random number generator
setstate()	Restores the internal state of the random number generator
getrandbits()	Returns a number representing the random bits
randrange()	Returns a random number between the given range
randint()	Returns a random number between the given range
choice()	Returns a random element from the given sequence
choices()	Returns a list with a random selection from the given sequence
shuffle()	Takes a sequence and returns the sequence in a random order
sample()	Returns a given sample of a sequence
random()	Returns a random float number between 0 and 1

Datetime

- The **Datetime** module supplies classes for manipulating dates and times.it helps us identify and process time-related elements like dates, hours, minutes, seconds, days of the week, months, years, etc. It offers various services like managing time zones and daylight savings time. It can work with timestamp data. It can extract the day of the week, day of the month, and other date and time formats from strings.

Datetime Methods

https://docs.python.org/3/library/datetime.html

https://www.w3schools.com/python/python_datetime.asp

OS

- This module provides a portable way of using operating system dependent functionality. If you just if you want to manipulate paths, see the os.path module, and if you want to read all the lines in all the files on the command line see the fileinput module. For creating temporary files and directories see the tempfile module, and for high-level file and directory handling see the shutil module.

OS

```
>>> import OS
>>> os.walk(path)
Names of files for path
>>> os.path.split(path)
('Dir','file')
>>> os.path.join('Dir','file'))
(Path)
```

OS

```
>>> os.listdir(path)
```

List of files for path

```
>>> os.path.isdir(path)
```

True if Dir/file/link, Fales if not

```
>>> os.scandir(root)
```

List of object in dictatory

```
>>> os.rename(path1, path2)
```

Rename the file or directory path1 to path2. If path2 exists, the operation will fail with an OSError subclass in a number of cases.

OS Methods

https://docs.python.org/3/library/os.html#os-file-dir

Shutil

- The **shutil** module offers a number of high-level operations on files and collections of files. In particular, functions are provided which support file copying and removal. For operations on individual files, see also the os module.

Shutil

```
>>> shutil.copyfile(path1, path2)
```

Copy the contents (no metadata) of the file named path1 to a file named path2 and return path2 in the most efficient way possible.

```
>>> shutil.copy(path1, path2,)
```

Copies the file path1 to the file or directory path2. path1 and dst should be path-like objects or strings. If path2 specifies a directory, the file will be copied into path2 using the base filename from path1. If path2 specifies a file that already exists, it will be replaced. Returns the path to the newly created file

Shutil

```
>>> shutil.move (path1, path2)
```

Recursively move a file or directory (path1) to another location (path2).

```
>>> shutil.copytree (path1, path2)
```

Recursively copy an entire directory tree rooted at path1 to a directory named path2 and return the destination directory. All intermediate directories needed to contain path2 will also be created by default

Shutil Methods

https://docs.python.org/3/library/shutil.html

Homework

1. Create a Python script to generate a guessing number game, allowing the user a limited number of guesses, using input from user and randint function, and will not work on Wednesdays

NumPy

- The **NumPy** (Numeric Python) package provides basic routines for manipulating large arrays and matrices of numeric data.
- NumPy are open-source, and therefore provide a free Matlab alternative.
- The package are popular among scientists, researchers and engineers who want to apply various mathematical methods on large datasets.

NumPy’s main object is the **Array**

- NumPy’s main object is the homogeneous multidimensional **array**.
- It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers.
- In Numpy dimensions are called ***axes***.
- The number of axes is ***rank***.
  - *A **vector** is an array of rank 1*
  - *A 2D **matrix** is an array of rank 2.*

The Array object – Creation

```
>>> import numpy as np
>>> a = np.array([0, 1, 2])                    # 1 x 3 array
array([0,1,2])
```

0	1	2

```
>>> b = np.array([[0, 1, 2], [3, 4, 5]])       # 2 x 3 array
array([[0,1,2],
       [3,4,5]])
```

0	1	2
3	4	5

Array – Attributes & Methods

```
>>> MAT = np.array([[0, 1, 2], [3, 4, 5]]) # Creates a 2 x 3 array
array([[0,1,2],
       [3,4,5]])
>>> MAT.ndim  #number of dimensions
2
>>> MAT.shape  #dimension sizes
(2,3)
>>> len(MAT) # returns the size of the first dimension
2
>>> MAT.T          #Transpose
array([[0, 3],
     [1, 4],
     [2, 5]])`
```

0	1	2
3	4	5

0	3
1	4
2	5

Creating an Array of a specific type

```
>>> Mat = np.array(range(10), float)
>>> Mat
array([ 0., 1., 2., 3., 4., 5., 6., 7., 8., 9.])
>>> Mat.dtype
dtype('float64')
```

Creating arrays containing number sequences

```
>>> np.arange(10)
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> np.arange(3,9)
array([3, 4, 5, 6, 7, 8])
>>> np.arange(2,9,2) # start, end (exclusive), step
array([2, 4, 6, 8])
```

Like range in Python, but returns a numpy array object

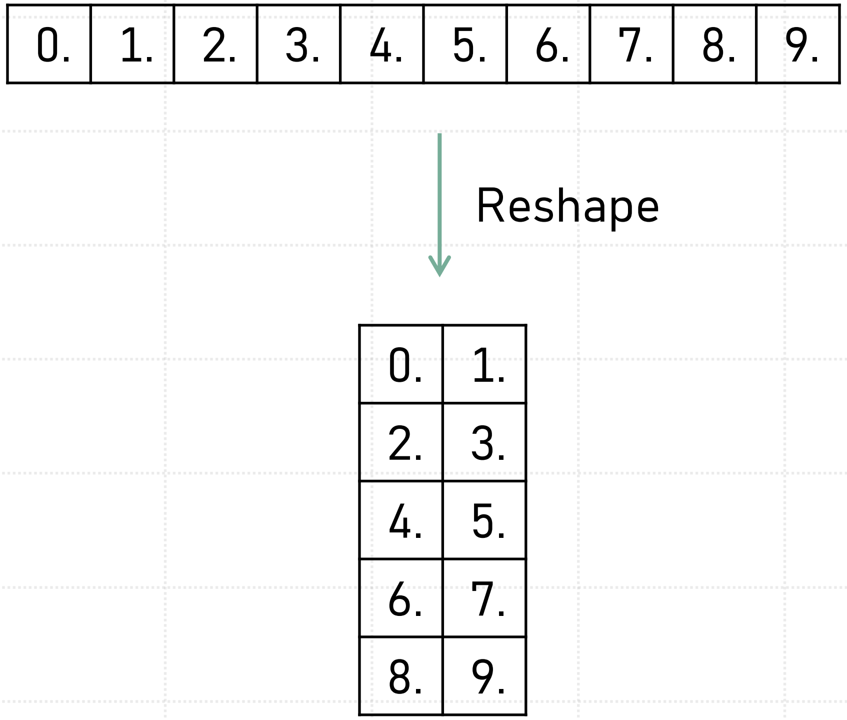
Creating arrays containing number sequences

```
# slicing is similar to standard lists
>>> print (np.arange(1, 5.5, 0.5)[3:1:-1])
[ 2.5  2. ]
>>> print(np.linspace(1,5,9))  # start, end, number of points
[ 1.   1.5  2.   2.5  3.   3.5  4.   4.5  5. ]
# Arrays with random values between 0 and 1
>>> print (np.random.rand(5))
[ 0.53844313  0.3384871   0.5763536   0.29159273
0.43938366]
```

Additional ways to create arrays in NumPy:
http://docs.scipy.org/doc/numpy-noitaerc-yarra-senituor#lmth.noitaerc-yarra.senituor/ecnerefer/1.10.1

Reshaping an Array

```
>>> Mat
array([ 0., 1., 2., 3., 4., 5., 6., 7., 8., 9.])

>>> Mat = a.reshape((5, 2))
Mat array([[ 0., 1.],
        [ 2., 3.],
        [ 4., 5],
        [ 6., 7.],
          [ 8., 9.]])
>>> Mat.shape
(5, 2)
```



Array Arithmetic

Arithmetic operators on arrays apply *elementwise*. A new array is created and filled with the result.

```
>>> x = np.array([1,5,2])
>>> y = np.array([7,4,1])
>>> x + y
array([8, 9, 3])
>>> x * y   # element by element multiplication ! Use np.dot(x,y) for matrix multiplication.
array([ 7, 20,  2])
>>> x - y
array([-6,  1,  1])
>>> x / y
array([0, 1, 2])
>>> x % y
array([1, 1, 0])
```

Note that here, x and y have the same size

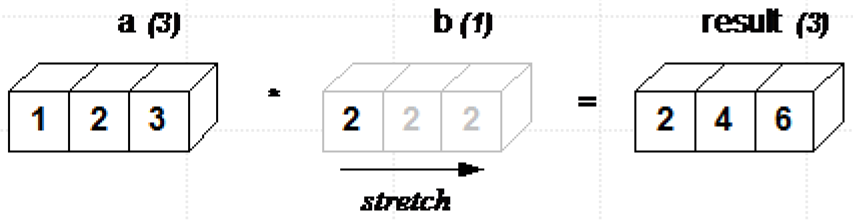
Broadcasting

- NumPy operations are usually done element-by-element which requires two arrays to have exactly the same shape:

```
>>> a = array([1.0,2.0,3.0])
>>> b = array([2.0,2.0,2.0])
>>> a * b
array([ 2.,  4.,  6.])
```

- But this will also work:

```
>>> a = array([1.0,2.0,3.0])
>>> b = 2.0
>>> a * b
array([ 2.,  4.,  6.])
```



Comparisons and Boolean operations

```
>>> a = np.random.random_integers(0, 10,
15)
array([ 4,  9,  1,  7,  0,  9,  8, 10,  1,  0,  7,  7,  9,
5,  1])
```

Comparisons and Boolean operations

```
>>> a = np.random.random_integers(0, 10, 15)
>>> b = np.random.random_integers(0, 10, 15)
>>> comp1 = a==b
>>> comp1
array([False, False, False, False, False,  True, False, False, False,   False,
False, False, False, False, False], dtype=bool)
>>> comp1.any()
True
>>> comp1.all()
False
```

Any is true?

Are all true?

Get the True indices

Fancy indexing: Logical indexing

```
>>> a = np.random.random_integers(0, 20,
  15)
>>> (a % 3 == 0)
```

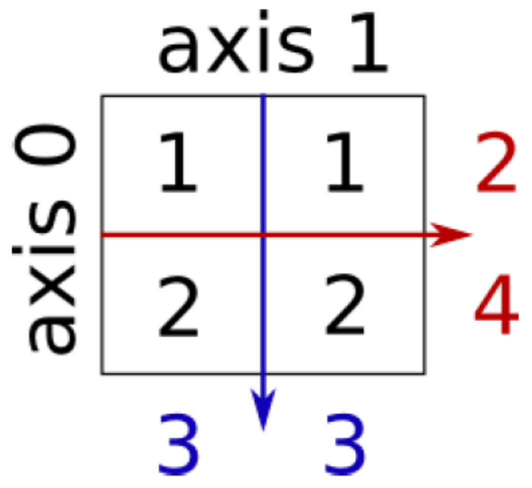
```
array([False, True, False, True, False, False, False, True,
False,True, True, False, True, False, False], dtype=bool)
```

Fancy indexing: Logical indexing

```
>>> a = np.random.random_integers(0, 20, 15)


>>> (a % 3 == 0)


# extract sub-array: this is a copy!
>>> extract_from_a = a[m]


# change sub-array
>>> a[m] = -1
```

Reductions

```
>>> x = np.array([[1, 1], [2, 2]])
array([[1, 1],
       [2, 2]])
>>> x.sum() # works on the entire matrix
6

>>> x.sum(axis=0) # sum over the columns (first dimension)
array([3, 3])

>>> x.sum(axis=1) # sum over the rows (second dimension)
array([2, 4])
>>> x[0, :].sum(), x[1, :].sum()
(2, 4)
```



Also works with x.min, x.max, x.mean etc.

Sorting along an axis

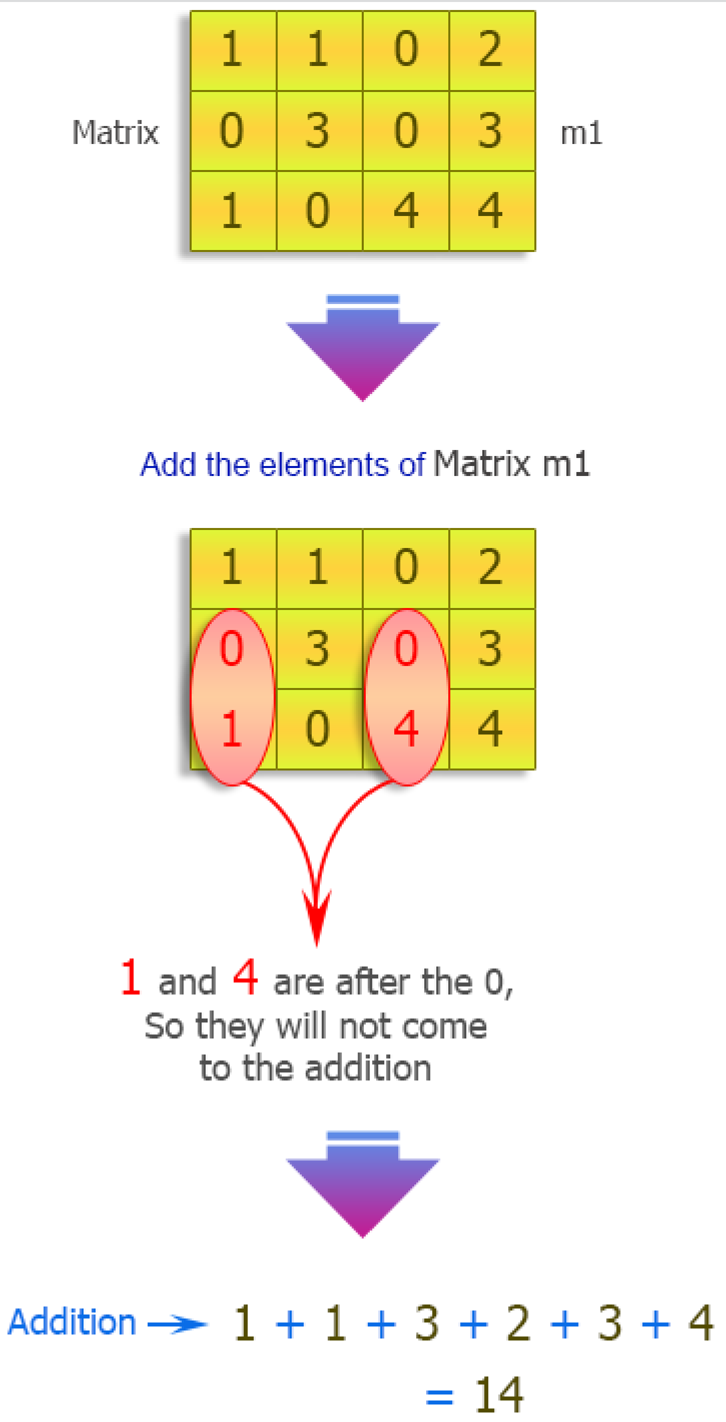
```
>>> a = np.array([[4, 3, 5], [1, 2, 1]])
>>> b = np.sort(a, axis=1)
>>> b
array([[3, 4, 5],
[1, 1, 2]])
```

Questions?

Hands On

Sum example

```
>>> m = [[1, 1, 0, 2],
         [0, 3, 0, 3],
         [1, 0, 4, 4]]

A = np.array(m)
element_sum = 0
for p in range(len(A)):
    for q in range(len(A[p])):
        if A[p][q] == 0 and p < len(A)-1:
            A[p+1][q] = 0
        element_sum += A[p][q]
print(element_sum)

14
```



Degrees example

```
---#Fahrenheit---

fvalues[32 ,99.91 ,34 ,45.21 ,12 ,0] =
F = np.array(fvalues)
print(("Values in Fahrenheit degrees:"
print(F)
print(("Values in Centigrade  degrees:"
print(np.round((((2,(32/9*5 -9 /F*5

---#Centigrade---

cvalues[0 ,37.73 ,1.11 ,7.34 ,11.11- ,17.78-] =
C = np.array(cvalues)
print(("Values in Centigrade degrees:"
print(C)
print(("Values in Fahrenheit  degrees:"
print(np.round((((2,(32 +5 /C*9
```

$$C = (5 (F - 32)) / 9$$

$$F = (9C + (32 * 5)) / 5$$

Plan for today

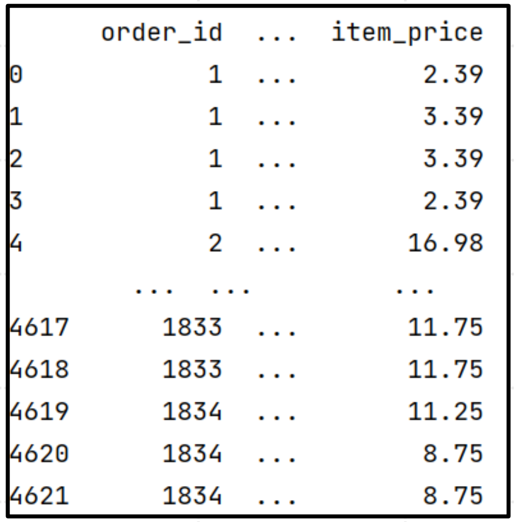
- **Numpy**
- **Pandas**

Pandas

- **Pandas** is a library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

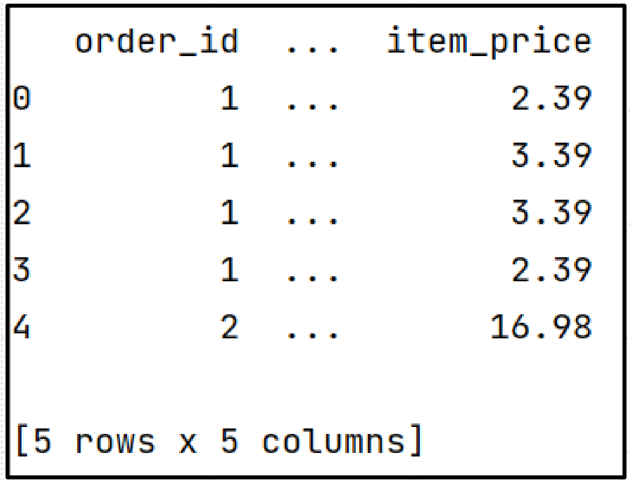
Read CSV-head()

```
>>> chiptole=pd.read_csv("chipotle.csv")

>>> print(chiptole)
```



```
      order_id  ...  item_price
0            1  ...        2.39
1            1  ...        3.39
2            1  ...        3.39
3            1  ...        2.39
4            2  ...       16.98
        ...  ...          ...
4617      1833  ...       11.75
4618      1833  ...       11.75
4619      1834  ...       11.25
4620      1834  ...        8.75
4621      1834  ...        8.75
```

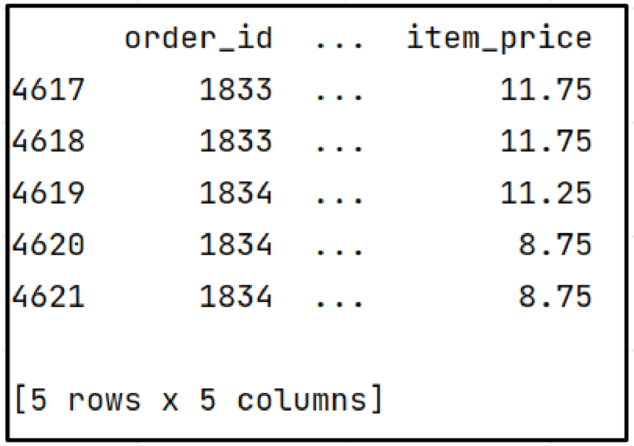
```
>>> print(chiptole.head(5))
```



```
   order_id  ...  item_price
0         1  ...        2.39
1         1  ...        3.39
2         1  ...        3.39
3         1  ...        2.39
4         2  ...       16.98

[5 rows x 5 columns]
```

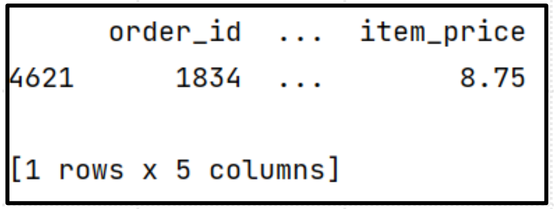
Read CSV-tail()

```
>>> chiptole=pd.read_csv("chipotle.tsv", sep = '\t')

>>> print(chipotle.tail())
```



```
      order_id  ...  item_price
4617      1833  ...       11.75
4618      1833  ...       11.75
4619      1834  ...       11.25
4620      1834  ...        8.75
4621      1834  ...        8.75

[5 rows x 5 columns]
```

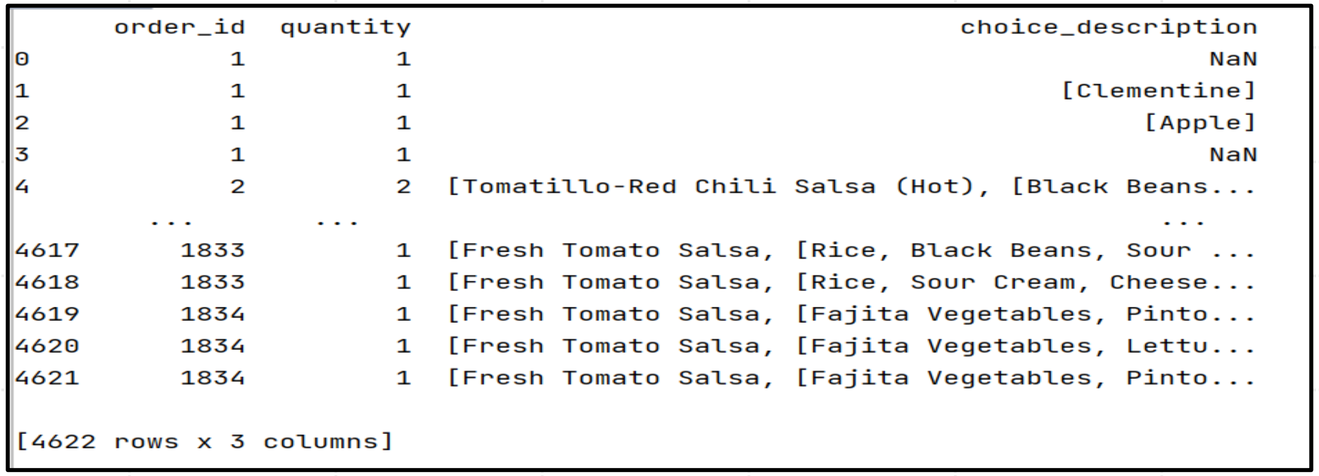
```
>>> print(chiptole.tail(1))
```



```
      order_id  ...  item_price
4621      1834  ...        8.75

[1 rows x 5 columns]
```

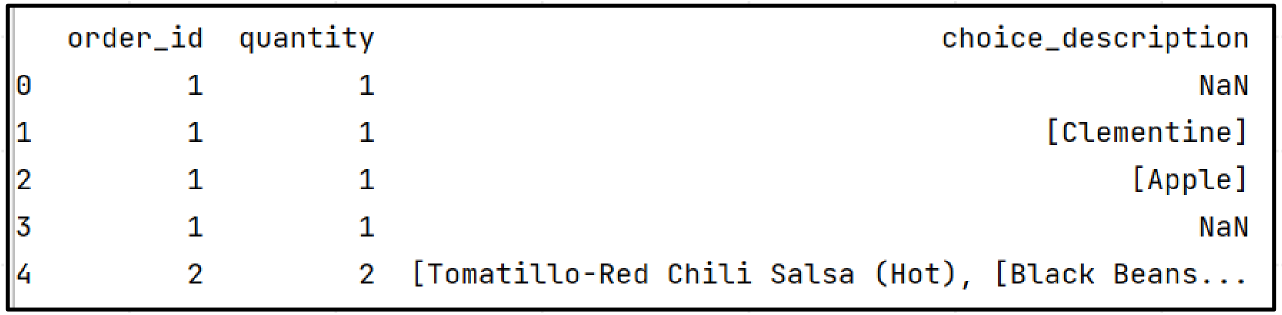
Read CSV-select

>>> chiptole[["order_id", "quantity", "choice_description"]]



```
      order_id  quantity                               choice_description
0            1         1                                              NaN
1            1         1                                     [Clementine]
2            1         1                                          [Apple]
3            1         1                                              NaN
4            2         2  [Tomatillo-Red Chili Salsa (Hot), [Black Beans...
       ...       ...                                              ...
4617      1833         1  [Fresh Tomato Salsa, [Rice, Black Beans, Sour ...
4618      1833         1  [Fresh Tomato Salsa, [Rice, Sour Cream, Cheese...
4619      1834         1  [Fresh Tomato Salsa, [Fajita Vegetables, Pinto...
4620      1834         1  [Fresh Tomato Salsa, [Fajita Vegetables, Lettu...
4621      1834         1  [Fresh Tomato Salsa, [Fajita Vegetables, Pinto...

[4622 rows x 3 columns]
```

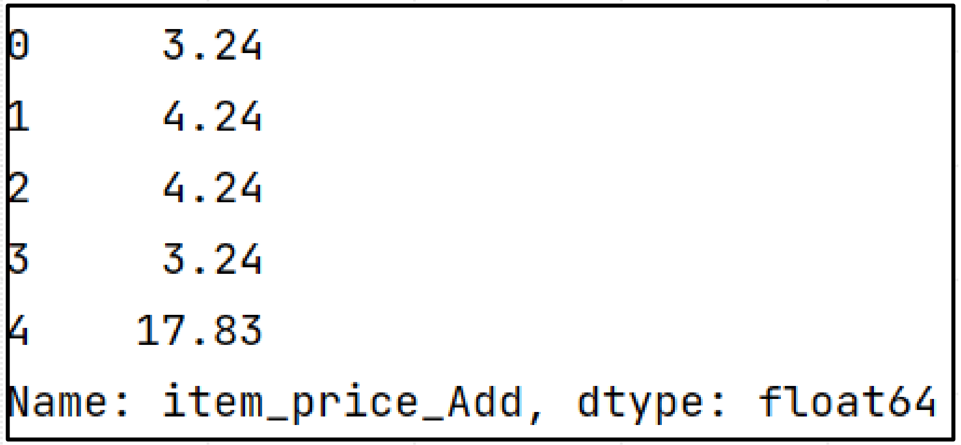
>>>chiptole[["order_id","quantity","choice_description"]].head()



```
   order_id  quantity                               choice_description
0         1         1                                              NaN
1         1         1                                     [Clementine]
2         1         1                                          [Apple]
3         1         1                                              NaN
4         2         2  [Tomatillo-Red Chili Salsa (Hot), [Black Beans...
```

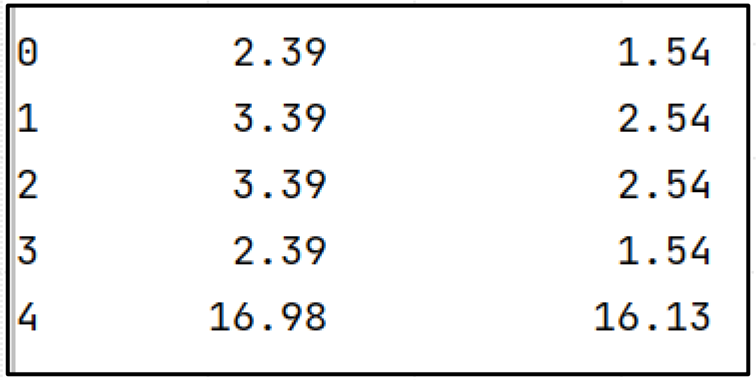
Read CSV-Calculation

```
>>> chiptole["item_price_Add"] = chiptole["item_price"].add(0.85)
>>> chiptole["item_price_Add"].head()
```



```
0     3.24
1     4.24
2     4.24
3     3.24
4    17.83
Name: item_price_Add, dtype: float64
```

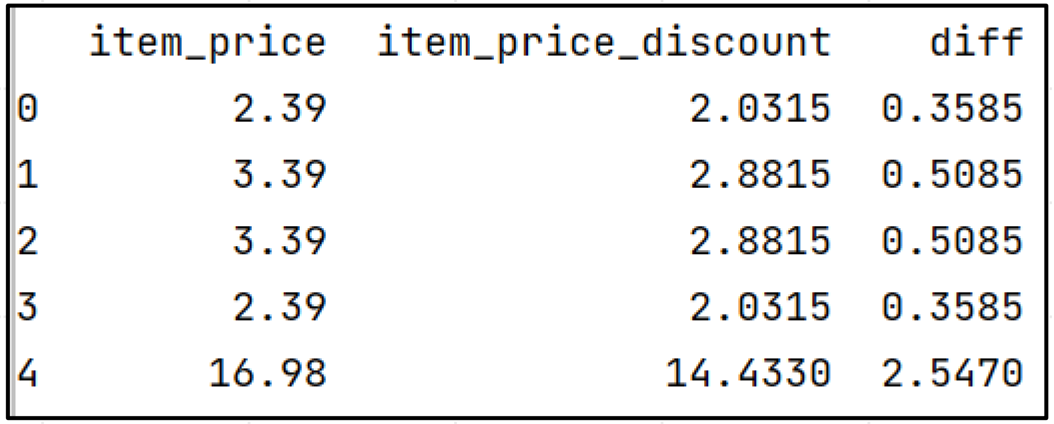
```
>>> chiptole["item_price_Sub"] = chiptole["item_price"].sub(0.85)
>>> chiptole[["item_price","item_price_Sub"]].head()
```



```
0         2.39             1.54
1         3.39             2.54
2         3.39             2.54
3         2.39             1.54
4        16.98            16.13
```

Read CSV-Insert column

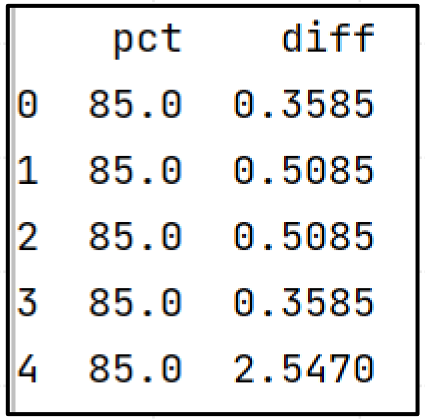
```
>>> chiptole.insert(loc = 4,
 column = "diff",
 value = chiptole["item_price"] - chiptole["item_price_discount"])
>>> chiptole[["item_price","item_price_discount","diff"]].head()
```



```
   item_price  item_price_discount    diff
0        2.39               2.0315  0.3585
1        3.39               2.8815  0.5085
2        3.39               2.8815  0.5085
3        2.39               2.0315  0.3585
4       16.98              14.4330  2.5470
```

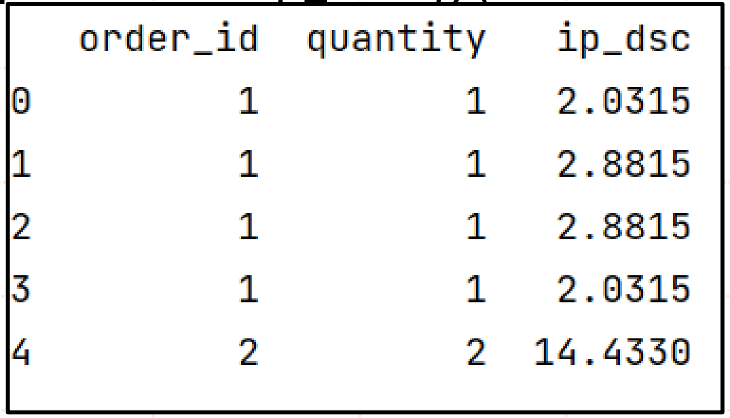
Read CSV-Assigning/Rename

```
>>>chiptole2=chiptole.assign(pct = chiptole["item_price_discount"]
/ chiptole["item_price"] * 100).head()

>>> print(chiptole2[["pct","diff"]])
```



```
    pct    diff
0  85.0  0.3585
1  85.0  0.5085
2  85.0  0.5085
3  85.0  0.3585
4  85.0  2.5470
```

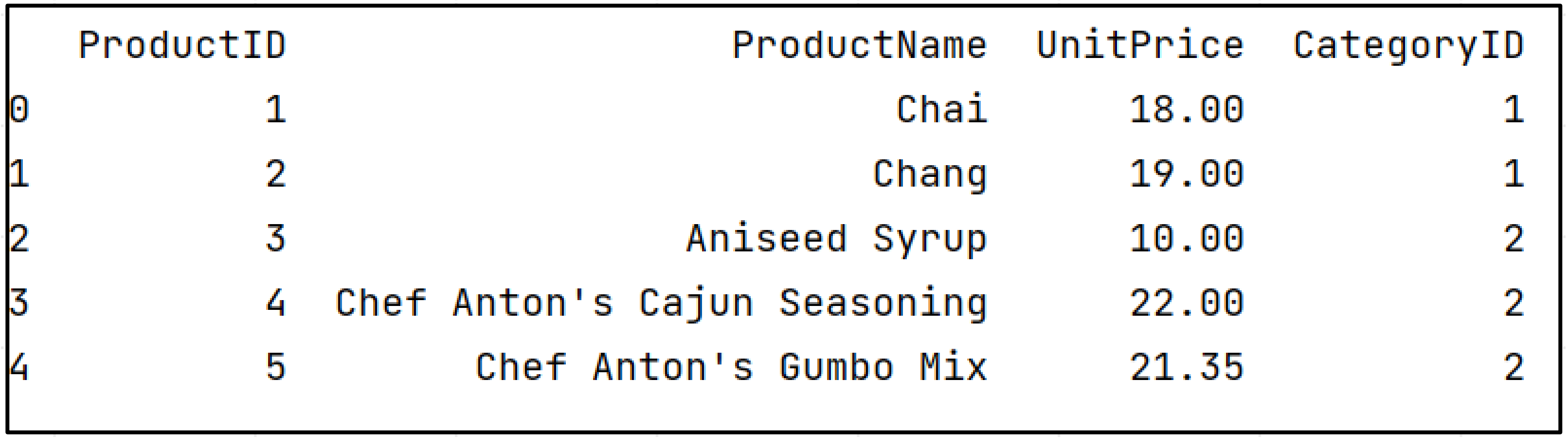
```
>>>chiptole[["order_id", "quantity", "item_price_discount"]]\
.rename(columns = {"item_price_discount":"ip_dsc"})\
.head()
```



```
   order_id  quantity   ip_dsc
0         1         1   2.0315
1         1         1   2.8815
2         1         1   2.8815
3         1         1   2.0315
4         2         2  14.4330
```

Read excel

```
>>> products=pd.read_excel(r"products.xlsx")

>>> print(products.head())
```



```
   ProductID                      ProductName  UnitPrice  CategoryID
0          1                             Chai      18.00           1
1          2                            Chang      19.00           1
2          3                    Aniseed Syrup      10.00           2
3          4   Chef Anton's Cajun Seasoning       22.00           2
4          5           Chef Anton's Gumbo Mix      21.35           2
```

To excel/csv

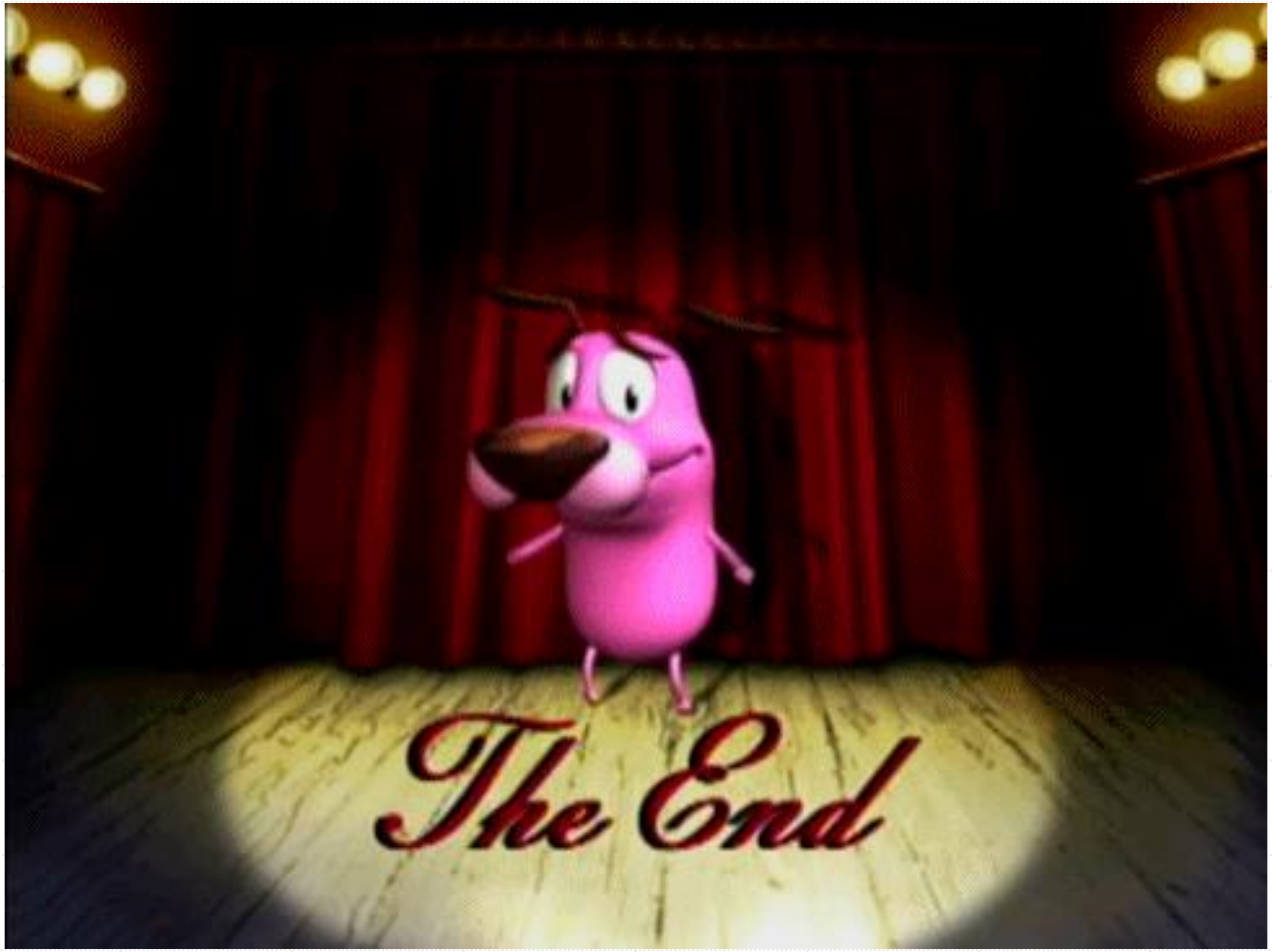
```
>>> DF1=np.random.randint(100,200,8)
>>> DF2=np.random.randint(100,200,8)

>>> df = pd.DataFrame([DF1[0:len(DF1)], DF2[0:len(DF2)]],
                      index=['X', 'Y'],
                      columns=range(1, len(DF2)+1))

>>> df.to_excel('Data1.xlsx')
>>> df.to_csv('Data1.csv')
```

Questions?

Hands On

Homework

1. Create a Python script that take a list of dictionaries and create an excel file that including the id ,Date and S_Id columns by the dictionary's keys(use the .item() to extract the values from the dictionaries)


The End