Pandas

Just a taste

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

```
In [2]: !dir *.csv
```

```
 Volume in drive C is OS
 Volume Serial Number is 8669-AC6F

 Directory of C:\David\my_repos_dwb\upgraded-waffle\Sec_04

11/04/2023  11:19 AM                61 salaries.csv
               1 File(s)             61 bytes
               0 Dir(s)  689,731,747,840 bytes free
```

OUTPUT should be

```
Volume in drive C is OS
 Volume Serial Number is XXXX-XXXX

 Directory of C:\XXX\my_repos_dwb\upgraded-waffle\Sec_04

11/04/2023  11:19 AM                61 salaries.csv
               1 File(s)             61 bytes
               0 Dir(s)  ,XXX,XXX bytes free
```

```
In [3]: pwd
```

```
Out[3]: 'C:\\David\\my_repos_dwb\\upgraded-waffle\\Sec_04'
```

OUTPUT should be

```
'C:\\XXX\\my_repos_dwb\\upgraded-waffle\\Sec_04'
```

In [4]:
```
!type salaries.csv
```

```
Name,Salary,Age
John,50000,34
Sally,120000,45
Alyssa,80000,27
```

OUTPUT should be

```
Name,Salary,Age
John,50000,34
Sally,120000,45
Alyssa,80000,27
```

Starting where the lecture does (except the import)

In [5]:
```
pd.read_csv('salaries.csv')
```

Out[5]:

	Name	Salary	Age
0	John	50000	34
1	Sally	120000	45
2	Alyssa	80000	27

In [6]:
```
print(pd.read_csv('salaries.csv'))
```

```
      Name  Salary  Age
0     John   50000   34
1    Sally  120000   45
2  Alyssa   80000   27
```

In [7]:
```
df = pd.read_csv('salaries.csv')
```

In [8]:
```
df['Salary']
```

Out[8]:
```
0     50000
1    120000
2     80000
Name: Salary, dtype: int64
```

In [9]:
```
df[['Salary', 'Name']]
```

Out[9]:

	Salary	Name
0	50000	John
1	120000	Sally
2	80000	Alyssa

In [10]:
```
df['Salary']
```

Out[10]:
```
0     50000
1    120000
2     80000
Name: Salary, dtype: int64
```

In [11]:
```
df['Salary'].max()
```

Out[11]:
```
120000
```

In [12]:
```
df.describe()
```

Out[12]:

	Salary	Age
count	3.000000	3.000000
mean	83333.333333	35.333333
std	35118.845843	9.073772
min	50000.000000	27.000000
25%	65000.000000	30.500000
50%	80000.000000	34.000000
75%	100000.000000	39.500000
max	120000.000000	45.000000

In [13]:
```
df['Name', 'Salary']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\.conda\envs\tfdeeplearning\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tole
rance)
   2441             try:
-> 2442                 return self._engine.get_loc(key)
   2443             except KeyError:

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: ('Name', 'Salary')

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-13-829ad399aef2> in <module>()
----> 1 df['Name', 'Salary']

~\.conda\envs\tfdeeplearning\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   1962             return self._getitem_multilevel(key)
   1963         else:
-> 1964             return self._getitem_column(key)
   1965 
   1966     def _getitem_column(self, key):

~\.conda\envs\tfdeeplearning\lib\site-packages\pandas\core\frame.py in _getitem_column(self, key)
   1969         # get column
   1970         if self.columns.is_unique:
-> 1971             return self._get_item_cache(key)
   1972 
   1973         # duplicate columns & possible reduce dimensionality

~\.conda\envs\tfdeeplearning\lib\site-packages\pandas\core\generic.py in _get_item_cache(self, item)
   1643         res = cache.get(item)
   1644         if res is None:
-> 1645             values = self._data.get(item)
   1646             res = self._box_item_values(item, values)
   1647             cache[item] = res

~\.conda\envs\tfdeeplearning\lib\site-packages\pandas\core\internals.py in get(self, item, fastpath)
   3588
   3589             if not isnull(item):
-> 3590                 loc = self.items.get_loc(item)
   3591             else:
   3592                 indexer = np.arange(len(self.items))[isnull(self.items)]

~\.conda\envs\tfdeeplearning\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tole
rance)
   2442                 return self._engine.get_loc(key)
   2443             except KeyError:
-> 2444                 return self._engine.get_loc(self._maybe_cast_indexer(key))
   2445
   2446         indexer = self.get_indexer([key], method=method, tolerance=tolerance)

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: ('Name', 'Salary')
```

OUTPUT should be

```
A big error. We need the headers in a list, and they're now "posing as" an index

The error ends with

KeyError: ('Name', 'Salary')
```

In [14]:
```
df[['Name', 'Age']] # Has an HTML table as output
```

Out[14]:

	Name	Age
0	John	34
1	Sally	45
2	Alyssa	27

In [15]:
```
print(df[['Name', 'Age']])
```

```
     Name  Age
0    John   34
1   Sally   45
2  Alyssa   27
```

In [16]:
```
df['Age']
```

Out[16]:
```
0    34
1    45
2    27
Name: Age, dtype: int64
```

In [17]:
```
df['Age'].mean()
```

Out[17]:
```
35.333333333333336
```

In [18]:
```
df['Age'].describe()
```

Out[18]:
```
count     3.000000
mean     35.333333
std       9.073772
min      27.000000
25%      30.500000
50%      34.000000
75%      39.500000
max      45.000000
Name: Age, dtype: float64
```

In [19]:
```
# We can use filters here, too
df['Salary'] > 60000
```

Out[19]:
```
0    False
1     True
2     True
Name: Salary, dtype: bool
```

In [20]:
```
# Look at the original to see if that makes sense
df
```

Out[20]:

	Name	Salary	Age
0	John	50000	34
1	Sally	120000	45
2	Alyssa	80000	27

In [21]:
```
# Get entries instead of just booleans
myfilter_codealong = df['Salary'] > 60000

df[myfilter_codealong] # output is HTML table
```

Out[21]:

	Name	Salary	Age
1	Sally	120000	45
2	Alyssa	80000	27

In [22]:
```
print(df[myfilter_codealong])
```

```
     Name  Salary  Age
1   Sally  120000   45
2  Alyssa   80000   27
```

In [23]:
```
df['Age'] > 30
```
Out[23]:
```
0     True
1     True
2    False
Name: Age, dtype: bool
```

In [24]:
```
age_filter =df['Age'] > 30
```

In [25]:
```
df[age_filter] # Output will be HTML table
```
Out[25]:

	Name	Salary	Age
0	John	50000	34
1	Sally	120000	45

In [26]:
```
print(df[age_filter])
```
```
    Name  Salary  Age
0   John   50000   34
1  Sally  120000   45
```

In [27]:
```
df[df['Age'] > 30] # output will be HTML table
```
Out[27]:

	Name	Salary	Age
0	John	50000	34
1	Sally	120000	45

In [28]:
```
print(df[df['Age'] > 30])
```
```
    Name  Salary  Age
0   John   50000   34
1  Sally  120000   45
```

How we will mostly use Pandas

In [29]:
```
df2 = pd.read_csv('salaries.csv')
```

In [30]:
```
df2.as_matrix()
```

Out[30]:
```
array([['John', 50000, 34],
       ['Sally', 120000, 45],
       ['Alyssa', 80000, 27]], dtype=object)
```

That's all for now!

https://www.udemy.com/course/complete-guide-to-tensorflow-for-deep-learning-with-python/learn/lecture/7982582 (https://www.udemy.com/course/complete-guide-to-tensorflow-for-deep-learning-with-python/learn/lecture/7982582)