5.1 - DATA WRANGLING with PANDAS

## **5.1.3 - Merging DataFrames with pandas**

Chapter-1: Preparing data

|  |  |  |
| --- | --- | --- |
| Reading multiple data files | | |
| Import tools | * pd.read\_csv() * pd.read\_excel() * pd.read\_html() * pd.read\_json() |  |
| Separate file | pd.read\_csv() | dataframe0 = pd.read\_csv('sales-jan-2015.csv') |
| Multiple files | for loop | filenames = ['sales-jan-2015.csv', 'sales-feb-2015.csv']  dataframes = []  for f in filenames:  dataframes.append(pd.read\_csv(f)) |
| comprehension | dataframes = [pd.read\_csv(f) for f in filenames] |
| glob | from glob import glob  filenames = glob('sales\*.csv')  dataframes = [pd.read\_csv(f) for f in filenames] |
| Reindexing DataFrames | | |
| importing | index\_col= | w\_mean = pd.read\_csv('quarterly\_mean\_temp.csv', index\_col='Month') |
| Type of index | .index | print(w\_mean.index)  Index(['Apr', 'Jan', 'Jul', 'Oct'], dtype='object', name='Month') |
| Reindexing | .reindex() | ordered = ['Jan', 'Apr', 'Jul', 'Oct']  w\_mean2 = w\_mean.reindex(ordered) |
| from a DataFrame Index | w\_mean.reindex(w\_max.index) |
| Sorting index | .sort\_index() | w\_mean2.sort\_index() |
|  | ascending=False | reverse alphabetical order by specifying the additional keyword argument |
| Arithmetic with Series & DataFrames | | |
| Scalar multiplication | \* | weather.loc['2013-07-01':'2013-07-07', 'PrecipitationIn'] \* 2.54 |
| Dividing | .divide() | week1\_range.divide(week1\_mean, axis='rows') |
| Percentage changes | .pct\_change() | week1\_mean.pct\_change() \* 100 |
| Adding | + | bronze + silver |
| .add() | bronze.add(silver) |
| Using a fill\_value | fill\_value=0 | bronze.add(silver, fill\_value=0) |
| Chaining .add() |  | bronze.add(silver, fill\_value=0).add(gold, fill\_value=0) |

Chapter-2: Concatenating data

|  |  |  |
| --- | --- | --- |
| Appending & concatenating Series | | |
|  | .append() | ● Invocation: s1.append(s2)  ● Stacks rows of s2 below s1  ● Method for Series & DataFrames |
|  | .concat() | ● Invocation: pd.concat([s1, s2, s3])  ● Can stack row-wise or column-wise  ● pandas module function |
| Using .append() |  | east = northeast.append(south). ## satirlari alta yapistirir, iki index’lidir |
| Using .reset\_index() | .reset\_index() | new\_east = northeast.append(south).reset\_index(drop=True) |
| Using concat() |  | east = pd.concat([northeast, south]) ## iki index’lidir |
| Using ignore\_index | ignore\_index= | new\_east = pd.concat([northeast, south], ignore\_index=True) |
| Appending & concatenating DataFrames | | |
| Appending Dataframes | .append() | population.append(unemployment)  ### satirlari oldugu gibi altina yapistirir, tekrar eden indexler olabilir. |
| Concenating rows | .concat (axis=0) | pd.concat([population, unemployment], axis=0)  ## satir birlestirmesi yapar, tekrar eden indexler olabilir |
| Concatenating columns | .concat (axis=1) | pd.concat([population, unemployment], axis=1)  ## sutun birlestirmesi yapar, ayni indexleri tek satirda gosterir |
| Concatenation, keys, & MultiIndexes | | |
| Using multi-index | keys=[] | rain1314 = pd.concat([rain2013, rain2014], keys=[2013, 2014], axis=0) |
| rain1314 = pd.concat([rain2013, rain2014], keys=[2013, 2014], axis='columns') |
| Accessing a multi-index | .loc[] | print(rain1314.loc[2014]) |
| pd.concat() with dict |  | rain\_dict = {2013: rain2013, 2014: rain2014}  rain1314 = pd.concat(rain\_dict, axis='columns') |
| Outer & inner joins | | |
| Using with arrays | np.arange().reshape() | A = np.arange(8).reshape(2,4) + 0.1  print(A)  [[ 0.1 1.1 2.1 3.1]  [ 4.1 5.1 6.1 7.1]] |
| Stacking arrays horizontally | np.hstack()  np.concatenate(axis=1) | np.hstack([B, A])  np.concatenate([B, A], axis=1) |
| Stacking arrays horizontally | np.vstack()  np.concatenate(axis=0) | np.vstack([A, C])  np.concatenate([A, C], axis=0) |
| Manipulating data as arrays | np.concatenate() | population\_array = np.array(population). ## index bilgisi gider  unemployment\_array = np.array(unemployment)  print(np.concatenate([population\_array, unemployment\_array], axis=1)) |
| Joins |  | ● Joining tables: Combining rows of multiple tables  ● Outer join: Union of index sets (all labels, no repetition)  Missing fields filled with NaN  ● Inner join: Intersection of index sets (only common labels) |
| Concatenation & inner | join='inner' | pd.concat([population, unemployment], axis=1, join='inner')  ## sadece iki set arasinda kesisen satirlari alir |
| Concatenation & outer join | join='outer' | pd.concat([population, unemployment], axis=1, join='outer')  ## birlestirilen datasetlerin tum satirlarini alir |
| Inner join on other axis | join='inner', axis=0 | pd.concat([population, unemployment], join='inner', axis=0) |

Chapter-3: Merging data

|  |  |  |
| --- | --- | --- |
| Merging dataframes | | |
| Merging | .merge() | pd.merge(population, cities). ## ornekte ayni indexe ait bilgiler kullanilmis |
| Merging all columns |  | pd.merge(bronze, gold) |
| Merging on | on= | pd.merge(bronze, gold, on='NOC'). # index mi acaba? |
| Merging on multiple columns |  | pd.merge(bronze, gold, on=['NOC', 'Country']) |
| Using suffixes | suffixes= | pd.merge(bronze, gold, on=['NOC', 'Country'], suffixes=['\_bronze', '\_gold'])  ## sutunlarin varolan basliklarina eklenti yapar |
| Specifying columns to merge | left\_on= , right\_on= | pd.merge(counties, cities, left\_on='CITY NAME', right\_on='City') |
| Joining DataFrames | | |
| Merge with “how=” | how= | pd.merge(bronze, gold, on=['NOC', 'Country'], how='inner')  how=’inner’ -🡪 kesisenleri alir  how=’left’ -🡪 soldaki DF’in tum satirlari ve sagdakinin kesisenleri  how=’right’ -🡪 sagdaki DF’in tum satirlari ve soldakinin kesisenleri  how=’outer’ -🡪 her iki DF’in tum satirlari |
| Using .join | .join() | population.join(unemployment, how= 'right') # “how” hususu ayni |
| Ordered merges | | |
| Using merge\_ordered() | merge\_ordered() | pd.merge\_ordered(hardware, software)  # sorted\_values’un tarihleri arka arkaya dizer, “ordered” tarih sirasi yapar |

Chapter-4: Case Study - Summer Olympics

|  |  |  |
| --- | --- | --- |
|  | .copy() |  |
|  | .ffill() | weather3 = weather1.reindex(year).ffill() |
|  | .last() | aggregation method to select the last element when resampling. |
|  | "%s\_top5.csv" % | file\_name = "%s\_top5.csv" % medal  ### The expression "%s\_top5.csv" % medal evaluates as a string with the value of medal replacing %s in the format string. |
|  | pd.IndexSlicep | A slicer pd.IndexSlice is required when slicing on the inner level of a MultiIndex. |
|  |  | idx = pd.IndexSlice  # Print all the data on medals won by the United Kingdom  print(medals\_sorted.loc[idx[:,'United Kingdom'],:]) |
|  | pd.merge\_asof() |  |
|  | .expanding() |  |
|  | pd.melt() |  |