Reading: Reference guide: Pandas tools for structuring a dataset

As you've learned, there are far too many Python functions to memorize all of them. That's why, as every data professional will tell you, you'll be using reference sheets and coding libraries nearly every day in your data analysis work.

The following reference guide will help you identify the most common Pandas tools used for structuring data. Note that this is just for reference. For detailed information on how each method works, including explanations of every parameter and examples, refer to the linked documentation.

Save this course item

You may want to save a copy of this guide for future reference. You can use it as a resource for additional practice or in your future professional projects. To access a downloadable version of this course item, click the link below and select "Use Template."

Reference guide: Pandas tools for structuring a dataset

OR

If you don't have a Google account, you can download the item directly from the attachment below.

Reference guide: Pandas tools for structuring a dataset

Combine data

Note that for many situations that require combining data, you can use a number of different functions, methods, or approaches. Usually you're not limited to a single "correct" function. So if these functions and methods seem very similar, don't worry! It's because they are! The best way to learn them, determine what works best for you, and understand them is to use them!

df.merge()

- A method available to the **DataFrame** class.
- Use **df.merge()** to take columns or indices from other dataframes and combine them with the one to which you're applying the method.
- Example:
- **Note:** The following code block is not interactive.

```
df1.merge(df2, how='inner', on=['month','year'])
```

pd.concat()

- A pandas function to combine series and/or dataframes
- Use **pd.concat()** to join columns, rows, or dataframes along a particular axis
- Example:
- **Note:** The following code block is not interactive.

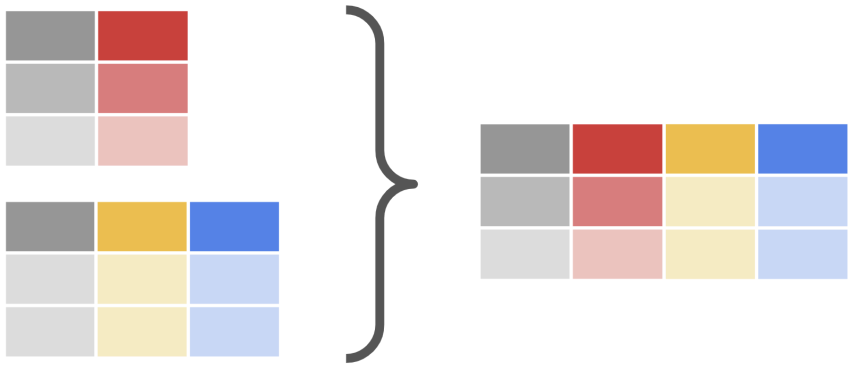
```
df3 = pd.concat([df1.drop(['column_1','column_2'], axis=1), df2])
```

df.join()

- A method available to the **DataFrame** class.
- Use **df.join()** to combine columns with another dataframe either on an index or on a key column. Efficiently join multiple **DataFrame** objects by index at once by passing a list.
- Example:
- **Note:** The following code block is not interactive.

```
df1.set_index('key').join(df2.set_index('key'))
```

Visual representation of a combination:



Extract or select data

df[[columns]]

- Use **df[[columns]]** to extract/select columns from a dataframe.
  Example:

```
print(df)
print()
df[['animal', 'legs']]
```

Output:

```
     animal     class     color    legs
0    cardinal   Aves      red      2
1    gecko      Reptilia  green    4
2    raven      Aves      black    2

        animal     legs
0       cardinal   2
1       gecko      4
2       raven      2
```

[df.select_dtypes()]

- A method available to the **DataFrame** class.
- Use **df.select_dtypes()** to return a subset of the dataframe's columns based on the column dtypes (e.g., **float64**, **int64**, **bool**, **object**, etc.).

Example:

```
print(df)
print()
df2 = df.select_dtypes(include=['int64'])
df2
```

Output:

```
       animal     class     color    legs
0      cardinal   Aves      red      2
1      gecko      Reptilia  green    4
2      raven      Aves      black    2

     legs
0    2
1    4
2    2
```

Visual representation of extraction:

Filter data

Recall from Course 2 that Boolean masks are used to filter dataframes.

df[condition]

- Use **df[condition]** to create a Boolean mask, then apply the mask to the dataframe to filter according to selected condition.
- Example:

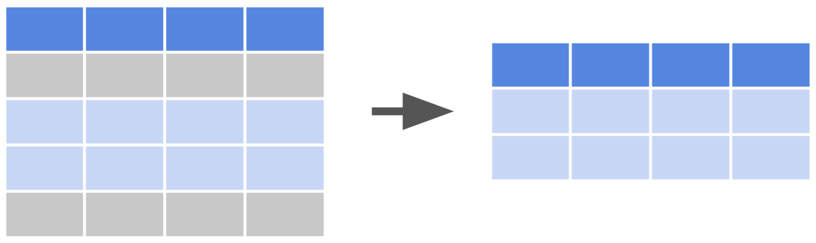
```
print(df)
print()
df[df['class']=='Aves']
```

Output:

```
      animal     class      color    legs
0     cardinal   Aves       red       2
1     gecko      Reptilia   green     4
2     raven      Aves       black     2

      animal     class    color   legs
0      cardinal  Aves     red      2
2      raven     Aves     black    2
```

Visual representation of filtering:



Sort data

[df.sort_values()]

- A method available to the **DataFrame** class.
- Use **df.sort_values()** to sort data according to selected parameters.
- Example:

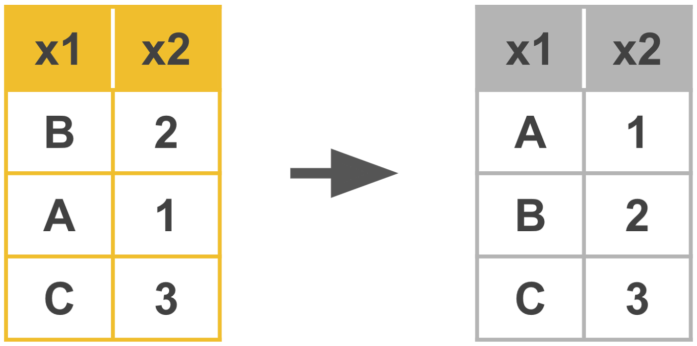
```
print(df)
print()
df.sort_values(by=['legs'], ascending=False)
```

Output:

```
      animal     class     color    legs
0     cardinal   Aves      red       2
1     gecko      Reptilia  green     4
2     raven      Aves      black     2

       animal    class       color     legs
1      gecko     Reptilia    green      4
0      cardinal  Aves        red        2
2      raven     Aves        black      2
```

Visual representation of sorting:



Slice data

[df.iloc[]]

- Use **df.iloc[]** to slice a dataframe based on an integer index location.
- Examples:

  df.iloc[5:10, 2:] → selects only rows 5 through 9, at columns 2+
  df.iloc[5:10] → selects only rows 5 through 9, all columns
  df.iloc[1, 2] → selects value at row 1, column 2
  df.iloc[[0, 2], [2, 4]] → selects only rows 0 and 2, at columns 2 and 4

[df.loc[]]

- Use **df.loc[]** to slice a dataframe based on a label or Boolean array.
- Example:

```
print(df)
print()
df.loc[:, ['color', 'class']]
```

Output:

```
     animal     class     color    legs
0    cardinal   Aves      red       2
1    gecko      Reptilia  green     4
2    raven      Aves      black     2

      color     class
0     red       Aves
1     green     Reptilia
2     black     Aves
```

Key takeaways

The tools in this reference guide are foundational to structuring data, including filtering, sorting, merging, and slicing. You will find yourself using them throughout your career as a data professional.

Resources for more information

Refer to these links for more details on Python functions and their various parameters.

- Pandas documentation to describe parameters in Python functions
- W3schools provides explanations for Python functions in an easy-to-understand way