Python Data Frames and Matrices Cheat Sheet

Hamed Ahmadinia, Ph.D.
Hamed.Ahmadinia@metropolia.fi

Arrays and Matrices

Creating Arrays:

```
import numpy as np
arr = np.array([1, 2, 3])  # Convert list to array
zeros = np.zeros((3,3))    # 3x3 matrix of zeros
ones = np.ones((2,4))      # 2x4 matrix of ones
np.arange(0, 10, 2)     # Evenly spaced values from 0 to 8
np.linspace(0, 1, 5)    # 5 evenly spaced values from 0 to 1
np.eye(3)                  # 3x3 Identity matrix
```

Reshaping Arrays:

```
arr.reshape((2, 3)) # Reshape 1D array to 2D (2x3)
arr.flatten()     # Flatten multi-dimensional array to 1D
arr.T               # Transpose: Swap rows and columns
arr[:, np.newaxis]  # Add a new axis for reshaping
```

NumPy Operations

- **NaN (Not a Number):** Represents missing or invalid data.
- **Inf and -Inf (Infinity):** Represents extremely large or small values.
- **np.isinf(arr)** to check for infinity.

Mathematical Operations:

```
np.add(a, b)       # Element-wise addition
np.subtract(a, b)  # Element-wise subtraction
np.multiply(a, b)  # Element-wise multiplication
np.divide(a, b)    # Element-wise division
np.dot(A, B)       # Matrix multiplication
np.linalg.inv(A)   # Compute inverse of matrix A
np.linalg.det(A)   # Compute determinant of matrix A
np.sum(A, axis=0)  # Sum along columns
np.mean(A)         # Compute mean of array
```

Concatenation and Sorting:

```
np.concatenate((a, b), axis=0) # Row-wise join
np.vstack((a, b))  # Stack arrays vertically
np.hstack((a, b))  # Stack arrays horizontally
np.sort(arr)       # Sort array in ascending order
np.argsort(arr)    # Get indices of sorted elements
```

Pandas DataFrames

Creating a DataFrame:

```
import pandas as pd
df = pd.DataFrame({'A': [1,2,3], 'B': [4,5,6]})
```

Selecting and Filtering Data:

```
df['A']         # Select column A
df.loc[0]       # Select first row
df[df['A'] > 2] # Filter rows
```

Data Editing and Transformation

Modifying Data:

```
df['C'] = df['A'] + df['B']  # Create new column
df.drop('B', axis=1)        # Remove column B
df.fillna(0)                # Replace NaN with 0
```

Sorting and Grouping:

```
df.sort_values('A')    # Sort by column A
df.groupby('A').sum() # Group and aggregate
```

Merging and Concatenation

Merging DataFrames:

```
pd.merge(df1, df2, on='key')    # Merge on key
pd.concat([df1, df2], axis=0)  # Append rows
```

Time-Series Data

What is Time-Series Data?
Time-series data consists of observations recorded at successive time intervals.

Key Functions for Time-Series Handling

Function	Description
`to_datetime()`	Converts a column to datetime format
`date_range()`	Generates a range of dates
`set_index()`	Sets the date column as index
`resample()`	Changes data frequency (e.g., daily to monthly)
`ffill()`	Forward-fills missing values
`bfill()`	Backward-fills missing values

Handling Dates

Converting and Indexing Dates:

```
pd.to_datetime(df['date']) # Convert column to datetime
df.set_index('date', inplace=True) # Set column as index
df['2024-01':'2024-02'] # Select data within range
```

Resampling Time-Series Data

Down-Sampling: Converting Daily Data to Monthly Aggregates

```
df.resample('M').sum()   # Sum values for each month
df.resample('M').mean() # Compute monthly average
```

Up-Sampling: Converting Daily Data to Hourly (Filling Gaps)

```
df.resample('H').ffill()  # Forward-fill missing values
df.resample('H').bfill()  # Backward-fill missing values
```

References: VanderPlas (2016), McKinney (2017)