INFO5002: Intro to Python for Info Sys

Pandas

pandas.pydata.org

Northeastern University

Slides created by: Zachary Doucet

Even more power

- Pandas adds more data structures and power over Numpy.

```
pip install pandas
```

```
import pandas as pd
```

Series and DataFrame as primitive

- Series: labelled one-dimensional array holding data of any type; integers, strings, Python object's.

```
s = pd.Series([4, 5, 12, np.nan, 32, 18])
```

```
0     4.0
1     5.0
2    12.0
3     NaN
4    32.0
5    18.0
dtype: float64
```

- DataFrame: two-dimensional data structure that acts like a table with rows and columns.

```
df = pd.DataFrame(np.array, index=row_names, columns=colum_names)
```

	A	B	C	D
2013-01-01	-0.365031	0.701977	0.381228	1.564787
2013-01-02	0.345290	-0.739007	-0.178305	1.069980
2013-01-03	0.675831	0.886833	-1.258269	1.183045
2013-01-04	0.448686	-0.578519	0.427933	-0.159340
2013-01-05	1.464388	2.221634	1.273367	1.046402
2013-01-06	-1.091492	0.479029	-2.464129	-2.946307

- Can also pass as a dictionary.

```
df2 = pd.DataFrame(
    {
        "A": 1.0,
        "B": pd.Timestamp("20130102"),
        "C": pd.Series(1, index=list(range(4)), dtype="float32"),
        "D": np.array([3] * 4, dtype="int32"),
        "E": pd.Categorical(["test", "train", "test", "train"]),
        "F": "foo",
    }
)
```

	A	B	C	D	E	F
0	1.0	2013-01-02	1.0	3	test	foo
1	1.0	2013-01-02	1.0	3	train	foo
2	1.0	2013-01-02	1.0	3	test	foo
3	1.0	2013-01-02	1.0	3	train	foo

Useful functions and attributes

- df.head(n=5): return the *n* first rows.
- df.tail(n=5): return the *n* last rows.
- df.index: return the row labels.
- df.columns: return the col labels.
- df.to_numpy(): return a Numpy representation.

- df.describe(): shows quick statistical summary.
- df.T: transpose
- df.sort_index(axis=0, ascending=True)
- df.sort_values(by, axis=0, ascending=True)

	A	B	C
count	100.000000	100.000000	100.000000
mean	102.366338	100.033401	99.982868
std	26.092785	25.816671	26.665102
min	31.517570	21.936161	22.466281
25%	84.552389	82.723932	80.393764
50%	103.406103	100.215987	101.635768
75%	119.650715	114.003662	117.620123
max	171.914097	170.454752	168.754208

Indexing

- Pass in a single argument in brackets to get a series of the corresponding column. `df["C"]`
- Pass in a slice to get matching rows.

row col

```
df.iloc[7:10, 1:3]
```

- If you want to work with direct indexing can use **iloc**.

Conditional selection

- Similar to numpy.

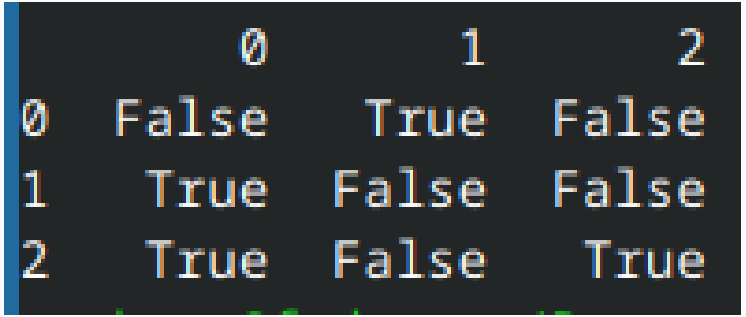
```
df[df["A"]>18]
```

- Can use **isin()** for non-number data.

```
df[df2["C"].isin(["rain", "storm"])]
```

Missing data

- df.dropna(): drop any row with missing data.
- df.fillna(value=None): fill any missing data with *value*.
- df.isna(): returns a new DataFrame where True appears in each cell with missing data.



	0	1	2
0	False	True	False
1	True	False	False
2	True	False	True

Importing/Exporting

- pd.read_csv(“filepath.csv”): to read a csv and load as DataFrame.
- df.to_csv(“filepath.csv”): save DataFrame to csv.
- pd.read_parquet(“filepath.parquet”)
- df.to_parquet(“filepath.parquet”)
- pd.read_excel(“filepath.xlsx”)
- df.to_excel(“filepath.xlsx”)