Applied Data Analytics

Pandas basics

DataFrames: Indexing and slicing

Hans-Martin von Gaudecker

Example: Egypt

- Index: Year
- Columns: Population and GDP per capita

year	pop	gdpPercap
1952	22.2	1418.8
1977	38.8	2785.5
2002	73.3	4754.6

Indexing rows using .loc[] / .iloc[]

```
[1] egypt.loc[1977]
[1] pop            38.8
    gdpPercap    2785.5
    Name: 1977, dtype: float64

[2] egypt.iloc[1]
[2] pop            38.8
    gdpPercap    2785.5
    Name: 1977, dtype: float64
```

- Same syntax as for Series
- Now returns a Series
- Name of the Series is the index' value

Indexing columns using []

```
[3] egypt["pop"]
```

```
[3] year
    1952    22.2
    1977    38.8
    2002    73.3
    Name: pop, dtype: float64
```

```
[4] egypt["gdpPercap"]
```

```
[4] year
    1952    1418.8
    1977    2785.5
    2002    4754.6
    Name: gdpPercap, dtype: float64
```

- Single `[ ]` with column label (typically a string)
- Returns a Series
- Name of the Series is the column label

Slicing columns using [[]]

```
[5] egypt[["gdpPercap", "pop"]]
```

```
[5]       gdpPercap   pop
    year
    1952     1418.8  22.2
    1977     2785.5  38.8
    2002     4754.6  73.3
```

- Double `[[]]` with column labels separated by commas
- Returns a DataFrame
- Order of columns is the one we put into `[[]]`

Slicing columns using [[]]

```
[6] egypt[["gdpPercap"]]
```

```
[6]       gdpPercap
    year
    1952     1418.8
    1977     2785.5
    2002     4754.6
```

```
[7] egypt[[]]
```

```
[7] Empty DataFrame
    Columns: []
    Index: [1952, 1977, 2002]
```

- Double `[[]]` with a single column label
- Returns a DataFrame with one column
- Double `[[]]`, empty
- Returns a DataFrame with no columns

Indexing elements using .loc[row, col]

```
[8] egypt.loc[1977, "pop"]
```

```
[8] np.float64(38.8)
```

- `.loc[]` allows you to specify both row and column (separate by comma)
- Then returns a single element