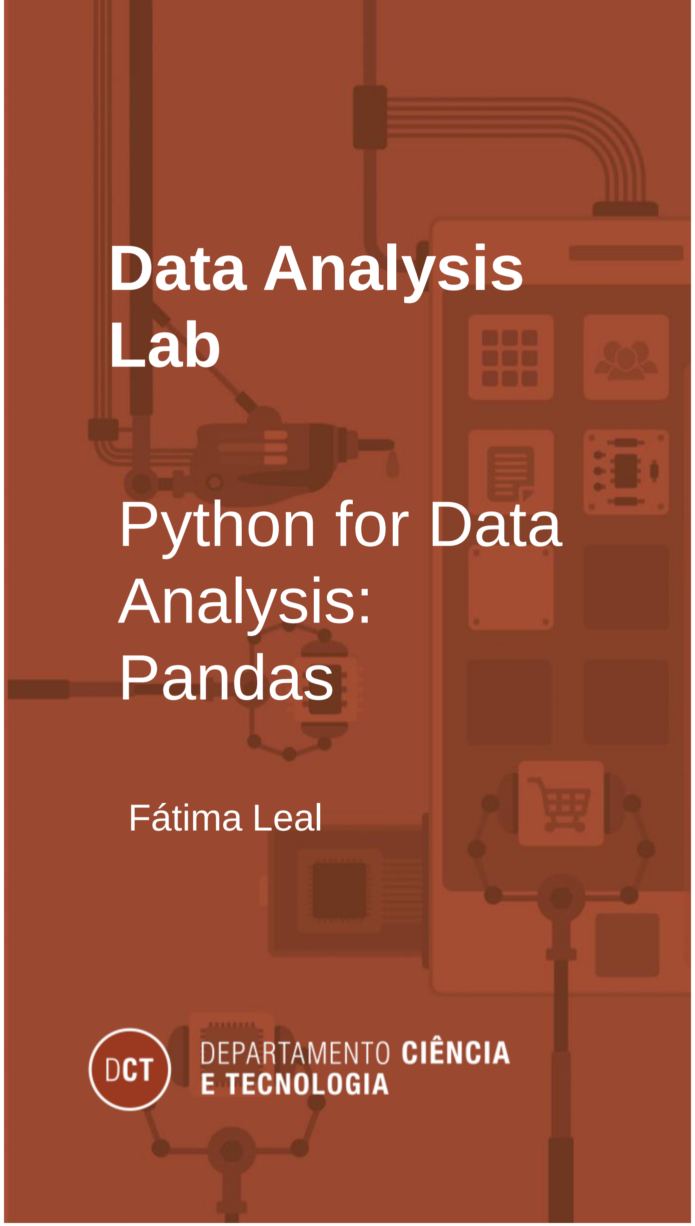


Data Analysis Lab

Python for Data Analysis: Pandas

Fátima Leal

DCT DEPARTAMENTO CIÊNCIA E TECNOLOGIA

UPT UNIVERSIDADE PORTUCALENSE

Content

- Python for Data Analysis
- Practical Examples
- WorkSheet 3

Python: Pandas

- Pandas is a newer package built on top of NumPy
- Provides an efficient implementation of a ***DataFrame***
- DataFrames are essentially multidimensional arrays with attached row and column labels
- Heterogeneous types and/or missing data
- Explore Series and DataFrames
- Provides tools for data manipulation: reshaping, merging, sorting, slicing, aggregation, etc.
- Allows handling missing data

Python: Pandas

- Taking the data structure from the mini recommendation project

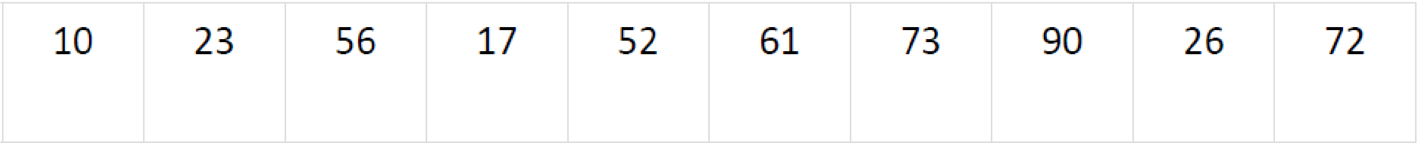
```
Import pandas as pd
users = dict()
users = {1: [5,3,5,2,4,3,3], 2: [3,3,4,5,3,2,5], 3:
[4,4,5,5,2,2,3]}

dfUsers = pd.DataFrame(users)
```

- What do you see as result?
- Give names to columns
- Give names to rows (index)

Python: Pandas Series Object

- Series is a one-dimensional array like structure with homogeneous data. For example, the following series is a collection of integers 10, 23, 56,….

- …

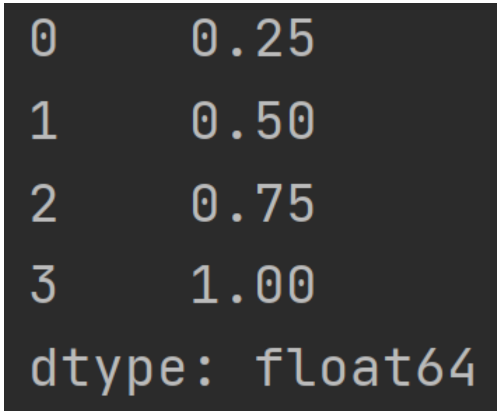


10	23	56	17	52	61	73	90	26	72

Python: Pandas Series Object

Pandas Series is a one-dimensional array of indexed data. It can be created from a list or array as follows:

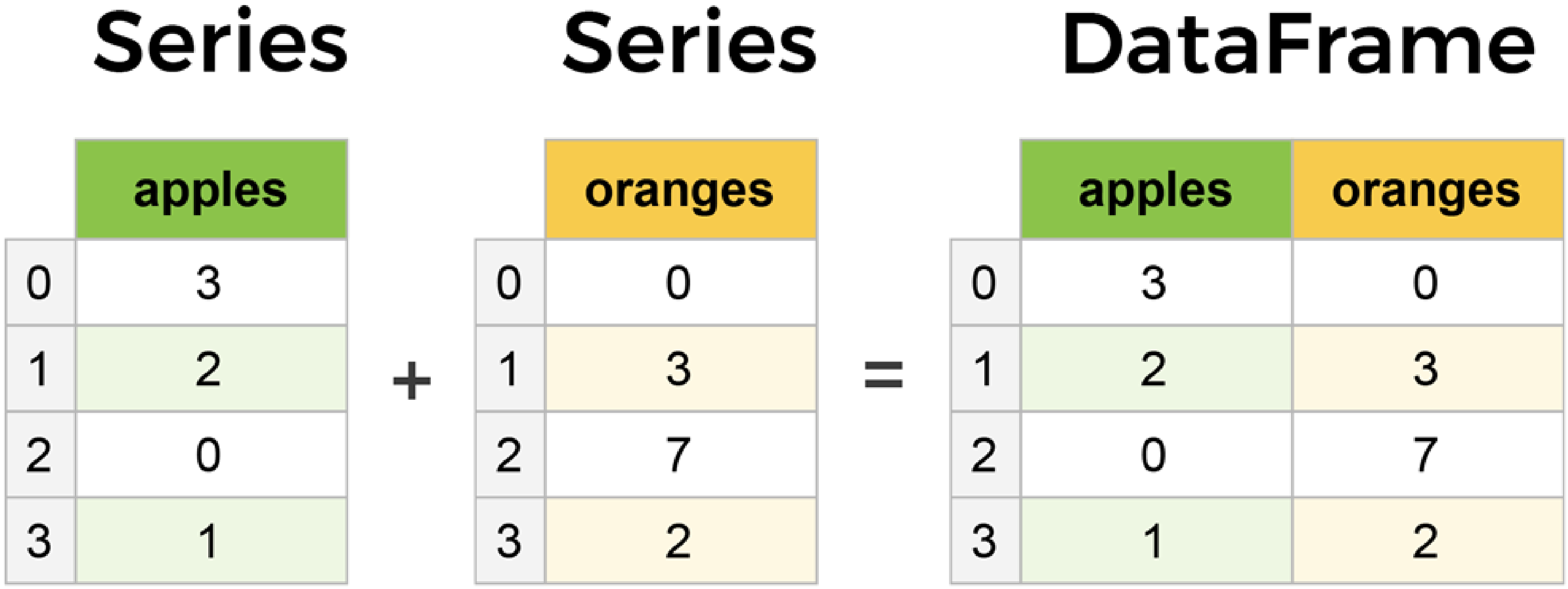
```
data = pd.Series([0.25, 0.5, 0.75, 1.0])
print(data)
```



```
0     0.25
1     0.50
2     0.75
3     1.00
dtype: float64
```

```
data.values #?

data[1] #?

data[1:3] #?

data.index #?
```

Python: Pandas Series Object



Series

	apples
0	3
1	2
2	0
3	1

+

Series

	oranges
0	0
1	3
2	7
3	2

=

DataFrame

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

Python: Pandas DataFrame

- DataFrame is a two-dimensional array with heterogeneous data. For example,

Name	Age	Gender	Rating
Steve	32	Male	3.45
Lia	28	Female	4.6
Vin	45	Male	3.9
Katie	38	Female	2.78

Python: Pandas DataFrame

- Two-dimensional, size-mutable and potentially heterogeneous tabular data

```
users = {1: [5,3,5,2,4,3,3], 2: [3,3,4,5,3,2,5], 3:
[4,4,5,5,2,2,3]}
dfUsers = pd.DataFrame(users).T
```

- Experiment the following commands and see the results:

```
df.head() #?

df.tail(3) #?

df.describe() #?

df.T#?

df.size#?
```

```
df.sort_index(axis=1, ascending=False) #?

df.sort_values(by="b") #?

df[“a"] #?

df[0:3] #?
```

Python: Pandas DataFrame

- A pandas DataFrame can be created using various inputs like:
  - Lists
  - Dict
  - Series
  - Numpy arrays
  - From files
  - Another data frame

```
import pandas as pd
 data = [_______]
 df = pd.DataFrame(data)
 print df
```

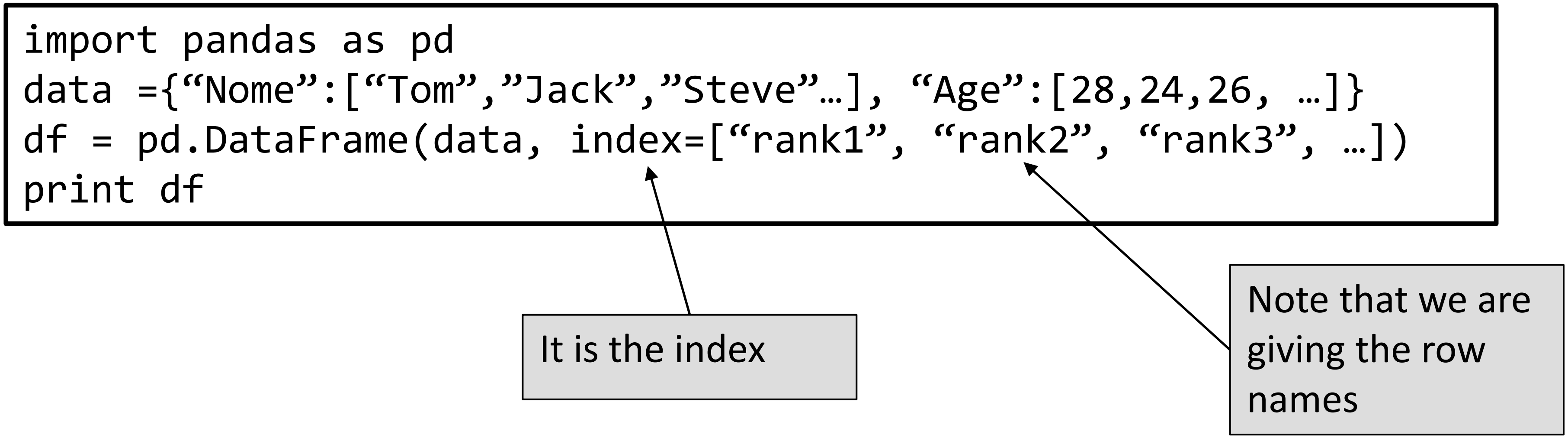
```
import pandas as pd
data = [[‘Joao’, 16],[],…]
df = pd.DataFrame(data, columns=[“Name”, “Age”])
print df
```

Note that we are giving the columns names

Columns labels

Python: Pandas DataFrame

- More examples:



```
import pandas as pd
data ={“Nome”:[“Tom”,”Jack”,”Steve”…], “Age”:[28,24,26, …]}
df = pd.DataFrame(data, index=[“rank1”, “rank2”, “rank3”, …])
print df
```

It is the index

Note that we are giving the row names

Pandas DataFrame Atributes

Python objects have *attributes* and *methods*

df.attribute	description
dtypes	list the types of the columns
columns	list the column names
axes	list the row labels and column names
ndim	number of dimensions
size	number of elements
shape	return a tuple representing the dimensionality
values	numpy representation of the data

Pandas DataFrame Methods

Unlike attributes, python methods have parenthesis.

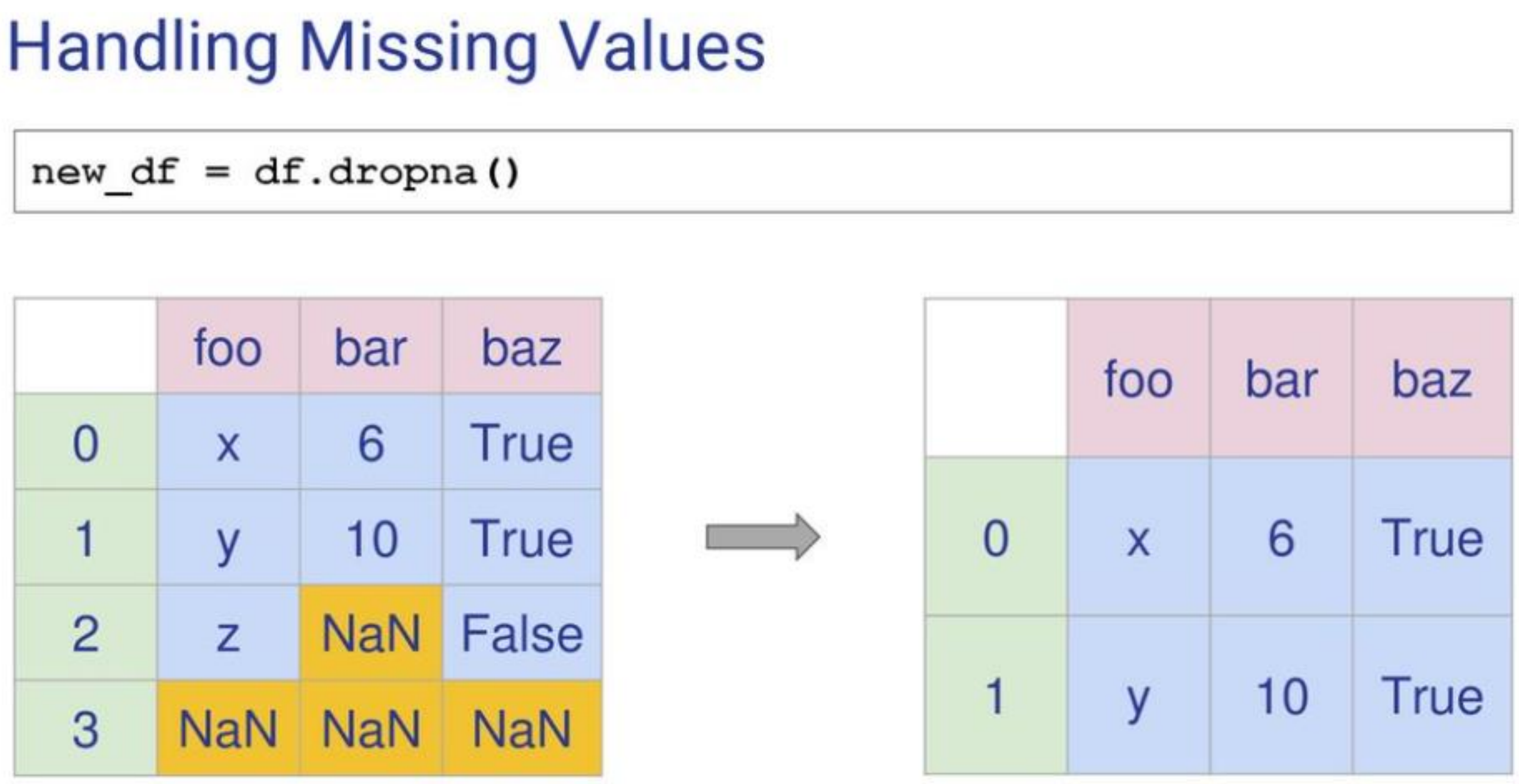
df.method()	description
head([n]), tail([n])	first/last n rows
describe()	generate descriptive statistics (for numeric columns only)
max(), min()	return max/min values for all numeric columns
mean(), median()	return mean/median values for all numeric columns
std()	standard deviation
sample([n])	returns a random sample of the data frame
dropna()	drop all the records with missing values

Pandas DataFrame

There are several methods to deal with missing values in a data frame:

df.method()	description
dropna()	Drop missing observations
dropna(how='all')	Drop observations where all cells is NA
dropna(axis=1, how='all')	Drop column if all the values are missing
dropna(thresh = 5)	Drop rows that contain less than 5 non-missing values
fillna(0)	Replace missing values with zeros
isnull()	returns True if the value is missing
notnull()	Returns True for non-missing values

Pandas DataFrame



Handling Missing Values

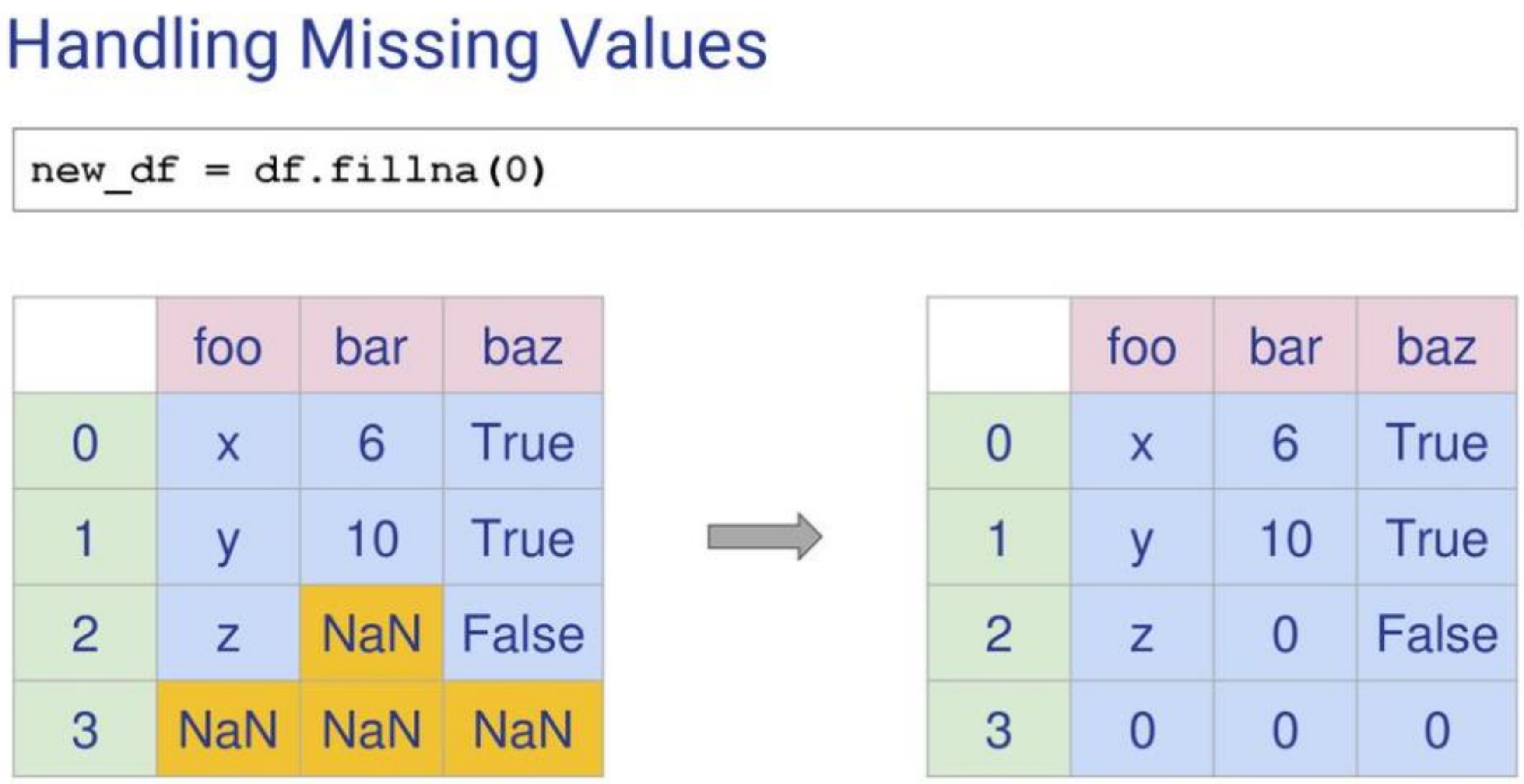
```
new_df = df.dropna()
```

	foo	bar	baz
0	x	6	True
1	y	10	True
2	z	NaN	False
3	NaN	NaN	NaN

→

	foo	bar	baz
0	x	6	True
1	y	10	True

Pandas DataFrame



Handling Missing Values

```
new_df = df.fillna(0)
```

	foo	bar	baz
0	x	6	True
1	y	10	True
2	z	NaN	False
3	NaN	NaN	NaN

→

	foo	bar	baz
0	x	6	True
1	y	10	True
2	z	0	False
3	0	0	0

Pandas DataFrame

- Aggregation - computing a summary statistic about each group, i.e.
  - compute group sums or means
  - compute group sizes/counts

- Common statistic functions:
  - min, max
  - count, sum, prod
  - mean, median, mode, mad
  - std, var

```
df.mean()
```

```
df.min()
```

```
df.std()
```

```
df.describe()
```

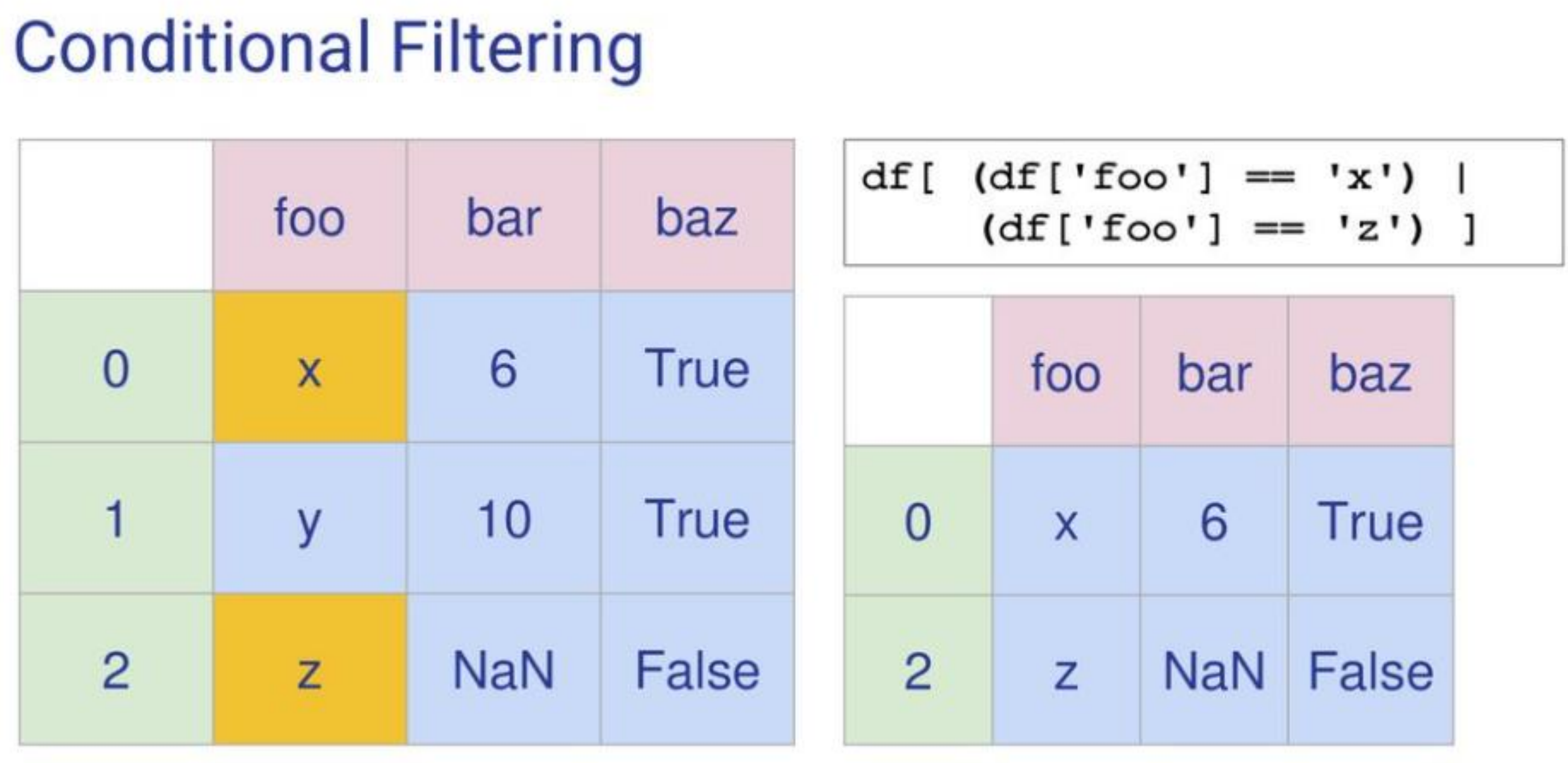
Pandas DataFrame

- **Select a column** in a Data Frame
  - *Method 1*: Subset the data frame using column name: `df[‘a’]`
  - *Method 2*: Use the column name as an attribute: `df.a`

- **Filtering**
  - `df[df['a']>3]`

Any Boolean operator can be used to subset the data:
> greater; >= greater or equal;
< less; <= less or equal;
== equal; != not equal;

Pandas DataFrame



Conditional Filtering

	foo	bar	baz
0	x	6	True
1	y	10	True
2	z	NaN	False

```
df[ (df['foo'] == 'x') |
     (df['foo'] == 'z') ]
```

	foo	bar	baz
0	x	6	True
2	z	NaN	False

Pandas DataFrame

- **Slicing**: there are several ways to subset a DataFrame:
  - one or more columns
  - one or more rows
  - a subset of rows and columns
  - Rows and columns can be selected by their position or label

- When **selecting one column**, it is possible to use **single set of brackets**, but the resulting object **will be a Series** (not a DataFrame):

```
#Select the column a:
df[‘a’]
```

Pandas DataFrame

- **Loc and iloc**
  - loc gets rows (and/or columns) with particular labels
  - iloc gets rows (and/or columns) at integer locations

```
loc[row_label, column_label]
iloc[row_position, column_position]
```

```
#Select the column a:
df.loc[:,‘a’]
df.iloc[:,0]
```

Pandas DataFrame

- If we need to select a range of rows, using their labels we can use method `loc`:

```
#Select rows by their labels:
df.loc[1:3,[‘a’,’b’]]
```

- We can sort the data by a value in the column. By default, the sorting will occur in ascending order and a new data frame is return.

```
#Create a new data frame sorted by the column b
df.sort_values(by='b')
```

Python: Pandas DataFrame

- Two-dimensional, size-mutable and potentially heterogeneous tabular data

```
users = {1: [5,3,5,2,4,3,3], 2: [3,3,4,5,3,2,5], 3:
[4,4,5,5,2,2,3]}
dfUsers = pd.DataFrame(users).T
```

- Experiment the following commands and see the results:

```
df.head() #?

df.tail(3) #?

df.describe() #?

df.T#?

df.size#?
```

```
df.sort_index(axis=1, ascending=False) #?

df.sort_values(by="b") #?

df[“a"] #?

df[0:3] #?
```

Python: Pandas DataFrame with files

Pandas DataFrame

- Pandas and files:

```
movies = pd.read_csv("IMDB-Movie-Data.csv")
```

- Experiment:

```
movies.head()
movies.tail()
movies.info()
movies.shape
movies.columns
```

- Analyse the columns and rows and the corresponding titles

Pandas DataFrame

Exercise:

- Rename each column name to lowercase

```
movies.columns = [col.lower() for col in movies]
print(movies.columns)
```

Pandas DataFrame

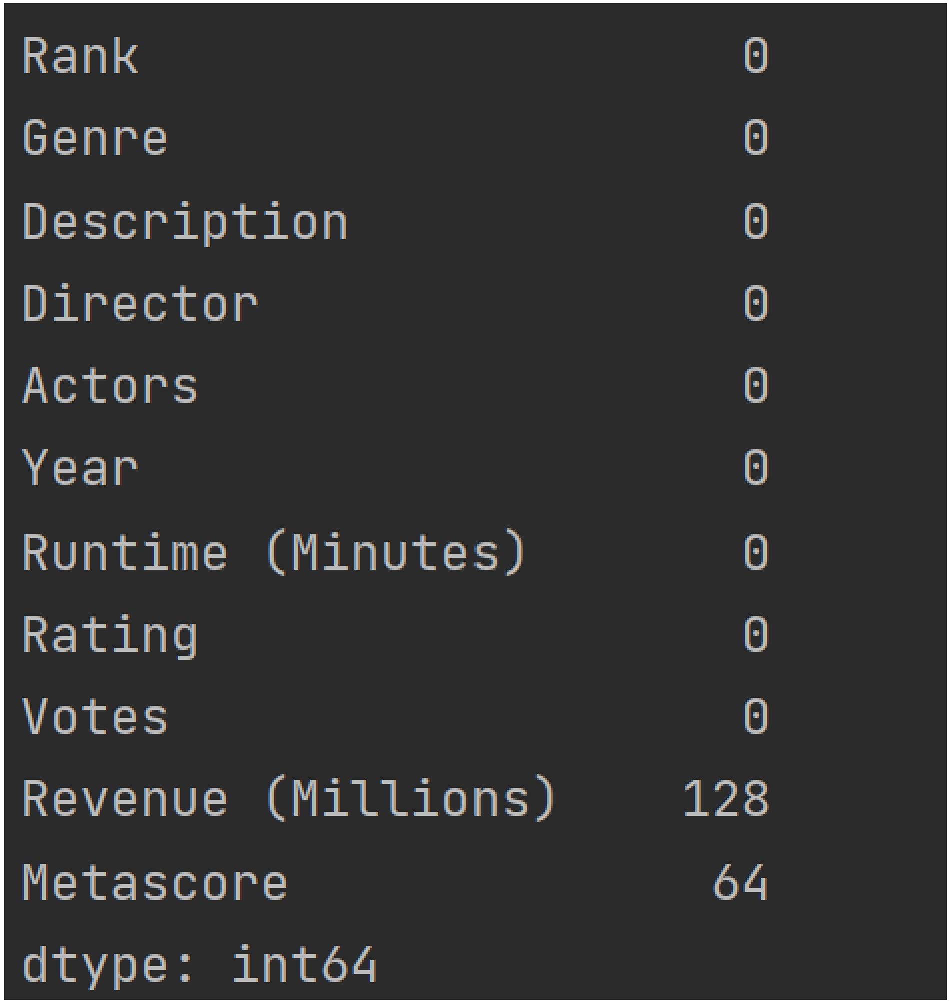
- To count the number of nulls in each column we use an aggregate function for summing:

```
movies.isnull().sum()
```

- .isnull() just by itself isn't very useful, and is usually used in conjunction with other methods, like sum().

Pandas DataFrame

- 128 missing values for revenue_millions
- 64 missing values for metascore



```
Rank                    0
Genre                   0
Description             0
Director                0
Actors                  0
Year                    0
Runtime (Minutes)       0
Rating                  0
Votes                   0
Revenue (Millions)    128
Metascore              64
dtype: int64
```

Pandas DataFrame

- Exercise
  - Remove null values
  - Remove only the columns with null values

```
movies.dropna()
movies.dropna(axis=1)
```

Pandas DataFrame

1. Calculate the average of the revenue column

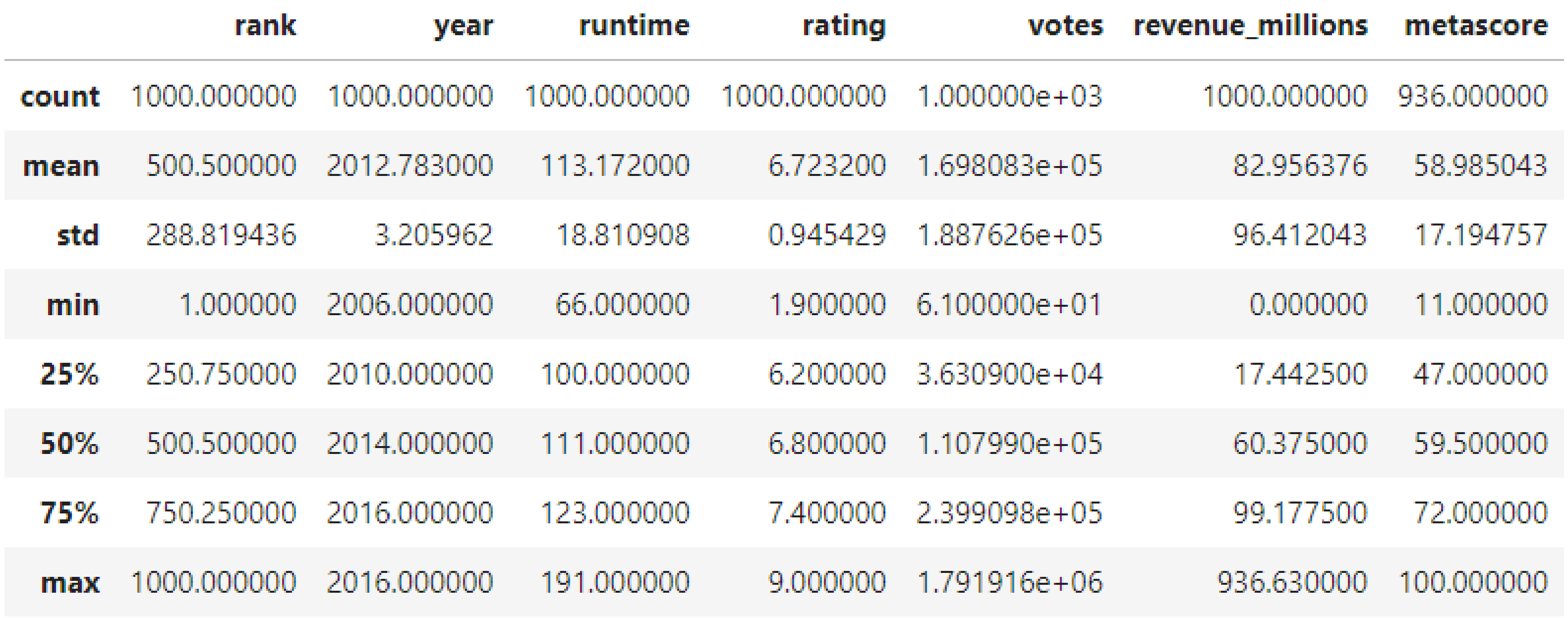
2. Replace the null values with the average changing the values of the dataFrame

3. Check if the the values were actully modified in the data frame

```
revenue = movies[‘Revenue (Millions)’]
revenue_mean= revenue.mean()
revenue.fillna(revenue_mean, inplace=True)
movies.isnull().sum()
```

Pandas DataFrame

- Using describe() on an entire DataFrame we can get a summary of the distribution of continuous variables



	rank	year	runtime	rating	votes	revenue_millions	metascore
count	1000.000000	1000.000000	1000.000000	1000.000000	1.000000e+03	1000.000000	936.000000
mean	500.500000	2012.783000	113.172000	6.723200	1.698083e+05	82.956376	58.985043
std	288.819436	3.205962	18.810908	0.945429	1.887626e+05	96.412043	17.194757
min	1.000000	2006.000000	66.000000	1.900000	6.100000e+01	0.000000	11.000000
25%	250.750000	2010.000000	100.000000	6.200000	3.630900e+04	17.442500	47.000000
50%	500.500000	2014.000000	111.000000	6.800000	1.107990e+05	60.375000	59.500000
75%	750.250000	2016.000000	123.000000	7.400000	2.399098e+05	99.177500	72.000000
max	1000.000000	2016.000000	191.000000	9.000000	1.791916e+06	936.630000	100.000000

Pandas DataFrame

- .describe() can also be used on a categorical variable to get the count of rows, unique count of categories, top category, and freq of top category:

```
movies[‘genre’].describe()

movies['Genre'].value_counts().head(10)
```

Pandas DataFrame

Exercises

1. Using slicing locate the Prometheus movie and see the corresponding details
2. Locate all movies among Prometheus and Sing
3. Select the movies directed by Ridley Scott
4. Select the movies with a rating greater than or equal to 8.0
5. Select movies directed Christopher Nolan OR Ridley Scott
6. All movies that were released between 2005 and 2010 and have a rating above 8.0

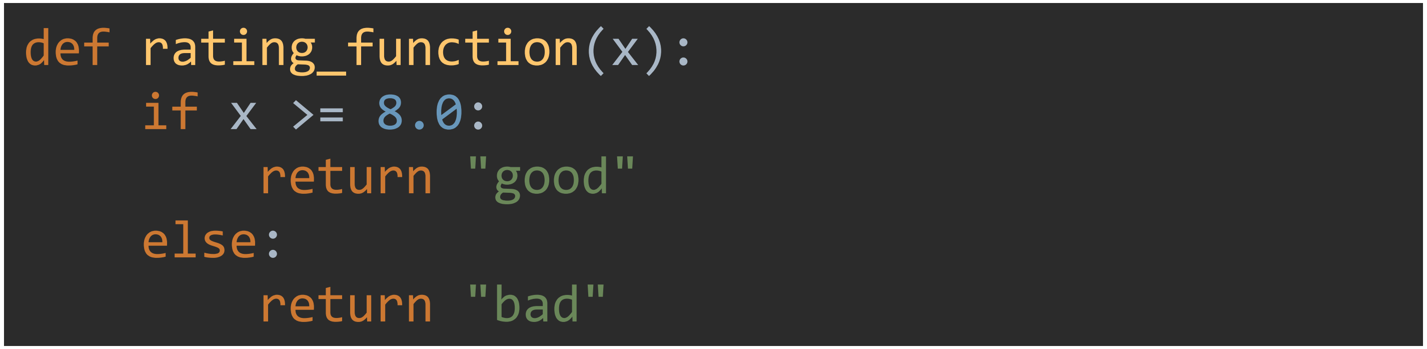
Pandas DataFrame

- Possible solutions

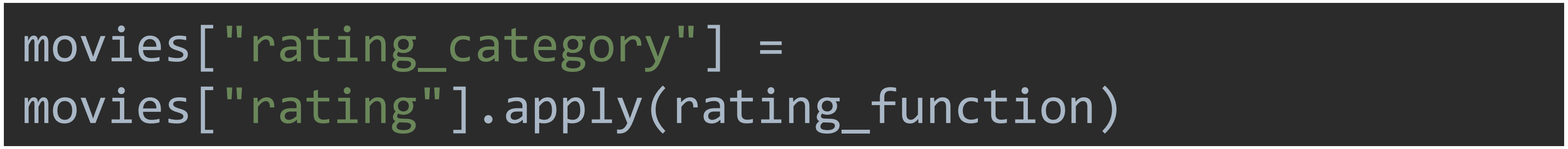
```
#1
movies.loc["Prometheus"]
#2
movies.loc['Prometheus':'Sing’]
#3
movies[movies['director'] == "Ridley Scott“]
#4
movies[movies['rating'] >= 8.0].head(3)
#5
movies_df[(movies_df['director'] == 'Christopher Nolan') |
(movies_df['director'] == 'Ridley Scott')].head()
#6
movies_df[
    ((movies_df['year'] >= 2005) & (movies_df['year'] <= 2010))
    & (movies_df['rating'] > 8.0))]
```

Pandas DataFrame

- Applying functions



```
def rating_function(x):
    if x >= 8.0:
        return "good"
    else:
        return "bad"
```



```
movies["rating_category"] =
movies["rating"].apply(rating_function)
```

Pandas DataFrame

Exercises

- Create a new column in your data frame which classifies with 0 and 1 and -1 the revenues.
  - 0 for the revenues equal to the avegere
  - 1 for the reveneus higher than the average
  - -1 for the revenus lesser than the average

- Create a textual review and using the OpenAI try to classify the sentiment as positive or negative review

UPT

UNIVERSIDADE
PORTUCALENSE

Do conhecimento à prática.