Analytics Jumpstart

Intro to commonly used *pandas* methods

Nashville Software School

Goals for today

- Learn about the *pandas* library
- See some common *pandas* methods
- Work on coding tasks

Introduction

Python is a a general purpose programming language and is widely used for web applications, scientific computing, machine learning, and data analysis.

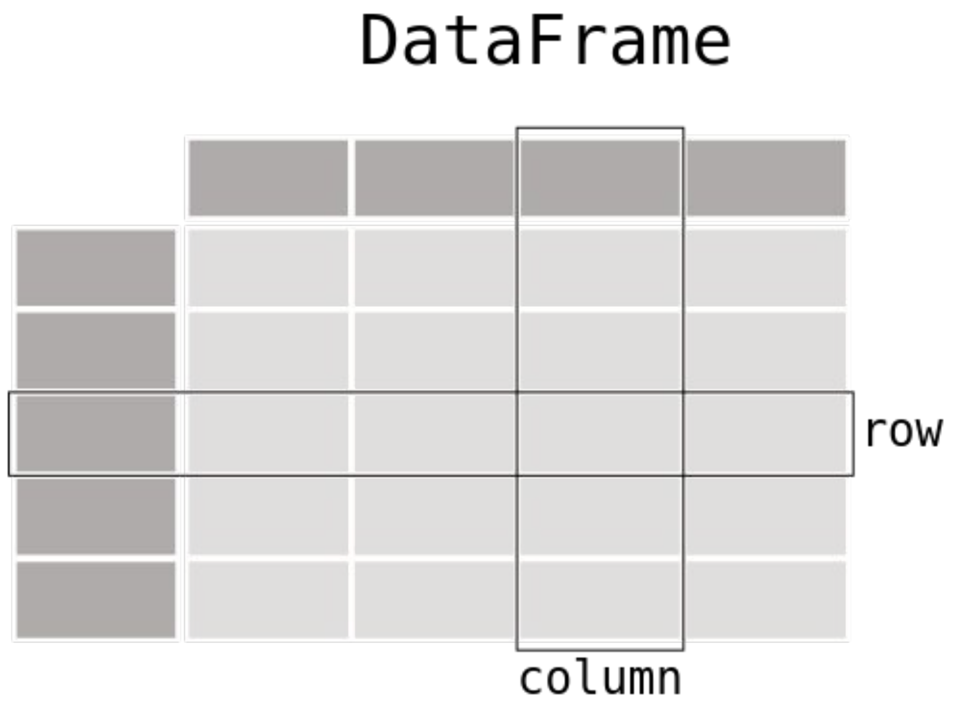
We'll be making use of the *pandas* library, an open source data analysis and manipulation tool.

In Python, a **library** bundles together functions and objects that have a common functionality (like data analysis).

The *pandas* library was installed when you installed Anaconda.

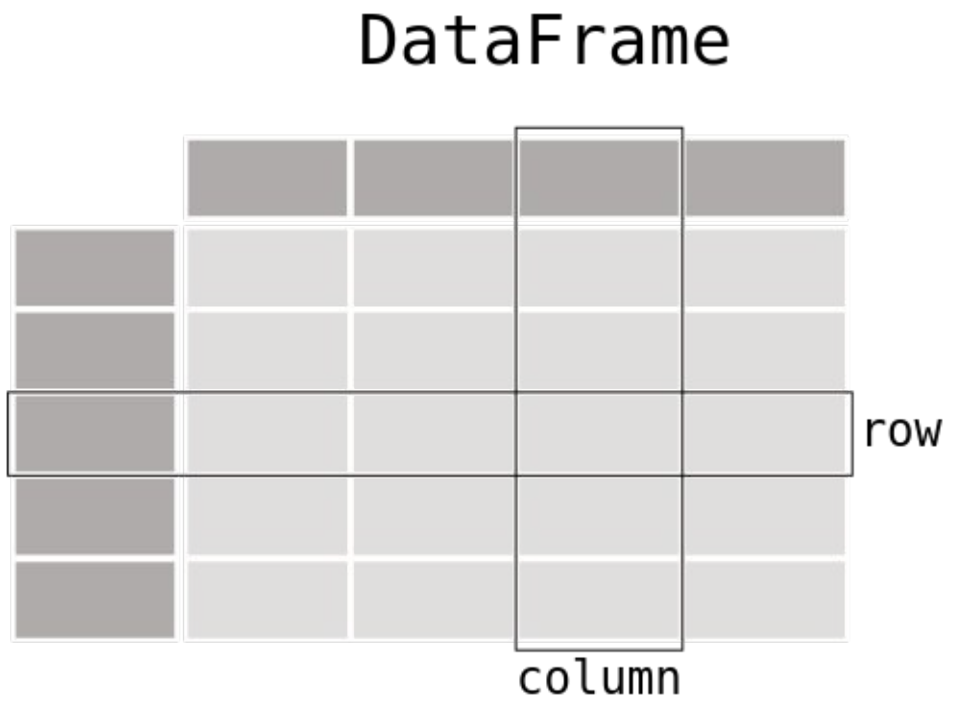
Introduction

The primary tool for working with data in *pandas* is the **DataFrame**, an object which can hold tabular data.

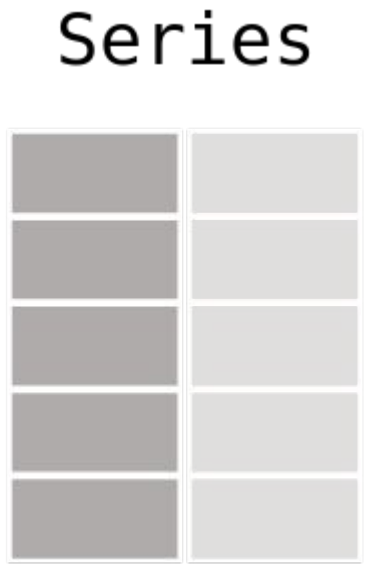


Introduction

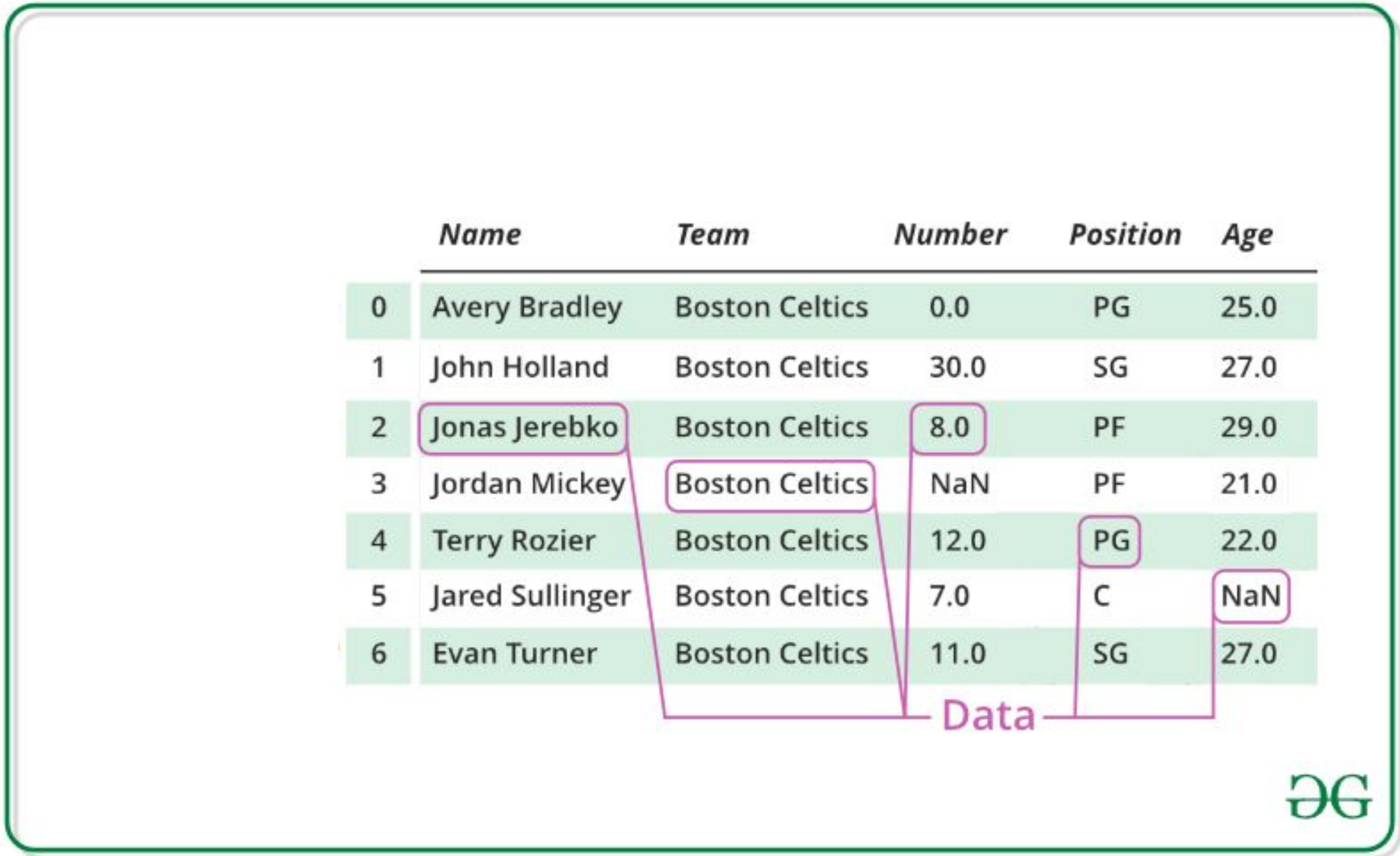
The primary tool for working with data in *pandas* is the **DataFrame**, an object which can hold tabular data.



Each column of a DataFrame is a *pandas* **Series**.



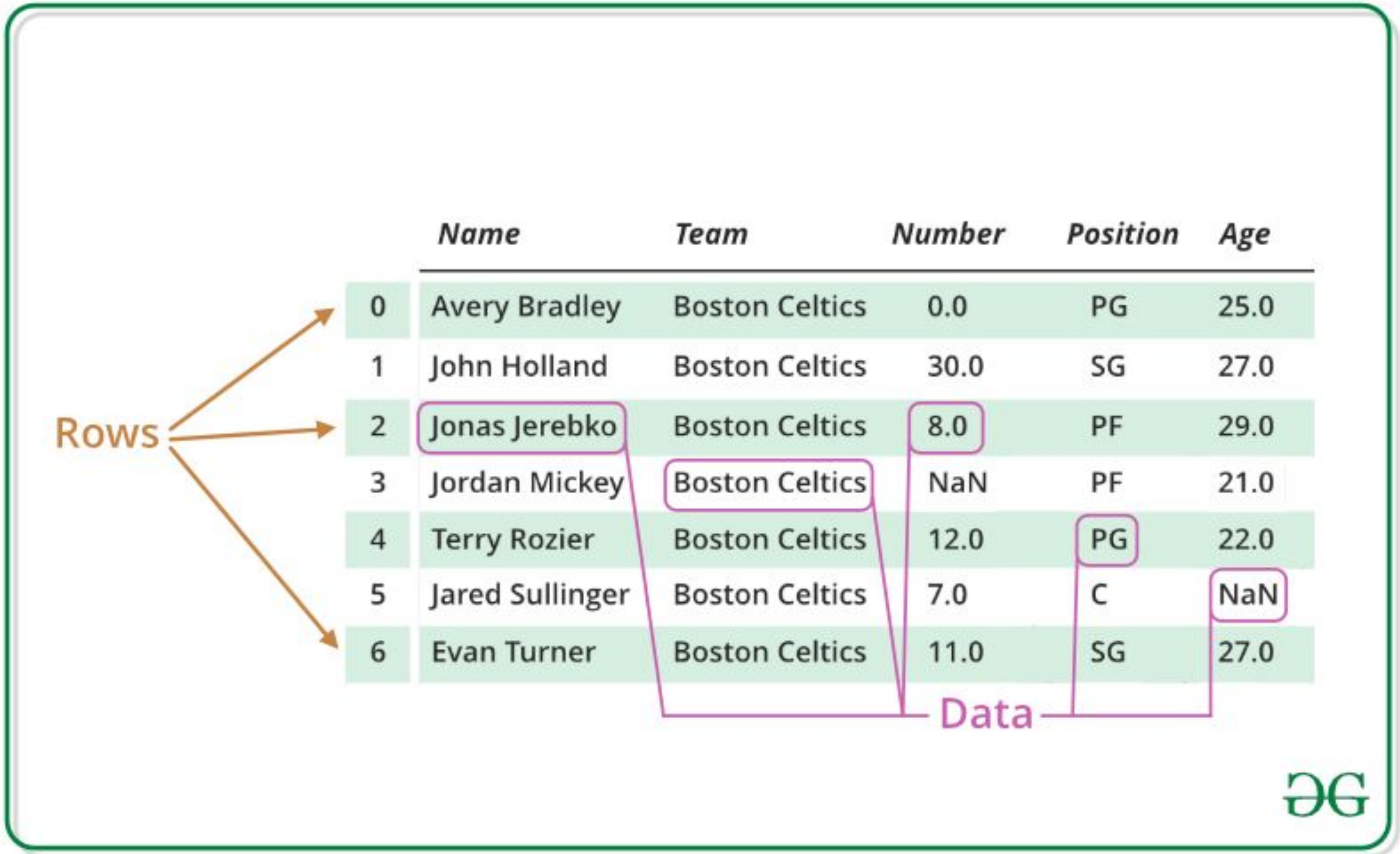
Introduction



	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

https://www.geeksforgeeks.org/python-pandas-dataframe/

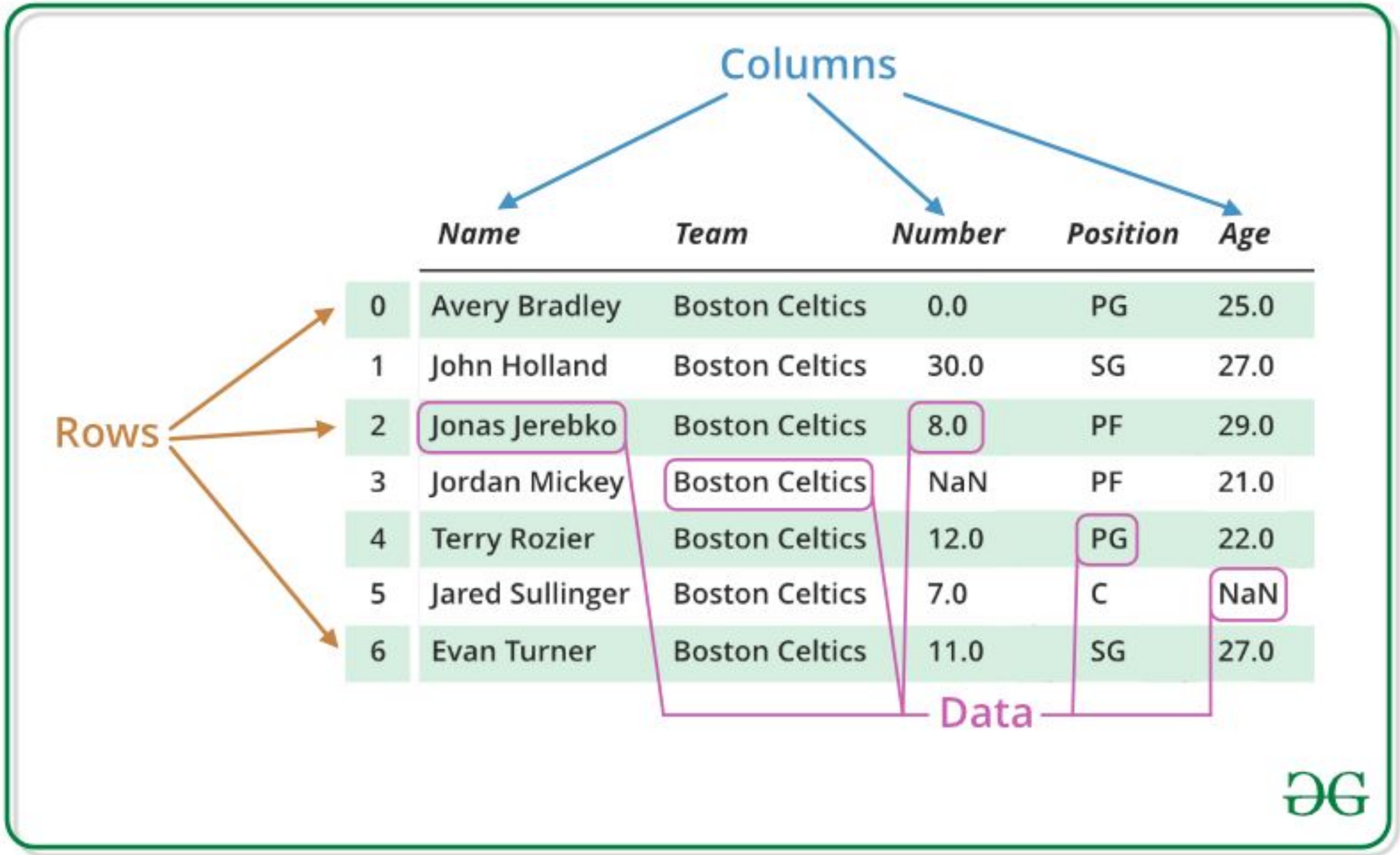
Introduction



	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

https://www.geeksforgeeks.org/python-pandas-dataframe/

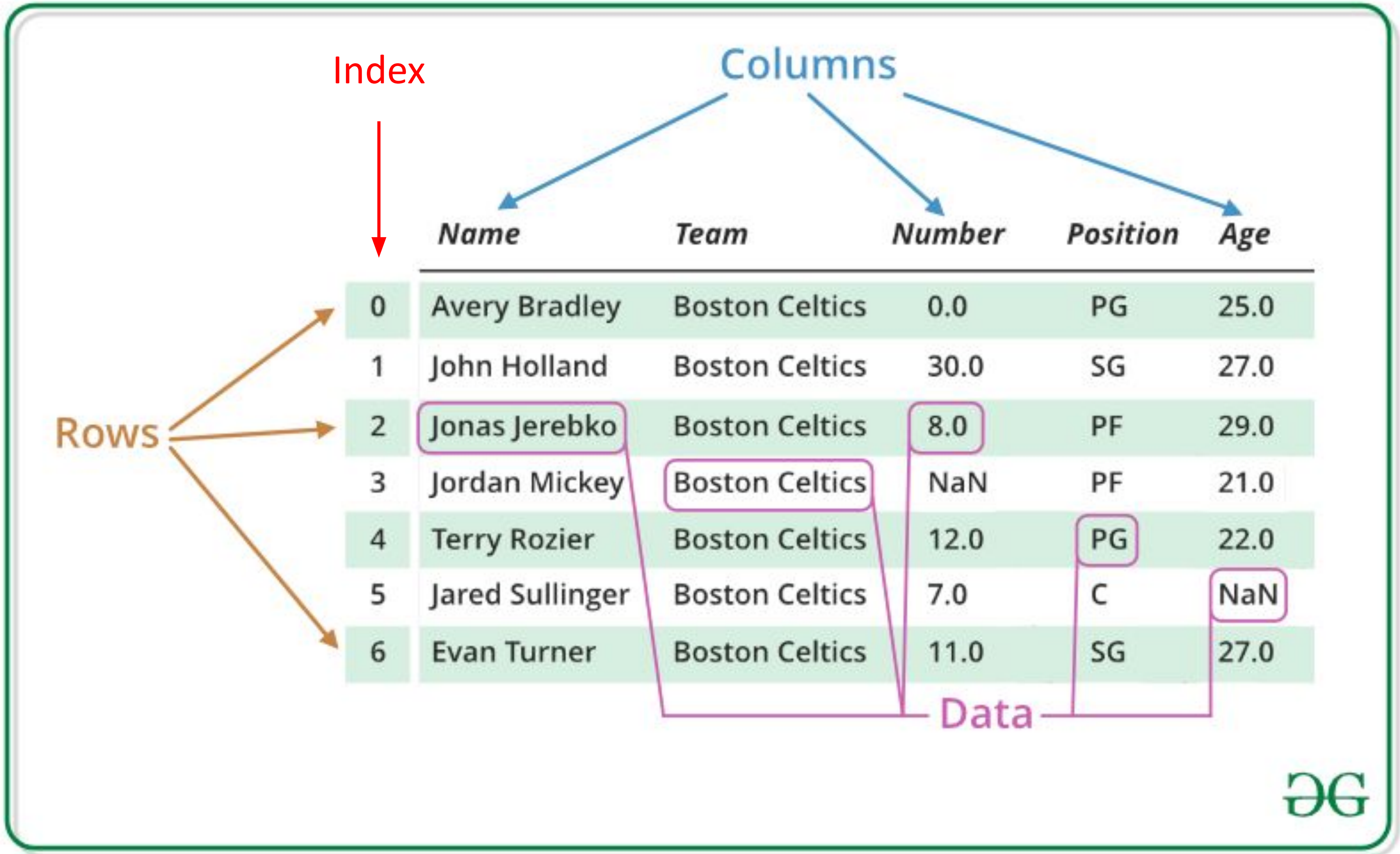
Introduction



	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

https://www.geeksforgeeks.org/python-pandas-dataframe/

Introduction



	Name	*Team*	*Number*	*Position*	*Age*
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

https://www.geeksforgeeks.org/python-pandas-dataframe/

Introduction

To make use of *pandas*, we'll import it using the command

```
import pandas as pd
```

Introduction

To make use of *pandas*, we'll import it using the command

```
import pandas as pd
```

Allows us to use the components of *pandas*.

Introduction

To make use of *pandas*, we'll import it using the command

```
import pandas as pd
```

This is an **alias**, meaning that when referring to *pandas* objects, we will use the abbreviation *pd* instead of the full word.

Introduction

We will make use of **functions**, which you can think of as a way to direct Python to do certain actions, like read data in or modify a DataFrame.

Introduction

We will make use of **functions**, which you can think of as a way to direct Python to do certain actions, like read data in or modify a DataFrame.

When calling a function, you give the name of the function followed by a set of parentheses, which includes any **arguments** that you need to pass in.

Introduction

We will make use of **functions**, which you can think of as a way to direct Python to do certain actions, like read data in or modify a DataFrame.

When calling a function, you give the name of the function followed by a set of parentheses, which includes any **arguments** that you need to pass in.

```
pd.read_csv('../data/public_art.csv')
```

Introduction

We will make use of **functions**, which you can think of as a way to direct Python to do certain actions, like read data in or modify a DataFrame.

When calling a function, you give the name of the function followed by a set of parentheses, which includes any **arguments** that you need to pass in.

```
pd.read_csv('../data/public_art.csv')
```

The name of the function is *read_csv*.

This function reads in data from a **csv (comma separated values)** file into a *pandas* DataFrame.

Introduction

We will make use of **functions**, which you can think of as a way to direct Python to do certain actions, like read data in or modify a DataFrame.

When calling a function, you give the name of the function followed by a set of parentheses, which includes any **arguments** that you need to pass in.

```
pd.read_csv('../data/public_art.csv')
```

It is part of the *pandas* library.

Introduction

We will make use of **functions**, which you can think of as a way to direct Python to do certain actions, like read data in or modify a DataFrame.

When calling a function, you give the name of the function followed by a set of parentheses, which includes any **arguments** that you need to pass in.

```
pd.read_csv('../data/public_art.csv')
```

We need parentheses since we're calling a function.

Introduction

We will make use of **functions**, which you can think of as a way to direct Python to do certain actions, like read data in or modify a DataFrame.

When calling a function, you give the name of the function followed by a set of parentheses, which includes any **arguments** that you need to pass in.

```
pd.read_csv('../data/public_art.csv')
```

Here, we need an argument, which tells the function where to find our data.

Introduction

We will make use of **functions**, which you can think of as a way to direct Python to do certain actions, like read data in or modify a DataFrame.

When calling a function, you give the name of the function followed by a set of parentheses, which includes any **arguments** that you need to pass in.

```
art = pd.read_csv('../data/public_art.csv')
```

We can **assign** the result of this function to a **variable** named art which will allow us to reuse it.

Introduction

Function can be part of a library (like *pd.read_csv()*) or they can be built in to objects. In this case, we call them **methods**.

Introduction

Function can be part of a library (like *pd.read_csv()*) or they can be built in to objects. In this case, we call them **methods**.

```
art.head()
```

Introduction

Function can be part of a library (like *pd.read_csv()*) or they can be built in to objects. In this case, we call them **methods**.

```
art.head()
```

The name of the method.

This function returns the first five rows of a DataFrame.

Introduction

Function can be part of a library (like *pd.read_csv()*) or they can be built in to objects. In this case, we call them **methods**.

```
art.head()
```

It is a built-in function of *pandas* DataFrames. This is the name of the DataFrame we are working with.

Introduction

Function can be part of a library (like *pd.read_csv()*) or they can be built in to objects. In this case, we call them **methods**.

```
art.head()
```

It does not require any arguments, but still needs parentheses since it is a function.

Introduction

Dataframes also have additional characteristics called **attributes** or **properties** which are also accessed with a period but do not require parentheses.

Introduction

Dataframes also have additional characteristics called **attributes** or **properties** which are also accessed with a period but do not require parentheses.

For example, the *shape* attribute contains the dimensions (number of rows and number of columns) of a DataFrame.

```
    art.shape
```

Now, let’s look at some commonly-used *pandas* tools which we’ll practice in our first notebook.

NASHVILLE SOFTWARE SCHOOL

pandas – https://pandas.pydata.org/pandas-docs/stable/reference/index.html

Importing Data

- **pd.read_csv()** – read a comma delimited file; good practice is to look at the raw file in a text editor (like Visual Studio Code, not Excel); additional arguments may be needed to handle extra rows at the top and extra data (footnotes) at the bottom.

pandas – https://pandas.pydata.org/pandas-docs/stable/reference/index.html

Importing Data

- **pd.read_csv()** – read a comma delimited file; good practice is to look at the raw file in a text editor (like Visual Studio Code, not Excel); additional arguments may be needed to handle extra rows at the top and extra data (footnotes) at the bottom.

Inspecting

- **df.head()** – looks at the top of the DataFrame; 5 rows by default
- **df.tail()** - looks at the bottom of the DataFrame; 5 rows by default
- **df.shape** – returns the dimensions of a DataFrame: (number of rows, number of columns)
- **df.info()** – method to get information about the DataFrame

pandas – https://pandas.pydata.org/pandas-docs/stable/reference/index.html

Modifying

- **df.columns** – column labels attribute
- **df.rename()** – rename values (can pass in a dictionary with existing columns as the key and new ones as the values)
- **df.drop()** – drop the specified labels (either rows or columns) from the DataFrame

pandas – https://pandas.pydata.org/pandas-docs/stable/reference/index.html

Modifying

- **df.columns** – column labels attribute
- **df.rename()** – rename values (can pass in a dictionary with existing columns as the key and new ones as the values)
- **df.drop()** – drop the specified labels (either rows or columns) from the DataFrame

Summarizing

- **series.unique()** – returns the unique values in a column
- **series.nunique()** - returns the *number* of unique elements in a column
- **series.value_counts()** - returns the unique elements in a column and the number of appearances of each

pandas – https://pandas.pydata.org/pandas-docs/stable/reference/index.html

Modifying

- **df.columns** – column labels attribute
- **df.rename()** – rename values (can pass in a dictionary with existing columns as the key and new ones as the values)
- **df.drop()** – drop the specified labels (either rows or columns) from the DataFrame

Summarizing

- **series.unique()** – returns the unique values in a column
- **series.nunique()** - returns the *number* of unique elements in a column
- **series.value_counts()** - returns the unique elements in a column and the number of appearances of each

Slicing/Filtering

- **df.loc[]** – pass in row name and column name to access data at that location
- **df[[ ]]** - creates a slice (subset) of the DataFrame including just the columns passed in

Notice that these last **accessors** use square brackets instead of parentheses.

Let’s open our first shared notebook so we can see these in action:

notebook_01_public_art_part_1.ipynb