Class 6:
Introduction to Pandas

Python for Data Analysts: Method & Tools

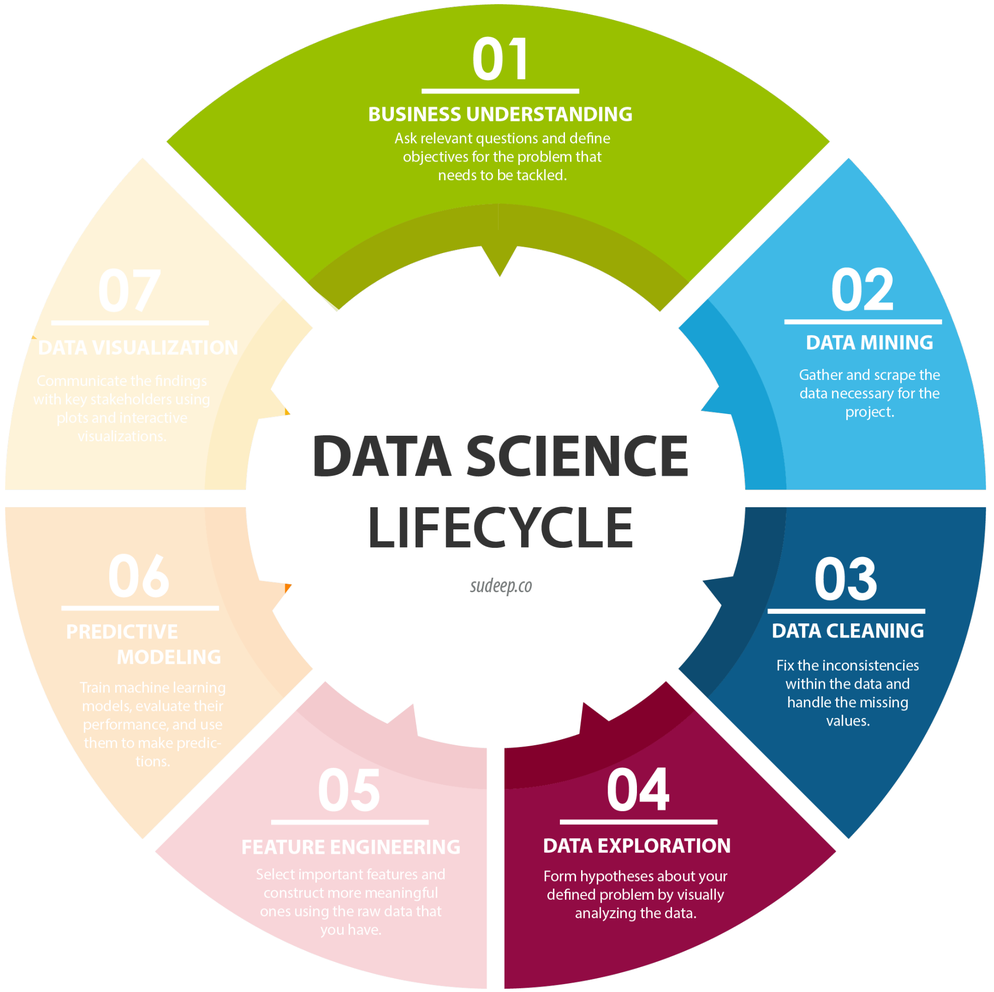
Felipe Dominguez - Adjunct Professor - Jan 23, 2023

Today's Class

- While Exercises
- What is Pandas?
- Advantages of Pandas
- Pandas DataFrames
  - Import / Export
  - Explore your data
  - Access Data Frames

Data Science - Lifecycle

- 60-80% of a Data Scientist's time is spent in collecting, cleaning, exploring, and preparing the data for analysis



- Data Science Salary

- Data Analyst Salary

Source: bultinboston

What is Pandas?

Pandas

- PANel DAta - PANDA
- World class open source project (2008)
- Home to the DataFrame
- Documentation —> https://pandas.pydata.org/docs/

Pandas

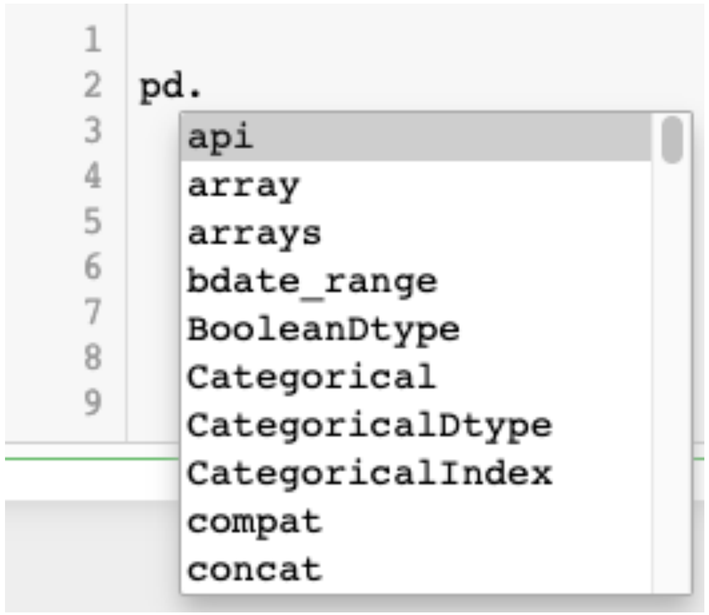
- Built on top of Numpy
  - Use of linear algebra
  - Numpy structures are computationally faster than lists
  - Several DS packages rely on Numpy
  - Data types restrictions

Pandas

```
# Code 2.1.
# import pandas library
import pandas as pd

```

- Easier to reference pandas functions and methods



- You can access all pandas functions
  - Check **documentation** or ask for **help**

Pandas

```
1 help(pd.isna)
```

```
Help on function isna in module pandas.core.dtypes.missing:

isna(obj)
    Detect missing values for an array-like object.

    This function takes a scalar or array-like object and indicates
    whether values are missing (``NaN`` in numeric arrays, ``None`` or ``NaN``
    in object arrays, ``NaT`` in datetimelike).
```

```
For Series and DataFrame, the same type is returned, containing booleans.

>>> df = pd.DataFrame([['ant', 'bee', 'cat'], ['dog', None, 'fly']])
>>> df
     0     1    2
0  ant   bee  cat
1  dog  None  fly
>>> pd.isna(df)
       0      1      2
0  False  False  False
1  False   True  False

>>> pd.isna(df[1])
0    False
1     True
Name: 1, dtype: bool
```

This function evaluates each element of a data frame and returns True or False if the element is empty or not

Pandas DataFrames

Pandas - Data Frame

- Spreadsheet-like structure (Tabular)
  - 2-Dimensional labeled data structure (columns & rows)
  - Lists, dicts, series, 2-numpy array, external files (**excel, csv, SQL**), other DataFrame
  - The prefix ‘df’ is generally used when declaring DataFrames variables

```
1  df = pd.DataFrame([['ant', 'bee', 'cat'],
2                     ['dog', None, 'fly']])
3  df
```

	0	1	2
0	ant	bee	cat
1	dog	None	fly

- To create a new Data Frame use pd.DataFrame() and store it in a new variable (**df_**)

Import / Export DataFrames

Pandas - Data Frame Import

- Import a csv file

```
1 help(pd.read_csv)
```

```
Help on function read_csv in module pandas.io.parsers.readers:

read_csv(filepath_or_buffer: 'FilePath | ReadCsvBuffer[bytes] | ReadCsvBuffer[str]', sep=<no_default>, delimiter=Non
e, header='infer', names=<no_default>, index_col=None, usecols=None, squeeze=None, prefix=<no_default>, mangle_dupe_c
ols=True, dtype: 'DtypeArg | None' = None, engine: 'CSVEngine | None' = None, converters=None, true_values=None, fals
e_values=None, skipinitialspace=False, skiprows=None, skipfooter=0, nrows=None, na_values=None, keep_default_na=True,
na_filter=True, verbose=False, skip_blank_lines=True, parse_dates=None, infer_datetime_format=False, keep_date_col=Fa
lse, date_parser=None, dayfirst=False, cache_dates=True, iterator=False, chunksize=None, compression: 'CompressionOpt
ions' = 'infer', thousands=None, decimal: 'str' = '.', lineterminator=None, quotechar='"', quoting=0, doublequote=Tru
e, escapechar=None, comment=None, encoding=None, encoding_errors: 'str | None' = 'strict', dialect=None, error_bad_li
nes=None, warn_bad_lines=None, on_bad_lines=None, delim_whitespace=False, low_memory=True, memory_map=False, float_pr
ecision=None, storage_options: 'StorageOptions' = None)
    Read a comma-separated values (csv) file into DataFrame.
```

Pandas - Data Frame Import

- Import a csv file - Write the correct path

```
# Code 2.1.4. Import csv file
# storing path file
path_file = "/Users/fadominguez/Documents/GitHub/intro-to-python-2023/classes/6. Class 6/__resources/world_cup_ma

# transform csv file into DataFrame
df_wcup   = pd.read_csv(filepath_or_buffer = path_file, # path to file or URL
                        sep = ",",                      # separator character
                        header = 0)                     # first row headers

# output DataFrame
df_wcup
```

	ID	Year	Date	Stage	Home Team	Home Goals	Away Goals	Away Team	Win Conditions	Host Team	Winner
0	1	1930	7/13/30	Group stage	France	4	1	Mexico	NaN	False	France
1	2	1930	7/13/30	Group stage	United States	3	0	Belgium	NaN	False	United States
2	3	1930	7/14/30	Group stage	Yugoslavia	2	1	Brazil	NaN	False	Yugoslavia
3	4	1930	7/14/30	Group stage	Romania	3	1	Peru	NaN	False	Romania
4	5	1930	7/15/30	Group stage	Argentina	1	0	France	NaN	False	Argentina

Pandas - Data Frame Import

- Import a csv file - Shorten the path

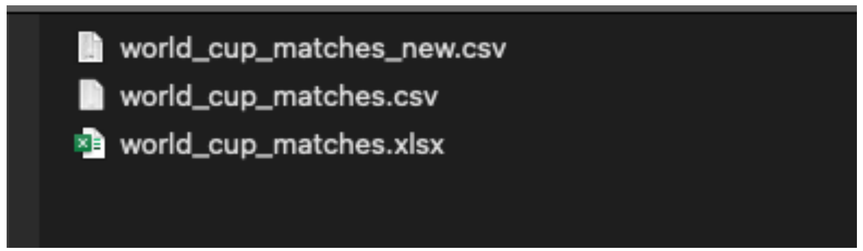
```
2  # Code 2.1.4. Import csv file
3  # storing path file
4  path_file = "./__resources/world_cup_matches.csv"
5
6  # transform csv file into DataFrame
7  df_wcup   = pd.read_csv(filepath_or_buffer = path_file, # path to file or URL
8                          sep = ",",                      # separator character
9                          header = 0)                     # first row headers
10
11 # output DataFrame
12 df_wcup
```

	ID	Year	Date	Stage	Home Team	Home Goals	Away Goals	Away Team	Win Conditions	Host Team	Winner
0	1	1930	7/13/30	Group stage	France	4	1	Mexico	NaN	False	France
1	2	1930	7/13/30	Group stage	United States	3	0	Belgium	NaN	False	United States
2	3	1930	7/14/30	Group stage	Yugoslavia	2	1	Brazil	NaN	False	Yugoslavia
3	4	1930	7/14/30	Group stage	Romania	3	1	Peru	NaN	False	Romania
4	5	1930	7/15/30	Group stage	Argentina	1	0	France	NaN	False	Argentina

Pandas - Data Frame Export

- Write the correct path!

```
2  df_wcup.to_csv("./__resources/world_cup_matches_new.csv")
```



Exploring DataFrames

DataFrame - Exploring

- df = pd.DataFrame()

DataFrame Method	Description
df	Output several rows and columns of a dataset
df.info()	Access column names, number of non-missing values, and variable types
df.head()	Access the first ‘n’ rows of a DataFrame
df.tail()	Access the las n rows of a DataFrame
df.shape()	Access the dimension of a DataFrame
df.columns	Output the column names of a DataFrame
df.count()	Counts the number of non-missing values in a dataset

DataFrame - Exploring .head()

- First check — csv document was imported correctly
- Brief glance at the quality of the data and its features
- First understanding of your dataset

```
1  # Code 2.2.1.
2
3  # Change n to whatever number you like.
4  df_wcup.head(n = 5)
```

	ID	Year	Date	Stage	Home Team	Home Goals	Away Goals	Away Team	Win Conditions	Host Team	Winner
0	1	1930	1930-07-13	Group stage	France	4	1	Mexico	NaN	False	France
1	2	1930	1930-07-13	Group stage	United States	3	0	Belgium	NaN	False	United States
2	3	1930	1930-07-14	Group stage	Yugoslavia	2	1	Brazil	NaN	False	Yugoslavia
3	4	1930	1930-07-14	Group stage	Romania	3	1	Peru	NaN	False	Romania
4	5	1930	1930-07-15	Group stage	Argentina	1	0	France	NaN	False	Argentina

DataFrame - Exploring .info()

- Information about a DataFrame
  - Column name, data type, non-null values, memory usage

- Columns can only host **one type of data types**
  - If Pandas find more than one dtype in a column, it will transform the entire column to a string ('object')

```
1  # Code 2.2.2.
2  df_wcup.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 900 entries, 0 to 899
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ID              900 non-null    int64
 1   Year            900 non-null    int64
 2   Date            900 non-null    datetime64[ns]
 3   Stage           900 non-null    object
 4   Home Team       900 non-null    object
 5   Home Goals      900 non-null    int64
 6   Away Goals      900 non-null    int64
 7   Away Team       900 non-null    object
 8   Win Conditions  62 non-null     object
 9   Host Team       900 non-null    bool
 10  Winner          900 non-null    object
dtypes: bool(1), datetime64[ns](1), int64(4), object(5)
memory usage: 71.3+ KB
```

Exploring DataFrames - Practice

Accessing DataFrames

Data Frame Access - Slice

- Slice DataFrames by rows - Returns a DataFrame
  - DataFrame[start_row : end_row]

```
1  # Code 3.1.
2  # Slicing the elements from 5 to 9 index.
3  sliced_df = df_wcup[5:10]
4  sliced_df
```

	ID	Year	Date	Stage	Home Team	Home Goals	Away Goals	Away Team	Win Conditions	Host Team	Winner
5	6	1930	1930-07-16	Group stage	Chile	3	0	Mexico	NaN	False	Chile
6	7	1930	1930-07-17	Group stage	Yugoslavia	4	0	Bolivia	NaN	False	Yugoslavia
7	8	1930	1930-07-17	Group stage	United States	3	0	Paraguay	NaN	False	United States
8	9	1930	1930-07-18	Group stage	Uruguay	1	0	Peru	NaN	True	Uruguay
9	10	1930	1930-07-19	Group stage	Chile	1	0	France	NaN	False	Chile

Data Frame Access - Subset columns

- Slice DataFrames by rows
  - pd.Series:
    - DataFrame[“column_name”]
    - DataFrame.column_name
  - pd.DataFrame()
    - DataFrame[[“column_name”]]

```
# Code 2.3.1.2.
winner_serie = df_wcup['Winner']
print(type(winner_serie))
print(winner_serie)
```

```
<class 'pandas.core.series.Series'>
0             France
1      United States
2         Yugoslavia
3            Romania
4          Argentina
           ...
895          Croatia
896           France
897          Croatia
898          Belgium
899           France
Name: Winner, Length: 900, dtype: object
```

```
# Code 2.3.1.3.
winner_df = df_wcup[['Winner']]
print(type(winner_df))
winner_df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
```

	Winner
0	France
1	United States
2	Yugoslavia
3	Romania
4	Argentina

Data Frame Access - Subset columns

- pd.DataFrame()
  - DataFrame[[“c_name1” , “c_name_2”, …, “c_name_n”]]

```
1  # Code 2.3.1.4.
2  winner_df = df_wcup[['Winner', 'Date', 'Home Team']]
3  print(type(winner_df))
4  winner_df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
```

	Winner	Date	Home Team
0	France	1930-07-13	France
1	United States	1930-07-13	United States
2	Yugoslavia	1930-07-14	Yugoslavia
3	Romania	1930-07-14	Romania
4	Argentina	1930-07-15	Argentina

Data Frame Access - Subset

- Can I combine slice and subset?
  - Yes, but....

```
1 # Code 3.2.
2 # Not recommended method to subset and slice a DataFrame
3 df_wcup[['Date', 'Winner']][:2]
```

	Date	Winner
0	1930-07-13	France
1	1930-07-13	United States

Note

The Python and NumPy indexing operators `[]` and attribute operator `.` provide quick and easy access to pandas data structures across a wide range of use cases. This makes interactive work intuitive, as there's little new to learn if you already know how to deal with Python dictionaries and NumPy arrays. However, since the type of the data to be accessed isn't known in advance, directly using standard operators has some optimization limits. For production code, we recommended that you take advantage of the optimized pandas data access methods exposed in this chapter.

Data Frame Access - .loc()

- Subset and slice based on **row index** and **column names**
- DataFrames are mutable, columns index might change. Using the column name you ensure consistency.
- You provide yourself and others context
- Does not change the original DataFrame, but returns a new DataFrame.

DataFrame.loc[s_row : e_row , [“c_name1”, ..., “c_name_n”]]

DataFrame.loc[s_row : e_row , [“c_name1” : “c_name_n”]]

```
# Code 3.2.1.1.
# Use of .loc()
# Subset 10 rows and the column Winner, Date, and Home Team from df_wcup
df_loc = df_wcup.loc[ 20:30 , ["Winner", "Date", "Home Team"]]
df_loc
```

	Winner	Date	Home Team
20	Switzerland	1934-05-27	Switzerland
21	Sweden	1934-05-27	Sweden
22	Germany	1934-05-27	Germany
23	Spain	1934-05-27	Spain
24	Italy	1934-05-27	Italy
25	Czechoslovakia	1934-05-27	Czechoslovakia
26	Czechoslovakia	1934-05-31	Czechoslovakia
27	Germany	1934-05-31	Germany
28	Spain	1934-05-31	Italy
29	Austria	1934-05-31	Austria
30	Italy	1934-06-01	Italy

Data Frame Access - .iloc()

- Subset and slice based on **row index** and **column index**

- Column names not necessarily tell you what they are, you might no have labels, or labels might be too large or complicated.

- Using numbers instead of names is more efficient and easier to loop over.

DataFrame.loc[s_row : e_row , [c_number_1, ..., c_number_n]]

DataFrame.loc[s_row : e_row , [c_number_1 : c_number_n]]

```
1  # Code 3.2.2.2.
2  # Subsetting with iloc() method
3  # Subset 10 rows and the column Winner, Date, and Home Team from df_wcup
4  df_iloc = df_wcup.iloc[ 20:30 , [10, 2, 4]]
5  df_iloc
```

	Winner	Date	Home Team
20	Switzerland	1934-05-27	Switzerland
21	Sweden	1934-05-27	Sweden
22	Germany	1934-05-27	Germany
23	Spain	1934-05-27	Spain
24	Italy	1934-05-27	Italy
25	Czechoslovakia	1934-05-27	Czechoslovakia
26	Czechoslovakia	1934-05-31	Czechoslovakia
27	Germany	1934-05-31	Germany
28	Spain	1934-05-31	Italy
29	Austria	1934-05-31	Austria

Data Frame Access - Conditions

- Similar to np arrays!
- Pandas evaluate each element and returns a True/False depending on the condition
- We can set the condition inside a DataFrame

```
   1  df_wcup['Home Goals'] > 2
```

```
0       True
1       True
2      False
3       True
4      False
       ...
895    False
896    False
897    False
898    False
899     True
Name: Home Goals, Length: 900, dtype: bool
```

Data Frame Access - Conditions

- We can set the condition inside a DataFrame

- Return a new DataFrame, must save it into a new DataFrame

```
# Code 3.3.1.

# Condition to slice the world cup df
df_brazil_w = df_wcup.loc[:, :][df_wcup['Winner'] == "Brazil"]

df_brazil_w.head()
```

	ID	Year	Date	Stage	Home Team	Home Goals	Away Goals	Away Team	Win Conditions	Host Team	Winner
11	12	1930	1930-07-20	Group stage	Brazil	4	0	Bolivia	NaN	False	Brazil
40	41	1938	1938-06-05	Round of 16	Brazil	6	5	Poland	Extra time	False	Brazil
48	49	1938	1938-06-14	Quarter-finals	Brazil	2	1	Czechoslovakia	NaN	False	Brazil
51	52	1938	1938-06-19	Third place	Brazil	4	2	Sweden	NaN	False	Brazil
53	54	1950	1950-06-24	First round	Brazil	4	0	Mexico	NaN	True	Brazil

Data Frame Access - Conditions

- We can set the condition inside a DataFrame
- Return a new DataFrame, must save it into a new DataFrame
- You can slice based on columns and rows at the same time

```
1  # Code 3.3.1.
2
3  # Condition to slice the world cup df
4  df_brazil_w = df_wcup.loc[:, :][df_wcup['Winner'] == "Brazil"]
5
6  df_brazil_w.head()
```

	ID	Year	Date	Stage	Home Team	Home Goals	Away Goals	Away Team	Win Conditions	Host Team	Winner
11	12	1930	1930-07-20	Group stage	Brazil	4	0	Bolivia	NaN	False	Brazil
40	41	1938	1938-06-05	Round of 16	Brazil	6	5	Poland	Extra time	False	Brazil
48	49	1938	1938-06-14	Quarter-finals	Brazil	2	1	Czechoslovakia	NaN	False	Brazil
51	52	1938	1938-06-19	Third place	Brazil	4	2	Sweden	NaN	False	Brazil
53	54	1950	1950-06-24	First round	Brazil	4	0	Mexico	NaN	True	Brazil

Data Frame Access - Conditions

- We can set the condition inside a DataFrame

- Return a new DataFrame, must save it into a new DataFrame

```
# Code 3.3.3.

df_brazil_w = df_wcup[(df_wcup['Winner'] == "Brazil") & (df_wcup['Stage'] == "Final")]
df_brazil_w
```

	ID	Year	Date	Stage	Home Team	Home Goals	Away Goals	Away Team	Win Conditions	Host Team	Winner
135	136	1958	1958-06-29	Final	Sweden	2	5	Brazil	NaN	True	Brazil
167	168	1962	1962-06-17	Final	Brazil	3	1	Czechoslovakia	NaN	False	Brazil
231	232	1970	1970-06-21	Final	Brazil	4	1	Italy	NaN	False	Brazil
515	516	1994	1994-07-17	Final	Brazil	0	0	Italy	Brazil win on penalties (3 - 2)	False	Brazil
643	644	2002	2002-06-30	Final	Germany	0	2	Brazil	NaN	False	Brazil

Data Frame Access - Add new values

- New value **must have the same dimensions!**

- Best practice: Keep the **original DataFrame unchanged**. Copy and create a new DataFrame

```
1  df = pd.DataFrame([[1, "a"], [2, "b"], [3,"c"]],
2                    columns = ["numbers", "letters"])
3
4  df_copy = df.copy()
5
6  df_copy["special_characters"] = [".", "/", "!"]
7
8  df_copy
```

	numbers	letters	special_characters
0	1	a	.
1	2	b	/
2	3	c	!

Data Frame Access - Add new values

```
1  df = pd.DataFrame([[1, "a"], [2, "b"], [3,"c"]],
2                    columns = ["numbers", "letters"])
3
4  df_copy = df
5
6  df_copy["special_characters"] = [".", "/", "!"]
7
8  df_copy
```

	numbers	letters	special_characters
0	1	a	.
1	2	b	/
2	3	c	!

```
1  df
```

	numbers	letters	special_characters
0	1	a	.
1	2	b	/
2	3	c	!

Data Frame Access - Add new values

- Use mathematical operations between features
  - **Feature Engineer**

```
# Code 3.4.2.
# Copy the original df
df_b_copy_2 = df_brazil_w.copy()

# Creating total goals column as the sum between Home Goals & Away Goals
df_b_copy_2['Total Goals'] = df_b_copy_2['Home Goals'] + df_b_copy_2['Away Goals']
df_b_copy_2
```

	ID	Year	Date	Stage	Home Team	Home Goals	Away Goals	Away Team	Win Conditions	Host Team	Winner	Total Goals
135	136	1958	1958-06-29	Final	Sweden	2	5	Brazil	NaN	True	Brazil	7
167	168	1962	1962-06-17	Final	Brazil	3	1	Czechoslovakia	NaN	False	Brazil	4
231	232	1970	1970-06-21	Final	Brazil	4	1	Italy	NaN	False	Brazil	5
515	516	1994	1994-07-17	Final	Brazil	0	0	Italy	Brazil win on penalties (3 - 2)	False	Brazil	0
643	644	2002	2002-06-30	Final	Germany	0	2	Brazil	NaN	False	Brazil	2

Data Frame Access - Update Values

- Update values based on column name
- The new value must have the same dimensions

```
1 df_copy["special_characters"] = ["=", "+", "*"]
2
3 df_copy
```

	numbers	letters	special_characters
0	1	2	=
1	2	b	+
2	3	c	*

Data Frame Access - Update Values

- Update values based on conditions

```
# Code 3.4.1.
# Copy the original df
df_b_copy = df_brazil_w.copy()
# replacing the value only on those observations where 'Win Conditions' is null (NaN) (using .loc)
df_b_copy.loc[df_b_copy['Win Conditions'].isnull(), ["Win Conditions"]] = 'Full Time'
df_b_copy
```

	ID	Year	Date	Stage	Home Team	Home Goals	Away Goals	Away Team	Win Conditions	Host Team	Winner
135	136	1958	1958-06-29	Final	Sweden	2	5	Brazil	Full Time	True	Brazil
167	168	1962	1962-06-17	Final	Brazil	3	1	Czechoslovakia	Full Time	False	Brazil
231	232	1970	1970-06-21	Final	Brazil	4	1	Italy	Full Time	False	Brazil
515	516	1994	1994-07-17	Final	Brazil	0	0	Italy	Brazil win on penalties (3 - 2)	False	Brazil
643	644	2002	2002-06-30	Final	Germany	0	2	Brazil	Full Time	False	Brazil