Cheatsheets / **Getting Started with Python for Data Science** codecademy

Cleaning and Transforming Columns

Pandas DataFrame .info() Method

The pandas DataFrame method `.info()` displays a table of information for each column.

```
parks.info()
```

```
parks.info()
```

#	Column	Non-Null Count	Dtype
0	index	72 non-null	int64
1	Park	72 non-null	object
2	Location	72 non-null	object
3	AnnualPassPrice	72 non-null	int64

- `#` indicates the column index number
- `Column` refers to the column name
- `Non-Null Count` is the number of non-missing values in the column
- `DType` is the column's data type

Dropping Columns in a Pandas DataFrame

The pandas `.drop()` method is used to remove irrelevant columns from a DataFrame. This method has two keywords:

- `labels` takes a list of column names to drop
- `axis=1` tells pandas we want to drop columns (not rows)

	index	Park	Location
0	1	Great Smoky Mountains	Gatlinburg, TN
1	2	Zion	Springdale, UT
2	3	Yellowstone	Jackson, WY

The code snippet drops the `index` column to produce

	Park	Location
0	Great Smoky Mountains	Gatlinburg, TN
1	Zion	Springdale, UT
2	Yellowstone	Jackson, WY

```
# Drop the index column
drop_columns = ['index']
nationalparks.drop(labels=drop_columns,
axis=1)
```

codecademy

Renaming Columns in a Pandas DataFrame

	index	Park	Year2019
0	1	Great Smoky Mountains	12547743
1	2	Zion	4488268
2	3	Yellowstone	4020288

```
# Rename the Park column to National Park
column_mapper = {'Park': 'National Park'}
parks.rename(mapper=column_mapper, axis=1)
```

The pandas `.rename()` method renames columns in a DataFrame. There are two particularly important keywords for `.rename()`:

- `mapper` takes a dictionary mapping the old column names (as keys) to the new column names (as values)
- `axis=1` tells pandas to rename the columns axis

Arithmetic Operators in Python

Python has built-in **arithmetic operators** for performing calculations, including

- Addition (+),
- Subtraction (-),
- Multiplication (*)
- Division (/)

Like mathematics, Python uses parentheses to control the order of operations in a calculation.

```
100 + 10
# Output: 110

100 - 10
# Output: 90

100 * 10
# Output: 1000

100 / 10
# Output: 10

(100 + 10) / (10)
# Output: 11.0
```

codecademy

Rounding Numbers in Python

The `round()` function in Python rounds a number to a certain number of decimals using the following syntax:

```
round(numeric_variable, numbe
```

```
pi = 3.14159
# Round pi to 4 decimals
round(pi, 4)
# Output: 3.1416
```

Pandas Column Calculations

In pandas, arithmetic operators like `+`, `-`, `/`, and `*` can be applied to all the rows of a column at once.

Here's a sample DataFrame `parks`.

	Park	Area_SqMi
0	Great Smoky Mountains	816.3
1	Zion	229.1
2	Yellowstone	3468.4

The code snippet produces the following DataFrame:

	Park	Area_SqMi	Area_Sc
0	Great Smoky Mountains	816.3	2114.2
1	Zion	229.1	593.36
2	Yellowstone	3468.4	8983.1

```
# convert miles to km using column
multiplication
parks['Area_SqKm'] = parks['Area_SqMi']
* 2.59
```

codecademy

Splitting a Column in a Pandas DataFrame

The pandas method `.str.split(pat='x', expand=True)` will split the information in a text column into multiple columns using `'x'` as a delimiter. Common delimiters include commas (`,`), colons (`:`), and dashes (`-`).

	Location
0	Gatlinburg, TN
1	Springdale, UT
2	Jackson, WY

The keyword argument `expand=True` creates a DataFrame containing the split information that can be accessed through pandas indexing.

	0	**1**
0	Gatlinburg	TN
1	Springdale	UT
2	Jackson	WY

```
# Split the Location column on the comma
delimiter
parks['Location'].str.split(pat=',',
expand=True)
```

codecademy

Combining Columns in a Pandas DataFrame

The Series method .str.cat() combines text from two columns into a single string:

```
df['Combined'] = df['Column1'
     df['Column2'],
     sep=',')
```

- .cat() places the text in Column2 after the text in Column1
- sep=',' places a comma ',' after the text from Column1 and before the text from Column2

	City	State
0	Gatlinburg	TN
1	Springdale	UT
2	Jackson	WY

The code snippet produces the following Location column:

	Location
0	Gatlinburg, TN
1	Springdale, UT
2	Jackson, WY

```
# Combine the `City` and `State` columns
into a single column `Location`
parks['Location'] =
parks['City'].str.cat(
  parks['State'],
  sep=', ')
```

codecademy

Transforming Text Columns in Pandas with .lower(), .upper(), and .title()

Pandas can alter text case using

- .str.lower() - converts all text to lowercase
- .str.upper() - converts all text to uppercase
- .str.title() - converts all text to titles

	Park
0	Great Smoky Mountains
1	Zion
2	Yellowstone

Convert Park to lowercase and uppercase

	Park	.str.lower()	.
0	Great Smoky Mountains	great smoky mountains	G M
1	Zion	zion	Z
2	Yellowstone	yellowstone	Y

```
# Convert to lowercase
parks['Park'].str.lower()

# Convert to uppercase
parks['Park'].str.upper()
```

codecademy

Find-and-Replace in Pandas

	Before	After
0	Great.Smoky.Mountains	Great Smoky Mountai
3	Grand.Canyon	Grand Canyon
4	Rocky.Mountain	Rocky Mountai

The pandas method .str.replace() performs a find-and-replace on each row of a series. Every section of text that matches the string passed to pat will be replaced by the string passed to repl .

```
df['Column'] = df['Column'].st
                        pat='old_pa
                        repl='new_p
                        regex=False
```

```
# Replace periods '.' with spaces
parks['Park'] =
parks['Park'].str.replace(
        pat='.',
  repl=' ',
        regex=False)
```

Missing Data in Pandas

Missing or null values in a pandas DataFrame are often represented with a NaN value.

	Park	Location	Annua
0	Great Smoky Mountains	Gatlinburg, TN	40.0
1	Zion	NaN	70.0
2	Yellowstone	Jackson, WY	NaN

- Zion has a missing Location value
- Yellowstone has a missing AnnualPassPrice value

codecademy

Changing Data Types in Pandas

The pandas method `.astype()` converts the type of a column from one type to another. The new type is specified within the parentheses:

- `float64` for decimals
- `int64` for integers
- `object` for text/objects
- `category` for categorical data

	Park	Area
0	Great Smoky Mountains	‘816.3’
1	Zion	‘229.1’
2	Yellowstone	‘3468.4’

```
# Convert `Area` from object to float
parks['Area'] =
parks['Area'].astype('float64')
```

Print Share