Working with Data in Python Cheat Sheet

Reading and writing files

Package/Method | **Description** | **Syntax and Code Example**

File opening modes — Different modes to open files for specific operations.

Syntax: r (reading) w (writing) a (appending) + (updating: read/write) b (binary, otherwise text)

```
Examples: with open("data.txt", "r") as file: content = file.read() print(content) with open("output.txt", "w") as file: file.write("Hello, world!") with open("log.txt", "a") as file: file.write("Log entry: Something happened.") with open("data.txt", "r+") as file: content = file.read() file.write("Updated content: " + content)</td>
```

Copied!

File reading methods — Different methods to read file content in various ways.

Syntax:

```
file.readlines() # reads all lines as a list
readline() # reads the next line as a string
file.read() # reads the entire file content as a string
```

Copied!

Example:

```
with open("data.txt", "r") as file:
    lines = file.readlines()
    next_line = file.readline()
    content = file.read()
```

Copied!

File writing methods — Different write methods to write content to a file.

Syntax:

```
file.write(content) # writes a string to the file
file.writelines(lines) # writes a list of strings to the file
```

Copied!

Example:

```
lines = ["Hello\n", "World\n"]
with open("output.txt", "w") as file:
    file.writelines(lines)
```

Copied!

Iterating over lines — Iterates through each line in the file using a `loop`.

Syntax:

```
for line in file: # Code to process each line
```

Copied!

Example:

```
with open("data.txt", "r") as file:
for line in file: print(line)
```

Copied!

Open() and close() — Opens a file, performs operations, and explicitly closes the file using the close() method.

Syntax:

```
file = open(filename, mode) # Code that uses the file
file.close()
```

Copied!

Example:

```
file = open("data.txt", "r")
content = file.read()
file.close()
```

Copied!

with open() — Opens a file using a with block, ensuring automatic file closure after usage.

Syntax:

```
with open(filename, mode) as file: # Code that uses the file
```

Copied!

Example:

```
with open("data.txt", "r") as file:
content = file.read()
```

Copied!

Pandas

Package/Method	Description	Syntax and Code Example
.read_csv()	Reads data from a `.CSV` file and creates a DataFrame.	Syntax: dataframe_name = pd.read_csv("filename.csv") Example: df = pd.read_csv("data.csv")

.read_excel()	Reads data from an Excel file and creates a DataFrame.	Syntax:
		`dataframe_name = pd.read_excel("filename.xlsx")`
		Copied!
		Example:
		`df = pd.read_excel("data.xlsx")`
		Copied!

.to_csv()	Writes DataFrame to a CSV file.	Syntax:
		`dataframe_name.to_csv("output.csv", index=False)`
		Copied!
		Example:
		`df.to_csv("output.csv", index=False)`
		Copied!

Access Columns	Accesses a specific column using [] in the DataFrame.	Syntax:
		`dataframe_name["column_name"] # Accesses single column`
		`dataframe_name[["column1", "column2"]] # Accesses multiple columns`
		Copied!
		Example:
		`df["age"]`
		`df[["name", "age"]]`
		Copied!

describe()	Generates statistics summary of numeric columns in the DataFrame.	Syntax:
		`dataframe_name.describe()`
		Copied!
		Example:
		`df.describe()`
		Copied!

drop()	Removes specified rows or columns from the DataFrame. axis=1 indicates columns. axis=0 indicates rows.	Syntax:
		`dataframe_name.drop(["column1", "column2"], axis=1, inplace=True)`
		`dataframe_name.drop(index=[row1, row2], axis=0, inplace=True)`
		Copied!
		Example:
		`df.drop(["age", "salary"], axis=1, inplace=True) # Will drop columns`
		`df.drop(index=[5, 10], axis=0, inplace=True) # Will drop rows`
		Copied!

dropna()	Removes rows with missing NaN values from the DataFrame. axis=0 indicates rows.	Syntax:
		`dataframe_name.dropna(axis=0, inplace=True)`
		Copied!
		Example:
		`df.dropna(axis=0, inplace=True)`
		Copied!

duplicated()	Duplicate or repetitive values or records within a data set.	Syntax:
		`dataframe_name.duplicated()`
		Copied!
		Example:
		`duplicate_rows = df[df.duplicated()]`
		Copied!

Filter Rows	Creates a new DataFrame with rows that meet specified conditions.	Syntax: `filtered_df = dataframe_name[(Conditional_statements)]` Copied!

		Example: `filtered_df = df[(df["age"] > 30) & (df["salary"] < 50000)` Copied!

groupby()	Splits a DataFrame into groups based on specified criteria, enabling subsequent aggregation, transformation, or analysis within each group.	Syntax:
		`grouped = dataframe_name.groupby(by, axis=0, level=None, as_index=True,`
		`sort=True, group_keys=True, squeeze=False, observed=False, dropna=True)`
		Copied!
		Example:
		`grouped = df.groupby(["category", "region"]).agg({"sales": "sum"})`
		Copied!

head()	Displays the first n rows of the DataFrame.	Syntax:
		`dataframe_name.head(n)`
		Copied!
		Example:
		`df.head(5)`
		Copied!

Import pandas	Imports the Pandas library with the alias pd.	Syntax: `import pandas as pd` Copied! Example: `import pandas as pd` Copied!

info()	Provides information about the DataFrame, including data types and memory usage.	Syntax:
		`dataframe_name.info()`
		Copied!
		Example:
		`df.info()`
		Copied!

merge()	Merges two DataFrames based on multiple common columns.	Syntax:
		`merged_df = pd.merge(df1, df2, on=["column1", "column2"])`
		Copied!
		Example:
		`merged_df = pd.merge(sales, products, on=["product_id", "category_id"])`
		Copied!

print DataFrame	Displays the content of the DataFrame.	Syntax:
		`print(df) # or just type df`
		Copied!
		Example:
		`print(df)`
		`df`
		Copied!

replace()	Replaces specific values in a column with new values.	Syntax:
		`dataframe_name["column_name"].replace(old_value, new_value, inplace=True)`
		Copied!
		Example:
		`df["status"].replace("In Progress", "Active", inplace=True)`
		Copied!

tail()	Displays the last n rows of the DataFrame.	Syntax:

		`dataframe_name.tail(n)`
		Copied!
		Example:
		`df.tail(5)`
		Copied!

Numpy

Package/Method	Description	Syntax and Code Example
Importing NumPy	Imports the NumPy library.	Syntax: `import numpy as np` Copied! Example: `import numpy as np` Copied!

np.array()	Creates a one or multi-dimensional array,	Syntax:
		`array_1d = np.array([list1 values]) # 1D Array`
		`array_2d = np.array([[list1 values], [list2 values]]) # 2D Array`
		Copied!
		Example:
		`array_1d = np.array([1, 2, 3]) # 1D Array`
		`array_2d = np.array([[1, 2], [3, 4]]) # 2D Array`
		Copied!

Numpy Array Attributes	- Calculates the mean of array elements
	- Calculates the sum of array elements
	- Finds the minimum value in the array
	- Finds the maximum value in the array
	- Computes dot product of two arrays
		Example:
		`np.mean(array)`
		`np.sum(array)`
		`np.min(array`
		`np.max(array)`
		`np.dot(array_1, array_2)`
		Copied!

Skills Network