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.tx

File reading methods	Different methods to read file content in various ways.	Syntax:
		
		```python\nfile.readlines() # reads all lines as a list\nreadline() # reads the next line as a string\nfile.read() # reads the entire file content as a string\n```
		
		Example:
		
		```python\nwith open("data.txt", "r") as file:\n lines = file.readlines()\n next_line = file.readline()\n content = file.read()\n```

File writing methods	Different write methods to write content to a file.	Syntax:
		
		```python\nfile.write(content) # writes a string to the file\nfile.writelines(lines) # writes a list of strings to the file\n```
		
		Example:
		
		```python\nlines = ["Hello\n", "World\n"]\nwith open("output.txt", "w") as file:\n file.writelines(lines)\n```

Iterating over lines	Iterates through each line in the file using a `loop`.	Syntax:
		
		```python\nfor line in file: # Code to process each line\n```
		
		Example:
		
		```python\nwith open("data.txt", "r") as file:\n for line in file: print(line)\n```

		Syntax:
| Open() and close() | Opens a file, performs operations, and explicitly closes the file using the close() method. | ```
file = open(filename, mode) # Code that uses the file
file.close()
```<br><br>Example:<br><br>```
file = open("data.txt", "r")
content = file.read()
file.close()
``` |
| with open() | Opens a file using a with block, ensuring automatic file closure after usage. | Syntax:<br><br>```
with open(filename, mode) as file: # Code that uses the file
```<br><br>Example:<br><br>```
with open("data.txt", "r") as file:
    content = file.read()
``` |

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:<br><br>```
dataframe_name = pd.read_excel("filename.xlsx")
```<br><br>Example:<br><br>```
df = pd.read_excel("data.xlsx")
``` |
| .to_csv() | Writes DataFrame to a CSV file. | Syntax:<br><br>```
dataframe_name.to_csv("output.csv", index=False)
```<br><br>Example:<br><br>```
df.to_csv("output.csv", index=False)
``` |
| Access Columns | Accesses a specific column using [] in the DataFrame. | Syntax:<br><br>```
dataframe_name["column_name"] # Accesses single column
dataframe_name[["column1", "column2"]] # Accesses multiple columns
```<br><br>Example:<br><br>```
df["age"]
df[["name", "age"]]
``` |

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

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)```
		
		Example:
		
		```df.drop(["age", "salary"], axis=1, inplace=True) # Will drop columns```
		```df.drop(index=[5, 10], axis=0, inplace=True) # Will drop rows```

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

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

Filter Rows	Creates a new DataFrame with rows that meet specified conditions.	Syntax:
		
		```filtered_df = dataframe_name[(Conditional_statements)]```
		
		Example:
		
		```filtered_df = df[(df["age"] > 30) & (df["salary"] < 50000)]```

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

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

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

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

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

Package/Method	Description	Syntax and Code Example

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

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

tail()	Displays the last n rows of the DataFrame.	Syntax:
		
		```dataframe_name.tail(n)```
		
		Example:
		
		```df.tail(5)```

Numpy

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

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```
		
		Example:
		
		```array_1d = np.array([1, 2, 3]) # 1D Array```
		```array_2d = np.array([[1, 2], [3, 4]]) # 2D Array```

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:
		
		``python
		np.mean(array)
		np.sum(array)
		np.min(array)
		np.max(array)
		np.dot(array_1, array_2)
		``

Skills Network