

Working with Data in Python Cheat Sheet

Reading and writing files

Package/Method	Description
File opening modes	Different modes to open files for specific operations.

```

Syntax: r (reading) w (writing) a (appending) b (binary) t (text)
1. 1
Example: 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", "rb") as file: content = file.read() file.write("Updated content: " + content)
Copy
```

```

Syntax:
1. 1
2. 2
3. 3
Example:
1. 1
2. readlines() # reads all lines as a list
3. readline() # reads the next line as a string
4. file.read() # reads the entire file content as a string
Copy
```

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

```

Example:
1. 1
2. 2
3. 3
4. 4
5. with open("data.txt", "r") as file:
6. lines = file.readlines()
7. new_lines = file.readlines()
8. content = file.read()
Copy
```

```

Syntax:
1. 1
2. 2
Example:
1. file.write(content) # writes a string to the file
2. file.writelines(lines) # writes a list of strings to the file
Copy
```

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

```

Example:
1. 1
2. 2
3. 3
4. 4
5. lines = ["line1\n", "line2\n"]
6. with open("output.txt", "w") as file:
7. file.writelines(lines)
Copy
```

```

Syntax:
1. 1
2. for line in file: # Code to process each line
Copy
```

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

```

Example:
1. 1
2. 2
3. 3
4. with open("data.txt", "r") as file:
5. for line in file: print(line)
Copy
```

```

Syntax:
1. 1
2. 2
Example:
1. file = open(filename, mode) # Code that uses the file
2. file.close()
Copy
```

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

```

Example:
1. 1
2. 2
3. 3
4. file = open("data.txt", "r")
5. content = file.read()
6. file.close()
Copy
```

```

Syntax:
1. 1
2. with open(filename, mode) as file: # Code that uses the file
Copy
```

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

```

Example:
1. 1
2. 2
3. with open("data.txt", "r") as file:
4. content = file.read()
Copy
```

Pandas

Package/Method	Description
read_csv()	Reads data from a .CSV file and creates a DataFrame.
read_excel()	Reads data from an Excel file and creates a DataFrame.
to_csv()	Writes DataFrame to a CSV file.

```

Syntax and Code Example
Syntax: dataframe_name = pd.read_csv("filename.csv") Example: df = pd.read_csv("data.csv")
Syntax:
1. 1
2. dataframe_name = pd.read_excel("filename.xlsx")
Copy
```

```

Example:
1. 1
2. df = pd.read_csv("data.xlsx")
Syntax:
1. 1
2. dataframe_name.to_csv("output.csv", index=False)
Copy
```

```

Example:
1. 1
2. df.to_csv("output.csv", index=False)
Copy
```

```

Syntax:
1. 1
2. 2
3. dataframe_name["column_name"] # Extracts single column
4. dataframe_name[["column1", "column2"]] # Extracts multiple columns
Copy
```

Access Columns Accesses a specific column using [] in the DataFrame.

```

Example:
1. 1
2. 2
3. df["name", "age"]
Copy
```

```

Syntax:
1. 1
2. dataframe_name.describe()
Copy
```

Describe() Generates statistics summary of numeric columns in the DataFrame.

```

Example:
1. 1
2. df.describe()
Copy
```

```

Syntax:
1. 1
2. 2
3. dataframe_name.drop(["column1", "column2"], axis=1, inplace=True)
4. dataframe_name.drop(index=row, row, axis=0, inplace=True)
Copy
```

drop() Removes specified rows or columns from the DataFrame. axis=1 indicates columns. axis=0 indicates rows.

```

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

```

Syntax:
1. 1
2. dataframe_name.dropna(inplace=True)
Copy
```

dropna() Removes rows with missing NaN values from the DataFrame. axis=0 indicates rows.

```

Example:
1. 1
2. df.dropna(inplace=True)
Copy
```

```

Syntax:
1. 1
2. dataframe_name.duplicated()
Copy
```

duplicated() Duplicate or repetitive values or records within a data set.

```

Example:
1. 1
2. duplicated_rows = df[df.duplicated()]
Copy
```

```

Syntax:
1. 1
2. filtered_df = dataframe_name[(condition1_statement)]
Copy
```

Filter Rows Creates a new DataFrame with rows that meet specified conditions.

```

Example:
1. 1
2. filtered_df = df[df["age"] > 30 & (df["salary"] < 10000)
Copy
```

```

Syntax:
1. 1
2. grouped = dataframe_name.groupby(by, axis=0, as_index=False, sort=False, group_keys=True, squeeze=False, observed=False, dropna=True)
Copy
```

groupby() Splits a DataFrame into groups based on specified criteria, enabling subsequent aggregation, transformation, or analysis within each group.

```

Example:
1. 1
2. grouped = df.groupby(["category", "region"]).agg({"sales": "sum"})
Copy
```

```

Syntax:
1. 1
2. dataframe_name.head(n)
Copy
```

head() Displays the first n rows of the DataFrame.

```

Example:
1. 1
2. df.head(5)
Copy
```

```

Syntax:
1. 1
2. import pandas as pd
Copy
```

Import pandas Imports the Pandas library with the alias pd.

info() Provides information about the DataFrame, including data types and memory usage.

merge() Merges two DataFrames based on multiple common columns.

print DataFrame Displays the content of the DataFrame.

replace() Replaces specific values in a column with new values.

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

Numpy		
Package/Method	Description	Syntax and Code Example
Importing Numpy	Imports the Numpy library.	<div>Syntax:</div> <pre>1. Import numpy as np</pre> <div>Example:</div> <pre>1. 1 2. Import numpy as np</pre>
		<div>Syntax:</div> <pre>1. 1 2. 1</pre> <div>Example:</div> <pre>1. array_1d = np.array([1000, 1000]) # 1D array 2. array_2d = np.array([[1000, 1000], [1000, 1000]]) # 2D array</pre>
np.array()	Creates a one or multi-dimensional array.	<div>Example:</div> <pre>1. 1 2. 1 3. array_1d = np.array([1, 1, 1]) # 1D array 4. array_2d = np.array([[1, 1], [1, 1]]) # 2D array</pre>
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 the product of two arrays	<div>Example:</div> <pre>1. 1 2. 1 3. 1 4. 1 5. 1</pre> <div>Example:</div> <pre>1. np.mean(array) 2. np.sum(array) 3. np.min(array) 4. np.max(array) 5. np.dot(array_1, array_2)</pre>

```
1. 1
2. Import pandas as pd
3.
4. 1
5. DataFrame_name.info()
6.
7. 1
8. df.info()
9.
10. 1
11. merged_df = pd.merge(df1, df2, on=['column1', 'column2'])
12.
13. 1
14. merged_df = pd.merge(column, products, on=['product_id', 'category_id'])
15.
16. 1
17. print(dfT) # or just type df
18.
19. 1
20. 1
21. 1
22. print(dfT)
23. df
24.
25. 1
26. DataFrame_name['column_name'].replace(old_value, new_value, inplace=True)
27.
28. 1
29. df['column'].replace("in progress", "Active", inplace=True)
30.
31. 1
32. DataFrame_name.tail(n)
33.
34. 1
35. df.tail(n)
```

