12/27/24, 12:56 PM about:blank

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.				
File reading methods	Different methods to read file content in various ways.	<pre>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 Example: with open("data.txt", "r") as file: lines = file.readlines() next_line = file.readline() content = file.read()</pre>			
File writing methods	Different write methods to write content to a file.	<pre>Syntax: file.write(content) # writes a string to the file file.writelines(lines) # writes a list of strings to the file Example: lines = ["Hello\n", "World\n"] with open("output.txt", "w") as file: file.writelines(lines)</pre>			
Iterating over lines	Iterates through each line in the file using a 'loop'.	Syntax: for line in file: # Code to process each line Example: with open("data.txt", "r") as file: for line in file: print(line)			
Open() and close()	Opens a file, performs operations, and explicitly closes the file using the close() method.	<pre>Syntax: file = open(filename, mode) # Code that uses the file file.close() Example: file = open("data.txt", "r") content = file.read() file.close()</pre>			
with open()	Opens a file using a with block, ensuring automatic file closure after usage.				

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.	<pre>Syntax: dataframe_name = pd.read_excel("filename.xlsx") Example: df = pd.read_excel("data.xlsx")</pre>
.to_csv()	Writes DataFrame to a CSV file.	Syntax: dataframe_name.to_csv("output.csv", index=False)

about:blank 1/3

 	1	Example:
		<pre>df.to_csv("output.csv", index=False)</pre>
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 Example: 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.	<pre>Syntax: filtered_df = dataframe_name[(Conditional_statements)] Example: filtered_df = df[(df["age"] > 30) & (df["salary"] < 50000)</pre>
groupby()	Splits a DataFrame into groups based on specified criteria, enabling subsequent aggregation, transformation, or analysis within each group.	<pre>Syntax: grouped = dataframe_name.groupby(by, axis=0, level=None, as_index=True, sort=True, group_keys=True, squeeze=False, observed=False, dropna=True) Example: grouped = df.groupby(["category", "region"]).agg({"sales": "sum"})</pre>
head()	Displays the first n rows of the DataFrame.	Syntax: dataframe_name.head(n) Example: df.head(5)
Import pandas	Imports the Pandas library with the alias pd.	Syntax: import pandas as pd Example: import pandas as pd
info()	Provides information about the DataFrame, including data types and memory usage.	Syntax: dataframe_name.info()

	Example:	
		df.info()
merge()	Merges two DataFrames based on multiple common columns.	<pre>Syntax: merged_df = pd.merge(df1, df2, on=["column1", "column2"]) Example: merged_df = pd.merge(sales, products, on=["product_id", "category_id"])</pre>
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

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



 $\hbox{@}$ IBM Corporation. All rights reserved.

about:blank 3/3