All-in-One PyData Cheat Sheets

Python NumPy

pandas seaborn

All-in-One PyData Cheat Sheets: Python, NumPy, Pandas, Seaborn & Anaconda

Pandas Cheat Sheet

Pandas is a powerful data manipulation and analysis library for Python. It provides a flexible and efficient way to work with structured data, such as spreadsheets, databases, and CSV files. Pandas allows you to perform tasks such as data cleaning, data exploration, data manipulation, and data visualization.

If you’re new to pandas, it can be overwhelming to learn all the different methods and functions that the library provides. That’s where a cheat sheet comes in handy. A cheat sheet is a concise reference guide that provides a quick overview of the most commonly used pandas methods and functions. It can save you time and help you be more productive by providing a quick reference to commonly used methods and functions.

The pandas cheat sheet I provided is divided into different themes and presented in a table format. The themes include importing data, data exploration, data cleaning, data manipulation, and data visualization. Each table contains a list of commonly used pandas methods and functions for that theme. For example, the data exploration table contains methods for showing the first and last n rows of data, data shape, data types, summary statistics, and more.

The cheat sheet is not meant to be an exhaustive reference guide to pandas. It’s designed to provide a quick reference to commonly used methods and functions.

Cheat Sheet

Importing Data

Task	Code
Import pandas	import pandas as pd

Import data from CSV file	df = pd.read_csv('filename.csv')
Import data from Excel file	df = pd.read_excel('filename.xlsx')

Import data from SQL database	Import sqlite3
	conn = sqlite3.connect('database.db')
	df = pd.read_sql_query('SELECT * FROM tablename', conn)

Data Exploration

Task	**Code**
Show first n rows of data	df.head(n)
Show last n rows of data	df.tail(n)
Show data shape	df.shape
Show data types	df.dtypes
Show summary statistics	df.describe()
Show unique values in a column	df['column'].unique()
Show number of unique values in a column	df['column'].nunique()
Show value counts in a column	df['column'].value_counts()
Show correlation matrix	df.corr()

Data Cleaning

Task	Code
Rename columns	df.rename(columns={‘old_name’:’new_name’})
Drop columns	df.drop(columns=[‘column1’, ‘column2’])
Drop rows with missing values	df.dropna()
Fill missing values with a constant	df.fillna(value)
Fill missing values with the mean	df.fillna(df.mean())
Replace values in a column	df[‘column’].replace(old_value, new_value)
Remove duplicates	df.drop_duplicates()

Data Manipulation

Task	Code
Select columns	df[[‘column1’, ‘column2’]]
Filter rows by a condition	df[df[‘column’] > value]
Sort data by one or more columns	df.sort_values(by=[‘column1’, ‘column2’])
Group data by a column and calculate statistics	df.groupby(‘column’).agg({‘column1’: ‘mean’, ‘column2’: ‘sum’})

Merge two dataframes by a common column	pd.merge(df1, df2, on='column')
Pivot table	pd.pivot_table(df, values='value', index='row', columns='column', aggfunc='mean')
Apply a function to a column	df['column'].apply(function)
Apply a function to a row	df.apply(function, axis=1)

Data Visualization

Task	Code
Line plot	df.plot(x='column1', y='column2', kind='line')
Bar plot	df.plot(x='column1', y='column2', kind='bar')
Histogram	df['column'].hist()
Scatter plot	df.plot(x='column1', y='column2', kind='scatter')
Box plot	df.boxplot(column='column')
Heatmap	sns.heatmap(df.corr(), annot=True)
Pairplot	sns.pairplot(df)

Seaborn Cheat Sheet

Introduction

Seaborn is a popular data visualization library in Python that is used to create beautiful and informative statistical graphics. It is built on top of the Matplotlib library and provides a high-level interface for creating attractive and informative visualizations. Seaborn is widely used in data science, machine learning, and statistical analysis.

To help users get started with Seaborn, we've created cheat sheet that provides a quick reference guide to the most commonly used functions and methods in Seaborn. The cheat sheet is designed to be a handy reference guide for users who are new to Seaborn or who need a quick reminder of the syntax and parameters of various functions.

This Seaborn cheat sheet covers a wide range of topics, including data visualization, statistical plotting, color palettes, and data manipulation. It includes examples of how to create various types of plots, such as scatter plots, line plots, bar plots, and heatmaps.

Importing Seaborn

```
import seaborn as sns
```

Setting the Style

```
sns.set_style(style=None, rc=None)
```

Parameter	Description
style	Name of style to use, or None to reset to default
rc	Dictionary of parameter values to set

Loading Datasets

```
sns.load_dataset(name, cache=True, data_home=None, **kws)
```

Parameter	Description

name	Name of dataset to load
cache	Whether to cache downloaded file
data_home	Directory to cache downloaded files
kws	Additional keyword arguments to pass to pandas.read_csv()

Plotting Functions

Relational Plots

sns.relplot(x=None, y=None, hue=None, size=None, style=None, data=None, kind='scatter', **kwargs)

Parameter	Description
x	Column name for x-axis
y	Column name for y-axis
hue	Column name for color grouping
size	Column name for size grouping
style	Column name for style grouping
data	DataFrame to use
kind	Type of plot to draw
kwargs	Additional keyword arguments to pass to the plotting function

Categorical Plots

sns.catplot(x=None, y=None, hue=None, data=None, kind='strip', **kwargs)

Parameter	Description
x	Column name for x-axis
y	Column name for y-axis
hue	Column name for color grouping
data	DataFrame to use
kind	Type of plot to draw
kwargs	Additional keyword arguments to pass to the plotting function

Distribution Plots

sns.displot(x=None, y=None, hue=None, data=None, kind='hist', **kwargs)

Parameter	Description
x	Column name for x-axis
y	Column name for y-axis
hue	Column name for color grouping
data	DataFrame to use
kind	Type of plot to draw
kwargs	Additional keyword arguments to pass to the plotting function

Regression Plots

sns.lmplot(x=None, y=None, hue=None, data=None, **kwargs)

Parameter	Description
x	Column name for x-axis

y	Column name for y-axis
hue	Column name for color grouping
data	DataFrame to use
kwargs	Additional keyword arguments to pass to the plotting function

Matrix Plots

sns.heatmap(data=None, **kwargs)

Parameter	Description
data	DataFrame to use
kwargs	Additional keyword arguments to pass to the plotting function

Customizing Plots

Color Palettes

sns.color_palette(palette=None, n_colors=None, desat=None)

Parameter	Description
palette	Name of palette to use
n_colors	Number of colors in the palette
desat	Saturation factor for colors

Color Maps

sns.color_palette(palette=None, n_colors=None, desat=None)

Parameter	Description

palette	Name of palette to use
n_colors	Number of colors in the palette
desat	Saturation factor for colors

Axis Labels

ax.set(xlabel=None, ylabel=None)

Parameter	**Description**
xlabel	Label for x-axis
ylabel	Label for y-axis

Titles

ax.set_title(label=None, fontdict=None, loc=None, pad=None, **kwargs)

Parameter	**Description**
label	Text for title
fontdict	Dictionary of font properties
loc	Location of title
pad	Padding between title and plot
kwargs	Additional keyword arguments to pass to matplotlib.text.Text()

Legends

ax.legend(*args, **kwargs)

Parameter	**Description**
args	Artist objects to include in legend
kwargs	Additional keyword arguments to pass to matplotlib.legend.Legend()

NumPy Cheat Sheet

NumPy is a popular Python library that is used for scientific computing. It is an open-source library that provides support for large, multi-dimensional arrays and matrices, along with a wide range of mathematical functions to operate on these arrays. NumPy is widely used in data science, machine learning, and scientific research.

NumPy is built on top of the Python programming language and provides a powerful set of tools for working with numerical data. It is designed to be fast and efficient, making it ideal for working with large datasets. NumPy is also highly optimized for numerical operations, making it a popular choice for scientific computing.

Cheat Sheet

Function	Description
numpy.array()	Creates a numpy array
numpy.zeros()	Creates an array of zeros
numpy.ones()	Creates an array of ones
numpy.arange()	Creates an array with evenly spaced values
numpy.linspace()	Creates an array with evenly spaced values within a specified range
numpy.random.rand()	Creates an array of random values between 0 and 1
numpy.random.randn()	Creates an array of random values from a normal distribution
numpy.reshape()	Reshapes an array
numpy.transpose()	Transposes an array

numpy.concatenate()	Concatenates two or more arrays
numpy.split()	Splits an array into multiple sub-arrays
numpy.max()	Returns the maximum value in an array
numpy.min()	Returns the minimum value in an array
numpy.mean()	Returns the mean of an array
numpy.median()	Returns the median of an array
numpy.std()	Returns the standard deviation of an array
numpy.var()	Returns the variance of an array
numpy.dot()	Computes the dot product of two arrays
numpy.sum()	Returns the sum of an array
numpy.prod()	Returns the product of an array
numpy.absolute()	Returns the absolute value of an array
numpy.exp()	Returns the exponential value of an array
numpy.log()	Returns the natural logarithm of an array
numpy.sin()	Returns the sine of an array
numpy.cos()	Returns the cosine of an array
numpy.tan()	Returns the tangent of an array
numpy.arcsin()	Returns the inverse sine of an array
numpy.arccos()	Returns the inverse cosine of an array
numpy.arctan()	Returns the inverse tangent of an array

Python Basics Cheat Sheet

Python is a high-level programming language that is widely used in various fields such as web development, data science, artificial intelligence, and more. It is known for its simplicity, readability, and ease of use, making it a popular choice for beginners and experts alike.

Here are some basic concepts of Python that every beginner should know:

1. Variables: Variables are used to store data in Python. They can hold different types of data such as numbers, strings, and lists.

2. Data Types: Python has several built-in data types such as integers, floats, strings, and booleans. Each data type has its own set of operations and methods.

3. Control Structures: Control structures are used to control the flow of a program. Python has if-else statements, loops, and functions that allow you to execute code based on certain conditions.

4. Functions: Functions are reusable blocks of code that perform a specific task. They can take input parameters and return output values.

5. Modules: Python has a vast library of modules that can be imported into your code to extend its functionality. Some popular modules include NumPy, Pandas, and Matplotlib.

6. Object-Oriented Programming: Python supports object-oriented programming, which allows you to create classes and objects that encapsulate data and behavior.

Variables and Data Types

Syntax	Description
x = 5	Assigns the value 5 to the variable x
y = ""Hello""	Assigns the string ""Hello"" to the variable y
z = True	Assigns the boolean value True to the variable z

a = [1, 2, 3]	Assigns a list of integers to the variable a
b = {""name"": ""John"", ""age"": 30}	Assigns a dictionary to the variable b

Operators

Syntax	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
//	Floor division
%	Modulus
**	Exponentiation
=	Assignment
==	Equality
!=	Not equal
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
and	Logical and
or	Logical or
not	Logical not

Control Flow

Conditional Statements

Syntax	Description
if condition:	Executes the code block if the condition is true
elif condition:	Executes the code block if the previous condition(s) are false and this condition is true
else:	Executes the code block if all previous conditions are false

Loops

Syntax	Description
for variable in iterable:	Executes the code block for each item in the iterable
while condition:	Executes the code block while the condition is true

Functions

Syntax	Description
def function_name(parameters):	Defines a function with the given name and parameters
return value	Returns the value from a function

Input and Output

Syntax	Description
print(value)	Prints the value to the console
input(prompt)	Prompts the user for input and returns the entered value

Modules

Syntax	Description
import module_name	Imports the module with the given name
from module_name import function_name	Imports the specified function from the module
as alias	Renames the imported module or function with the given alias

Exceptions

Syntax	Description
try:	Executes the code block
except exception_type:	Executes the code block if the specified exception is raised
finally:	Executes the code block regardless of whether an exception was raised

Classes

Syntax	Description
class class_name:	Defines a class with the given name
def __init__(self, parameters):	Defines the constructor for the class
def method_name(self, parameters):	Defines a method for the class

Anaconda Cheat Sheet

Anaconda is a popular open-source distribution of the Python and R programming languages. It is designed to simplify the process of data science and machine learning by providing a comprehensive set of tools and libraries for data analysis, visualization, and modeling. Anaconda is widely used by data scientists, researchers, and developers to build and deploy data-driven applications.

One of the key features of Anaconda is its package management system, which allows users to easily install, update, and manage hundreds of data science packages and libraries. Anaconda also includes a powerful integrated development environment (IDE) called Spyder, which provides a user-friendly interface for writing, testing, and debugging Python and R code.

Installation

Command	Description
conda install package_name	Install a package
conda update package_name	Update a package
conda remove package_name	Remove a package
conda create --name env_name	Create a new environment
conda activate env_name	Activate an environment
conda deactivate	Deactivate the current environment
conda info	Display information about the current installation
conda list	List all installed packages

Environments

Command	Description
conda create --name env_name	Create a new environment

conda activate env_name	Activate an environment
conda deactivate	Deactivate the current environment
conda env list	List all environments
conda env export > environment.yml	Export an environment to a YAML file
conda env create -f environment.yml	Create an environment from a YAML file

Packages

Command	Description
conda install package_name	Install a package
conda update package_name	Update a package
conda remove package_name	Remove a package
conda search package_name	Search for a package
conda list	List all installed packages
conda list --explicit > packages.txt	Export a list of installed packages to a text file
conda install --file packages.txt	Install packages from a text file

Channels

Command	Description
conda config --add channels channel_name	Add a new channel
conda config --remove channels channel_name	Remove a channel
conda config --show channels	Show all channels
conda install -c channel_name package_name	Install a package from a specific channel

Jupyter Notebook

Command	Description
jupyter notebook	Start the Jupyter Notebook server
jupyter notebook --notebook-dir=path/to/notebooks	Start the Jupyter Notebook server with a specific directory
jupyter notebook --port=8888	Start the Jupyter Notebook server on a specific port
jupyter notebook --no-browser	Start the Jupyter Notebook server without opening a browser
jupyter notebook list	List all running Jupyter Notebook servers
jupyter nbconvert --to=pdf notebook.ipynb	Convert a notebook to PDF format
jupyter nbconvert --to=html notebook.ipynb	Convert a notebook to HTML format

Conda Forge

Command	Description
conda install -c conda-forge package_name	Install a package from the Conda Forge channel
conda config --add channels conda-forge	Add the Conda Forge channel
conda config --remove channels conda-forge	Remove the Conda Forge channel
conda search -c conda-forge package_name	Search for a package in the Conda Forge channel