Cheat Sheet: Python for Data Science, AI & Development

Estimated reading time: 12 minutes

Package/Method	Description	Code Example

.read_csv()	Reads data from a `.CSV` file and creates a DataFrame.	Syntax:
		`dataframe_name = pd.read_csv("filename.csv")`
		Copied!
		Example:
		`df = pd.read_csv("data.csv")`
		Copied!

.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:

Package/Method	Description	Code 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!

Accessing Values	You can access the values in a dictionary using their corresponding keys.	Syntax:
		`Value = dict_name["key_name"]`
		Copied!
		Example:
		`name = person["name"]`
		`age = person["age"]`
		Copied!

Add or modify	Inserts a new key-value pair into the	Syntax: `dict_name[key] = value` Copied!

Package/Method	Description	Code Example

	dictionary. If the key already exists, the value will be updated; otherwise, a new entry is created.	Example:
		`person["Country"] = "USA" # A new entry will be created.`
		`person["city"] = "Chicago" # Update the existing value for the same key`
		Copied!

add()	Elements can be added to a set using the \`add()\` method. Duplicates are automatically removed, as sets only store unique values.	Syntax:
		`set_name.add(element)`
		Copied!
		Example:
		`fruits.add("mango")`
		Copied!

AND	Returns \`True\` if both statement1 and statement2 are \`True\`. Otherwise, returns \`False\`.	Syntax: `statement1 and statement2` Copied! Example:

Package/Method	Description	Code Example

		`marks = 90`
		`attendance_percentage = 87`
		`if marks >= 80 and attendance_percentage >= 85:`
		`print("qualify for honors")`
		`else:`
		`print("Not qualified for honors")`
		`# Output = qualify for honors`
		Copied!

Class Definition	Defines a blueprint for creating objects and defining their attributes and behaviors.	Syntax:
		`class ClassName: # Class attributes and methods`
		Copied!
		Example:
		`class Person:`
		`def __init__(self, name, age):`
		`self.name = name`
		`self.age = age`
		Copied!

clear()	The clear() method	Syntax:

Package/Method	Description	Code Example

	empties the dictionary, removing all key-value pairs within it. After this operation, the dictionary is still accessible and can be used further.	`dict_name.clear()`
		Copied!
		Example:
		`grades.clear()`
		Copied!

clear()	The `clear()` method removes all elements from the set, resulting in an empty set. It updates the set in-place.	Syntax:
		`set_name.clear()`
		Copied!
		Example:
		`fruits.clear()`
		Copied!

Comments	Comments are lines of text that are ignored by the Python interpreter when	`# This is a comment` Copied!

Package/Method	Description	Code Example
	executing the code.	

Concatenation	Combines (concatenates) strings.	Syntax:
		`concatenated_string = string1 + string2`
		Copied!
		Example:
		`result = "Hello" + " John"`
		Copied!

copy()	Creates a shallow copy of the dictionary. The new dictionary contains the same key-value pairs as the original, but they remain distinct objects in memory.	Syntax:
		`new_dict = dict_name.copy()`
		Copied!
		Example:
		`new_person = person.copy()`
		`new_person = dict(person) # another way to create a copy of dictionary`
		Copied!

copy()	The `copy()` method creates a shallow copy of the set. Any	Syntax: `new_set = set_name.copy()` Copied! Example:

Package/Method	Description	Code Example
	modifications to the copy won't affect the original set.	`new_fruits = fruits.copy()` Copied!

Creating a Dictionary	A dictionary is a built-in data type that represents a collection of key-value pairs. Dictionaries are enclosed in curly braces {}.	Example:
		`dict_name = {} #Creates an empty dictionary`
		`person = { "name": "John", "age": 30, "city": "New York"}`
		Copied!

Data Types	- Integer - Float - Boolean - String	Example: `x=7`

Package/Method	Description	Code Example

		# Integer Value
		y=14
		# Float Value
		is_valid = True
		# Boolean Value
		is_valid = False
		# Boolean Value
		F_Name = "John"
		# String Value
		Copied!

Define Function	A `function` is a reusable block of code that performs a specific task or set of tasks when called.	Syntax: def function_name(parameters): # Function body Copied! Example: def greet(name): print("Hello,", name) Copied!

Defining Sets	A set is an unordered collection of unique elements. Sets are enclosed in curly braces `{}`. They are useful for storing	Example:
		empty_set = set() #Creating an Empty
		Set fruits = {"apple", "banana", "orange"}
		Copied!

Package/Method	Description	Code Example
	distinct values and performing set operations.	

del	Removes the specified key-value pair from the dictionary. Raises a KeyError if the key does not exist.	Syntax:
		`del dict_name[key]`
		Copied!
		Example:
		`del person["Country"]`
		Copied!

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

discard()	Use the `discard()` method to remove a specific element from the set. Ignores if	Syntax:
		`set_name.discard(element)`
		Copied!
		Example:
		`fruits.discard("apple")`
		Copied!

Package/Method	Description	Code Example
	the element is not found.	

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	Syntax:

Package/Method	Description	Code Example

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

Equal(==)	Checks if two values are equal.	Syntax:
		`variable1 == variable2`
		Copied!
		Example 1:
		`5 == 5`
		`returns True`
		Copied!
		Example 2:
		`age = 25 age == 30`
		`returns False`
		Copied!

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)`
		Copied!
		Examples:

Package/Method	Description	Code 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", "r+") as file: content = file.read()`
		`file.write("Updated content: " + content)`
		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!

Package/Method	Description	Code Example

		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!

Filter Rows	Creates a new DataFrame	Syntax:

Package/Method	Description	Code Example

	with rows that meet specified conditions.	`filtered_df = dataframe_name[(Conditional_statements)]`
		Copied!
		Example:
		`filtered_df = df[(df["age"] > 30) & (df["salary"] < 50000)`
		Copied!

For Loop	A `for` loop repeatedly executes a block of code for a specified number of iterations or over a sequence of elements (list, range, string, etc.).	Syntax:
		`for variable in sequence: # Code to repeat`
		Copied!
		Example 1:
		`for num in range(1, 10):`
		`print(num)`
		Copied!
		Example 2:
		`fruits = ["apple", "banana", "orange", "grape", "kiwi"]`
		`for fruit in fruits:`
		`print(fruit)`
		Copied!

Package/Method	Description	Code Example

Function Call	A function call is the act of executing the code within the function using the provided arguments.	Syntax:
		`function_name(arguments)`
		Copied!
		Example:
		`greet("Alice")`
		Copied!

Greater Than or Equal To(>=)	Checks if the value of variable1 is greater than or equal to variable2.	Syntax:
		`variable1 >= variable2`
		Copied!
		Example 1:
		`5 >= 5 and 9 >= 5`
		`returns True`
		Copied!
		Example 2:
		`quantity = 105`
		`minimum = 100`
		`quantity >= minimum`
		`returns True`
		Copied!

Package/Method	Description	Code Example

Greater Than(>)	Checks if the value of variable1 is greater than variable2.	Syntax:
		`variable1 > variable2`
		Copied!
		Example 1:
		`9 > 6`
		`returns True`
		Copied!
		Example 2:
		`age = 20`
		`max_age = 25`
		`age > max_age`
		`returns False`
		Copied!

groupby()	Splits a DataFrame into groups based on specified criteria, enabling subsequent aggregation, transformation, or	Syntax:
		`grouped = dataframe_name.groupby(by, axis=0, level=None,`
		`as_index=True,`

Package/Method	Description	Code Example

	analysis within each group.	`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!

If Statement	Executes code block \`if\` the condition is \`True\`.	Syntax:
		`if condition: #code block for if statement`
		Copied!
		Example:
		`if temperature > 30:`
		`print("It's a hot day!")`
		Copied!

If-Elif-Else	Executes the first code	Syntax:

Package/Method	Description	Code Example

	block if condition1 is `True`, otherwise checks condition2, and so on. If no condition is `True`, the else block is executed.	`if condition1:`
		`# Code if condition1 is True`
		`elif condition2:`
		`# Code if condition2 is True`
		`else:`
		`# Code if no condition is True`
		Copied!
		Example:
		`score = 85 # Example score`
		`if score >= 90:`
		`print("You got an A!")`
		`elif score >= 80:`
		`print("You got a B.")`
		`else:`
		`print("You need to work harder.")`

Package/Method	Description	Code Example
		`# Output = You got a B.` Copied!

If-Else Statement	Executes the first code block if the condition is `True`, otherwise the second block.	Syntax:
		`if condition: # Code, if condition is True`
		`else: # Code, if condition is False`
		Copied!
		Example:
		`if age >= 18:`
		`print("You're an adult.")`
		`else:`
		`print("You're not an adult yet.")`
		Copied!

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

Package/Method	Description	Code Example
		`import numpy as np` Copied! Example: `import numpy as np` Copied!

Indexing	Accesses character at a specific index.	Example:
		`my_string="Hello"`
		`char = my_string[0]`
		Copied!

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

issubset()	The `issubset()` method checks if the current set is a subset of another set. It returns True if all elements of the current	Syntax:
		`is_subset = setissubset(set2)`
		Copied!
		Example:
		`is_subset = fruits.issubset(colors)`
		Copied!

Package/Method	Description	Code Example
	set are present in the other set, otherwise False.	

issuperset()	The `issuperset()` method checks if the current set is a superset of another set. It returns True if all elements of the other set are present in the current set, otherwise False.	Syntax:
		`is_superset = setissuperset(set2)`
		Copied!
		Example:
		`is_superset = colors.issuperset(fruits)`
		Copied!

items()	Retrieves all key-value pairs as tuples and converts them into a list of tuples. Each tuple consists of a	Syntax:
		`items_list = list(dict_name.items())`
		Copied!
		Example:
		`info = list(person.items())`
		Copied!

Package/Method	Description	Code Example
	key and its corresponding value.	

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!

key existence	You can check for the existence of a key in a dictionary using the in keyword	Example:
		`if "name" in person:`
		`print("Name exists in the dictionary.")`
		Copied!

keys()	Retrieves all keys from the dictionary and converts them into a list. Useful for iterating or	Syntax:
		`keys_list = list(dict_name.keys())`
		Copied!
		Example:
		`person_keys = list(person.keys())`
		Copied!

Package/Method	Description	Code Example
	processing keys using list methods.	

len()	Returns the length of a string.	Syntax:
		`len(string_name)`
		Copied!
		Example:
		`my_string="Hello"`
		`length = len(my_string)`
		Copied!

Less Than or Equal To(<=)	Checks if the value of variable1 is less than or equal to variable2.	Syntax:
		`variable1 <= variable2`
		Copied!
		Example 1:
		`5 <= 5 and 3 <= 5`
		`returns True`
		Copied!
		Example 2:
		`size = 38`
		`max_size = 40`

Package/Method	Description	Code Example

		`size <= max_size`
		`returns True`
		Copied!

Less Than(<)	Checks if the value of variable1 is less than variable2.	Syntax:
		`variable1 < variable2`
		Copied!
		Example 1:
		`4 < 6`
		`returns True`
		Copied!
		Example 2:
		`score = 60`
		`passing_score = 65`
		`score < passing_score`
		`returns True`
		Copied!

Loop Controls	`break` exits the loop prematurely. `continue` skips the rest of the current	Syntax:

Package/Method	Description	Code Example

	iteration and moves to the next iteration.	`for: # Code to repeat`
		`if # boolean statement`
		`break`
		`for: # Code to repeat`
		`if # boolean statement`
		`continue`
		Copied!
		Example 1:
		`for num in range(1, 6):`
		`if num == 3:`
		`break`
		`print(num)`
		Copied!
		Example 2:
		`for num in range(1, 6):`
		`if num == 3:`
		`continue`
		`print(num)`
		Copied!

Package/Method	Description	Code Example

lower()	Converts string to lowercase.	Example:
		my_string="Hello"
		uppercase_text = my_string.lower()
		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!
NOT	Returns `True` if variable is `False`, and vice versa.	Syntax: !variable Copied! Example: !isLocked returns True if the variable is False (i.e., unlocked). Copied!
Not Equal(!=)	Checks if two values	Syntax: variable1 != variable2

Package/Method	Description	Code Example

	are not equal.	Copied!
		Example:
		`a = 10`
		`b = 20`
		`a != b`
		`returns True`
		Copied!
		Example 2:
		`count=0`
		`count != 0`
		`returns False`
		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:

Package/Method	Description	Code 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	Example:
		`np.mean(array)`
		`np.sum(array)`
		`np.min(array)`
		`np.max(array)`
		`np.dot(array_1, array_2)`
		Copied!

Object Creation	Creates an instance of a class (object) using the class constructor.	Syntax:
		`object_name = ClassName(arguments)`
		Copied!
		Example:
		`person1 = Person("Alice", 25)`
		Copied!

Open() and close()	Opens a file, performs operations, and	Syntax:

Package/Method	Description	Code Example

	explicitly closes the file using the close() method.	`file = open(filename, mode) # Code that uses the file`
		`file.close()`
		Copied!
		Example:
		`file = open("data.txt", "r")`
		`content = file.read()`
		`file.close()`
		Copied!

OR	Returns `True` if either statement1 or statement2 (or both) are `True`. Otherwise, returns `False`.	Syntax:
		`statement1 || statement2`
		Copied!
		Example:
		`"Farewell Party Invitation"`
		`Grade = 12 grade == 11 or grade == 12`
		`returns True`
		Copied!

pop()	The `pop()` method removes and returns an arbitrary element	Syntax: `removed_element = set_name.pop()` Copied! Example:

Package/Method	Description	Code Example
	from the set. It raises a `KeyError` if the set is empty. Use this method to remove elements when the order doesn't matter.	`removed_fruit = fruits.pop()` Copied!

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

print()	Prints the message or variable inside `()`.	Example:
		`print("Hello, world")`
		`print(a+b)`
		Copied!

Python Operators	- Addition (+): Adds	Example:

Package/Method	Description	Code Example

	two values together.	`x = 9 y = 4`
		`result_add= x + y # Addition`
		`result_sub= x - y # Subtraction`
		`result_mul= x * y # Multiplication`
		`result_div= x / y # Division`
		`result_fdiv= x // y # Floor Division`
		`result_mod= x % y # Modulo</td>`
		Copied!

range()	Generates a sequence of numbers within a specified range.	Syntax:
		`range(stop)`
		`range(start, stop)`
		`range(start, stop, step)`
		Copied!
		Example:
		`range(5) #generates a sequence of integers from 0 to`

Package/Method	Description	Code Example

		`range(2, 10) #generates a sequence of integers from 2 to`
		`range(1, 11, 2) #generates odd integers from 1 to`
		Copied!

remove()	Use the `remove()` method to remove a specific element from the set. Raises a `KeyError` if the element is not found.	Syntax:
		`set_name.remove(element)`
		Copied!
		Example:
		`fruits.remove("banana")`
		Copied!

replace()	Replaces substrings.	Example:
		`my_string="Hello"`
		`new_text = my_string.replace("Hello", "Hi")`
		Copied!

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

Package/Method	Description	Code Example
		Example: `df["status"].replace("In Progress", "Active", inplace=True)` Copied!

Return Statement	`Return` is a keyword used to send a value back from a function to its caller.	Syntax:
		`return value`
		Copied!
		Example:
		`def add(a, b): return a + b`
		`result = add(3, 5)`
		Copied!

Set Operations	Perform various operations on sets: `union`, `intersection`, `difference`, `symmetric difference`.	Syntax:
		`union_set = setunion(set2)`
		`intersection_set = setintersection(set2)`
		`difference_set = setdifference(set2)`
		`sym_diff_set = setsymmetric_difference(set2)`
		Copied!
		Example:

Package/Method	Description	Code Example

		combined = fruits.union(colors)
		common = fruits.intersection(colors)
		unique_to_fruits = fruits.difference(colors)
		sym_diff = fruits.symmetric_difference(colors)
		Copied!

Slicing	Extracts a portion of the string.	Syntax: substring = string_name[start:end] Copied! Example: my_string="Hello" substring = my_string[0:5] Copied!

split()	Splits a string into a list based on a delimiter.	Example:
		my_string="Hello"
		split_text = my_string.split(",")
		Copied!

strip()	Removes leading/trailing whitespace.	Example: my_string="Hello"

Package/Method	Description	Code Example
		`trimmed = my_string.strip()` Copied!

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

Try-Except Block	Tries to execute the code in the try block. If an exception of the specified type occurs, the code in the except block is executed.	Syntax:
		`try: # Code that might raise an exception except`
		`ExceptionType: # Code to handle the exception`
		Copied!
		Example:
		`try:`
		`num = int(input("Enter a number: "))`
		`except ValueError:`
		`print("Invalid input. Please enter a valid number.")`
		Copied!

Package/Method	Description	Code Example

Try-Except with Else Block	Code in the `else` block is executed if no exception occurs in the try block.	Syntax:
		`try: # Code that might raise an exception except`
		`ExceptionType: # Code to handle the exception`
		`else: # Code to execute if no exception occurs`
		Copied!
		Example:
		`try:`
		`num = int(input("Enter a number: "))`
		`except ValueError:`
		`print("Invalid input. Please enter a valid number")`
		`else:`
		`print("You entered:", num)`
		Copied!

Try-Except with Finally Block	Code in the `finally` block always	Syntax:

Package/Method	Description	Code Example

	executes, regardless of whether an exception occurred.	`try: # Code that might raise an exception except`
		`ExceptionType: # Code to handle the exception`
		`finally: # Code that always executes`
		Copied!
		Example:
		`try:`
		`file = open("data.txt", "r")`
		`data = file.read()`
		`except FileNotFoundError:`
		`print("File not found.")`
		`finally:`
		`file.close()`
		Copied!

update()	The update() method merges the provided dictionary into the	Syntax: `dict_name.update({key: value})` Copied! Example:

Package/Method	Description	Code Example
	existing dictionary, adding or updating key-value pairs.	`person.update({"Profession": "Doctor"})` Copied!

update()	The `update()` method adds elements from another iterable into the set. It maintains the uniqueness of elements.	Syntax:
		`set_name.update(iterable)`
		Copied!
		Example:
		`fruits.update(["kiwi", "grape"])`
		Copied!

upper()	Converts string to uppercase.	Example:
		`my_string="Hello"`
		`uppercase_text = my_string.upper()`
		Copied!

values()	Extracts all values from the dictionary and converts	Syntax: `values_list = list(dict_name.values())` Copied! Example:

Package/Method	Description	Code Example
	them into a list. This list can be used for further processing or analysis.	`person_values = list(person.values())` Copied!

Variable Assignment	Assigns a value to a variable.	Syntax:
		`variable_name = value`
		Copied!
		Example:
		`name="John" # assigning John to variable name`
		`x = 5 # assigning 5 to variable x`
		Copied!

While Loop	A `while` loop repeatedly executes a block of code as long as a specified condition remains `True`.	Syntax:
		`while condition: # Code to repeat`
		Copied!
		Example:
		`count = 0 while count < 5:`
		`print(count) count += 1`
		Copied!

with open()	Opens a file using a with	Syntax:

Package/Method	Description	Code Example

	block, ensuring automatic file closure after usage.	`with open(filename, mode) as file: # Code that uses the file`
		Copied!
		Example:
		`with open("data.txt", "r") as file:`
		`content = file.read()`