8 Best Python Cheat Sheets for Beginners and Intermediate Learners

Python Cheat Sheet can be really helpful when you're working on a project or trying a set of exercises related to a specific topic.

> *If you are just getting started with Data Science. I've got a few practical reads for you. One about The Best Python for Data Science Courses from World-Class Educators and One about Learning how to learn Data Science (with Python, Maths and Statistics).*

And now rather than explaining to you the importance of cheat sheets, why not just begin with the most useful Python resources available on the internet (for free) in the form of cheat sheet.

7+ Best Python Cheat Sheets ...

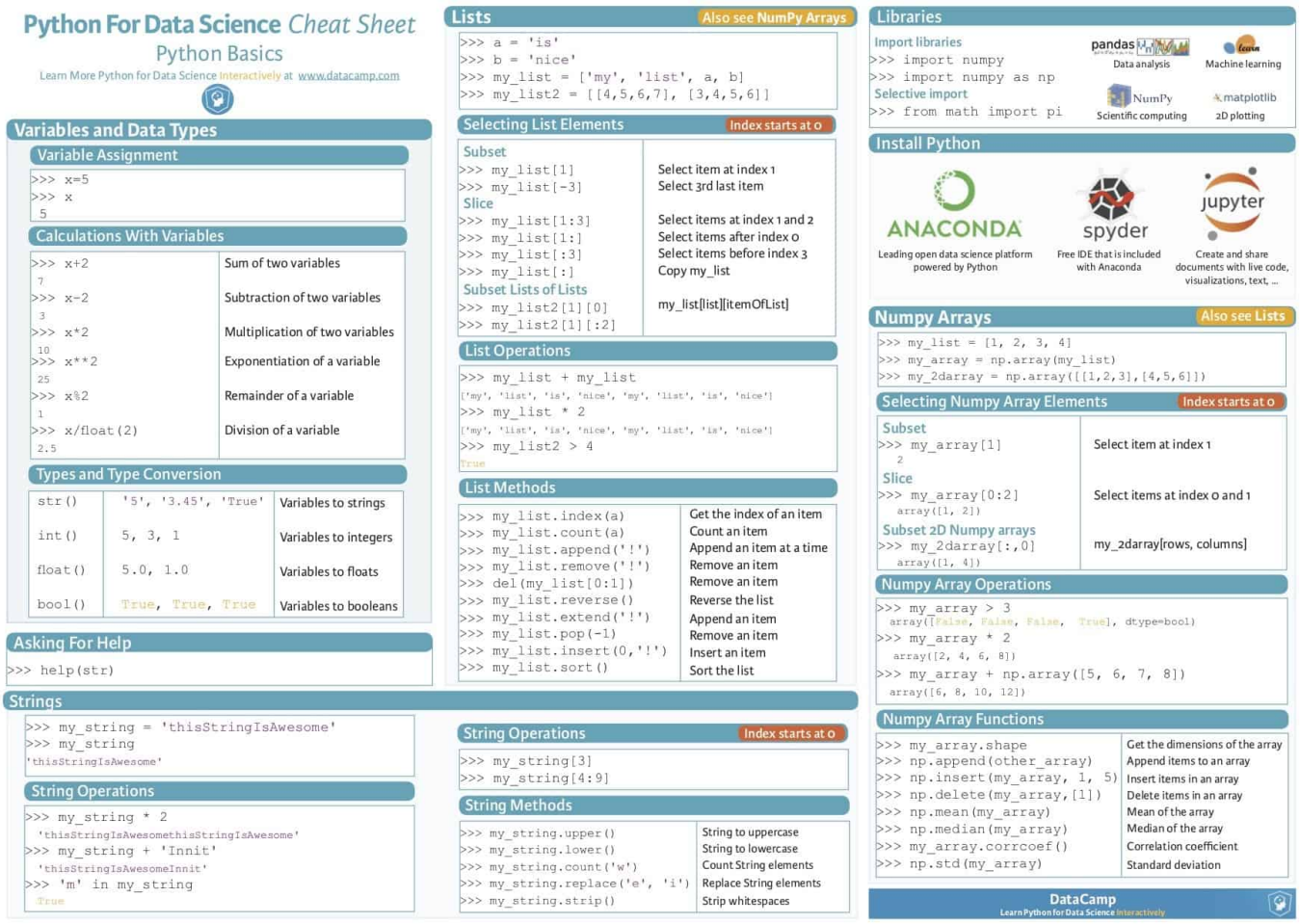
Here's a curated a list of Python Cheat Sheets and most commonly used Python Libraries. You'll be able to download them with ease and grasp the fundamentals for long-term benefits

---

1. Python for Data Science Cheat Sheet

This Python Cheat Sheet presents the Python basics that you need to do data science and will guide you through variables and data types, Strings, Lists, to eventually land at the fundamental package for scientific computing with Python, Numpy.

Also, This cheat sheet is free additional material that complements DataCamp's Intro to Python for Data Science course, where you learn by doing.



You can Download Pdf here

---

2. Python Cheat Sheet for Data Science: Basics

This cheat sheet is the companion to free Python Programming Beginner Course course offered by Dataquest which can start you on your data science journey.

You'll be able to practice reading files, Strings, Numeric Types, and Mathematical Operations, Lists and Dictionaries, Modules and Functions, Boolean Comparisons, Statements, and Loops.

Data Science Cheat Sheet

Python Basics

BASICS, PRINTING AND GETTING HELP

`x = 3` - Assign 3 to the variable x
`print(x)` - Print the value of x
`help(x)` - Show documentation for the `str` data type
`help(print)` - Show documentation for the `print()` function
`type(x)` - Return the type of the variable x (in this case, `int` for integer)

READING FILES

```
f = open("my_file.txt","r")
file_as_string = f.read()
```

- Open the file `my_file.txt` and assign its contents to `s`

```
import csv
f = open("my_dataset.csv","r")
csvreader = csv.reader(f)
csv_as_list = list(csvreader)
```

- Open the CSV file `my_dataset.csv` and assign its data to the list of lists `csv_as_list`

STRINGS

`s = "hello"` - Assign the string `"hello"` to the variable `s`

```
s = """She said,
"there's a good idea."
"""
```

- Assign a multi-line string to the variable `s`. Also used to create strings that contain both " and ' characters

`len(s)` - Return the number of characters in `s`
`s.startswith("hel")` - Test whether `s` starts with the substring `"hel"`
`s.endswith("lo")` - Test whether `s` ends with the substring `"lo"`
`"{} plus {} is {}".format(3,1,4)` - Return the string with the values `3`, `1`, and `4` inserted
`s.replace("e","z")` - Return a new string based on `s` with all occurances of `"e"` replaced with `"z"`
`s.split(" ")` - Split the string `s` into a list of strings, separating on the character `" "` and return that list

NUMERIC TYPES AND MATHEMATICAL OPERATIONS

`i = int("5")` - Convert the string `"5"` to the integer `5` and assign the result to `i`
`f = float("2.5")` - Convert the string `"2.5"` to the float value `2.5` and assign the result to `f`
`5 + 5` - Addition
`5 - 5` - Subtraction
`10 / 2` - Division
`5 * 2` - Multiplication
`3 ** 2` - Raise `3` to the power of `2` (or 3^2)
`27 ** (1/3)` - The 3rd root of `27` (or $\sqrt[3]{27}$)
`x += 1` - Assign the value of `x + 1` to `x`
`x -= 1` - Assign the value of `x - 1` to `x`

LISTS

`l = [100,21,88,3]` - Assign a list containing the integers `100`, `21`, `88`, and `3` to the variable `l`
`l = list()` - Create an empty list and assign the result to `l`
`l[0]` - Return the first value in the list `l`
`l[-1]` - Return the last value in the list `l`
`l[1:3]` - Return a slice (list) containing the second and third values of `l`
`len(l)` - Return the number of elements in `l`
`sum(l)` - Return the sum of the values of `l`
`min(l)` - Return the minimum value from `l`
`max(l)` - Return the maximum value from `l`
`l.append(16)` - Append the value `16` to the end of `l`
`l.sort()` - Sort the items in `l` in ascending order
`" ".join(["A","B","C","D"])` - Converts the list `["A", "B", "C", "D"]` into the string `"A B C D"`

DICTIONARIES

`d = {"CA":"Canada","GB":"Great Britain", "IN":"India"}` - Create a dictionary with keys of `"CA"`, `"GB"`, and `"IN"` and corresponding values of of `"Canada"`, `"Great Britain"`, and `"India"`
`d["GB"]` - Return the value from the dictionary `d` that has the key `"GB"`
`d.get("AU","Sorry")` - Return the value from the dictionary `d` that has the key `"AU"`, or the string `"Sorry"` if the key `"AU"` is not found in `d`
`d.keys()` - Return a list of the keys from `d`
`d.values()` - Return a list of the values from `d`
`d.items()` - Return a list of `(key, value)` pairs from `d`

MODULES AND FUNCTIONS

The body of a function is defined through indentation.

`import random` - Import the module `random`
`from math import sqrt` - Import the function `sqrt` from the module `math`

```
def calculate(addition_one,addition_two,
exponent=1,factor=1):
    result = (value_one + value_two) ** exponent * factor
    return result
```

- Define a new function `calculate` with two required and two optional named arguments which calculates and returns a result.

`addition(3,5,factor=10)` - Run the `addition` function with the values `3` and `5` and the named argument `10`

BOOLEAN COMPARISONS

`x == 5` - Test whether `x` is equal to `5`
`x != 5` - Test whether `x` is not equal to `5`
`x > 5` - Test whether `x` is greater than `5`
`x < 5` - Test whether `x` is less than `5`
`x >= 5` - Test whether `x` is greater than or equal to `5`
`x <= 5` - Test whether `x` is less than or equal to `5`
`x == 5 or name == "alfred"` - Test whether `x` is equal to `5` or `name` is equal to `"alfred"`
`x == 5 and name == "alfred"` - Test whether `x` is equal to `5` and `name` is equal to `"alfred"`
`5 in l` - Checks whether the value `5` exists in the list `l`
`"GB" in d` - Checks whether the value `"GB"` exists in the keys for `d`

IF STATEMENTS AND LOOPS

The body of if statements and loops are defined through indentation.

```
if x > 5:
    print("{} is greater than five".format(x))
elif x < 0:
    print("{} is negative".format(x))
else:
    print("{} is between zero and five".format(x))
```

- Test the value of the variable `x` and run the code body based on the value

```
for value in l:
    print(value)
```

- Iterate over each value in `l`, running the code in the body of the loop with each iteration

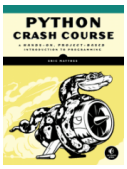
```
while x < 10:
    x += 1
```

- Run the code in the body of the loop until the value of `x` is no longer less than `10`

You can Download here

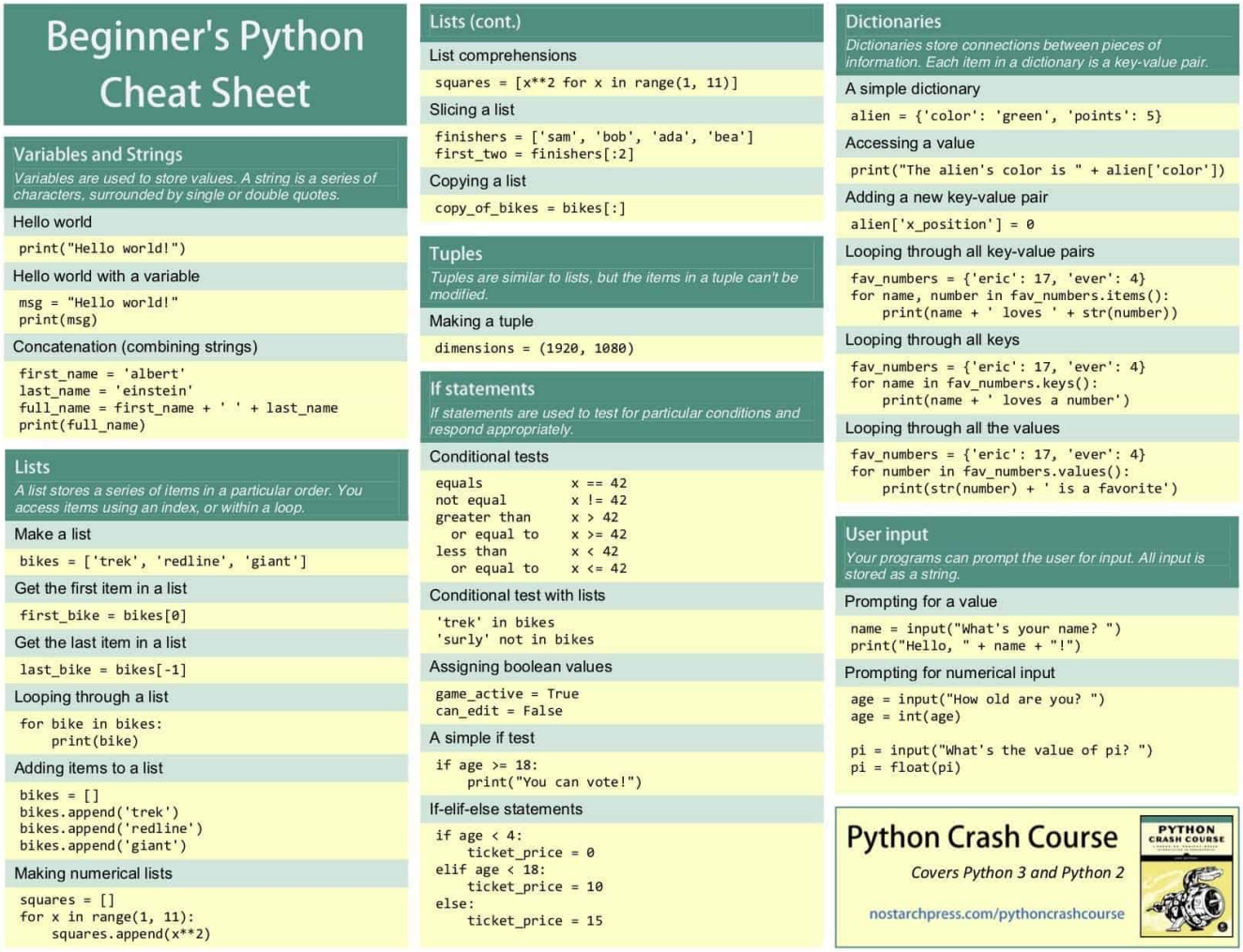
---

3. Python Crash Course



This Python Cheat is from the Book Python Crash Course which aims to remind you of syntax rules and grasp all the important concepts in Python programming as a beginner.

You will also create a game with Pygame, Create Data Visualization with PyGal and build Web Apps with Django.



Beginner's Python Cheat Sheet

Variables and Strings

Variables are used to store values. A string is a series of characters, surrounded by single or double quotes.

Hello world

```
print("Hello world!")
```

Hello world with a variable

```
msg = "Hello world!"
print(msg)
```

Concatenation (combining strings)

```
first_name = 'albert'
last_name = 'einstein'
full_name = first_name + ' ' + last_name
print(full_name)
```

Lists

A list stores a series of items in a particular order. You access items using an index, or within a loop.

Make a list

```
bikes = ['trek', 'redline', 'giant']
```

Get the first item in a list

```
first_bike = bikes[0]
```

Get the last item in a list

```
last_bike = bikes[-1]
```

Looping through a list

```
for bike in bikes:
    print(bike)
```

Adding items to a list

```
bikes = []
bikes.append('trek')
bikes.append('redline')
bikes.append('giant')
```

Making numerical lists

```
squares = []
for x in range(1, 11):
    squares.append(x**2)
```

Lists (cont.)

List comprehensions

```
squares = [x**2 for x in range(1, 11)]
```

Slicing a list

```
finishers = ['sam', 'bob', 'ada', 'bea']
first_two = finishers[:2]
```

Copying a list

```
copy_of_bikes = bikes[:]
```

Tuples

Tuples are similar to lists, but the items in a tuple can't be modified.

Making a tuple

```
dimensions = (1920, 1080)
```

If statements

If statements are used to test for particular conditions and respond appropriately.

Conditional tests

```
equals            x == 42
not equal         x != 42
greater than      x > 42
  or equal to     x >= 42
less than         x < 42
  or equal to     x <= 42
```

Conditional test with lists

```
'trek' in bikes
'surly' not in bikes
```

Assigning boolean values

```
game_active = True
can_edit = False
```

A simple if test

```
if age >= 18:
    print("You can vote!")
```

If-elif-else statements

```
if age < 4:
    ticket_price = 0
elif age < 18:
    ticket_price = 10
else:
    ticket_price = 15
```

Dictionaries

Dictionaries store connections between pieces of information. Each item in a dictionary is a key-value pair.

A simple dictionary

```
alien = {'color': 'green', 'points': 5}
```

Accessing a value

```
print("The alien's color is " + alien['color'])
```

Adding a new key-value pair

```
alien['x_position'] = 0
```

Looping through all key-value pairs

```
fav_numbers = {'eric': 17, 'ever': 4}
for name, number in fav_numbers.items():
    print(name + ' loves ' + str(number))
```

Looping through all keys

```
fav_numbers = {'eric': 17, 'ever': 4}
for name in fav_numbers.keys():
    print(name + ' loves a number')
```

Looping through all the values

```
fav_numbers = {'eric': 17, 'ever': 4}
for number in fav_numbers.values():
    print(str(number) + ' is a favorite')
```

User input

Your programs can prompt the user for input. All input is stored as a string.

Prompting for a value

```
name = input("What's your name? ")
print("Hello, " + name + "!")
```

Prompting for numerical input

```
age = input("How old are you? ")
age = int(age)

pi = input("What's the value of pi? ")
pi = float(pi)
```


Python Crash Course

Covers Python 3 and Python 2

nostarchpress.com/pythoncrashcourse

You can Download Pdf here

---

4. Python Cheat Sheet for Data Science: Intermediate

DATAQUEST

This cheat sheet assumes you are familiar with the content of the Python Basic

Cheat Sheet from Dataquest.

This Python cheat sheet provides in-depth focus on Lists, Strings, Range, Dictionaries, Sets, Regular Expressions, List Comprehension, Functions for Looping, DateTime, Random, Counter and Try Except.

Data Science Cheat Sheet

Python - Intermediate

KEY BASICS, PRINTING AND GETTING HELP

This cheat sheet assumes you are familiar with the content of our Python Basics Cheat Sheet

`s` - A Python string variable
`i` - A Python integer variable
`f` - A Python float variable
`l` - A Python list variable
`d` - A Python dictionary variable

LISTS

`l.pop(3)` - Returns the fourth item from `l` and deletes it from the list
`l.remove(x)` - Removes the first item in `l` that is equal to `x`
`l.reverse()` - Reverses the order of the items in `l`
`l[1::2]` - Returns every second item from `l`, commencing from the `1`st item
`l[-5:]` - Returns the last 5 items from `l` specific axis

STRINGS

`s.lower()` - Returns a lowercase version of `s`
`s.title()` - Returns `s` with the first letter of every word capitalized
`"23".zfill(4)` - Returns `"0023"` by left-filling the string with `0`'s to make it's length `4`.
`s.splitlines()` - Returns a list by splitting the string on any newline characters.

Python strings share some common methods with lists

`s[:5]` - Returns the first `5` characters of `s`
`"fri" + "end"` - Returns `"friend"`
`"end" in s` - Returns `True` if the substring `"end"` is found in `s`

RANGE

Range objects are useful for creating sequences of integers for looping.

`range(5)` - Returns a sequence from `0` to `4`
`range(2000,2018)` - Returns a sequence from `2000` to `2017`
`range(0,11,2)` - Returns a sequence from `0` to `10`, with each item incrementing by `2`
`range(0,-10,-1)` - Returns a sequence from `0` to `-9`
`list(range(5))` - Returns a list from `0` to `4`

DICTIONARIES

`max(d, key=d.get)` - Return the key that corresponds to the largest value in `d`
`min(d, key=d.get)` - Return the key that corresponds to the smallest value in `d`

SETS

`my_set = set(l)` - Return a `set` object containing the unique values from `l`
`len(my_set)` - Returns the number of objects in `my_set` (or, the number of unique values from `l`)
`a in my_set` - Returns `True` if the value `a` exists in `my_set`

REGULAR EXPRESSIONS

`import re` - Import the Regular Expressions module
`re.search("abc",s)` - Returns a `match` object if the regex `"abc"` is found in `s`, otherwise `None`
`re.sub("abc","xyz",s)` - Returns a string where all instances matching regex `"abc"` are replaced by `"xyz"`

LIST COMPREHENSION

A one-line expression of a for loop

`[i ** 2 for i in range(10)]` - Returns a list of the squares of values from `0` to `9`
`[s.lower() for s in l_strings]` - Returns the list `l_strings`, with each item having had the `.lower()` method applied
`[i for i in l_floats if i < 0.5]` - Returns the items from `l_floats` that are less than `0.5`

FUNCTIONS FOR LOOPING

```
for i, value in enumerate(l):
  print("The value of item {} is {}".
  format(i,value))
```

- Iterate over the list `l`, printing the index location of each item and its value

```
for one, two in zip(l_one,l_two):
  print("one: {}, two: {}".format(one,two))
```

- Iterate over two lists, `l_one` and `l_two` and print each value

```
while x < 10:
  x += 1
```

- Run the code in the body of the loop until the value of `x` is no longer less than `10`

DATETIME

`import datetime as dt` - Import the `datetime` module
`now = dt.datetime.now()` - Assign `datetime` object representing the current time to `now`
`wks4 = dt.datetime.timedelta(weeks=4)` - Assign a `timedelta` object representing a timespan of 4 weeks to `wks4`
`now - wks4` - Return a `datetime` object representing the time 4 weeks prior to `now`
`newyear_2020 = dt.datetime(year=2020, month=12, day=31)` - Assign a `datetime` object representing December 25, 2020 to `newyear_2020`
`newyear_2020.strftime("%A, %b %d, %Y")` - Returns `"Thursday, Dec 31, 2020"`
`dt.datetime.strptime('Dec 31, 2020',"%b %d, %Y")` - Return a `datetime` object representing December 31, 2020

RANDOM

`import random` - Import the `random` module
`random.random()` - Returns a random float between `0.0` and `1.0`
`random.randint(0,10)` - Returns a random integer between `0` and `10`
`random.choice(l)` - Returns a random item from the list `l`

COUNTER

`from collections import Counter` - Import the `Counter` class
`c = Counter(l)` - Assign a `Counter` (dict-like) object with the counts of each unique item from `l`, to `c`
`c.most_common(3)` - Return the 3 most common items from `l`

TRY/EXCEPT

Catch and deal with Errors

`l_ints = [1, 2, 3, "", 5]` - Assign a list of integers with one missing value to `l_ints`

```
l_floats = []
for i in l_ints:
  try:
    l_floats.append(float(i))
  except:
    l_floats.append(i)
```

- Convert each value of `l_ints` to a float, catching and handling `ValueError: could not convert string to float:` where values are missing.

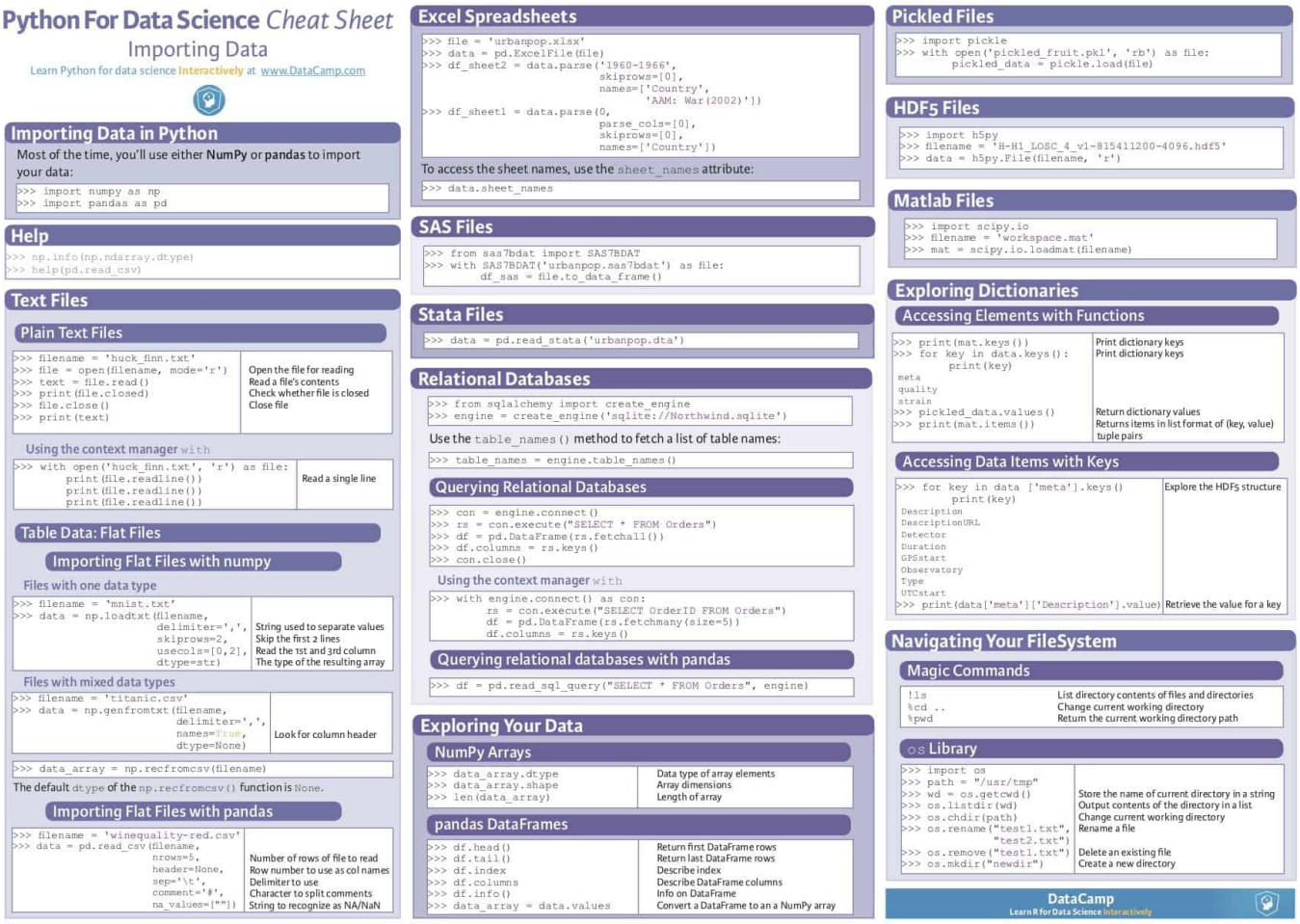
You can Download here

---

5. Importing Data in Python Cheat Sheet

This Python Cheat Sheet from Datacamp provides everything that you need to kickstart your data science learning with Python. Moreover, you'll have a handy reference guide to importing your data, from flat files to files native to other software, and relational databases.

You'll also learn how you can get data from files native to other software such as Excel spreadsheets, Stata, SAS and MATLAB files and relational databases.



You can Download Pdf here

---

6. Python NumPy Cheat Sheet

DATAQUEST

This cheat sheet assumes you are familiar with NumPy. If you're interested in learning NumPy, you can start learning about NumPy in Python Data

Science Course from Dataquest.

This Python Numpy Cheat Sheet will make you familiar with NumPy Array and how you can Import and Export Data for analysis. You'll also index data and retrieve results, using NumPy with Scalar Math, Vector Math, and Statistics will hold no secrets for you any longer.

Data Science Cheat Sheet

NumPy

KEY

We'll use shorthand in this cheat sheet

`arr` - A numpy Array object

IMPORTS

Import these to start

```
import numpy as np
```

IMPORTING/EXPORTING

`np.loadtxt('file.txt')` - From a text file

`np.genfromtxt('file.csv',delimiter=',')` - From a CSV file

`np.savetxt('file.txt',arr,delimiter=' ')` - Writes to a text file

`np.savetxt('file.csv',arr,delimiter=',')` - Writes to a CSV file

CREATING ARRAYS

`np.array([1,2,3])` - One dimensional array

`np.array([(1,2,3),(4,5,6)])` - Two dimensional array

`np.zeros(3)` - 1D array of length **3** all values **0**

`np.ones((3,4))` - **3x4** array with all values **1**

`np.eye(5)` - **5x5** array of **0** with **1** on diagonal (Identity matrix)

`np.linspace(0,100,6)` - Array of **6** evenly divided values from **0** to **100**

`np.arange(0,10,3)` - Array of values from **0** to less than **10** with step **3** (eg **[0,3,6,9]**)

`np.full((2,3),8)` - **2x3** array with all values **8**

`np.random.rand(4,5)` - **4x5** array of random floats between **0-1**

`np.random.rand(6,7)*100` - **6x7** array of random floats between **0-100**

`np.random.randint(5,size=(2,3))` - **2x3** array with random ints between **0-4**

INSPECTING PROPERTIES

`arr.size` - Returns number of elements in **arr**

`arr.shape` - Returns dimensions of **arr** (rows, columns)

`arr.dtype` - Returns type of elements in **arr**

`arr.astype(dtype)` - Convert **arr** elements to type **dtype**

`arr.tolist()` - Convert **arr** to a Python list

`np.info(np.eye)` - View documentation for **np.eye**

COPYING/SORTING/RESHAPING

`np.copy(arr)` - Copies **arr** to new memory

`arr.view(dtype)` - Creates view of **arr** elements with type **dtype**

`arr.sort()` - Sorts **arr**

`arr.sort(axis=0)` - Sorts specific axis of **arr**

`two_d_arr.flatten()` - Flattens 2D array **two_d_arr** to 1D

`arr.T` - Transposes **arr** (rows become columns and vice versa)

`arr.reshape(3,4)` - Reshapes **arr** to **3** rows, **4** columns without changing data

`arr.resize((5,6))` - Changes **arr** shape to **5x6** and fills new values with **0**

ADDING/REMOVING ELEMENTS

`np.append(arr,values)` - Appends **values** to end of **arr**

`np.insert(arr,2,values)` - Inserts **values** into **arr** before index **2**

`np.delete(arr,3,axis=0)` - Deletes row on index **3** of **arr**

`np.delete(arr,4,axis=1)` - Deletes column on index **4** of **arr**

COMBINING/SPLITTING

`np.concatenate((arr1,arr2),axis=0)` - Adds **arr2** as rows to the end of **arr1**

`np.concatenate((arr1,arr2),axis=1)` - Adds **arr2** as columns to end of **arr1**

`np.split(arr,3)` - Splits **arr** into **3** sub-arrays

`np.hsplit(arr,5)` - Splits **arr** horizontally on the **5**th index

INDEXING/SLICING/SUBSETTING

`arr[5]` - Returns the element at index **5**

`arr[2,5]` - Returns the 2D array element on index **[2][5]**

`arr[1]=4` - Assigns array element on index **1** the value **4**

`arr[1,3]=10` - Assigns array element on index **[1][3]** the value **10**

`arr[0:3]` - Returns the elements at indices **0,1,2** (On a 2D array: returns rows **0,1,2**)

`arr[0:3,4]` - Returns the elements on rows **0,1,2** at column **4**

`arr[:2]` - Returns the elements at indices **0,1** (On a 2D array: returns rows **0,1**)

`arr[:,1]` - Returns the elements at index **1** on all rows

`arr<5` - Returns an array with boolean values

`(arr1<3) & (arr2>5)` - Returns an array with boolean values

`~arr` - Inverts a boolean array

`arr[arr<5]` - Returns array elements smaller than **5**

SCALAR MATH

`np.add(arr,1)` - Add **1** to each array element

`np.subtract(arr,2)` - Subtract **2** from each array element

`np.multiply(arr,3)` - Multiply each array element by **3**

`np.divide(arr,4)` - Divide each array element by **4** (returns **np.nan** for division by zero)

`np.power(arr,5)` - Raise each array element to the **5**th power

VECTOR MATH

`np.add(arr1,arr2)` - Elementwise add **arr2** to **arr1**

`np.subtract(arr1,arr2)` - Elementwise subtract **arr2** from **arr1**

`np.multiply(arr1,arr2)` - Elementwise multiply **arr1** by **arr2**

`np.divide(arr1,arr2)` - Elementwise divide **arr1** by **arr2**

`np.power(arr1,arr2)` - Elementwise raise **arr1** raised to the power of **arr2**

`np.array_equal(arr1,arr2)` - Returns **True** if the arrays have the same elements and shape

`np.sqrt(arr)` - Square root of each element in the array

`np.sin(arr)` - Sine of each element in the array

`np.log(arr)` - Natural log of each element in the array

`np.abs(arr)` - Absolute value of each element in the array

`np.ceil(arr)` - Rounds up to the nearest int

`np.floor(arr)` - Rounds down to the nearest int

`np.round(arr)` - Rounds to the nearest int

STATISTICS

`np.mean(arr,axis=0)` - Returns mean along specific axis

`arr.sum()` - Returns sum of **arr**

`arr.min()` - Returns minimum value of **arr**

`arr.max(axis=0)` - Returns maximum value of specific axis

`np.var(arr)` - Returns the variance of array

`np.std(arr,axis=1)` - Returns the standard deviation of specific axis

`arr.corrcoef()` - Returns correlation coefficient of array

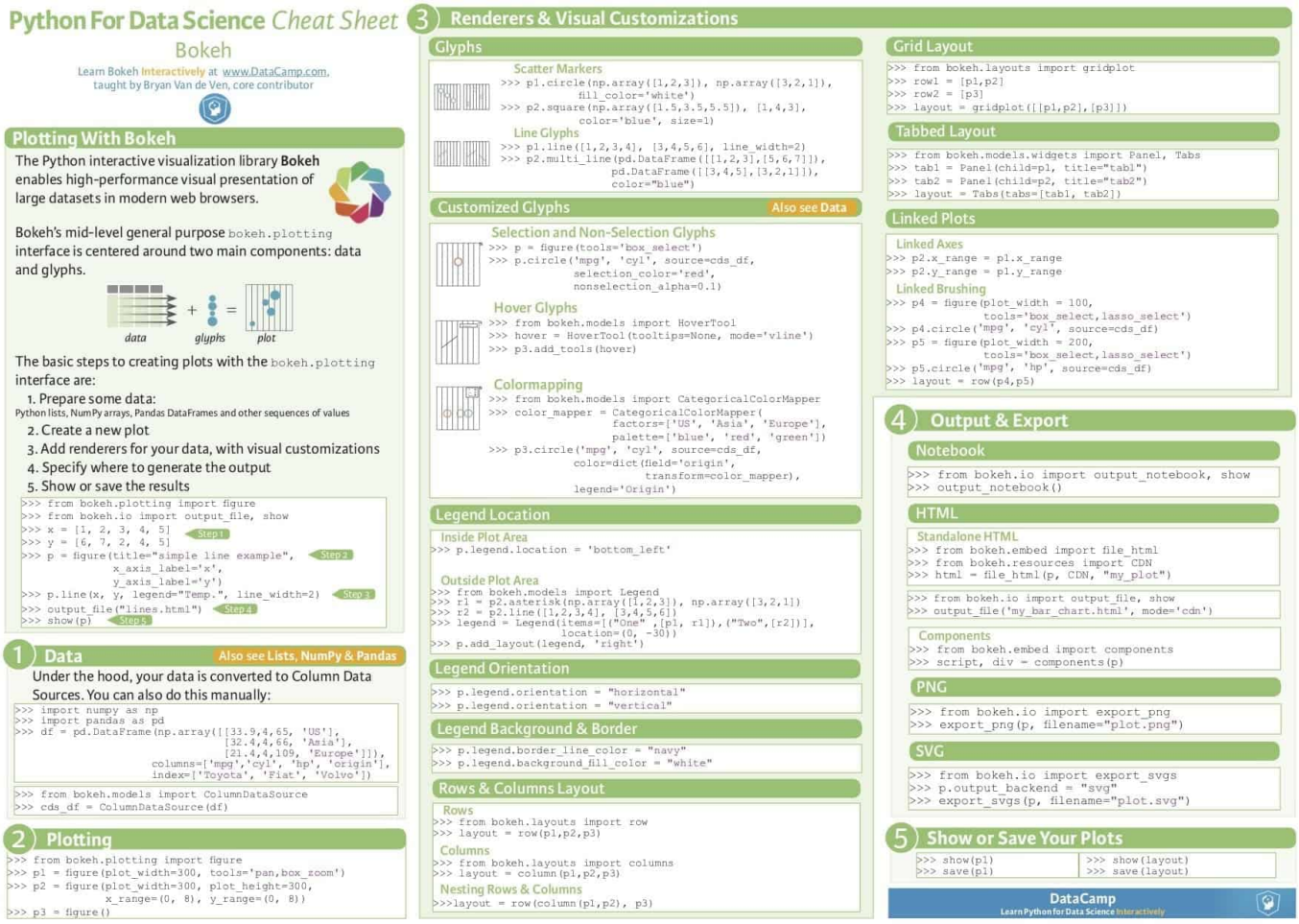
You can Download here

---

7. Python Data Visualization: Bokeh Cheat Sheet

This Python Cheat Sheet will guide you to interactive plotting and statistical charts with Bokeh. Python Bokeh Cheat Sheet is a free additional material for Interactive Data Visualization with Bokeh Course and is a handy one-page reference for those who need an extra push to get started with Bokeh.

This cheat sheet will walk you through making beautiful plots and also introduce you to the basics of statistical charts.



You can Download Pdf here

---

8. Python for Data Science: Pandas Cheat Sheet

Pandas is a data-centric Python package. It's common when first learning pandas to have trouble remembering all the functions and methods that you need, and it's

nice to have a handy reference.

We hope this cheat sheet will help you out! If you are interested in learning, you can signup for free and start learning Pandas for Data Science Course offered by Dataquest.

Data Science Cheat Sheet

Pandas

KEY

We'll use shorthand in this cheat sheet

`df` - A pandas DataFrame object

`s` - A pandas Series object

IMPORTS

Import these to start

```
import pandas as pd
import numpy as np
```

IMPORTING DATA

`pd.read_csv(filename)` - From a CSV file

`pd.read_table(filename)` - From a delimited text file (like TSV)

`pd.read_excel(filename)` - From an Excel file

`pd.read_sql(query, connection_object)` - Reads from a SQL table/database

`pd.read_json(json_string)` - Reads from a JSON formatted string, URL or file.

`pd.read_html(url)` - Parses an html URL, string or file and extracts tables to a list of dataframes

`pd.read_clipboard()` - Takes the contents of your clipboard and passes it to `read_table()`

`pd.DataFrame(dict)` - From a dict, keys for columns names, values for data as lists

EXPORTING DATA

`df.to_csv(filename)` - Writes to a CSV file

`df.to_excel(filename)` - Writes to an Excel file

`df.to_sql(table_name, connection_object)` - Writes to a SQL table

`df.to_json(filename)` - Writes to a file in JSON format

`df.to_html(filename)` - Saves as an HTML table

`df.to_clipboard()` - Writes to the clipboard

CREATE TEST OBJECTS

Useful for testing

`pd.DataFrame(np.random.rand(20,5))` - 5 columns and 20 rows of random floats

`pd.Series(my_list)` - Creates a series from an iterable `my_list`

`df.index = pd.date_range('1900/1/30', periods=df.shape[0])` - Adds a date index

VIEWING/INSPECTING DATA

`df.head(n)` - First `n` rows of the DataFrame

`df.tail(n)` - Last `n` rows of the DataFrame

`df.shape()` - Number of rows and columns

`df.info()` - Index, Datatype and Memory information

`df.describe()` - Summary statistics for numerical columns

`s.value_counts(dropna=False)` - Views unique values and counts

`df.apply(pd.Series.value_counts)` - Unique values and counts for all columns

SELECTION

`df[col]` - Returns column with label `col` as Series

`df[[col1, col2]]` - Returns Columns as a new DataFrame

`s.iloc[0]` - Selection by position

`s.loc[0]` - Selection by index

`df.iloc[0,:]` - First row

`df.iloc[0,0]` - First element of first column

DATA CLEANING

`df.columns = ['a','b','c']` - Renames columns

`pd.isnull()` - Checks for null Values, Returns Boolean Array

`pd.notnull()` - Opposite of `s.isnull()`

`df.dropna()` - Drops all rows that contain null values

`df.dropna(axis=1)` - Drops all columns that contain null values

`df.dropna(axis=1,thresh=n)` - Drops all rows have have less than `n` non null values

`df.fillna(x)` - Replaces all null values with `x`

`s.fillna(s.mean())` - Replaces all null values with the mean (mean can be replaced with almost any function from the statistics section)

`s.astype(float)` - Converts the datatype of the series to float

`s.replace(1,'one')` - Replaces all values equal to `1` with `'one'`

`s.replace([1,3],['one','three'])` - Replaces all `1` with `'one'` and `3` with `'three'`

`df.rename(columns=lambda x: x + 1)` - Mass renaming of columns

`df.rename(columns={'old_name': 'new_name'})` - Selective renaming

`df.set_index('column_one')` - Changes the index

`df.rename(index=lambda x: x + 1)` - Mass renaming of index

FILTER, SORT, & GROUPBY

`df[df[col] > 0.5]` - Rows where the `col` column is greater than `0.5`

`df[(df[col] > 0.5) & (df[col] < 0.7)]` - Rows where `0.7 > col > 0.5`

`df.sort_values(col1)` - Sorts values by `col1` in ascending order

`df.sort_values(col2,ascending=False)` - Sorts values by `col2` in descending order

`df.sort_values([col1,col2], ascending=[True,False])` - Sorts values by `col1` in ascending order then `col2` in descending order

`df.groupby(col)` - Returns a groupby object for values from one column

`df.groupby([col1,col2])` - Returns a groupby object values from multiple columns

`df.groupby(col1)[col2].mean()` - Returns the mean of the values in `col2`, grouped by the values in `col1` (mean can be replaced with almost any function from the statistics section)

`df.pivot_table(index=col1,values=[col2,col3],aggfunc=mean)` - Creates a pivot table that groups by `col1` and calculates the mean of `col2` and `col3`

`df.groupby(col1).agg(np.mean)` - Finds the average across all columns for every unique column 1 group

`df.apply(np.mean)` - Applies a function across each column

`df.apply(np.max, axis=1)` - Applies a function across each row

JOIN/COMBINE

`df1.append(df2)` - Adds the rows in `df1` to the end of `df2` (columns should be identical)

`pd.concat([df1, df2],axis=1)` - Adds the columns in `df1` to the end of `df2` (rows should be identical)

`df1.join(df2,on=col1,how='inner')` - SQL-style joins the columns in `df1` with the columns on `df2` where the rows for `col` have identical values. `how` can be one of `'left'`, `'right'`, `'outer'`, `'inner'`

STATISTICS

These can all be applied to a series as well.

`df.describe()` - Summary statistics for numerical columns

`df.mean()` - Returns the mean of all columns

`df.corr()` - Returns the correlation between columns in a DataFrame

`df.count()` - Returns the number of non-null values in each DataFrame column

`df.max()` - Returns the highest value in each column

`df.min()` - Returns the lowest value in each column

`df.median()` - Returns the median of each column

`df.std()` - Returns the standard deviation of each column

You can Download here

Thanks for making it to the end

Share & Learn! Do add your favourite Python Cheat Sheet in the comments below.

You may also be interested in checking the list of Machine Learning Cheat Sheets in Python and Maths or listening to a Python Podcast to bootstrap your knowledge in Python.

If you liked this article, i've got few practical reads for you.

Statistics for Data Science

Best and Affordable Data Science Courses

Skills in Python for Every Data Scientist

I've also got this Data Science Newsletter that you might be into. I send a tiny email once or twice every quarter with some useful resource I've found.

Don't worry, I hate spam as much as you. Feel free to subscribe. ☟

Banner Image Credits: https://uxdesign.cc/ui-cheat-sheet-text-fields-2152112615f8