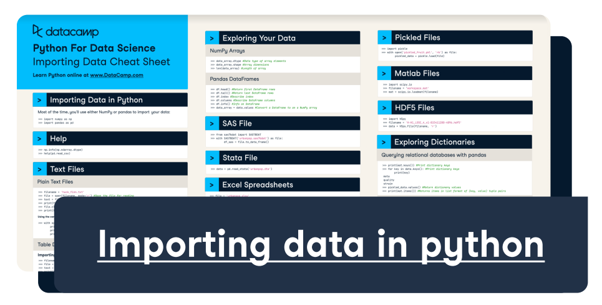
# Python Basics

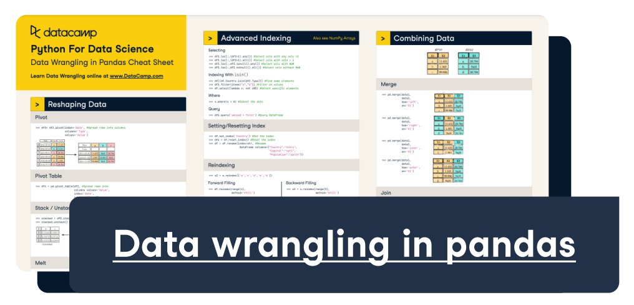
Python Cheat Sheet for Beginners

**Learn Python online at www.DataCamp.com**

## > How to use this cheat sheet

Python is the most popular programming language in data science. It is easy to learn and comes with a wide array of powerful libraries for data analysis. This cheat sheet provides beginners and intermediate users a guide to starting using python. Use it to jump-start your journey with python. If you want more detailed Python cheat sheets, check out the following cheat sheets below:


Importing data in python


Data wrangling in pandas

## > Accessing help and getting object types

```
1 + 1 # Everything after the hash symbol is ignored by Python
help(max) # Display the documentation for the max function
type('a') # Get the type of an object — this returns str
```

## > Importing packages

Python packages are a collection of useful tools developed by the open-source community. They extend the capabilities of the python language. To install a new package (for example, pandas), you can go to your command prompt and type in `pip install pandas`. Once a package is installed, you can import it as follows.

```
import pandas # Import a package without an alias
import pandas as pd # Import a package with an alias
from pandas import DataFrame # Import an object from a package
```

## > The working directory

The working directory is the default file path that python reads or saves files into. An example of the working directory is "C://file/path". The os library is needed to set and get the working directory.

```
import os # Import the operating system package
os.getcwd() # Get the current directory
os.setcwd("new/working/directory") # Set the working directory to a new file path
```

## > Operators

### Arithmetic operators

```
102 + 37 # Add two numbers with +                22 // 7 # Integer divide a number with //
102 - 37 # Subtract a number with -              3 ** 4 # Raise to the power with **
4 * 6 # Multiply two numbers with *              22 % 7 # Returns 1 # Get the remainder after
22 / 7 # Divide a number by another with /       division with %
```

### Assignment operators

```
a = 5 # Assign a value to a
x[0] = 1 # Change the value of an item in a list
```

### Numeric comparison operators

```
3 == 3 # Test for equality with ==               3 >= 3 # Test greater than or equal to with >=
3 != 3 # Test for inequality with !=             3 < 4  # Test less than with <
3 > 1  # Test greater than with >                3 <= 4 # Test less than or equal to with <=
```

### Logical operators

```
~(2 == 2)          # Logical NOT with ~       (1 >= 1) | (1 < 1) # Logical OR with |
(1 != 1) & (1 < 1) # Logical AND with &       (1 != 1) ^ (1 < 1) # Logical XOR with ^
```

## > Getting started with lists

A list is an ordered and changeable sequence of elements. It can hold integers, characters, floats, strings, and even objects.

### Creating lists

```
# Create lists with [], elements separated by commas
x = [1, 3, 2]
```

### List functions and methods

```
x.sorted(x) # Return a sorted copy of the list e.g., [1,2,3]
x.sort() # Sorts the list in-place (replaces x)
reversed(x) # Reverse the order of elements in x e.g., [2,3,1]
x.reversed() # Reverse the list in-place
x.count(2) # Count the number of element 2 in the list
```

### Selecting list elements

Python lists are zero-indexed (the first element has index 0). For ranges, the first element is included but the last is not.

```
# Define the list
x = ['a', 'b', 'c', 'd', 'e']                 x[1:3] # Select 1st (inclusive) to 3rd (exclusive)
x[0] # Select the 0th element in the list     x[2:] # Select the 2nd to the end
x[-1] # Select the last element in the list   x[:3] # Select 0th to 3rd (exclusive)
```

### Concatenating lists

```
# Define the x and y lists      x + y # Returns [1, 3, 6, 10, 15, 21]
x = [1, 3, 6]                   3 * x # Returns [1, 3, 6, 1, 3, 6, 1, 3, 6]
y = [10, 15, 21]
```

## > Getting started with dictionaries

A dictionary stores data values in key-value pairs. That is, unlike lists which are indexed by position, dictionaries are indexed by their keys, the names of which must be unique.

### Creating dictionaries

```
# Create a dictionary with {}
{'a': 1, 'b': 4, 'c': 9}
```

### Dictionary functions and methods

```
x = {'a': 1, 'b': 2, 'c': 3} # Define the x ditionary
x.keys() # Get the keys of a dictionary, returns dict_keys(['a', 'b', 'c'])
x.values() # Get the values of a dictionary, returns dict_values([1, 2, 3])
```

### Selecting dictionary elements

```
x['a'] # 1 # Get a value from a dictionary by specifying the key
```

## > NumPy arrays

NumPy is a python package for scientific computing. It provides multidimensional array objects and efficient operations on them. To import NumPy, you can run this Python code `import numpy as np`

### Creating arrays

```
# Convert a python list to a NumPy array
np.array([1, 2, 3]) # Returns array([1, 2, 3])
# Return a sequence from start (inclusive) to end (exclusive)
np.arange(1,5) # Returns array([1, 2, 3, 4])
# Return a stepped sequence from start (inclusive) to end (exclusive)
np.arange(1,5,2) # Returns array([1, 3])
# Repeat values n times
np.repeat([1, 3, 6], 3) # Returns array([1, 1, 1, 3, 3, 3, 6, 6, 6])
# Repeat values n times
np.tile([1, 3, 6], 3) # Returns array([1, 3, 6, 1, 3, 6, 1, 3, 6])
```

## > Math functions and methods

All functions take an array as the input.

```
np.log(x) # Calculate logarithm           np.quantile(x, q) # Calculate q-th quantile
np.exp(x) # Calculate exponential         np.round(x, n) # Round to n decimal places
np.max(x) # Get maximum value             np.var(x) # Calculate variance
np.min(x) # Get minimum value             np.std(x) # Calculate standard deviation
np.sum(x) # Calculate sum
np.mean(x) # Calculate mean
```

## > Getting started with characters and strings

```
# Create a string with double or single quotes
"DataCamp"

# Embed a quote in string with the escape character \
"He said, \"DataCamp\""

# Create multi-line strings with triple quotes
"""
A Frame of Data
Tidy, Mine, Analyze It
Now You Have Meaning
Citation: https://mdsr-book.github.io/haikus.html
"""

str[0] # Get the character at a specific position
str[0:2] # Get a substring from starting to ending index (exclusive)
```

### Combining and splitting strings

```
"Data" + "Framed" # Concatenate strings with +, this returns 'DataFramed'
3 * "data " # Repeat strings with *, this returns 'data data data '
"beekeepers".split("e") # Split a string on a delimiter, returns ['b', '', 'k', '', 'p', 'rs']
```

### Mutate strings

```
str = "Jack and Jill" # Define str
str.upper() # Convert a string to uppercase, returns 'JACK AND JILL'
str.lower() # Convert a string to lowercase, returns 'jack and jill'
str.title() # Convert a string to title case, returns 'Jack And Jill'
str.replace("J", "P") # Replaces matches of a substring with another, returns 'Pack and Pill'
```

## > Getting started with DataFrames

Pandas is a fast and powerful package for data analysis and manipulation in python. To import the package, you can use `import pandas as pd`. A pandas DataFrame is a structure that contains two-dimensional data stored as rows and columns. A pandas series is a structure that contains one-dimensional data.

### Creating DataFrames

```
# Create a dataframe from a dictionary       # Create a dataframe from a list of dictionaries
pd.DataFrame({                               pd.DataFrame([
 'a': [1, 2, 3],                              {'a': 1, 'b': 4, 'c': 'x'},
 'b': np.array([4, 4, 6]),                    {'a': 1, 'b': 4, 'c': 'x'},
 'c': ['x', 'x', 'y']                         {'a': 3, 'b': 6, 'c': 'y'}
})                                           ])
```

### Selecting DataFrame Elements

Select a row, column or element from a dataframe. Remember: all positions are counted from zero, not one.

```
# Select the 3rd row
df.iloc[3]
# Select one column by name
df['col']
# Select multiple columns by names
df[['col1', 'col2']]
# Select 2nd column
df.iloc[:, 2]
# Select the element in the 3rd row, 2nd column
df.iloc[3, 2]
```

### Manipulating DataFrames

```
# Concatenate DataFrames vertically            # Calculate the mean of each column
pd.concat([df, df])                            df.mean()
# Concatenate DataFrames horizontally          # Get summary statistics by column
pd.concat([df,df],axis="columns")              df.agg(aggregation_function)
# Get rows matching a condition                # Get unique rows
df.query('logical_condition')                  df.drop_duplicates()
# Drop columns by name                         # Sort by values in a column
df.drop(columns=['col_name'])                  df.sort_values(by='col_name')
# Rename columns                               # Get rows with largest values in a column
df.rename(columns={"oldname": "newname"})      df.nlargest(n, 'col_name')
# Add a new column
df.assign(temp_f=9 / 5 * df['temp_c'] + 32)
```