



Python Cheat Sheet



Python | Pandas
Data Analysis
Data Visualization

by Frank Andrade



Python Basics Cheat Sheet

Here you will find all the Python core concepts you need to know before learning any third-party library.

Data Types

Integers (int): `1`
Float (float): `1.2`
String (str): `"Hello World"`
Boolean: `True/False`
List: `[value1, value2]`
Dictionary: `{key1:value1, key2:value2, ...}`

Numeric Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Exponent
%	Modulus
//	Floor division

Comparison Operators

==	Equal to
!=	Different
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

String methods

`string.upper()`: converts to uppercase
`string.lower()`: converts to lowercase
`string.title()`: converts to title case
`string.count('l')`: counts how many times "l" appears
`string.find('h')`: position of the "h" first occurrence
`string.replace('o','u')`: replaces "o" with "u"

Variables

Variable assignment:
`message_1 = "I'm learning Python"`
`message_2 = "and it's fun!"`

String concatenation (+ operator):
`message_1 + ' ' + message_2`

String concatenation (f-string):
`f'{message_1} {message_2}'`

List

Creating a list:
`countries = ['United States', 'India', 'China', 'Brazil']`

Create an empty list:
`my_list = []`

Indexing:
`>>> countries[0]`
`United States`

`>>> countries[3]`
`Brazil`

`>>> countries[-1]`
`Brazil`

Slicing:
`>>> countries[0:3]`
`['United States', 'India', 'China']`

`>>> countries[1:]`
`['India', 'China', 'Brazil']`

`>>> countries[:2]`
`['United States', 'India']`

Adding elements to a list:
`countries.append('Canada')`
`countries.insert(0, 'Canada')`

Nested list:
`nested_list = [countries, countries_2]`

Remove element:
`countries.remove('United States')`
`countries.pop()` #removes and returns value
`del countries[0]`

Creating a new list:
`numbers = [4, 3, 10, 7, 1, 2]`

Sorting a list:
`>>> numbers.sort()`
`[1, 2, 3, 4, 7, 10]`

`>>> numbers.sort(reverse=True)`
`[10, 7, 4, 3, 2, 1]`

Update value on a list:
`>>> numbers[0] = 1000`
`>>> numbers`
`[1000, 7, 4, 3, 2, 1]`

Copying a list:
`new_list = countries[:]`
`new_list_2 = countries.copy()`

Built-in Functions

Print an object:
`print("Hello World")`

Return the length of x:
`len(x)`

Return the minimum value:
`min(x)`

Return the maximum value:
`max(x)`

Returns a sequence of numbers:
`range(x1,x2,n)` # from x1 to x2 (increments by n)

Convert x to a string:
`str(x)`

Convert x to an integer/float:
`int(x)`
`float(x)`

Convert x to a list:
`list(x)`

Dictionary

Creating a dictionary:

```
my_data = {'name': 'Frank', 'age': 26}
```

Create an empty dictionary:

```
my_dict = {}
```

Get value of key "name":

```
>>> my_data["name"]  
'Frank'
```

Get the keys:

```
>>> my_data.keys()  
dict_keys(['name', 'age'])
```

Get the values:

```
>>> my_data.values()  
dict_values(['Frank', 26])
```

Get the pair key-value:

```
>>> my_data.items()  
dict_items([('name', 'Frank'), ('age', 26)])
```

Adding/updating items in a dictionary:

```
my_data['height'] = 1.7  
my_data.update({'height': 1.8,  
               'languages': ['English', 'Spanish']})
```

```
>>> my_data  
{'name': 'Frank',  
  'age': 26,  
  'height': 1.8,  
  'languages': ['English', 'Spanish']}
```

Remove an item:

```
my_data.pop('height')  
del my_data['languages']  
my_data.clear()
```

Copying a dictionary:

```
new_dict = my_data.copy()
```

If Statement

Conditional test:

```
if <condition>:  
    <code>  
elif <condition>:  
    <code>  
  
...  
else:  
    <code>
```

Example:

```
if age >= 18:  
    print("You're an adult!")
```

Conditional test with list:

```
if <value> in <list>:  
    <code>
```

Loops

For loop:

```
for <variable> in <list>:  
    <code>
```

For loop and enumerate list elements:

```
for i, element in enumerate(<list>):  
    <code>
```

For loop and obtain dictionary elements:

```
for key, value in my_dict.items():  
    <code>
```

While loop:

```
while <condition>:  
    <code>
```

Data Validation

Try-except:

```
try:  
    <code>  
except <error>:  
    <code>
```

Loop control statement:

```
break: stops loop execution  
continue: jumps to next iteration  
pass: does nothing
```

Functions

Create a function:

```
def function(<params>):  
    <code>  
    return <data>
```

Modules

Import module:

```
import module  
module.method()
```

OS module:

```
import os  
os.getcwd()  
os.listdir()  
os.makedirs(<path>)
```

Special Characters

#	Comment
\n	New Line

Boolean Operators

and	logical AND
or	logical OR
not	logical NOT

Boolean Operators (Pandas)

&	logical AND
	logical OR
~	logical NOT

Below there are my guides, tutorials and complete Python courses:

- [Medium Guides](#)
- [YouTube Tutorials](#)
- [Udemy Courses](#)

Pandas Cheat Sheet

Pandas provides data analysis tools for Python. All of the following code examples refer to the dataframe below.

df =

	col1	col2
A	1	4
B	2	5
C	3	6

← axis 1

← axis 0

Getting Started

Import pandas:

```
import pandas as pd
```

Create a series:

```
s = pd.Series([1, 2, 3],
               index=['A', 'B', 'C'],
               name='col1')
```

Create a dataframe:

```
data = [[1, 4], [2, 5], [3, 6]]
index = ['A', 'B', 'C']
df = pd.DataFrame(data, index=index,
                  columns=['col1', 'col2'])
```

Read a csv file with pandas:

```
df = pd.read_csv('filename.csv')
```

Advanced parameters:

```
df = pd.read_csv('filename.csv', sep=',',
                 names=['col1', 'col2'],
                 index_col=0,
                 encoding='utf-8',
                 nrows=3)
```

Selecting rows and columns

Select single column:

```
df['col1']
```

Select multiple columns:

```
df[['col1', 'col2']]
```

Show first n rows:

```
df.head(2)
```

Show last n rows:

```
df.tail(2)
```

Select rows by index values:

```
df.loc['A'] df.loc[['A', 'B']]
```

Select rows by position:

```
df.loc[1] df.loc[1:]
```

Data wrangling

Filter by value:

```
df[df['col1'] > 1]
```

Sort by one column:

```
df.sort_values('col1')
```

Sort by columns:

```
df.sort_values(['col1', 'col2'],
               ascending=[False, True])
```

Identify duplicate rows:

```
df.duplicated()
```

Identify unique rows:

```
df['col1'].unique()
```

Swap rows and columns:

```
df = df.transpose()
df = df.T
```

Drop a column:

```
df = df.drop('col1', axis=1)
```

Clone a data frame:

```
clone = df.copy()
```

Connect multiple data frames vertically:

```
df2 = df + 5 #new dataframe
pd.concat([df, df2])
```

Merge multiple data frames horizontally:

```
df3 = pd.DataFrame([[1, 7], [8, 9]],
                   index=['B', 'D'],
                   columns=['col1', 'col3'])
```

#df3: new dataframe

Only merge complete rows (INNER JOIN):

```
df.merge(df3)
```

Left column stays complete (LEFT OUTER JOIN):

```
df.merge(df3, how='left')
```

Right column stays complete (RIGHT OUTER JOIN):

```
df.merge(df3, how='right')
```

Preserve all values (OUTER JOIN):

```
df.merge(df3, how='outer')
```

Merge rows by index:

```
df.merge(df3, left_index=True,
         right_index=True)
```

Fill NaN values:

```
df.fillna(0)
```

Apply your own function:

```
def func(x):
    return 2**x
df.apply(func)
```

Arithmetics and statistics

Add to all values:

```
df + 10
```

Sum over columns:

```
df.sum()
```

Cumulative sum over columns:

```
df.cumsum()
```

Mean over columns:

```
df.mean()
```

Standard deviation over columns:

```
df.std()
```

Count unique values:

```
df['col1'].value_counts()
```

Summarize descriptive statistics:

```
df.describe()
```

Hierarchical indexing

Create hierarchical index:
`df.stack()`

Dissolve hierarchical index:
`df.unstack()`

Aggregation

Create group object:
`g = df.groupby('col1')`

Iterate over groups:
`for i, group in g:
 print(i, group)`

Aggregate groups:
`g.sum()
g.prod()
g.mean()
g.std()
g.describe()`

Select columns from groups:
`g['col2'].sum()
g[['col2', 'col3']].sum()`

Transform values:
`import math
g.transform(math.log)`

Apply a list function on each group:
`def strsum(group):
 return ''.join([str(x) for x in group.value])

g['col2'].apply(strsum)`

Data export

Data as NumPy array:
`df.values`

Save data as CSV file:
`df.to_csv('output.csv', sep=',')`

Format a dataframe as tabular string:
`df.to_string()`

Convert a dataframe to a dictionary:
`df.to_dict()`

Save a dataframe as an Excel table:
`df.to_excel('output.xlsx')`

Pivot and Pivot Table

Read csv file 1:
`df_gdp = pd.read_csv('gdp.csv')`

The pivot() method:
`df_gdp.pivot(index="year",
 columns="country",
 values="gdppc")`

Read csv file 2:
`df_sales=pd.read_excel(
 'supermarket_sales.xlsx')`

Make pivot table:
`df_sales.pivot_table(index='Gender',
 aggfunc='sum')`

Make a pivot tables that says how much male and female spend in each category:

```
df_sales.pivot_table(index='Gender',  
                      columns='Product line',  
                      values='Total',  
                      aggfunc='sum')
```

Visualization

The plots below are made with a dataframe with the shape of `df_gdp` (pivot() method)

Import matplotlib:
`import matplotlib.pyplot as plt`

Start a new diagram:
`plt.figure()`

Scatter plot:
`df.plot(kind='scatter')`

Bar plot:
`df.plot(kind='bar',
 xlabel='data1',
 ylabel='data2')`

Lineplot:
`df.plot(kind='line',
 figsize=(8,4))`

Boxplot:
`df['col1'].plot(kind='box')`

Histogram over one column:
`df['col1'].plot(kind='hist',
 bins=3)`

Piechart:
`df.plot(kind='pie',
 y='col1',
 title='Population')`

Set tick marks:
`labels = ['A', 'B', 'C', 'D']
positions = [1, 2, 3, 4]
plt.xticks(positions, labels)
plt.yticks(positions, labels)`

Label diagram and axes:
`plt.title('Correlation')
plt.xlabel('Nunstück')
plt.ylabel('Slotermeyer')`

Save most recent diagram:
`plt.savefig('plot.png')
plt.savefig('plot.png', dpi=300)
plt.savefig('plot.svg')`

Below there are my guides, tutorials and complete Python courses:

- [Medium Guides](#)
- [YouTube Tutorials](#)
- [Udemy Courses](#)