Python Cheatsheet

Thomas Haslwanter
thomas.haslwanter@fh-ooe.at
August 27, 2026

Data Types

('pi', 3.14) tuple: mainly used for function returns

[1, 2, 3] list: can be `append`-ed, replaced, ...

{'x':x, 'y':y} dictionary `my_dict`:

```
my_dict['x']
my_dict['x']
my_dict.keys()
my_dict.values()
```

np.array([1,2,3]) NumPy array (1d)

np.array([[1,2,3], [4,5,6]]) array (2d)

pd.DataFrame{my_dict} Pandas dataframe `df`

```
df.x or df['x']
df.head(), df.tail()
df.to_csv(out_file)
df = pd.read_csv(in_file)
```

Slicing

```
txt = 'Hello, world'
```

`txt[:]`	copy of variable
`txt[0]`	`'H'`
`txt[:3]`	`'Hel'`
`txt[-3:]`	`'rld'`
`txt[1:3]`	`'el'` (last letter NOT included!)
`mat[0]`	`[1,2,3]`
`mat[0, :2]`	`[1,2]`
`mat[:, 0]`	`[1,4]`
`mat[0].T`	`[1,2,3]`

`np.atleast\_2d(mat[0]).T` => column vector

Loops

```
for ii in range(5):
  print(f'The square of {ii} is {ii**2}')

for color in colors:
  print(color)

for ii, color in enumerate(colors):
  print(f'Color {ii}: {color}')

people = ['Peter', 'Paul', 'Mary']
colors = ['red', 'green', 'blue']
for person, color in zip(people, colors):
  print(f'{person} likes {color}')

with open(out_file, 'a') as fh:
  fh.write('This appends a final line.')

# use 'w' for write, 'r' for read
```

NumPy & SciPy

```
# Generate 1D-arrays = vectors
np.array(my_list)
np.arange(start, stop, step)  # `stop` is NOT included!
np.linspace(start, stop, num_pts)
np.ones(num_pts)  # same with `zeros`
np.ones_like(other_array)
row_vector = np.r_[1, 2, 3]

# Generate matrices
2d_vector = np.atleast_2d(row_vector)
ones = np.ones( (rows, cols) )
np.column_stack([a, b])

# Vector and matrix operations
dot = mat_a @ mat_b
dot = a @ b
cross = np.cross(a, b)
element_wise = mat_a * mat_b
squared = mat ** 2
column_vector = 2d_vector.T # only 2d-arrays!!!
(rows, cols) = my_array.shape   # The brackets are
    ↪ optional here

# Smoothing, differentiation
smoothed_or_diffed = scipy.signal.savgol_filter(x,
    ↪ window_size, polyorder, deriv, dt)
diff = np.diff(vector)  # one element shorter
```

Pandas

```
df = pd.read_csv(in_file, delim=)
df.head()
df.tail()
df.columns()
df.to_csv(out_file)
df['col'] # select a column
df.col    # equivalent`
df.iloc[row, col] # select an element
```

Programs

```
""" Every file must have a header, and it has to use
    ↪ tripple quotes """

# author: [your name]
# date: [current date]

# Import standard packages
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# These other packages
from scipy import signal
import pingouin as pg


# Define all functions
def my_function(in_data: np.ndarray) -> tuple:[np.ndarray,
    ↪  np.ndarray]
  """ Every function has to have a header to, also in
      ↪ tripple quotes
  The type-hints, as well as the detailed spec of
      ↪ parameters, are optional

  Parameters
  ----------
  in_data: some values

  Returns
  -------
    out_data: (in_data, squared_data)
    """

    x = in_data
    x_squared = in_data**2

    return (x, x_squared)


if __name__ == '__main__':
    # Here comes the main script
    # This typically also tests/uses your functions
    x = np.arange(10)
    data, squared = np.my_function(x)
```

Plotting

```
out_file = 'my_file.png'  # or `jpg`
plt.plot(x, y, label='raw signal')
plt.plot(x, y_squared = 'squared')
plt.legend()
plt.savefig(out_file, dpi=300)

# ALWAYS let the user know if a file has been generated or
    ↪  changed!!
print(f'Image saved to {out_file}')
```

Miscellaneous commands

```
import os

os.path.abspath(os.curdir())
os.chdir(new_dir)

results = pg.linear_regression(x, y)  # line-fits

# Formatted strings
txt = f'The value of pi is {np.pi:.3f}'
print(txt)
```