Lecture 8: Files

Morten Rieger Hannemose, Vedrana Andersen Dahl

Fall 2023

Today's lecture

1. Files (20 min)
2. String formatting (10 min)
3. Plotting, sneak peek (10 min)

   (No coding example today)

Practical information

- Mid-term exam (test); Roughly corresponding to half an exam.
- (By the end of the course we will have a test of DE system.)
- Nicolai og Hans Henrik: *Programing for all* DTU 116/81 Saturday 28/10, from 9:00

Course overview

Weeks 1-7

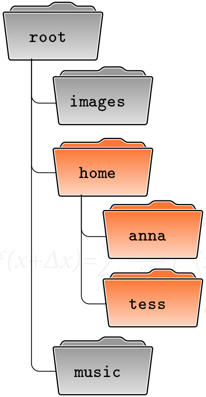
- Data types: `int`, `float`, `str`, `list`, `dict`, `tuple`
- Functions: void, fruitful, arguments, return types
- Flow control: conditions, loops
- In connection with sequences (string, list, tuple): indexing
- In connection with lists and strings: methods
- Problem solving
- Coding: testing, debugging

Weeks 8-13

- Files: reading and writing (today!)
- Object-oriented programming (2 weeks)
- Numpy, matplotlib, and other useful modules
- Efficiency and style
- The goodies
- Wrapping up

Files, folders, and paths

- Information on computers is stored in files.
- File: data treated as an entity, identified by filename (a string), usually ended by an extension.
- Path: a string that describes where the file is stored in a hierarchical system of folders.



Today:

- We can read and write files from Python.
- Focus on text files and csv-files (which are also text files)
- Exercises: examples of how an otherwise tedious task may be solved using programming, e.g. searching in many or large files.

Filenames and paths are strings

```python
import os

print(f'CWD: {os.getcwd()}')
print(os.listdir())

filename = os.path.join(os.getcwd(), 'new_file.txt')
print(filename)

if os.path.isfile(filename):
    print(f'File {filename} exists')
```

- Current working directory.
- Relative and absolute paths.
  - Relative: relates to cwd. Uses `..` to move one level up.
  - Absolute: starts with root directory.
- Dependency on the operative system.
  - Windows: `C:\Users\Vand\Desktop\course`
  - Unix-based (inc. macOS): `/Users/vand/Desktop/course`
- Module `os` – operating system interfaces.

Reading and writing text files

```python
with open('my_file.txt', 'w') as f:
    f.write('This is a new file\n')
    f.write('with two lines\n')

with open('my_file.txt') as f:
    lines = f.readlines()

print(lines)
```

- Function `open` opens a file and returns a file object.
- Statement `with` handles unexpected situations. No need to close a file.
- Reading methods: `read()`, `readline()`, `readlines()`
- Writing methods: `write()`, `writelines()`
- An escape character: backslash \ followed by a character. Most importantly: `\n`.
- Book Section 14.10, `repr()`: a printable representation
- Other readers and writers, for example, provided by `csv` module.

String formatting

```python
a = 17

print(a, '/3 =', a/3)
print(str(a) + '/3 = ' + str(a/3))

print(f'{a}/3 = {a/3}')  # using f-string
print(f'{a}/3 = {a/3:0.3}')  # custom format

print(f'{a=}')  # useful for debugging

bike_code = 217
bike_code_str = f'{bike_code:04}'
print(bike_code_str)
```

- Section 14.3 in the book: Format operator % (old string formatting)
- New string formatting (Python>3.6): f-strings
  - Prefixed with 'f' or 'F'
  - Replacement fields delimited by curly braces , evaluated at run time.

Simple plotting

```python
import matplotlib.pyplot as plt
import random

nr_points = 1000
x_values = list(range(nr_points))
y_values = [0]
for i in range(1, nr_points):
    y_values.append(y_values[-1] + random.randint(0, 10) - 5)

plt.plot(x_values, y_values)
plt.show()
```

- Matplotlib: Visualization with Python
- Things get slightly complicated and depend on how Python is used: your personal settings, operative system, Python interpreter... Therefore, Matplotlib has different backends.
- In week 11 we, focus on plotting.
- Goal for now: Generate a simple plot.
- To learn more: `https://matplotlib.org/stable/users/explain/quick_start.html`

Additional code shown during lecture

backslash.py

```python
print('ja\nnej')
print('ABCDEFG\b\b\b\b\b--')
print('ABCDEFG\r---')
```

writing_text.py

```python
f = open('test.txt', 'w')
f.write('I am writing to a file!\n')
f.close()
```

This is an example of writing a file without `with` statement. It is advisable to use `with` statement when writing and reading files.