目　录

致谢

当前文档 《Python 3 Module Examples（英文版）》 由 进击的皇虫 使用 书栈(BookStack.CN) 进行构建，生成于 2018-04-30。

书栈(BookStack.CN) 仅提供文档编写、整理、归类等功能，以及对文档内容的生成和导出工具。

文档内容由网友们编写和整理，书栈(BookStack.CN) 难以确认文档内容知识点是否错漏。如果您在阅读文档获取知识的时候，发现文档内容有不恰当的地方，请向我们反馈，让我们共同携手，将知识准确、高效且有效地传递给每一个人。

同时，如果您在日常生活、工作和学习中遇到有价值有营养的知识文档，欢迎分享到 书栈(BookStack.CN) ，为知识的传承献上您的一份力量！

如果当前文档生成时间太久，请到 书栈(BookStack.CN) 获取最新的文档，以跟上知识更新换代的步伐。

文档地址：http://www.bookstack.cn/books/Python3_Module_Examples

书栈官网：http://www.bookstack.cn

书栈开源：https://github.com/TruthHun

分享，让知识传承更久远！ 感谢知识的创造者，感谢知识的分享者，也感谢每一位阅读到此处的读者，因为我们都将成为知识的传承者。

Introduction

Python 3 Module Examples

Sources of this document can be found on https://github.com/krother/Python3_Module_Examples

Purpose of this e-book

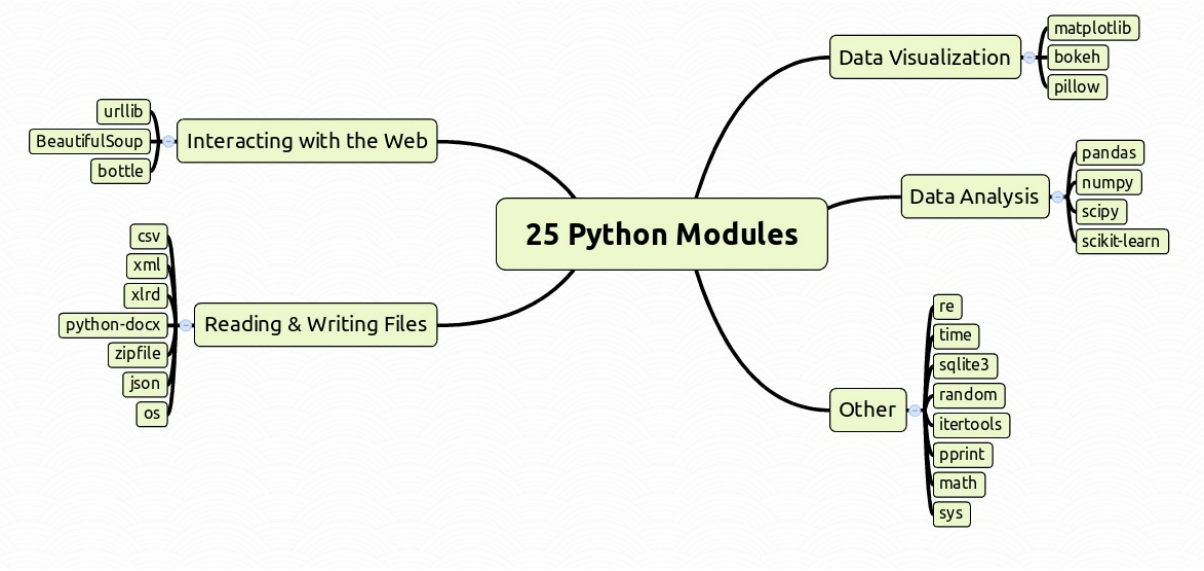
This e-book contains my favourite Python modules. Every module comes with a *brief description* and a *code example*.

This document is for you if:

- you know a little bit of Python already
- you would like to know what Python modules are there
- you find the amount of Python modules overwhelming
- you find the full documentation too heavy to begin
- you would like to try a few simple examples

Have fun getting to know Python better!

Overview


25 Python Modules
Interacting with the Web
urllib
BeautifulSoup
bottle
Reading & Writing Files
csv
xml
xlrd
python-docx
zipfile
json
os
Data Visualization
matplotlib
bokeh
pillow
Data Analysis
pandas
numpy
scipy
scikit-learn
Other
re
time
sqlite3
random
itertools
pprint
math
sys

Reading and writing files

- `csv - reading and writing CSV files`
- `json - working with JSON files`
- `xml - parsing XML files`
- `zipfile - reading and writing .zip files`

csv - reading and writing CSV files

csv

What it is good for?

Read and write *comma-separated-value (CSV)* files.

The `csv` module reads and writes nested lists from/to CSV files. You can set a *field delimiter*, *quote character* and *line terminator* character. Note that when reading a line, all columns are in string format.

Installed with Python by default

yes

Example

Write a table with two rows to a CSV file:

```
import csv

data = [["first", 1, 234],
        ["second", 5, 678]]

outfile = open('example.csv', 'w')
writer = csv.writer(outfile, delimiter=';', quotechar='"')
writer.writerows(data)
outfile.close()
```

Read the file again:

```
for row in csv.reader(open('example.csv'), delimiter=';'):
    print(row)

```

```
['first', '1', '234']
['second', '5', '678']
```

Where to learn more?

https://docs.python.org/3/library/csv.html

json - working with JSON files

json

What it is good for?

Convert Python dictionaries to JSON and back.

The *JavaScript Object Notation (JSON)* is frequently used to send structured data around the web or store it painlessly in files. The `json` modules utilizes the similarity of the JSON format to Python dictionaries.

Installed with Python by default

yes

Example

Convert a dictionary to a JSON-formatted string:

```
1. import json
2.
3. data = {'first': 1, 'second': 'two', 'third': [3,4,5]}
4. jj = json.dumps(data)
5. print(jj)
6. '{"second": "two", "first": 1, "third": [3, 4, 5]}'
```

Convert JSON string back to a Python dictionary:

```
1. d = json.loads(jj)
2. print(d)
3.
4. {'second': 'two', 'first': 1, 'third': [3, 4, 5]}
```

Where to learn more?

https://docs.python.org/3/library/json.html

xml - parsing XML files

xml

What it is good for?

Parse XML files.

The `xml` module contains several XML parsers. They produce a tree of DOM objects for each tag that can be searched and allow access to attributes.

Installed with Python by default

yes

Example

Sample XML data:

```
<?xml version="1.0" encoding="UTF-8" ?>
<actor_list>
<actor name="Hamlet">the Prince of Denmark</actor>
<actor name="Polonius">Ophelias father</actor>
</actor_list>
```

Read an XML file and extract content from tags:

```
from xml.dom.minidom import parse

document = parse('hamlet.xml')

actors = document.getElementsByTagName("actor")
for act in actors:
    name = act.getAttribute('name')
    for node in act.childNodes:
```

```
        if node.nodeType == node.TEXT_NODE:
            print("{} - {}".format(name, node.data))

Hamlet - the Prince of Denmark
Polonius - Ophelias father
```

Where to learn more?

https://docs.python.org/3/library/xml.html

zipfile - reading and writing .zip files

zipfile

What it is good for?

Read and write `.zip` files.

You can add both existing files and strings to a zip file. If you are adding strings you need to specify the file name it is written to. When you extract files to a folder, the output folder is automatically created.

Installed with Python by default

yes

Example

Create a new zip archive and add files to it:

```
import zipfile
z = zipfile.ZipFile('archive.zip', 'w')
z.write('myfile.txt')                  # has to exist
z.writestr('test.txt', 'Hello World')   # new
z.close()
```

List contents of the newly created zip file:

```
z = zipfile.ZipFile('archive.zip')
print(z.namelist())
```

Extract a file to a new folder:

```
print(z.extract('test.txt', 'myfolder'))
```

```
2. z.close()
```

Where to learn more?

docs.python.org/3/library/zipfile.html

Data analysis

- numpy - fast matrix calculations
- pandas - comfortable handling of tables
- scipy - scientific calculations
- scikit-learn - Machine Learning

numpy - fast matrix calculations

numpy

What it is good for?

numpy makes it easy to work with matrices in Python.

Because it is implemented in C, `numpy` accelerates many calculations. It is also type-safe - all elements of a matrix have the same type. Many of the most powerful Python libraries like `pandas`, `scikit-learn` and `PILLOW` have been built on top of numpy.

Pre-installed on Anaconda?

yes

How to install it?

```
pip install numpy
```

Example

Creating a 4 x 2 matrix and adding 10 to each element

```
import numpy as np

vector = np.array([[0, 1, 2, 3], [4, 5, 6, 7]])
print(vector + 10)

[[10 11 12 13]
 [14 15 16 17]]

```

```
print(vector.shape)

(2, 4)
```

Where to learn more?

http://www.numpy.org/

pandas - comfortable handling of tables

pandas

What it is good for?

Analyze tabular data.

`pandas` is an extremely powerful library to analyze, combine and manipulate data in many thinkable (and some unthinkable) ways. The tables called *DataFrame* have many similarities to R. DataFrames have an index column and functions for plotting and reading from CSV or Excel files are included by default. Pandas uses `numpy` under the hood.

Installed with Python by default

no

Installed with Anaconda

yes

How to install it?

```
1. pip install pandas
```

Example

Create a table with characters and numbers.:

```
1. import pandas as pd
```

```
2.
3. hamlet = [['Hamlet', 1.76], ['Polonius', 1.52], ['Ophelia', 1.83], ['Claudius', 1.95]]
4. df = pd.DataFrame(data = hamlet, columns = ['name', 'size'])
5. print(df)
6.
7.        name  size
8. 0    Hamlet  1.76
9. 1  Polonius  1.52
10. 2   Ophelia  1.83
11. 3  Claudius  1.95
```

Sorted lines by name, filter by minimum size, print first two values and write a CSV file:

```
1. sorted = df.sort_values(by='name', ascending=False)
2. tall = sorted[sorted['size'] > 1.70]
3. print(tall.head(2))
4.
5.        name  size
6. 3  Claudius  1.95
7. 2   Ophelia  1.83
8.
9. df.to_csv('hamlet.csv', index=False, header=True)
```

Where to learn more?

http://pandas.pydata.org/

scipy - scientific calculations

scipy

What it is good for?

Scientific calculations.

scipy is a Python library for fitting functions and other kinds of numerical
analyses. You find functions for signal processing, Fourier Transform, generating random datasets and many more. Scipy uses `numpy` and `matplotlib` .

Installed with Python by default

no

Installed with Anaconda

yes

How to install it?

```
1. pip install scipy
```

Example

Define a square function; create noisy X/Y data using `numpy` :

```
1. def func(x, a, b):
```

```
    return a * x**2 + b

import numpy as np
x = np.linspace(-10, 10, 100)
y = func(x, 1, 5)
ynoise = y + 20 * np.random.laplace(size=len(x))
```

Fit the parameters of the function with noisy data:

```
from scipy.optimize import curve_fit
params, pcov = curve_fit(func, x , ynoise)
yfit = func(x, params[0], params[1])
```

Plot the outcome:

```
import matplotlib.pyplot as plt
fig = plt.figure
plt.plot(x, yfit, "k-")
plt.plot(x, ynoise, "bx")
plt.savefig('fit.png')
```

Where to learn more?

http://scipy.org/

scikit-learn - Machine Learning

scikit-learn

What it is good for?

Machine Learning.

The `scikit-learn` library contains regression and classification methods ranging from simple linear regression over logistic regression, Support Vector Machines and multiple clustering methods to sophisticated things like Random Forests. In addition, rich functions for validating predictions exist.

Installed with Python by default

no

Installed with Anaconda

yes

How to install it?

```
1. pip install scikit-learn
```

Example

Load one of the example datasets and divide it into a training and test set:

```
from sklearn import svm, datasets, cross_validation

iris = datasets.load_iris()
X_train, X_test, Y_train, Y_test = \
    cross_validation.train_test_split(iris.data, iris.target, \
    test_size=0.4, random_state=True)
```

Fit a Support Vector Machine model and test it:

```
svc = svm.SVC(kernel='linear', C=1.0, probability=True).fit(X_train, Y_train)
print(svc.score(X_test, Y_test))

0.983333333333
```

Do a five-fold cross-validation:

```
print(accuracy = cross_validation.cross_val_score(svc, X, Y, cv=5, scoring='accuracy'))

[ 0.96666667  1.          0.96666667  0.96666667  1.        ]
```

Where to learn more?

http://scipy.org/

Data visualisation

- `matplotlib - plotting diagrams`
- `pillow - image manipulation`

matplotlib - plotting diagrams

matplotlib

What it is good for?

Plotting diagrams.

`matplotlib` is capable of producing static images of all common types of diagrams in print quality: line plots, scatter plots, bar charts, pie charts, histograms, heat maps etc.

Installed with Anaconda

yes

How to install it:

```
pip install matplotlib
```

Example

Plot a square function:

```
from pylab import *

x = list(range(-10, 10))
y = [xval**2 for xval in x]
figure()
plot(x, y, 'bo') # blue circles
title('square function')
xlabel('x')
ylabel('$x^2$')
```

```
savefig('plot.png')
```

Where to learn more?

http://matplotlib.org/

pillow - image manipulation

PILLOW

What it is good for?

Image manipulation.

`PILLOW` is the inofficial successor to the **Python Imaging Library (** `PIL` **)**. It facilitates creating, cutting and applying various filters to pixel-based images.

Installed with Python by default

no

Installed with Anaconda

yes

How to install it?

```
pip install pillow
```

Example

Convert all `.png` images in the directory to half their size.

```
from PIL import Image
import os

```

```
4. for filename in os.listdir('.'):
5.     if filename.endswith('.png'):
6.         im = Image.open(filename)
7.         x = im.size[0] // 2
8.         y = im.size[1] // 2
9.         small = im.resize((x, y))
10.        small.save('sm_' + filename)
```

Where to learn more?

https://pillow.readthedocs.org

Interacting with the web

- `requests - improved retrieving web pages`
- `bs4 - parsing HTML pages`
- `paramiko - executing commands via SSH`

requests - improved retrieving web pages

requests

What it is good for?

Retrieving webpages.

`requests` sends HTTP requests to web pages and allows you to read their content. Most standard tasks are a lot easier compared to the standard module `urllib`. `requests` can sending data to web forms via HTTP GET and POST, submit files and manage cookies.

Installed with Python by default

no

Installed with Anaconda

yes

Example

Read the homepage of the author.

```
1. import requests
2.
3. r = requests.get('http://www.academis.eu')
4. print(r.text)
```

Search scientific articles on PubMed:

```
1. url = "http://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi"
```

```
2. param_dict = {'db':'pubmed', 'term':'escherichia', 'rettype':'uilist'}
3.
4. r = requests.get(url, params=param_dict)
5. print(r.text)
```

Where to learn more?

http://docs.python-requests.org/en/latest/index.html

bs4 - parsing HTML pages

BeautifulSoup

What it is good for?

Parsing HTML pages.

Beautiful soup is much, much easier to use than the default HTML parser installed with Python.

Installed with Python by default

no

Installed with Anaconda

no

How to install it?

```
pip install bs4
```

Example

Parsing list items out of a HTML document:

```
from bs4 import BeautifulSoup

html = """<html><head></head><body>
<h1>Hamlet</h1>
```

```
<ul class="cast">
  <li>Hamlet</li>
  <li>Polonius</li>
  <li>Ophelia</li>
  <li>Claudius</li>
</ul>
</body></html>"""

soup = BeautifulSoup(html, "lxml")

for ul in soup.find_all('ul'):
    if "cast" in ul.get('class', []):
        for item in ul.find_all('li'):
            print(item.get_text(), end=", ")
```

Where to learn more?

http://www.crummy.com/software/BeautifulSoup/bs4/doc/

paramiko - executing commands via SSH

paramiko

What it is good for?

Executing commands via SSH.

`paramiko` allows you to execute Unix commands on another machine by logging in via SSH.
The `fabric` module gives you a comfortable interface built on top of `paramiko`.

Installed with Python by default

no

Installed with Anaconda

yes

Example

List the directory on a remote machine.

```
1. from paramiko import SSHClient
2. client = SSHClient()
3. client.load_system_host_keys()
4. client.connect('ssh.example.com', username="username", password="password")
5. stdin, stdout, stderr = client.exec_command('ls -l')
```

WARNING

Do not hard-code your password inside your Python files. Better read it from a configuration file or environment variable. That way it is less likely that it gets into wrong hands accidentally

Where to learn more?

www.paramiko.org/

Essential standard modules

- `math - mathematical functions`
- `os - working with files and directories`
- `random - generating random numbers`
- `re - pattern matching in text`
- `time - working with dates and times`
- `sqlite3 - a simple SQL database`
- `sys - settings of the Python interpreter`
- `itertools - working with lists and generators`

math - mathematical functions

math

What it is good for?

math contains mathematical functions similar to a scientific calculator.

In the module, you find various trigonometric and exponential functions. In addition the two constants `pi` and `e` are included.

Installed with Python by default

yes

Example

Calculating a **square root**, a **sine**, an **exponential function** and a **logarithm**.

```
1. import math
2.
3. print(math.sqrt(49))
4.
5. print(math.sin(math.pi / 2))
6.
7. print(math.exp(1.0) == math.e)
8.
9. print(math.log(256,2))
```

Where to learn more?

https://docs.python.org/3/library/math.html

os - working with files and directories

os

What it is good for?

Working with files and directories.

The `os` module provides an easy way to interact with files, directories and other parts of your operating system. It contains many functions to list, change, copy, remove and examine files and directories.

Installed with Python by default

yes

Example

Change directory and list its contents:

```
import os

os.chdir('/home/krother/python_modules/')
os.listdir('.')

['sys.md', 'os.md', 'csv.md', 're.md', 'random.md',
'pprint.md', 'numpy.md', 'time.md', 'itertools.md',
'json.md', 'template.md', 'math.md', 'urllib.md']
```

Check whether a file exists:

```
os.path.exists('os.md')
```

Copy a file and remove it afterwards:

```
1. os.system('cp os.md copy.md')
2. os.remove('copy.md')
```

Where to learn more?

https://docs.python.org/3/library/os.html

random - generating random numbers

random

What it is good for?

Generate random numbers.

`random` contains generators for the most common distributions.

Installed with Python by default

yes

Example

Creating random integers

One most wanted function is to create random integers in a given range:

```
1. dice = random.randint(1,6)
```

Creating random floats

The `random()` functin generates float numbers between 0 and 1:

```
1. import random
```

```
print random.random()
```

Generate random numbers from a few distributions.

```
import random

random.randint(1,6)

random.random()

random.gauss(0.0, 1.0)
```

Shuffle a list

```
data = [1, 2, 3, 4]
random.shuffle(data)
```

Creating random lists

Random combinations of elements with repetition:

```
from random import choice

bases = ['A','C','G','T']
dna = [choice(bases) for i in range(20)]
print ''.join(dna)
```

When elements are to be picked without repetition, you would use, the `sample` function:

```
from random import sample

flavors = ['vanilla','banana','mint']
icecream = sample(flavors, 2)
```

Where to learn more?

https://docs.python.org/3/library/random.html

re - pattern matching in text

re

What it is good for?

Pattern matching in text.

The `re` module implements **Regular Expression**, a powerful syntax for searching patterns in text. Regular Expressions are available in most programming languages. You need to learn some special characters to build your own patterns.

Installed with Python by default

yes

Example

```
1. import re
2.
3. text = "the quick brown fox jumps over the lazy dog"
```

Search for `o` and show adjacent characters:

```
1. re.findall(".o.", text)
2. print(re.findall(".o.", text))
3.
4. ['row', 'fox', ' ov', 'dog']
```

Search for three-letter words enclosed by whitespace:

```
print(re.findall("\s(\wo\w)\s*", text))

['fox', 'dog']
```

Substitute any of `dflj` by a `w`:

```
print(re.sub("[dflj]", "w", text))

'the quick brown wox wumps over the wazy wog'
```

Check if `jumps` or `swims` occurs and return details:

```
print(re.search('jumps|swims', text))

<_sre.SRE_Match object; span=(20, 25), match='jumps'>
```

Where to learn more?

Online Games

- regexone.com/ - Learn regular expressions by simple, interactive examples. Great place to start.
- Regex crossword - Train proper use of single characters, wildcards and square brackets. Easy.
- Regex Practice Quiz 1 - exercises to try offline.
- Regex golf - Advanced exercises. Match as many phrases with as few key strokes as possible.

Reference

- Python Regex HOWTO
- docs.python.org/3/library/re.html
- Quick Reference - a reference sheet for looking up metacharacters. Uses the **Python syntax**.

Online Regex Testers

- Regex 101 - Shows matched text with explanation.
- Pythex - RegEx tester using the Python `re` module.
- regexpal - Uses JavaScript to highlight matches.

time - working with dates and times

time

What it is good for?

Simple handling of times and dates.

The functions in `time` return the time and date in a structured format that can be formated to custom strings.

Installed with Python by default

yes

Example

The `time` module offers functions for getting the current time and date.

```
1. import time
2.
3. print(time.asctime())
4.
5. print(time.strftime('%a %d.%m.', time.localtime()))
```

Wait for two seconds:

```
1. time.sleep(2)
```

The `datetime` module also helps to format dates:

```
1. date = datetime.date(2015, 12, 24)
2. date.strftime("%d.%m.%Y")
```

Dates can be converted to integer numbers:

```
1. date = datetime.date(2015, 12, 24)
2. number = date.toordinal()
```

and back

```
1. datetime.date.fromordinal(7)
```

Where to learn more?

- https://docs.python.org/3/library/time.html
- https://docs.python.org/3/library/time.html

sqlite3 - a simple SQL database

sqlite3

What it is good for?

Create and use a *SQLite* database.

SQLite databases are stored in files. For using the `sqlite3` module you don't need to install or set up anything. SQLite is sufficient only for small SQL databases, but Python modules for bigger databases look very similar.

Installed with Python by default?

yes

Example

Create a new database:

```
import sqlite3

DB_SETUP = '''
  CREATE TABLE IF NOT EXISTS person (
    id INTEGER,
    name VARCHAR(32),
    description TEXT
  );'''
db = sqlite3.connect('hamlet.db')
db.executescript(DB_SETUP)
```

Insert data:

```
query = 'INSERT INTO person VALUES (?,?,?)'
```

```
db.execute(query, (1, "Hamlet", "the prince of Denkmark"))
db.execute(query, (2, "Polonius", "Ophelias father"))
db.commit()
```

Submit a query:

```
query = '''SELECT name, description FROM person'''
result = db.execute(query)
print(list(result))

db.close()
```

Where to learn more?

docs.python.org/3.5/library/sqlite3.html

sys - settings of the Python interpreter

sys

What it is good for?

Settings of the Python interpreter itself.

The `sys` module provides an access point to the Python environment. You find there command line arguments, the import path settings, the standard input, output and error stream and many more.

Installed with Python by default

yes

Example

Command line parameters used when calling Python:

```
1. import sys
2. print(sys.argv)
```

Version of the Python interpreter:

```
1. print(sys.version)
```

Directories in which Python looks for modules:

```
1. print(sys.path)
```

Exit Python altogether:

```
1. sys.exit()
```

Where to learn more?

https://docs.python.org/3/library/sys.html

itertools - working with lists and generators

itertools

What it is good for?

Functions to work with lists and iterators.

Most functions in this module return *iterators*, so you can use their result once or convert it to a list.

Installed with Python by default

yes

Example

Concatenate a list:

```
import itertools
ch = itertools.chain([1,2],[3,4])
print(list(ch))

[1, 2, 3, 4]

print(list(itertools.repeat([1,2], 3)))

[[1, 2], [1, 2], [1, 2]]
```

Permutations and combinations of list elements:

```
p = itertools.permutations([1,2,3])
print(list(p))

[(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)]
```

```
5.
6. c = itertools.combinations([1,2,3], 2)
7. print(list(c))
8.
9. [(1, 2), (1, 3), (2, 3)]
```

Where to learn more?

https://docs.python.org/3/library/itertools.html

Challenges

Challenges

Grafik

Challenge	Kurzbeschreibung	Rating
Spirale	Zeichne eine Spirale	sehr leicht
Postcards	Zeichne eine Postkarte	leicht
Thumbnails	Erzeuge Thumbnails von Bildern	mittel
Rekursive Grafik	Zeichne rekursive Bilder	mittel
Film	Drehe einen kurzen Film	schwer

Daten

Challenge	Kurzbeschreibung	Rating
Babynamengenerator	Erzeuge zufällige Babynamen	leicht
Reguläre Ausdrücke	Suche Muster in Text	mittel
Google	Führe eine Google-Suche durch	mittel
Webrecherche	Lade Webseiten herunter	mittel

Webentwicklung

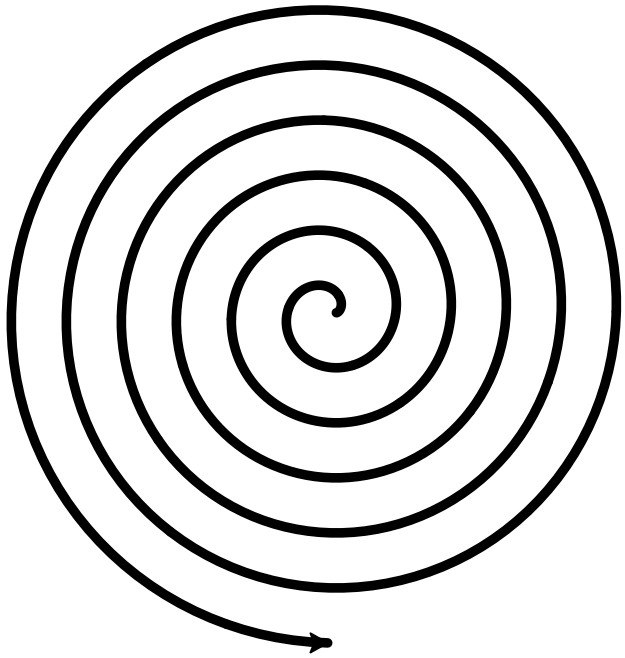
Challenge	Kurzbeschreibung	Rating
Daten anzeigen	Erstelle einen einfachen Bottle-Webserver	mittel
Blog	Erstelle eine Blog-Webseite mit Django	schwer

Spirale

Coding Challenge: Spiral

The Challenge

Write a program, that draws a spiral:



When your program draws a spiral with at least 3 loops, you have mastered this challenge.

What you can practise in this coding challenge

- loops

- The **Pillow** library in Python
- The **turtle** module in Python

Hints

- It is sufficient to draw the spiral as a series of short lines
- Where is it easier to start (inside or outside)?
- Both the Python modules `Pillow` and `turtle` are up to the task

Getting started

If you have no idea where to start, try the following Python script:

```
1. from turtle import forward, left
2. forward(50)
3. left(90)
4. forward(50)
```

Optional goals

- the line width grows thicker from the inside to the outside
- there is a color gradient along the spiral

Postkarte

Create a Postcard

Write a program that creates a postcard for the city of your choice.

1. Install Pillow

```
1. `pip install pillow`
```

(it is already installed with Anaconda)

2. Learn to know PIL

Exercise 2.1

Run the program `example01.py`. What does it do?

Exercise 2.2

Change the numbers in the program, so that you create a square-shaped image.

3. Drawing shapes

Exercise 3.1

Run the program `example02.py`. What does it do?

Exercise 3.2

Add a broad horizontal bar to an image of your favourite city.

Exercise 3.3

for fast students

Draw a 8-pointed star on an image of your favourite city.

Hint: You can compose such a star from squares, triangles or polygons.

4. Drawing text

Exercise 4.1

Run the program `example03.py`. What does it do?

Exercise 4.2

Write the text *“Welcome to (your city)”* to the shape from exercise 3.2.

5. Composing images

Exercise 5.1

Run the program `example04.py`. What does it do?

Exercise 5.2

Create a postcard composed of four smaller pictures, the horizontal bar and some text on it.

6. Applying filters

Exercise 6.1

Run the program `example05.py`. What does it do?

Exercise 6.2

Be creative!

License

Thumbnails

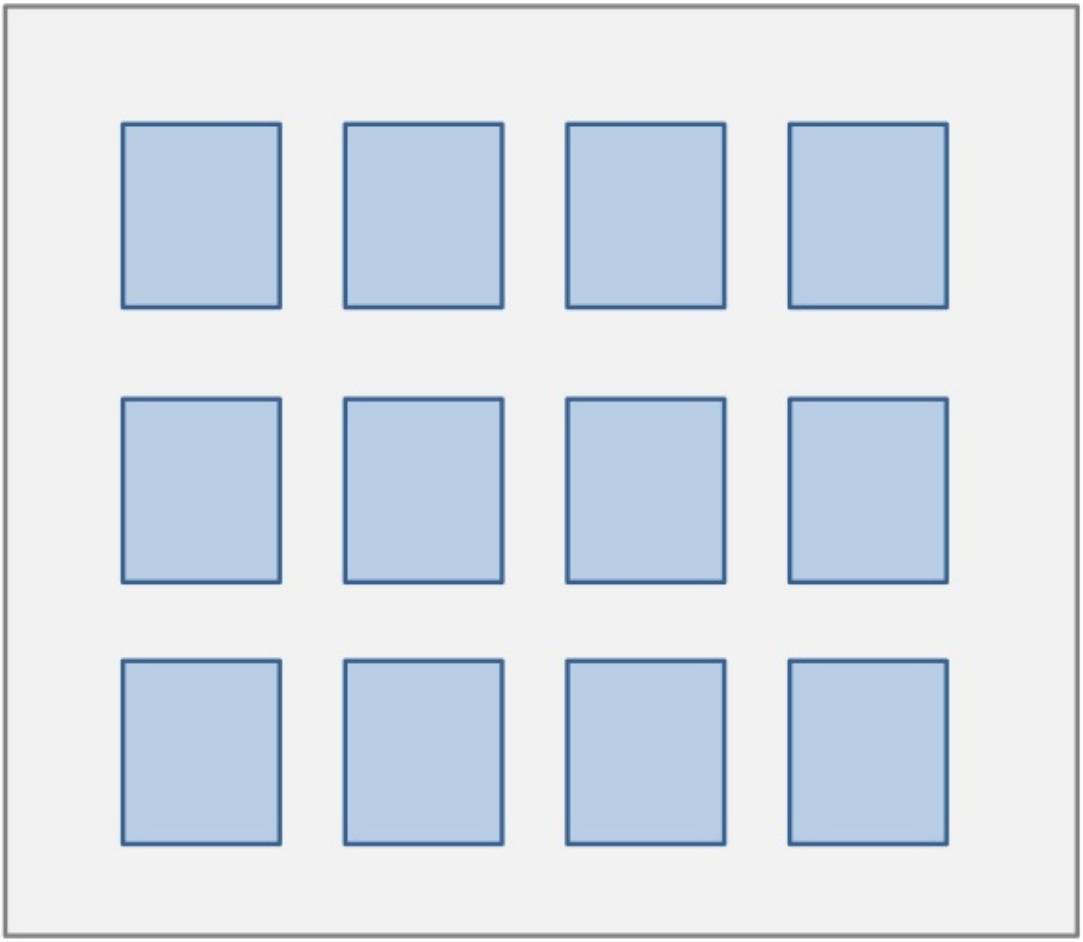
Thumbnail Generator

Problem description

“I have a big collection of photographs, and would like to search them. I have written a up keywords for each photograph in text files, e.g. ‘flower’, ‘beach’, ‘mountain’* etc. Now I need a program that can find all photographs for a given keyword.

The program should create a big picture that shows thumbnail images 100 pixels wide, like in the image below.”*

Rick Closeview , hobby photographer



Instructions

Implement a program according to the problem description. The following steps may be helpful:

1. Planning

Dissect the problem into at least three separate sub-problems. Write them down on paper (cards or a single sheet). Try to be as precise as possible. Answer resulting questions.

2. Divide the program into functions

Identify key functions your program needs to have. Decide what should be the input and output for each of them. Write them up as well.

3. Get the Graphics

The file `sample_photos.zip` contains a few example photographs. It also contains a file `pics.txt` in each subfolder. This is the input information for the program.

4. Explore Python libraries

You might want to take a closer look at:

- `os` for file handling
- `Pillow` for image manipulation

5. Write the program

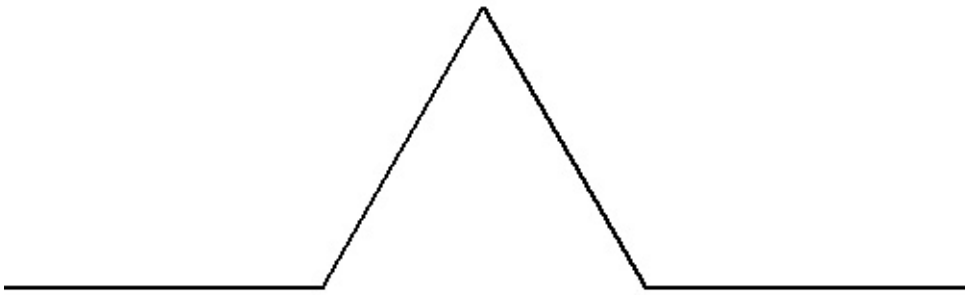
After these steps, start implementing.

Rekursive Grafik

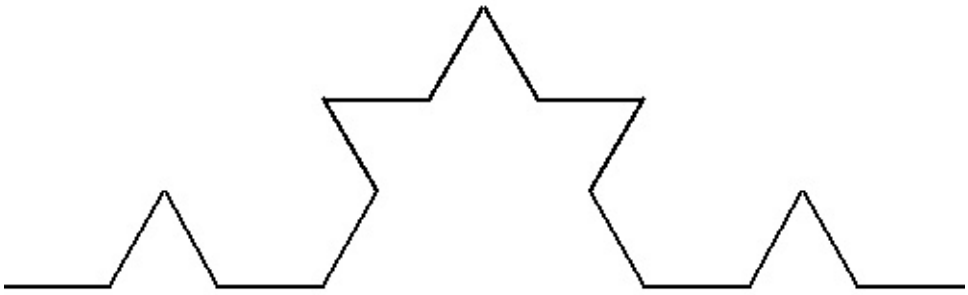
Coding Challenge: The Recursive Snowflake

The Challenge

Write a program, that for a parameter `n=1` creates a basic shape out of four straight lines:



When you increase `n`, each line gets replaced by a smaller version of the basic shape. For, instance with `n=2` you get:



And with `n=5`:

When your program is able to produce a picture for `0 < n < 6`, you have mastered this challenge.

What you can exercise in this coding challenge:

- Recursive algorithms
- The **Pillow** library in Python
- The **turtle** module in Python

Film

Create a Movie

Material:

- Your favourite image(s)
- Python + Pillow
- `MEncoder` or another program to create movies from a set of images.
- see the Flower Assembly Movie to get an idea how the result could look like.

Task

Write a program using the PIL library that creates a set of `.png` images. Generate the images and generate a movie from that using a non-Python tool (e.g. MEncoder). Remember that a movie typically has 25 frames per second.

Hints

- start with a very simple movie to make sure the assembly is working
- the Flower Assembly Movie was created by slowly disassembling the picture and playing the frames backwards

Creating a movie from frames on Windows

MEncoder requires files with the frames to have names like frame_000123.png so that they have the right order in the final movie.

1. Collect all frame images in one directory
2. copy Mencoder into that directory.
3. open a console (`Start -> Execute -> cmd`)
4. switch with `cd <directory_name>` to that directory
5. type

   ```
   mencoder “mf://*.png” -mf fps=25 -o output.avi -ovc lavc -lavcopts
   vcodec=mpeg4
   ```

Babynamengenerator

Babynamengenerator

Programmiere einen Babynamengenerator für unentschlossene Eltern.

Optionale Ziele:

- Das Programm gibt zufällig einen Namen aus einer vorgegebenen Liste aus
- Das Programm gibt zufällig einen Namen aus dem US-Melderegister aus
- Der Benutzer kann wahlweise Jungen- oder Mädchennamen auswählen
- Das Programm macht 10 Vorschläge
- Verwende als Liste möglicher Vornamen eine Datei aus dem US-Datensatz

Reguläre Ausdrücke

Reguläre Ausdrücke

Aufgabe 1

Finde Wörter mit `F` im Text. Führe dazu folgendes Beispiel aus:

```
1. import re
2.
3. text = "Es war einmal ein Ferkel, das hatte eine Flöte"
4. found = re.findall('(F\w+)[^\w]', text, re.IGNORECASE)
5. print(found)
```

Aufgabe 2

Was haben diese vier Bilder gemeinsam?

Bildquellen (links oben nach rechts unten):

- *By Source (WP:NFCC#4), Fair use*
- *Die Autorenschaft wurde nicht in einer maschinell lesbaren Form angegeben. Es wird Swarve~commonswiki als Autor angenommen, CC BY-SA 3.0*
- *Gaël Marziou from Grenoble, France - IMG_6266_DXO, CC BY 2.0*
- *Derfu, CC BY-SA 3.0*

Führe folgendes Codebeispiel aus:

```
import re
text = input("Was ist auf einem der Bilder zu sehen? ")
if re.search(r"^(\w+)i(\w+)[- ]\1o\2", text):
    print("stimmt!")
```

Aufgabe 3

Besuche die Seite www.regexone.com und führe einige der Übungen aus.

Aufgabe 4

Schreibe ein Programm, das E-Mail-Adressen in Text erkennt.

Verwende die Funktion `re.findall` und erstelle ein entsprechendes Suchmuster.

Auf Regex101 kannst Du den regulären Ausdruck testen.

Google

Google

Überprüfe mit einem Python-Programm, ob die Suchmaschine Google funktioniert.

Aufgabe 1

Lade die Webseite `www.google.de` mit Python herunter.

Aufgabe 2

Extrahiere den Titel der Seite aus dem HTML-Dokument und gib ihn auf dem Bildschirm aus.

Aufgabe 3

Führe eine Google-Suche aus Python durch. Recherchiere in der Dokumentation zum Modul `requests` danach, wie Du ein Formular über ein POST-request ausfüllen kannst.

Gib die Suchergebnisse auf dem Bildschirm aus.

Hinweise:

- Das Modul `requests` hilft beim Vorbereiten der Suchanfrage.
- Das Formular verwendet HTTP POST.
- Es gibt im Formular mindestens ein *‘hidden’*-Feld, das Du angeben mußt.

Webrecherche

Herunterladen von HTML-Seiten

In den folgenden Aufgaben laden wir Daten von einer statischen Webseite oder einem RSS-Feed herunter.

Aufgabe 1

Suche Dir eine der folgenden Seiten aus:

RSS-Feeds des Parlaments

- `https://www.parlament-berlin.de/de/Service/RSS-Feeds`

Biologische Datenbanken

- `http://ncbi.nlm.nih.gov`

Presse

- `http://www.reuters.com/tools/rss`

Demographie

- `http://www.gapminder.org`

Bücher

- `http://www.gutenberg.org`

Aufgabe 2

Zeige den Quelltext der Seite/des Feeds an. Versuche, markante Elemente zu finden, anhand derer Du den Titel, Links oder Inhalte erkennst.

Aufgabe 3

Extrahiere den Titel der Seite aus dem HTML-Dokument/Feed und gib ihn auf dem Bildschirm aus. Verwende dazu die normalen Stringfunktionen.

Aufgabe 4

Extrahiere Links oder andere Daten aus der Seite. Entwickle dazu eine Strategie, die entweder die Methoden von Strings, reguläre Ausdrücke oder einen fertigen Parser verwendet.

Aufgabe 5

Lade 10 der verlinkten Seiten herunter und speichere sie ab.

Nützliche Module

- `requests` zum Herunterladen von Webseiten
- `BeautifulSoup` zum Parsen von HTML-Seiten
- `scrapy` für beides zusammen

Webseite mit Strassennamen

Einen Webserver mit Flask bauen

In diesem Tutorial kannst Du eine eigene dynamische HTML-Seite erstellen und auf einem Webserver zum Laufen bringen. Wir verwenden dazu das Python-Modul `flask` .

`flask` ist vor allem für kleinere Webseiten geeignet. Eine Alternative ist `Django` , zu dem es das ausgezeichnete DjangoGirls Tutorial gibt.

1. Flask installieren

Installiere das Python-Modul `flask` mit `pip` :

```
1. pip install flask
```

2. Eine minimale Webseite starten

Erstelle ein Python-Programm `server.py`, und baue das Hello-World-Beispiel aus der Flask-Dokumentation nach.

Füge folgende Zeile zum Programm hinzu (funktioniert mit Anaconda besser als die in der Dokumentation angegebene Methode):

```
1. app.run()
```

Starte das Programm. Finde heraus, unter welcher HTTP-Adresse der Server läuft. Setze diese Adresse im Browser ein und prüfe, ob Dein Server erreichbar ist.

Tip:

Wir werden den Server im Verlauf des Tutorials noch sehr oft starten müssen. Stelle sicher, dass Du das Programm aus Deinem Editor oder von der Kommandozeile leicht anhalten und neu starten kannst.

3. HTML-Code einbinden

Die Flask-Funktionen können HTML-Code zurückgeben. Dies geht z.B. als String mit dreifachen Anführungszeichen. Folgender HTML-Code erzeugt eine Überschrift:

```
1. <h1>Unser Strassenverzeichnis</h1>
```

Mehr zu HTML-Elementen erfährst Du auf Selfhtml.org.

Baue die Überschrift in die Rückgabe der Python-Funktion ein. Starte den Server neu. Lade die Webseite im Browser neu. Prüfe, ob Deine Überschrift auf der Seite erscheint.

4. Eine Unterseite hinzufügen

Schreibe eine zweite Python-Funktion, die den Namen Deiner Strasse ausgibt. Verwende den Dekorator `@app.route`, um die Seite auf die URL `/zuhause` zu legen.

Starte den Server neu und rufe beide Unterseiten im Browser auf (`/` und `/zuhause`).

Hilfe:

In der Dokumentation unter **Routing**

5. Hyperlinks

Erstelle auf der Startseite einen Hyperlink, der auf die Unterseite verweist.

Dazu muss die Funktion `hello()` folgenden HTML-Code zurückgeben:

```
1. <a href="/zuhause">Meine Strasse anzeigen</a>
```

Starte den Server neu und prüfe, ob der Link funktioniert.

6. Ein Template hinzufügen

Es wird schnell beschwerlich, eine ganze HTML-Seite in unser Python-Skript zu kleben. Es ist besser, den HTML-Code in **Templates** zu speichern und diese einzubinden.

Erstelle eine Datei `templates/hello.html` , in die Du den folgenden HTML-Code einfügst:

```
1. <html>
2.   <head><title>Unser Strassenverzeichnis</title></head>
3.   <body>
4.     <h1>Unser Strassenverzeichnis</h1>
5.     <a href="/zuhause">Meine Strasse anzeigen</a>
6.   </body>
7. </html>
```

Binde das Template entsprechend dem Abschnitt Rendering Templates aus der Flask-Dokumentation hinzu.

Starte dann den Server neu und stelle sicher, dass der Inhalt des Templates angezeigt wird (**achte auf die Titelzeile des Browserfensters!**).

7. Variablen ins Template einbinden

Wir können aus den Python-Funktionen Daten an ein Template schicken, indem wir ein Dictionary zurückgeben:

```
1. return {'text': "Hallo" }
```

In den HTML-Templates kannst Du diese Variablen folgendermassen ansprechen:

```
1. {{ !text }}
```

8. Dynamische URLs

Du kannst in den URLs Platzhalter verwenden, deren Inhalt als Variable verfügbar ist. Schreibe eine Funktion, die einen Strassennamen in der URL erhält:

```
1. @app.route('/strasse/<strassenname>')
2. def strasse_anzeigen(strassenname):
3.     ...
```

Schreibe diese Funktion fertig und probiere die fertige Seite mit unterschiedlichen Strassennamen aus.

Achtung:

Mit Platzhaltern kann es leicht passieren, dass zwei Funktionen die gleiche URL ansprechen. Dies kann zu interessanten Fehlern führen, weil nicht sofort ersichtlich ist, welche Funktion Flask aufruft.

9. Geodaten bereitstellen

Als Datensatz verwenden wir das Strassennamenverzeichnis der Zeit. Die Daten liegen ursprünglich im Format **GeoJSON** vor. Wir verwenden eine Datei, die in das `.csv` -Format umgewandelt wurde, so dass Du es bequem mit `pandas` verwenden kannst.

Wähle einige Strassen aus der Datei aus und stelle diese auf der Webseite

als Tabelle dar. Lies dazu nach, wie Du eine `for` -Schleife im Template unterbringst.

10. Ein Formular erstellen

Recherchiere, wie Du ein Formular mit Flask erstellst. Baue ein Formular ein, in dem Du einen Strassennamen in ein Eingabefeld eingibst und über einen Knopf das Formular abschickst.

11. Kopf- und Fußzeilen

Du kannst Deine Templates auf mehrere Dateien aufteilen, um Redundanzen zu vermeiden. Erstelle eine Datei für Kopf- und Fusszeilen.

Wie das in Flask geht findest Du unter Template Inheritance.

12. CSS-Stylesheets einbinden

Hier kannst Du Typographie und Farben festlegen.

- Lies unter Selfhtml.org nach, wie CSS-Befehle aussehen.
- Baue einen CSS-Befehl in eines der Templates ein, der die Überschrift einfärbt.
- Erstelle eine Datei `static/style.css`, in die Du eine weitere CSS-Anweisung schreibst.
- Binde die CSS-Datei in Dein Template ein (schreibt Dozent an).
- Sage Flask, wo die CSS-Datei zu finden ist. Siehe Static Files

13. Eine Karte mit Folium zeichnen

Stelle die gefundenen Strassen als interaktive Karte dar. Probiere dazu zunächst ein Python-Skript mit dem Modul `folium` aus. Siehe https://github.com/krother/python_showcase/tree/master/map_markers.

14. Bootstrap

“Half the internet is built on Bootstrap”

Bootstrap ist eine Sammlung nützlicher CSS- und JavaScript-Elemente, mit

denen Du schnell eine Typographie hinbekommst, die auch auf Mobilgeräten gut aussieht.

Dazu ist eine Reihe von Schritten nötig:

- Binde die Bootstrap-Dateien in Deine Templates ein (siehe Anleitung)
- Probiere eine Vorlage aus der Dokumentation aus
- Probiere einzelne Elemente aus (unter *“Components”* findest Du einige, bei denen Du leicht siehst, ob sie funktionieren)

Anmerkung:

Oft ist es nicht wünschenswert, Dateien von einer Drittpartei einzubinden. Nachhaltiger ist es, diese als *statische Dateien* auf Deinem eigenen Server abzulegen.

15. Eine SQL-Datenbank verwenden

Es ist natürlich nicht grade effizient, bei jedem Neustart des Servers 900 MB Daten in den Speicher zu laden. Eine bessere Alternative ist eine **Datenbank**. Verwende das Python-Modul `sqlite3`, um eine Datenbank zu erstellen und die gewünschten Strassen aus einer Tabelle abzufragen.

16. Deployment auf einem öffentlichen Server

Lies Dir im Djangogirls-Tutorial durch, wie Du einen Server auf pythonanywhere anlegst. Probiere es aus! Es kostet nichts.

Hinweis:

Es ist sehr zu empfehlen, den Code mit `git` zu veröffentlichen, bevor Du Dich mit dem öffentlichen Server beschäftigst. Dadurch werden viele Einzelheiten deutlich einfacher.

Blog

Blog

Das Projekt besteht darin, eine eigene Webseite zu entwickeln und im Netz zu veröffentlichen. Das Django Girls Tutorial bietet eine ausgezeichnete Anleitung.

Ziel:

Die Webseite ist über einen Browser erreichbar.

Themen:

Server-Programmierung, HTML, Deployment

Python-Bibliotheken:

Django oder Bottle

Other useful libraries

Other useful libraries

General list of Python libraries

Awesome Python is a curated list of Python libraries and tools.

statsmodels

A statistics package

RPy2

A library connecting the R statistics program and Python.

Seaborn

An easier to use, more powerful interface for matplotlib

Browser-based Python

Brython is a JavaScript library that allows to place (limited) Python code on a web page.

PDFminer
PDF2text

Twython

https://apps.twitter.com/
Twitter Search API

Liste von Python-APIslist-of-python-apis/
ProgrammableWeb API-Katalog

scrapy - Python package to collect data from entire websites

matplotlib examples
matplotlib gallery
seaborn
D3.js
D3 gallery
Bokeh
ggplot

StatsModels

urllib

urllib

What it is good for?

Retrieving webpages.

`urllib` sends HTTP requests to web pages and allows you to read their content similar to files. Parameters can be passed as part of the URL like in a browser if the website is using the HTTP GET protocol.
(For forms using(HTTP POST) you need something different (Python example with ~5 lines)

Installed with Python by default

yes

Example

Read the homepage of the author.

```
1. from urllib import request, parse
2. url = 'http://www.academis.eu'
3. req = request.urlopen(url)
4. page = req.read()
5. print(len(page))
```

Where to learn more?

https://docs.python.org/3/library/zipfile.html

xml

xml

What is xml?

`xml` is a Python module for reading XML documents.

The XML documents are parsed to a tree-like structure of Python objects.

Exercises

Exercise 1

Store the following in an XML document:

```
<?xml version="1.0" encoding="UTF-8" ?>
<movieml name="XML example for teaching" author="Kristian Rother">

<movielist name="movies">
<movie id="1" name="Slumdog Millionaire (2008)">
   <rating value="8.0" votes="513804"></rating>
</movie>
<movie id="2" name="Planet of the Apes (1968)">
    <rating value="8.0" votes="119493"></rating>
</movie>
<movie id="3" name="12 Angry Men (1957)">
    <rating value="8.9" votes="323565"></rating>
</movie>
<movie id="4" name="Pulp Fiction (1994)">
    <rating value="8.9" votes="993081"></rating>
</movie>
<movie id="5" name="Schindler's List (1993)">
```

```
    <rating value="8.9" votes="652030"></rating>
</movie>
</movielist>

<movielist name="serials">
<movie id="6" name="Breaking Bad (2008)
    {To'hajiilee (#5.13)}">
    <rating value="9.7" votes="14799"></rating>
</movie>
<movie id="7" name="Game of Thrones (2011)
    {The Laws of Gods and Men (#4.6)}">
    <rating value="9.7" votes="13343"></rating>
</movie>
<movie id="8" name="Game of Thrones (2011)
    {The Lion and the Rose (#4.2)}">
    <rating value="9.7" votes="17564"></rating>
</movie>
<movie id="9" name="Game of Thrones (2011)
    {The Rains of Castamere (#3.9)}">
    <rating value="9.8" votes="27384"></rating>
</movie>
<movie id="10" name="Breaking Bad (2008)
    {Ozymandias (#5.14)}">
    <rating value="10.0" votes="55515"></rating>
</movie>
</movielist>

</movieml>
```

Exercise 2

Run the following program:

```
from xml.dom.minidom import parse

document = parse('movies.xml')

taglist = document.getElementsByTagName("movie")
for tag in taglist:
    mid = tag.getAttribute('id')
    name = tag.getAttribute('name')
    print(mid, name)

    subtags = tag.getElementsByTagName('rating')
    print(subtags)
```

Exercise 3

Calculate the total number of votes.

python-docx

python-docx

What it is good for?

Create, read and write Word documents.

A straightforward library to access documents in MS Word format. Available features include text, labeled sections, pictures and tables.

Installed with Python by default

no

Installed with Anaconda

no

How to install it?

```
pip install python-docx
```

Example

Create a Word document:

```
from docx import Document

d = Document()

```

```
d.add_heading('Hamlet')
d.add_heading('dramatis personae', 2)
d.add_paragraph('Hamlet, the Prince of Denmark')

d.save('hamlet.docx')
```

Read a Word document:

```
document = Document('hamlet.docx')
for para in document.paragraphs:
    print(para.text)
```

Where to learn more?

https://python-docx.readthedocs.org

xlrd

xlrd

What is xlrd?

`xlrd` is a Python library for reading Excel documents.

Installation

Write in the command line:

```
1. `pip install xlrd`
```

Exercises

Exercise 1

Create an Excel table similar to the data in the `.txt` files.

name	number
Jacob	34465
Michael	32025
Matthew	28569

Exercise 2

Execute the following program:

```
import xlrd

workbook = xlrd.open_workbook("example.xls")
sheet_names = workbook.sheet_names()
sheet = workbook.sheet_by_name(sheet_names[0])

for row_idx in range(0, sheet.nrows):
    print ('-'*40)
    print ('Row: %s' % row_idx)
    for col_idx in range(0, sheet.ncols):
        cell_obj = sheet.cell(row_idx, col_idx)
        print ('Column: [%s] cell_obj: [%s]' % (col_idx, cell_obj))
```

Exercise 3

Make the program work and print the data from the document to the screen.

Exercise 4

Calculate the sum of numeric values in the Python program.

xlrd

xlrd

What it is good for?

Read Excel documents.

The `xlrd` module reads a spreadsheet into a hierarchical data structure. On top are sheets, consisting of rows which in turn consist of columns. The corresponding Python module `xlwt` allows you to write Excel spreadsheets. You may also consider the newer `openpyxl` module.

Installed with Python by default?

no

Installed with Anaconda?

yes

How to install it?

```
pip install xlrd
```

Example

```
# Create a simple Excel file 'hamlet.xlsx' to begin with!
import xlrd

workbook = xlrd.open_workbook('hamlet.xlsx')
```

```
sheet_names = workbook.sheet_names()
sheet = workbook.sheet_by_name(sheet_names[0])

for row_idx in range(sheet.nrows):
    for col_idx in range(sheet.ncols):
        cell = sheet.cell(row_idx, col_idx)
        print(cell.value, end="\t")
    print()
```

Where to learn more?

http://www.python-excel.org/

Installing matplotlib for Python3

Installing matplotlib for Python3

1. create virtualenv for python3

```
export VIRTUALENV_PYTHON=/usr/bin/python3
mkvirtualenv plots
setvirtualenvproject ~/.virtualenvs/plots .
```

2. install pip3

```
sudo apt-get python-pip3
```

3. upgrade setuptools

```
pip3 install –upgrade setuptools
```

4. install dev package for libfreetype

```
sudo apt-get install libfreetype6
sudo apt-get install libfreetype6-dev
```

5. install matplotlib

```
pip3 install matplotlib
```

DETAILED STEPS IF STEP 4 FAILS

6. get matplotlib source from github

```
git clone https://github.com/matplotlib/matplotlib.git
```

7. install dependencies via pip3

```
pip3 -r requirements.txt
```

requirements.txt:

freetype-py==1.0.1

matplotlib==1.5.dev1

nose==1.3.6

numpy==1.9.2

pyparsing==2.0.3

python-dateutil==2.4.2

pytz==2015.2

six==1.9.0

8. install matplotlib

```
python setup.py build
python setup.py install
```

9. test

```
python
```

```
import pylab as plt
plt.plot(range(10))
plt.savefig('plot.png')
```

Exercises

Exercises

Exercise 1

Print today’s date and time of day in a custom format.

Regex and Text Processing exercises

Regex and Text Processing exercises

Wörter: Wie oft kommt Löw im Text vor?

Wörter in Tabelle schreiben

Lukas Podolski einlesen

Regex Crossword lösen

Warum manchmal nur ein Land?

Spielergebnisse parsen

Count number of words in a text.

Count most frequent character in a text.

Count which pairs of subsequent words occurs most often.

Calculate the entropy of some data.

Filter certain words out of a text.

pprint

pprint

What it is good for?

Print complex data structures nicely.

The `pprint` module prints lists, tuples, dictionary and other data structures. It tries to fit everything into one line. If this does not work, each entry gets its own line. You can control the line width.

Installed with Python by default?

yes

Example

Pretty-print data using a wide and a narrow width:

```
from pprint import pprint

data = {'second': 'two', 'first': 1, 'third': [3, 4, 5]}
pprint(data)

{'first': 1, 'second': 'two', 'third': [3, 4, 5]}

pprint(data, width=20)

{'first': 1,
 'second': 'two',
 'third': [3,
           4,
           5]}
```

Where to learn more?

https://docs.python.org/3/library/pprint.html

pandas

pandas

What is pandas?

`pandas` is a Python library for efficient handling of tables.

Installation

Write in the command line:

```
1. `pip install pandas`
```

Exercises

Exercise 1

Execute the following program:

```
1. import pandas as pd
2.
3. girls = []
4. for year in range(1880, 2015):
5.     fn = "names/yob%i.txt" % year
6.     df = pd.read_csv(PATH + fn, names=['gender', year], index_col=0)
7.     girl = df[df.gender=='F'][year]
```

```
    girls.append(girl)

girls = pd.DataFrame(girls).fillna(0)
```

Exercise 2

Write the content of the variables to the screen by adding `print` statements. Explain what it does.

Exercise 3

Add the following lines to the program. Explain them.

```
tgirls = girls.transpose()
tgirls['sum'] = girls.apply(sum)
```

Exercise 4

Add the following lines to the program. Explain them.

```
tgirls = tgirls[tgirls['sum'] >= 1000]
tgirls.to_csv('girls_1000.csv')
```

Exercise 5

Add lines to conduct a similar analysis of boys' names.

Display data as a HTML page

Display data as a HTML page

In this exercise you will display a table on a HTML-page by starting your own web server with `bottle` .

Warning: This exercise assumes you have basic knowledge on web servers and HTML pages. If you have never created a web page, it is probably easier to try the **DjangoGirls Tutorial** (which is much better documented).

1. Launch Bottle

Start a hello world web server using Bottle.

2. Add a template

Create a file `views/template_name.tpl` (a HTML file).
Add a template to the bottle server and connect it to the Python script by:

```
1. @view('template_name')
```

3. Supply variables

Fill the template with data from a variable by returning from the view:

```
1. return {'text': ... }
```

Add a directive to the HTML template like:

```
1. {{!text}}
```

4. Create headers and footers

Add a HTML header and footer to make the page nicer, include them in the template:

```
1. % include('header.tpl')
```

5. Data from CSV

Supply data to the form results from the CSV file `grosse_laender_2015.csv` .

6. Forms

Create a form where the user can enter a parameter. Use `form.py` as a starting point.

Bokeh

Bokeh

What it is good for?

Creating diagrams for the web.

Bokeh creates interactive and non-interactive diagrams that can be displayed as web pages (or parts thereof). Of course, plots can also be saved as images. Compared to `matplotlib` , Bokeh is not as mature yet, but shows many interesting features already

Installed with Python by default

no

Installed with Anaconda

no

How to install it?

```
pip install bokeh
```

Example

Plot a simple line plot, write it to a HTML file:

```
>>> from bokeh.plotting import figure, output_file, show

```

```
>>> x = [1, 2, 3, 4, 5]
>>> y = [6, 7, 2, 4, 5]

>>> output_file("lines.html", title="line plot example")

>>> p = figure(title="simple line example", x_axis_label='x', y_axis_label='y')
>>> p.line(x, y, legend="Temp.", line_width=2)
```

Display the results in a web browser:

```
>>> show(p)
```

Where to learn more?

http://bokeh.pydata.org