706.088 INFORMATIK 1

TESTING & PYTHONIC CODING

INHALT

- Testing - Theory
- Pytest, Unittest
- PyLint, Pep8, Tox
- Pythonic Code

WIEDERHOLUNG

WARUM TESTEN?

- Fehler früh finden
- Nachweis korrekter Software?
- Test-Driven-Development

TESTSCHRITTE IN SOFTWAREPROJEKTEN

- **Komponententest** (Unit-Test) ←
- Integrationstest
- Systemtest
- Abnahmetest

WELCHE FEHLER WERDEN GEFUNDEN?

- Programmierfehler
- Semantische Fehler
- *vielleicht* Arithmetische Fehler
- *vielleicht* Laufzeitfehler
- *vielleicht* Logikfehler
- **nicht** Designfehler

PYTEST, UNITTEST

- pytest ←
- Unittest, in der Standard Lib

auch bekannt als **PyUnit**

DEMO

pytest Beispiele

PEP8

- Python Code Style Guideline
- vorgeschlagen 2001 von Guido van Rossum
- Tools zum Aufräumen:
  - Flake8
  - PyLint
  - pycodestyle (formerly pep8)
  - mypy

PYLINT

- Codestandard check
- Fehlererkennung
- Refactoring Hilfe
  - Errorcodes

FLAKE8

- kombiniert (pyflakes und PEP-8 Checks)
- `python3 -m pip install flake8`
- `flake8 your_code.py`
  - » Errorcodes

TOX

- Tox will Testprozess für Python standardisieren
- Baut eigene virtualenv Umgebung für jeden Testlauf
- integriert pytest, pyflakes, pylint, pep8 etc.
- testen für verschiedene Python Versionen.

CONTINUOUS INTEGRATION

› Automatisches Testen und Deployen aus der Codeverwaltung

› git -> tests -> packaging -> push

PYTHONIC CODE

Code, der Eigentheiten von Python ausnützt und den Dialekt der Sprache ausschöpft.

Code Beispiele aus Raymond Hettingers Talk

SCHLEIFEN

```
for i in [0, 1, 2, 3, 4, 5, 6]:
  print(i**2)
```

```
for i in range(7):
  print(i**2)
```

SCHLEIFE

```
fruit_basket = ["banana", "apple", "pear", "plum"]

for i in range(len(fruit_basket)):
  print(fruit_basket[i])
```

```
for fruit in fruit_basket:
  print(fruit)
```

SCHLEIFE RÜCKWÄRTS

```
fruit_basket = ["banana", "apple", "pear", "plum"]

for i in range(len(fruit_basket)-1, -1, -1):
  print(fruit_basket[i])
```

```
for fruit in reversed(fruit_basket):
  print(fruit)
```

LISTEN MIT INDEX

```
fruit_basket = ["banana", "apple", "pear", "plum"]

for i in range(len(fruit_basket)):
  print(i, '=', fruit_basket[i])
```

```
for i, fruit in enumerate(fruit_basket):
  print(i, '=', fruit)
```

LISTEN SORTIERT

```
fruit_basket = ["banana", "apple", "plum", "pear"]

for fruit in sorted(fruit_basket):
  print(fruit)
```

SORTIEREN MIT SPEZIAL FUNKTION

```
fruit_basket = ["banana", "apple", "plum", "pear"]
# print( fruits sorted by length )
```

```
for fruit in sorted(fruit_basket, key=len):
  print(fruit)
```

LESEN BIS ZUM ENDZEICHEN

```
blocks = []
while True:
  block = f.read(32)
  if block == '':
    break
  blocks.append(block)
```

```
blocks = []
for block in iter(partial(f.read, 32), ''):
  blocks.append(block)
```

ENDE EINER SCHLEIFE

```
def find(seq, target):
  found = False
  for i, value in enumerate(seq):
    if value == target:
      found = True
      break
  if not found:
    return -1
  return i
```

```
def find(seq, target):
  for i, value in enumerate(seq):
    if value == target:
      break
  else:
    return -1
  return i
```

SCHLEIFE ÜBER 2 LISTEN

```
fruit_basket = ["banana", "apple", "pear", "plum", "cherry"]
kids = ["tom", "jenny", "william", "betty"]

n = min(len(kids), len(fruit_basket))
for i in range(n):
  print(kids[i], '=', fruit_basket[i])
```

```
for name, fruit in zip(kids, fruit_basket):
  print(name, '=', fruit)
```

LIST COMPREHENSIONS

```
result = []
for i in range(10):
  squares = i ** 2
  result.append(squares)
print(sum(result))
```

```
print(sum( [i**2 for i in range(10)] ))
```

```
# Generator Expression
print(sum( i**2 for i in range(10) ))
```

› PEP zu Generator Expressions

DICTIONARIES

```
d = {'banana': 'yellow', 'apple': 'red', 'pear': 'green'}

for k in d:
  print(k)
```

```
for k in d.keys():
  if k.startswith('p')
    del d[k]
```

```
# upper example changes stuff it's iterating over! That's bad!!
# better do this:
d = {k: d[k] for k in d if not k.startswith('p')}
```

DICTIONARIES: KEYS, VALUES

```
d = {'banana': 'yellow', 'apple': 'red', 'pear': 'green'}

for k in d:
  print(k, '=', d[k])
```

```
for k, v in d.items():
  print(k, '=', v)
```

```
for k, v in d.iteritems():
  print(k, '=', v)
```

DICTIONARIES ERSTELLEN

```
fruit_basket = ["banana", "apple", "pear", "plum"]
kids = ["tom", "jenny", "william", "betty"]

d = dict(zip(kids, fruit_basket))
```

FUNKTIONS SIGNATUREN

```
sort_input(input, 30, False)
```

```
sort_input(input, num=30, reversed=False)
```

SEQUENCE UNPACKING

```
name = student[0]
birthday = student[1]
field_of_study = student[2]

name, birthday, field_of_study  = student
```

STRINGS VERKNÜPFEN

```
kids = ["tom", "jenny", "william", "betty",
        "andrea", "tim", "angela"]

s = kids[0]
for kid in kids[1:]:
  s += ', ' + kid

print(', '.join(kids))
```

FRAGEN?

PRÜFUNG

2019-01-23 **i13** 20:00

FROHE FEIERTAGE


UND GUTEN RUTSCH!