Würgeschlange 3 - Lesson 3

Tobias Maschek, Viktor Reusch
https://github.com/jemx/wise1920-python

mit Materialien von Felix Döring, Felix Wittwer https://github.com/fsr/python-lessons

12. November 2019

Python-Kurs

Gliederung

1. Lists und Dicts - Déjà-vu?

2. Funktionen

   Parameterübergabe

Jetzt auch im Neuland

Folien jetzt auch unter `https://github.com/jemx/wise1920-python`

Lists und Dicts - Déjà-vu?

Déjà-vu, I've just been in this place before. — Aufgabe 3-1

- Es ist eine Liste mit Ziffern gegeben: `v = [1, 1, 2, 4, 3, 7, 3]`
- Erstellt ein dict, welches jede Ziffer (von 0 bis 9) als Key besitzt.
- Die Values sollen die absoluten Häufigkeiten der Ziffern in `v` sein.
- **Profis:** Die Values sollen die relativen Häufigkeiten der Ziffern in `v` sein.

Funktionen

Funktionsdefinition

Warum?

- Teilt Code in Abschnitte auf: übersichtlicher und einfacher lesbar
- Erlaubt einfache Kollaboration: jeder schreibt eine Funktion
- Ermöglicht Wiederbenutzung von Code an verschiedenen Stellen
- Macht *Rekursion* möglich

```
1  def square(i):
2      return i * i
3
4  print(square(5))  # 25
5
6  def return_function(return_early):
7      if return_early:
8          return
9      print("Hi")
10
11 return_function(True)
12 return_function(False)  # Hi
```

Scopes? Scopes :(

```
1 i = 5
2
3 def set_i(k):
4     i = k
5     print(i)
6
7 print(i)
8 set_i(42)
9 print(i)
```

Scopes? Scopes :)

```
1  i = 5
2
3  def set_i(k):
4      global i
5      i = k
6      print(i)
7
8  print(i)
9  set_i(42)
10 print(i)
```

Parameterübergabe

Call by Value

- Variablen-Inhalt wird in Parameter **kopiert**.
- Änderung am Parameter haben keinen Effekt auf die aufrufende Variable

```c
#include<stdio.h>

void use(int i) {
    i = 3;
}

int main() {
    int p = 0;
    use(p);
    printf("%d\n", p); // 0
}
```

Call by Reference

- Parameter ist **Referenz** auf die aufrufende Variable.
- Alle Änderungen am Parameter wirken auch auf die Variable.

```
#include <iostream>
using namespace std;

void change(int &i) {
    i = 3;
}

int main() {
    int p = 0;
    change(p);
    cout << p << endl; // 3
}
```

Call by Object

- Parameter hält eine Referenz auf das selbe **Objekt**, wie die aufrufende Variable.
- Änderungen am Objekt haben eine Wirkung.
- Änderungen am Parameter nicht.

```python
def use(i):
    i = 20

def change(l):
    l.append(12)

p = 10
use(p)
t = [10, 11]
change(t)
print(f"unchanged {p}, changed {t}")  # unchanged 10, changed [10, 11, 12]
```

Keyword Arguments

```
def control_flow(name, formal=True):
    if formal:
        print(f"Hallo Herr/Frau {name}!")
    else:
        print(f"Hi {name}")

control_flow("Meier")                # Hallo Herr/Frau Meier!
control_flow("Frank", False)         # Hi Frank
control_flow("Frank", formal=False) # Hi Frank
```

Aufgabe 3-2

1. Schreibe eine *Batman*-Funktion:
   - Die Funktion bekommt eine Zahl übergeben.
   - Die Zahl wird um eins verringert und es wird „Na “ ausgegeben.
   - Dann wird die Funktion rekursiv (mit der verringerten Zahl) aufgrufen.
   - Wenn die Zahl 0 erreicht, wird „Batman“ ausgegeben und die Rekursion beendet.
2. Schreibe eine Begrüßungsfunktion:
   - Die Funktion bekommt **optional** einen Namen übergeben.
   - Der Standard-Wert für den Namen ist „Welt“.
   - Grüße nun die Person mit dem übergebenen Namen.