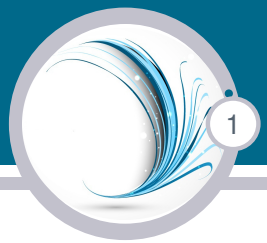
Institut für Informatik
Forschungsgruppe Datenbanken und Informationssysteme (DBIS)
Universität Innsbruck

Algorithmik und Programmieren

Martin Pichl
Datenbanken und Informationssysteme (DBIS)
dbis.uibk.ac.at
martin.pichl@uibk.ac.at

25. September 2017

Inhalt



Organisation
Zielgruppe

- “Einsteiger in die Informatik”
- Studierende ohne Programmierkenntnisse

Organisation
Teilnahme

- Die Teilnahme am Kurs ist freiwillig
- Gruppenarbeit ist ausdrücklich erwünscht
  - Helfen Sie anderen Studierenden, das hilft auch Ihnen
  - Erst wenn Sie die Lösungen erklären können, habe Sie das Problem wirklich verstanden!
- Stellen Sie Fragen!
- Melden Sie sich, wenn es zu schnell oder zu langsam geht

Organisation
Ablauf

- Theorieblock vor jeder Einheit
- Praktische Übungen
  - Sie lernen am meisten, wenn Sie selbst programmieren
  - Auch in Gruppen sollte jeder selbst programmieren

Organisation
Ziele

- Python kennen lernen
- Grundlagen der Programmierung
- Probleme zu abstrahieren und zu Lösen

Organisation
Termine

- Montag, von 14.15 – 16.00 Uhr
- Dienstag, von 14.15 – 17.00 Uhr
- ~~Mittwoch, von 11.15 – 13.00 Uhr und 14.15 – 16.00 Uhr~~
- Mittwoch, von 12.15 – 13.00 Uhr und 14.15 – 17.00 Uhr
- Donnerstag, von 11.15 – 13.00 Uhr
- Freitag, von 12.15 – 13.00 Uhr und 14.15 – 16.00 Uhr

Organisation
Kontakt

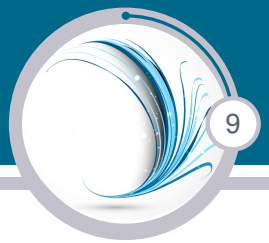
- Feedback und Fragen bitte an martin.pichl@uibk.ac.at
- Danke an Benedikt Hupfauf für das zur Zurverfügungstellen der Unterlagen

Theoretischer Input 0: Was ist Python?

Theoretischer Input 0
Was ist Python?

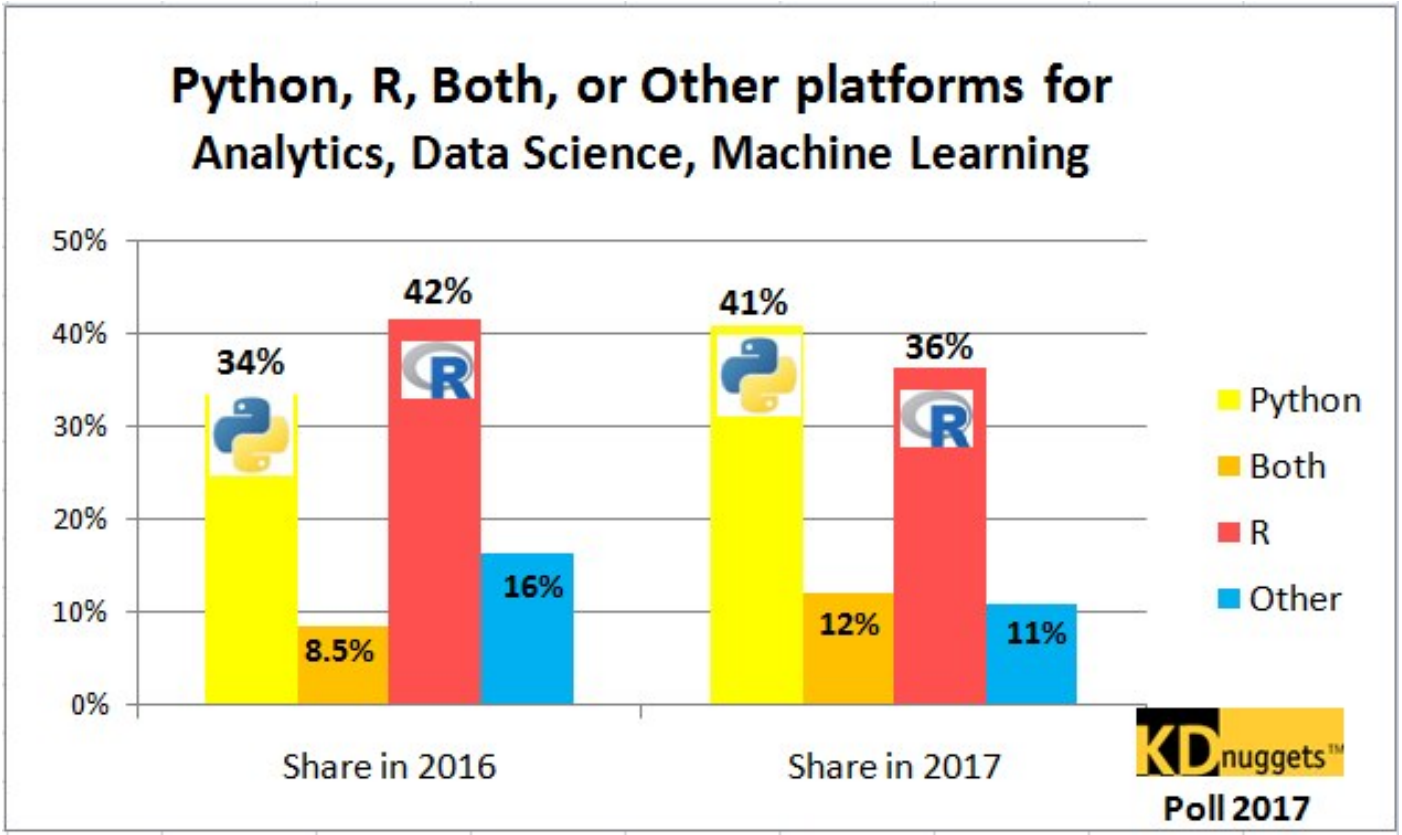
- Python ist eine junge Skriptsprache $\rightarrow$ es wird ein ein Interpreter benötigt
- Zwei Versionen sind verbreitet
  - Python 2: weit verbreitet, Zukunft?
  - Python 3: neue Features, semantisch korrekter
- Wir verwenden Python 3 in diesem Kurs

Theoretischer Input 0
Was ist Python?



- Open Source
- Plattformunabhängige Skriptsprache
- Einfach, verständlich und schnell zu erlernen
- Unterstützt verscheiden Paradigmen
- Umfassende Standardbibliothek
- Viele weitere nützliche Bibliotheken (!)
- Nachfrage nach Python Programmierern steigt :)

Theoretischer Input 0
Was ist Python?



Source: kdnuggets.com

Theoretischer Input 0
Hilfreiche Ressourcen

- Die Python Dokumentation docs.python.org
- Stack Overflow Community stackoverflow.com
- Stack Quora quora.com
- Online Bücher, Tutorials und Kurse:
  - A Byte of Python
  - The Python Tutorial
  - ...

Theoretischer Input I: Datentypen & Variablen

Theoretischer Input I
Datentypen

Python liefert Datentypen für die gängigsten Anwendungen (Built-in Types):

- Strings (Zeichenketten, Text): `'This is text!'`
- Integer (Ganzzahlen): `1, 0xFF, 0b100`
- Float (Gleitkommazahlen): `1.35, 1.56e-5` (entspricht $1.56 * 10^{-}5$)
- Complex (Komplexe Zahlen): `2+3j`
- Bool (Wahr/Falsch): `True, False`

Theoretischer Input I
Hello World

- Python bietet zwei Möglichkeiten ein Programm auszuführen
  - Interaktiv (Command Prompt / Shell)
  - Python Skript
- Schreiben Sie ihr erstes Hello World Programm

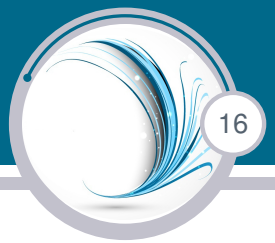
Theoretischer Input I
Variablen

- Konstanten (wie eben benutzt) sind nicht immer ausreichend
- Variablen können beliebige Werte zugewiesen werden
- In Python muss der Typ nicht vorher bekannt gegeben werden und kann sich zur Laufzeit ändern
- Der dynamischen Typ bestimmt welche Operationen für diese Variable zulässig sind

Theoretischer Input I
Variablennamen

- Gültige Variablennamen sind eine beliebige Kombination aus folgenden Zeichen: a-z, A-Z, 0-9 und _.
- Beachten Sie folgende Ausnahmen:
  - Variablennamen dürfen nicht mit einer Ziffer beginnen
  - Reservierte Schlüsselwörter:
    and, as, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while, with, yield
- Variablen sollten einen **selbsterklärenden** Namen haben

Theoretischer Input I
Ausdrücke



Sie können mit Ausdrücken Variablen Werte zuweisen. Hierfür stellt Python bereits viele Operatoren bereit, beispielsweise `+`, `*`, `<`, `=`, `==` uvm.. Beachten Sie, dass `=` einer Variable einen Wert zuweist, während `==` zwei Werte vergleicht ohne sie zu verändern.

```
>>> x = 2 + 1
>>> x = 2 * x
>>> print(x)
6
>>> x == 5
False
>>> print(x)
6
```

Theoretischer Input I
Operatoren Präzedenz

- Die Operatoren Präzedenz bestimmt, in welcher Reihenfolge die Operatoren angewandt werden
- Damit wird bestimmt wie Ausdrücke evaluiert werden
- Logische vs. Bitweise Operatoren
- The Python Language Reference 6.16. Operator precedence

Übung Input I
Variablen

- Unterlagen: http://bit.ly/2yBba4H

Theoretischer Input II: Kontrollfluss

Theoretischer Input II
Kontrollfluss if-Statement

- Anweisungen werden sequentiell ausgeführt
- Ist das nicht genug, kann der Kontrollfluss gesteuert werden
- Das `if`-Statement
  - Führt einen Code Block nur aus, wenn die Bedingung `<condition>` erfüllt ist:
  - Kann optimal mit einem `else`-Block ergänzt werden,der ausgeführt wird, wenn die Bedingung nicht erfüllt ist.

```
1 if <condition>:
2   <executed when condition true>
3 else:
4   <executed when condition false>
```

Theoretischer Input II
Kontrollfluss if-Statement Beispiel I


```
1 >>> if a < 0:
2 ...     print('a is negative')
3 ...     print('this line belongs to the same codeblock')
4 >>> print('this is a new code block and always executed')
```

- Im Gegensatz zu den meisten Programmiersprachen ist die Einrückung in Python nicht optional, sondern definiert einen Codeblock
- Sie können Code mit <tab> oder mit 2-4 Leerzeichen einrücken, sollten aber bei einer dieser Optionen bleiben.

Theoretischer Input II
Kontrollfluss if-Statement Beispiel II


```
1 >>> if a < 0:
2 ...   print('a is negative')
3 >>> elif a > 0:
4 ...   print('a is positive')
5 >>> else:
6 ...   print('a must be zero then')
```

- Das Schlüsselwort `elif <other condition>:` ist eine Kurzschreibweise für das häufig gebrauchte `else: if <other condition>:`

Theoretischer Input II
Kontrollfluss while-Statement

- Mit dem `while`-Statement wird ein Programmblock wiederholt solange die Bedingung `<condition>` erfüllt ist.
- Optional kann ein `else` Block angegeben werden, der einmal ausgeführt wird, sobald die Bedingung nicht mehr erfüllt ist.

```
1 wile <condition>:
2   <executed as long as condition is true>
3 else:
4   <executed once after the condition is false>
```

Theoretischer Input II
Kontrollfluss while-Statement Beispiel


```
1  >>> i = 0
2  >>> while i < 3:
3  ...     print('i is:', str(i))
4  ...     i = i + 1
5  >>> else:
6  ...     print('end')
7  ...
8  i is: 0
9  i is: 1
10 i is: 2
11 end
```

- Achten Sie darauf, dass die Bedingung nicht für immer `TRUE` sein darf, da die Schleife sonst endlos ist.
- Der Code Block des `else`-Statements wird immer genau einmal ausgeführt.
- Was passiert, wenn das `while`-Statement `print('i is: ' + str(i))` aufruft?

Theoretischer Input II
Kontrollfluss Break and Continue

- Eine Schleife muss nicht immer bis zum Ende ausgeführt werden.
- Mit dem Schlüsselwort `continue` wird der **aktuelle** Schleifendurchlauf abgebrochen.
- Mit dem Schlüsselwort `break` wir die Schleife sofort **beendet**; auch das `else`-Statement wird übersprungen

```
>>> i = 0
>>> while i < 1000:
...     i = i + 1
...     if i == 4: break
...     if i == 1: continue
...     print('i is:', str(i))
... else:
...     print('end')
...
i is: 2
i is: 3
```

Theoretischer Input III: Funktionen & Module

Theoretischer Input III
Funktionen

- Häufig verwendete Codeteile können in Funktionen ausgelagert werden
- Eine Funktion sollte immer genau einen Zweck erfüllen!
- Funktionen können, müssen aber keinen Rückgabewert haben (Schlüsselwort `return`).
- Die Parameterliste `<parameters>` kann beliebig lang, also auch leer, sein.

```
1 def <function name>(<parameters>):
2     <function code>
```

Theoretischer Input III
Funktionen Beispiel

Die folgenden Minimalbeispiele sollen nur zeigen wie Funktionen definiert werden können. Eine Funktion umfasst typischerweise mehr als nur eine Zeile Code.

```
>>> def mul(x , y):
...     return x * y
...
>>> mul(2,3)
6
```

```
>>> def printABC():
...     print('abcdefghijklmnopqrstuvxyz')
...
>>> printABC()
abcdefghijklmnopqrstuvxyz
```

Theoretischer Input III
Funktionen Beispiele

- Python bietet eine große Auswahl an Funktionen, die bereits vorimplementiert sind
- Funktionen die “zusammen gehören” werden in Module ausgelagert, etwa das Modul math, in dem Sie zahlreiche Funktionen aus der Mathematik finden.
- Werden Funktionen aus einem Modul benötigt, muss dieses explizit geladen werden.

```
1 >>> import math
2 >>> math.sin(45)
3 0.8509035245341184
4 >>> from math import sin
5 >>> sin(45)
6 0.8509035245341184
```

Bis zum nächsten Mal :)