Einführung in die Computerlinguistik – Einführung in Python (1)

Dozentin: Wiebke Petersen

10. Foliensatz

Compiler

"Ein Compiler (auch Übersetzer oder Kompilierer genannt) ist ein Computerprogramm, das ein in einer Quellsprache geschriebenes Programm – genannt Quellprogramm – in ein semantisch äquivalentes Programm einer Zielsprache (Zielprogramm) umwandelt.
Üblicherweise handelt es sich dabei um die Übersetzung eines von einem Programmierer in einer Programmiersprache geschriebenen Quelltextes in Assemblersprache, Bytecode oder Maschinensprache.
Das Übersetzen eines Quellprogramms in ein Zielprogramm durch einen Compiler wird auch als Kompilierung bezeichnet." (Aus `http://de.wikipedia.org/wiki/Compiler`)

Interpreter

"Ein Interpreter (im Sinne der Softwaretechnik) ist ein Computerprogramm, das einen Programm-Quellcode im Gegensatz zu Assemblern oder Compilern nicht in eine auf dem System direkt ausführbare Datei umwandelt, sondern den Quellcode einliest, analysiert und ausführt. Die Analyse des Quellcodes erfolgt also zur Laufzeit des Programms." (Aus `http://de.wikipedia.org/wiki/Interpreter`)

Python – Geschichte und mehr

- Ende der 1980er Jahre entwickelt von Guido van Rossum
- Ziel: Programmieren soll einfach sein und Spaß machen. Dazu setzt die Sprache auf eine besonders übersichtliche Syntax
- Python steht zwischen reinen Interpreter- und reinen Compiler-Sprachen: Programme werden zur Laufzeit in einen sog. Bytecode übersetzt, der dann vom Python-Interpreter ausgeführt wird
- Python-Interpreter (Freeware) für Windows: ActivePython
  `http://www.activestate.com/activepython/downloads`
- Unter Linux und Mac OS X ist Python in der Regel vorinstalliert
- Kurze Tutorials:
  `http://www.wspiegel.de/pykurs/`,
  `http://docs.python.org/tutorial/introduction.html`
- Empfehlenswertes Buch
  Allen B. Downey: *Think Python*. Green Tea Press, 3. Auflage, 2008
  (http://www.greenteapress.com/thinkpython/thinkpython.pdf).
- Notepad++: `http://sourceforge.net/projects/notepad-plus/files/`

Crashkurs: Syntax eines Python-Programms

```
1  import random
2  even = 0
3  odd = 0
4  for counter in range(0,10):
5    number = random.randint(0,100)
6    if ((number % 2) == 0):
7      print "%i is even" % (number)
8      even = even + 1
9    else:
10     print "%i is odd" % (number)
11     odd = odd + 1
12 print "Found %i even and %i odd numbers" % (even, odd)
```

1. Zeile 1: Laden einer Bibliothek mit Funktionen für Zufallszahlen
2. Zeile 3–4, 6, 9, 12: Variablenzuweisung
3. Zeile 5: for-Schleife und Aufruf der Funktion `range()`
4. Zeile 6–11: Verzweigung mit if/else (Kontrollstruktur)
5. Zeile 7, 10, 12: Bildschirmausgabe

Ausprobieren von Python

- Man muss nicht gleich ein ganzes Python-Programm schreiben
- Kleine Python-Anweisungen lassen sich direkt in den Python-Interpreter eingeben
- Dazu in einer Kommandozeile das Programm "`python`" starten
- Python fordert mit ">>> " die Eingabe einer Python-Anweisung an
- Beispiele:
  - `1 + 2` führt zur Ausgabe "3"
  - `round(5.5)` führt zur Ausgabe "6.0"
  - `x = 5` und `y = 6` und `min(x,y)` führt zur Ausgabe "5"

Schreiben von Python-Programmen

- Für echte Programme müssen die Anweisungen in eine Datei geschrieben werden, da Python interaktive Eingaben nach dem Schließen "vergisst"
- Python-Programme kann man mit jedem beliebigen Texteditor schreiben
- Speichern mit Endung "`.py`" (Beispiel "`script.py`")
- Ausführen von der Kommandozeile mit "`python script.py`"

"Hello world!"

```
1  # Print "Hello world!" to the screen
2  print "Hello world!"
```

- Kommentarzeichen: `#` (alles was auf `#` folgt wird nicht als zum Programm gehörig interpretiert)
- Kommentare sind wichtig, sie helfen den Code zu verstehen
- Gewöhnen Sie sich an, sorgfältig zu kommentieren!
- `print` ist eine Anweisung
- `Hello world!` ist das Argument von `print`
- Jede Anweisung bekommt eine eigene Zeile!
- In Python gibt es, im Gegensatz zu Sprachen wie Perl oder Prolog, kein besonderes Zeichen zum Zeilenende

Ausführen eines Programms (im CIP-Raum)

- Öffnen Sie die Datei `python.cmd`. Es öffnet sich ein DOS-Fenster
- Wechseln Sie zu dem Ordner in dem das Python-Programm liegt (Beispiel: `T:\EinfCL\Nachname\Python`):
  - Wechseln zum Laufwerk T: "`T:`"
  - Wechseln in den Unterordner "EinfCL\Nachname\Python": "`cd EinfCL\Nachname\Python`"
- Ausführen des Programms (Beispiel: "world.py"): "`python world.py`"

Übungseinheit (1): “Hello world!” selbst schreiben

1. Erstellen Sie einen Unterordner für Ihre Python-Programme. Beispiel: “T:\EinfCL\Nachname\Python”
2. Öffnen Sie einen Texteditor (z.B. Notepad++, Textpad)
3. Schreiben Sie ein “Hello world!” Programm
4. Nicht vergessen zu speichern
5. Testen Sie das Programm
6. Sollte Ihr Programm einmal nicht von selbst stoppen, so können Sie es jederzeit mit `Strg+c` abbrechen (Steuerung/Control und c)
7. Python selbst beenden Sie mit der Eingabe von `exit()` gefolgt von Enter

Textausgabe

```
1 print "Hello world!"
2 print "Hello" + " " + "world!"
3 print "Hello %s!" % ("world")
4 print "%s plus %s is %s" % (1, 2, 1+2)
```

Zeile 1 Die einfachste Form der Textausgabe: gibt eine fest definierte Zeichenkette aus

Zeile 2 Wie Zeile 1, allerdings mit Stringkonkatenation

Zeile 3 Interpolierte Textausgabe: gibt eine Zeichenkette mit einem Platzhalter `%s` aus. Der Platzhalter wird in der Ausgabe durch den Wert `"world"` ersetzt.

Zeile 4 Interpolierte Ausgabe mit mehreren Platzhaltern: die Platzhalter `%s` werden entsprechend ihrer Reihenfolge ersetzt

Texteingabe

```
1 input = raw_input()
2 print input
```

- Die Funktion `raw_input()` liest eine Eingabe von der Standardeingabe (Tastatur)
- In Zeile 1 wird die Benutzereingabe in der Variable `input` gespeichert

```
1 name = raw_input("What is your name? ")
2 print "Hello %s!" % (name)
```

- `raw_input()` wird hier mit einem zusätzlichen Argument `"What is your name? "` aufgerufen. Die Funktion gibt so zunächst das Argument auf dem Bildschirm aus und wartet dann auf eine Benutzereingabe. So lassen sich einfache Benutzereingaben umsetzen

Variablen

```
1  fruit1 = "apples"
2  fruit2 = "plums"
3  amount1 = 6
4  amount2 = 10
5  sum = amount1 + amount2
6  print "%s %s and %s %s are %s fruits." % \
7    (amount1, fruit1, amount2, fruit2, sum)
```

- Erst mit Variablen werden Programme variabel!
- Variablennamen bestehen aus Buchstaben, Ziffern und dem Unterstrich (keine Leerzeichen)
- Variablennamen beginnen nie mit einer Ziffer
- Groß- und Kleinschreibung wird unterschieden!
- Verwenden Sie *bedeutungsvolle* Bezeichnungen
- Variablen werden Werte zugeordnet
  - Zahlen
  - Strings (Zeichenketten)
  - ...

Ausdrücke

```
1  fruit1 = "apples"
2  fruit2 = "plums"
3  amount1 = 6
4  amount2 = 10
5  sum = amount1 + amount2
6  print "%s %s and %s %s are %s fruits." % \
7    (amount1, fruit1, amount2, fruit2, sum)
8  print "The juice could be called " + fruit1 + \
9    "-" + fruit2 + "-juice."
```

- Numerische Operationen: `+ - * / % **`
  - `a = 5`
  - `b = a + (3.5 * 2)`
  - `c = b * 2`
  - `d = c / 2`
- Konkatenation von Strings: `"Haus" + "bau"`

Übungseinheit (2): Python spielen

- Angenommen, Python führt das folgende Programm aus. Welchen Wert hat am Ende die Variable `change`?

```
1 last = 1000
2 current = 900
3 change = (current - last) / (last / 100)
```

- Was gibt das Programm jetzt aus?

```
1 print "Last: %i. Current: %i" % (last, current)
2 print "Change: %f percent" % (change)
```

- Der Platzhalter `%i` steht für eine ganze Zahl, `%f` für eine reelle Zahl

Übungseinheit (3): Kreisradius berechnen

1. Schreiben Sie ein Programm, das den User nach dem Radius eines Kreises fragt und den Umfang und die Fläche des Kreises berechnet und ausgibt.
2. Testen Sie das Programm
3. Wenn Sie noch Zeit haben, erweitern Sie das Früchteprogramm um eine Eingabe durch den User

Kontrollstrukturen: if/else

```
1 password = raw_input("Please enter password: ")
2 if (password == "42"):
3   print "Correct password! Welcome."
4 else:
5   print "Wrong password! Access denied."
```

- Die if/else-Kontrullstruktur ermöglicht Verzweigungen in Python-Programmen
- Codeblöcke werden in Python *eingerückt* und nicht, wie in Java, Perl oder C, durch geschweifte Klammern umschlossen. Sie beginnen mit einem Doppelpunkt
- Ein falsch oder unregelmäßig eingerücktes Programm kann *nicht ausgeführt* werden
- == ist der Vergleichsoperator. In diesem Fall wird geprüft, ob der Wert der Variablen `password` der Zeichenkette `42` entspricht

Schleifen: while

```
1 password = raw_input("Please enter password: ")
2 while (password != "42"):
3   print "Wrong password! Access denied."
4   password = raw_input("Please enter password: ")
5 print "Correct password! Welcome."
```

- Eine while-Schleife wird so lange ausgeführt, wie die Schleifenbedingung wahr ist. In diesem Fall wird die Schleife so lange ausgeführt, wie die Variable `password` nicht den Wert `"42"` beinhält
- `!=` ist der negierte Vergleichsoperator
- (Weitere Vergleichsoperatoren sind: < kleiner, <= kleiner oder gleich, > größer, >= größer oder gleich)

Übungseinheit (4)

1. Erweitern Sie ihr Kreisberechnungsprogramm: Verzichten Sie bei negativen Werten für den Radius auf die Berechnung und generieren Sie statt dessen eine Fehlermeldung für den User. Erstellen Sie dazu zwei Programme: eines unter Verwendung von if/else und eines unter Verwendung von while.
2. Wenn Sie Zeit haben, können Sie den Benutzer auch fragen, ob er die Werte für ein Quadrat oder einen Kreis berechnen möchte