Python für Linguisten

Dozentin: Wiebke Petersen & Co-Dozent: Valentin Heinz

0. Foliensatz
Hilfestellungen und FAQ

IDLE Python Gui: Editor

- die Anwendung besteht aus zwei Fenstern: dem Editor und der Shell. Letztere wird zuerst gestartet (siehe vorherige Abbildung)
- der Editor wird durch `File` → `New Window` gestartet
- dieser ist gut für größere Programme, diese können per `F5` ausgeführt werden, automatische Aufforderung zum Speichern
- einen Überblick über die Tastaturkürzel erhältst man durch die einzelnen Reiter, erstellen Sie einen Merkzettel
- Befehlsvervollständigung per `TAB`
- die Ausführung findet durch den Interpreter im Rahmen der Shell statt

IDLE Python Gui: Shell

- diese Shell ist **NICHT** die Python Interactive Shell, die im Startmenü aufgeführt ist!
- diese Shell ist das erste Fenster, das sich öffnet, wenn Python IDLE Gui aufgerufen wird.
- in der Shell werden Befehle eingeben, ausgeführt und per `STRG + P` und `STRG + N` kann im Befehlsverlauf "geblättert" werden.
- Befehle können mit `Strg + S` gespeichert werden, oder durch `File` → `Save`
- die Shell kann durch `Strg + F6` neu gestartet werden, dies kann auch durch den Reiter `Shell` → `Restart Shell` erfolgen
- unter dem Reiter `Edit` findest man hilfreiche Funktionalitäten, wie etwa das Zurücknehmen von Änderungen
- um das Syntaxhighlighting zu verstehen, kann man unter `Options` → `Configure IDLE` (neues Fenster öffnet sich) → `Highlighting` eine Vorschau aufrufen

Benennungskonventionen 1

- Dateien: benennen Sie ihre Dateien systematisch, etwa `p1_integer.py` für ein Skript aus der ersten Sitzung, in welchem mit Integern gearbeitet wurde. Es ist notwendig, **keine Leerzeichen!** zu verwenden!
- Kleinschreibung: immer **kleingeschrieben** werden:
  - Variablennamen
  - Funktionsdefinitionen und Funktionsaufrufe
  - Dateinamen
- (Natürlich kann es begründete Ausnahmen geben, etwa, wenn man sich auf CamelCaps geeinigt hat)

Benennungskonventionen 2

- Allgemein: vergeben Sie sprechende, englischsprachige Benennungen und bilden Sie diese schematisch:
  - Funktionsnamen aus Verb und Objekt: `print_debug` ist ein sinnvoller Name für eine Funktion, die Debugging-Output ausgibt
  - Variablennamen aus Inhalt und Datentyp: `token_list` für eine Liste die Token enthält
  - falls dies nicht möglich ist, beschreiben Sie die Funktion bzw. Tätigkeit: `counter` (oder per Programmiererkonvention auch `i`) als Benennung für einen Zähler
- falls Zweifel bestehen, konsultieren Sie den Styleguide PEP08 (`http://www.python.org/dev/peps/pep-0008/`)

Dokumentation – docstrings

- gewöhnen Sie sich an, sauber zu dokumentieren. Dies erhöht die Lebensdauer des Codes und erleichtert es auch anderen, mit diesem sinnvoll zu arbeiten.
- verwenden Sie docstrings (siehe `http://www.python.org/dev/peps/pep-0257/`):

```
def get_volume_box(length, height, depth):
    ''' (number, number, number) -> float
    Return the volume of a box
    >>> get_volume_box(1,1,5)
    5.0
    '''
```

Hilfreiche optionale Tools

- Folgende Tools sind hilfreich und werden die Codequalität deutlich erhöhen. Sie setzen aber eine eigene Einarbeitung voraus und sind bisher nicht Bestandteil des Kurses:
- mittels `pep8`, kann die Einhaltung des Styleguides überprüft werden
- mittels `pyflakes` werden Programme auf logische Fehler getestet, ohne sie auszuführen
- `pylint` verfügt über einen größeren Funktionsumfang als pyflakes, führt aber aus
- docstrings können automatisiert mittels `doctest` überprüft werden. Das bedeutet, das Code und Dokumentation übereinstimmen
- pep8: `https://pypi.python.org/pypi/pep8`
- pylint: `http://docs.python.org/3/faq/programming.html#is-there-a-tool-to-help-find-bugs-or-perform-static-analysis` bzw. `https://bitbucket.org/logilab/pylint/`
- pyflakes: `https://launchpad.net/pyflakes`
- doctest: `http://docs.python.org/3/library/doctest.html`

Fragen und Antworten – wird fortlaufend ergänzt

- Frage: Warum gibt `string_1 > string_2` ein Ergebnis zurück? Antwort: Warum nicht? Frage: nach welchem Kriterium geschieht dies?
- Antwort: Die Strings werden zeichenweise bzgl. ihres Rangs in der ASCII-Tabelle verglichen (früher bedeutet kleiner). Sind sie nicht identisch, ist der Vergleich abgeschlossen und der Rest wird nicht einbezogen. Sind sie identisch wird mit den nächsten Zeichen ebenso verfahren, bis diese sich unterscheiden oder keine weiteren verfügbar sind. Siehe auch: `http://docs.python.org/2/tutorial/datastructures.html#comparing-sequences-and-other-types`
- Bevor jemand fragt: Falls man verschiedene Objekte vergleicht (geht nur, falls keines eine Gleitkommazahl ist), werden diese alphabetisch nach dem Namen des Typs verglichen. D.h. z.B.: `list < str < tupel`

Fragen und Antworten

- Frage: Warum ergibt `3.4 * 3.4` nicht 11.5, sondern 11.559999999999999?
- Antwort: Gleitkommazahlen werden in Python intern binär repräsentiert. Dezimalrepräsentation ist auch möglich, siehe `http://docs.python.org/3/tutorial/stdlib2.html#decimal-floating-point-arithmetic`
- Frage: Wie kann ich in der IDLE-Python-GUI Zeilennummern einschalten?
- Antwort: Gar nicht, benutze z.B. Geany oder Notepad++. IDLE-Python besitzt aber einen Debug-Modus, der das Zeilenspringen erlaubt: `http://docs.python.org/2/library/idle.html`
- Frage: Gibt es eine Übersicht zu den grundlegenden Python-Funktionen, d.h. zur Standardlibrary? Antwort: `http://docs.python.org/3/tutorial/stdlib.html`
- Frage: Wie kann ich alle Vorkommen von 'foo' mit 'bar' ersetzen? Antwort: Importieren Sie das re-Modul. Programmcode:
  ```
  neuer_string = re.sub('foo', 'bar', alter_string)
  ```
- Frage: Ich möchte ein ß oder ein ö in einem Kommentar verwenden, bekomme aber eine Fehlermeldung. Antwort: Deklariere ein Encoding, wie utf-8 oder iso8859-1: Beispiel: `# -*- coding: iso-8859-1 -*-` Hintergrund: `http://www.python.org/dev/peps/pep-0263/`