

Einführung in Python

Arne Hüffmeier

1 Ziele der Vorlesung

2 Einstieg in Python

3 Ende

Was soll vermittelt werden?

Problemorientiertes Denken

Warum Python ?

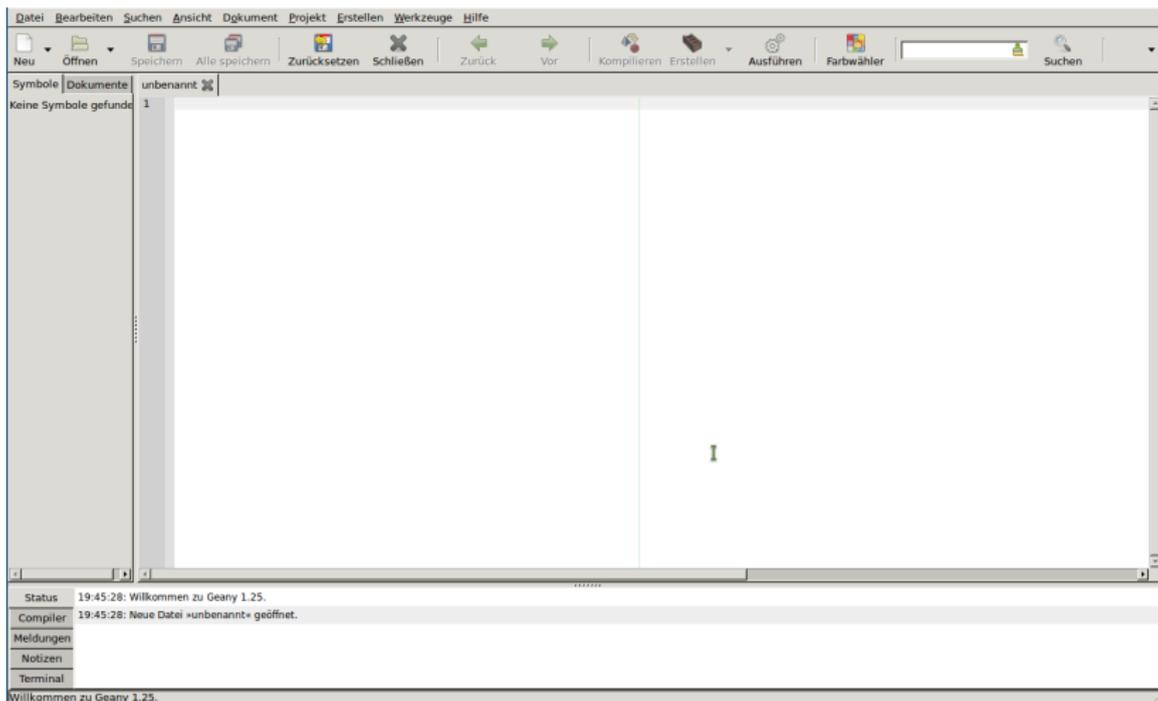
Vorteile

- einfache Syntax
- kein unnötiger Overhead
- relativ einfache Konstrukte
- viele Funktionen gibt es schon
- Plattformunabhängigkeit

Nachteile

- langsamere Ausführung

Geany einrichten



Geany einrichten

Bearbeiten > Einstellungen > Editor > Einrückung

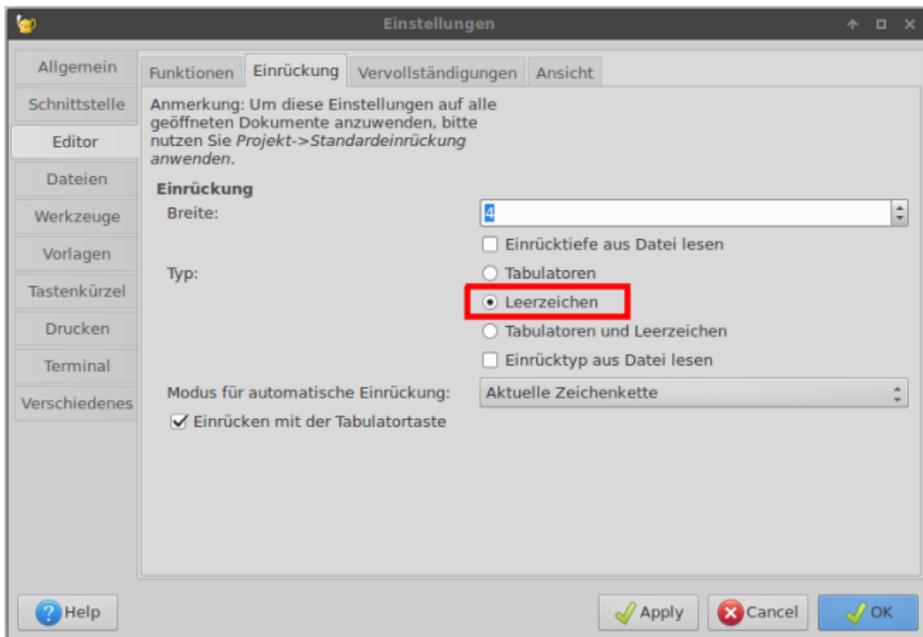


Abbildung: Aus Tabulatoren mache Leerzeichen

Geany einrichten

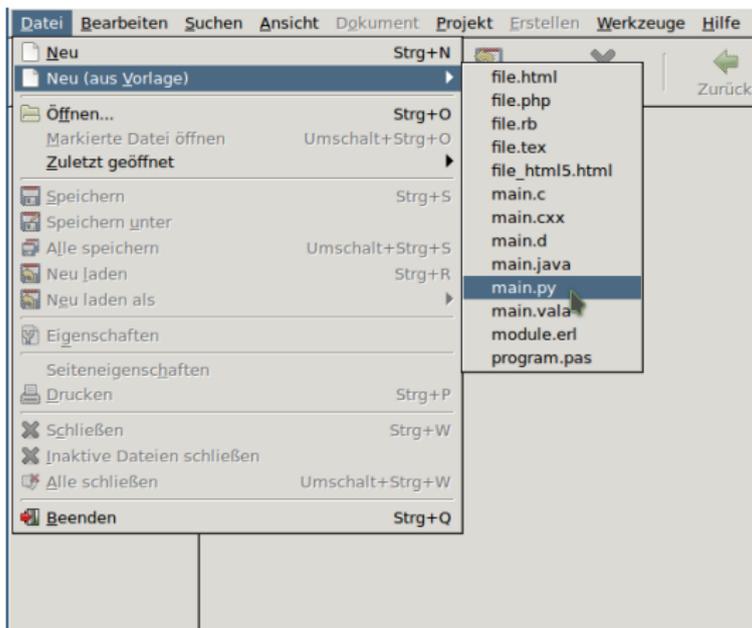


Abbildung: Neue Python Datei erstellen.

Geany einrichten

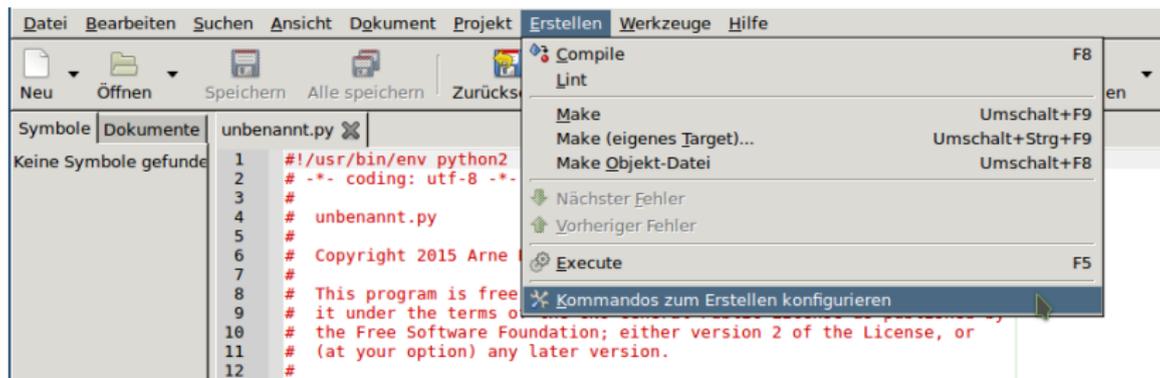


Abbildung: Kommandos zum Erstellen konfigurieren.

Geany einrichten

| # | Label | Kommando | Arbeitsverzeichnis | Zurücksetzen |
|--|---|---|--------------------|---|
| Kommandos für Python | | | | |
| 1. | C <code>ompile</code> | <code>python -m py_compile "%f"</code> | |  |
| 2. | | | |  |
| 3. | L <code>int</code> | <code>pep8 --max-line-length=80 "%f"</code> | |  |
| Regulärer Ausdruck für Fehlermeldungen: | | <code>(.+:){0-9}+:(0-9)+</code> | |  |
| Dateitypunabhängige Befehle | | | | |
| 1. | M <code>ake</code> | <code>make</code> | |  |
| 2. | M <code>ake</code> (eigenes T <code>arget</code>)... | <code>make</code> | |  |
| 3. | M <code>ake</code> O <code>bjekt</code> -Datei | <code>make %e.o</code> | |  |
| 4. | | | |  |
| Regulärer Ausdruck für Fehlermeldungen: | | | |  |
| <i>Notiz: Element 2 öffnet ein Dialog und fügt das Ergebnis am Ende des Kommandos an</i> | | | | |
| Befehle zum Ausführen | | | | |
| 1. | E <code>xecute</code> | <code>python "%f"</code> | |  |
| 2. | | | |  |
| <small>%d, %e, %f, %p, %l werden innerhalb der Kommando- und Verzeichnisfelder ersetzt - Details gibt es in der Dokumentation.</small> | | | | |
| | | | | <input type="button" value="Abbrechen"/> <input type="button" value="OK"/> |

Abbildung: Aus python python3 machen

Das Erste Programm

```
1 print("Hallo World")
```

Das *print* gibt an, dass etwas ausgegeben werden soll.
In den Anführungszeichen kann ein beliebiger Text stehen.

Was können wir jetzt damit machen?

- Wir können Text ausgeben ;-)
- Wir können rechnen.

Operationen

● `print(10 + 2)`

● `print(10 - 2)`

● `print(10 * 2)`

● `print(10 + 3 + 2)`

● `print(10 % 3)`

● `print(10 ** 3)`

Ein paar Infos am Rande

Infos über print

```
print("Hallo")  
print("Du_ Da")
```

Der *print* Befehl gibt eine Zeile aus. Somit würde das

Hallo

Du Da

ergeben. Man kann aber den Zeilenumbruch am Ende unterdrücken oder durch etwas anderes ersetzen.

```
print("hallo", end="")  
print("Du_ Da")
```

Wie Python das macht und warum das *end* nicht in " steht, klären wir bei dem Thema Funktionen.

Noch ein paar Infos am Rande

Infos über print

Es ist auch möglich mehrere Dinge in einem print auszugeben

```
print("Hallo", "Du", "Da")
```

Ausgabe

```
Hallo Du Da
```

Wenn man keine Leerzeichen haben will, kann man das so machen

```
print("Hallo"+"Du"+"Da")
```

oder so

```
print("Hallo", "Du", "Da", sep='')
```

Numerische Operationen

- + Addition
- - Subtraktion
- * Multiplikation
- / Division
- % Modulo (Division mit Rest)
- ** Potenz

Fragen ?

Fragen?

Geschafft

Nun habt ihr einen Einstieg in Python

Viel Spaß im Tutorium