Einführung in die computerlinguistische Programmierung mit Python

Willkommen im ersten Programmierkurs Ihres Computerlinguistikstudiums!

Hier erwerben Sie die Grundlagen im Programmieren, die Sie für Ihr weiteres Studium benötigen:

- Umgang mit einem professionellen Editor
- Datenstrukturen in Python
- Lesen und Schreiben von Dateien
- Zerlegung komplexer Aufgaben in unterschiedliche Teilprobleme, die nacheinander gelöst werden
- Erste Schritte in der Verarbeitung natürlicher Sprache
- ...

Wir arbeiten im Kurs mit der Programmiersprache Python.

Bevor es losgeht...

Die erste Programmiersprache zu lernen ist für die meisten Menschen schwierig, weil es viele unterschiedliche Tools und Zusammenhänge gibt, die man verstehen und sich merken muss. Deshalb möchten wir Sie einladen, so viele Fragen wie möglich zu stellen!

Um Ihnen das Stellen von Fragen so leicht wie möglich zu machen, haben wir für diesen Kurs ein anonymes Formular (https://forms.gle/sw7gN1PMmEG6GX1WA) erstellt.

Sie können jederzeit während des Unterrichts, in Pausen oder zwischen den Sitzungen Fragen an uns schicken, die wir dann bei der nächsten Gelegenheit im Unterricht klären.

Oft ist das direkte Nachfragen im Unterricht einfacher. Wählen Sie den Weg, der am besten zu Ihnen passt!

Python

Python existiert seit 1991 und ist heutzutage eine mächtige und verbreitete Sprache in der Wirtschaft, aber auch im wissenschaftlichen Bereich. Die Fähigkeiten, die Sie in diesem Kurs erwerben, können Sie also in beiden Bereichen später anwenden.

Python funktioniert auf Windows-, Mac- und Linuxsystemen und ist **vielseitig einsetzbar**. Die Syntax der Sprache ermöglicht verschiedene Programmierstile, z.B. objektorientiert, logisch, oder funktional. Ziel dieses Kurses ist es, Ihnen einen Einblick in die wichtigsten Funktionalitäten von Python zu bieten.

Eine Besonderheit der Sprache ist, dass in Python geschriebene Programme relativ **einfach zu lesen** sind -- verglichen mit anderen verbreiteten Programmiersprachen. Die folgenden Codeausschnitte geben jeweils die Worte "Hallo Welt!" auf dem Bildschirm aus, wobei die Java-Variante verschachtelter ist und etwas mehr Boilerplate-Code (https://de.wikipedia.org/wiki/Boilerplate#Programmierung) enthält.

Python

```python
print("Hallo Welt!")
```

Java

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Der Python-Interpreter

Gängige Programmiersprachen werden entweder **interpretiert** oder **kompiliert**. Die Unterscheidung bezieht sich darauf, in welcher Weise der geschriebene Programmcode (z.B. unser Hallo-Welt-Programm) in Maschinensprache übersetzt wird, die dann dafür sorgt, dass der Computer unsere Anweisungen befolgt.

Interpretierte Sprachen:

- Übersetzung erfolgt während der Laufzeit.
- Effekt: Programmierfehler werden evtl. erst mitten im Programm ersichtlich.
- Leicht zu nutzen: Dadurch, dass kein separater Schritt zum Übersetzen des Codes notwendig ist, können wir Programmieren und Testen nahtlos miteinander verbinden. (Großer Vorteil für Anfänger_innen!)
- **REPL** (Read-Evaluate-Print Loop) möglich: Wir können den Interpreter einfach starten und einzelne Befehle ausprobieren, statt ein komplettes (fehlerfreies) Programm schreiben und kompilieren zu müssen. (Großer Vorteil für Anfänger_innen!)

Kompilierte Sprachen:

- Übersetzung erfolgt in einem separaten Schritt; erst der kompilierte Code kann dann als Programm ausgeführt werden.
- Effekt: (Manche) Programmierfehler werden schon beim Kompilieren gefunden.
- Kompilierte Sprachen sind teilweise effizienter, da der Quellcode nur einmal gelesen und umgewandelt werden muss.

Wir nutzen Python als interpretierte Programmiersprache. Um Pythoncode ausführen zu können, müssen wir also einen **Python-Interpreter** installieren.

Schreiben können wir den Code in einem beliebigen Texteditor. Professionelle Editoren haben meist eingebaute Tools zum Ausführen von Interpretern.

Dieses Jupyter-Notebook kann übrigens Pythoncode direkt ausführen! Probieren Sie es einfach mal aus: Klicken Sie auf den Programmiercode im Kästchen unter dieser Zeile und drücken Sie dann `Ctrl + Enter`. Unterhalb des Kästchens erscheint dann das Ergebnis des Pythonbefehls. Wenn Sie mögen, können Sie den Code auch verändern. Dazu klicken Sie auf das Kästchen und tippen etwas anderes ein, bevor Sie das Kästchen mit `Ctrl + Enter` ausführen.

In [ ]:

```
print("Führ dich aus!")
```

Wenn der Befehl `print()` ausgeführt wird, zeigt der Interpreter den Inhalt der Klammern einfach an. Im Fall oben war das ein vordefinierter Text. Sie können aber auch komplexe Ausdrücke `print()` en: Dann wird der Inhalt der Klammern zuerst ausgewertet und danach wird `print()` auf das Resultat angewendet. Probieren Sie es aus und ändern Sie wieder den Inhalt der Klammern, damit ein anderes Ergebnis angezeigt wird:

In [ ]:

```
print(3 + 8)
```

In [ ]:

```
print("Hallo " + "Welt")
```

Der Interpreter versucht jede Zeile Ihres Codes zu übersetzen und auszuführen. Das ist problematisch, wenn Sie sich verprogrammieren...

In [ ]:

```
print(Hallo Welt!)
```

... oder wenn Sie sich Notizen in Ihrem Programm machen möchten.

In [ ]:

```
meinen Namen ausgeben:
print("Esther Seyffarth")
```

Notizen im Code sind aber dringend zu empfehlen, gerade in der ersten Zeit, damit Sie sich merken können, was Ihr Code bedeutet.

Notizen in Programmcode heißen **Kommentare** und können auf zwei verschiedene Weisen markiert werden.

In [ ]:

```
# meinen Namen ausgeben
print("Ben Burkhardt")
```

In [ ]:

```
"""
Ausgabe:
- mein Name
- meine Emailadresse
"""
print("Esther Seyffarth")
print("esther.seyffarth@hhu.de")
```

Wie Sie sehen, werden die Kommentare vom Interpreter einfach übersprungen. Kommentare, die mit `#` beginnen, gelten bis zum nächsten Zeilenumbruch. Kommentare, die mit `"""` oder `'''` beginnen, gelten über Zeilenumbrüche hinweg, bis das nächste Mal `"""` bzw. `'''` gefunden wird. An der Farbe der Schrift erkennen wir übrigens, dass Kommentare nicht wie Programmcode interpretiert werden:

In [ ]:

```
print("kein Kommentar!")
#print("kein Kommentar!")

############################
# print("kein Kommentar!") #
############################
```

Einige Gründe, Kommentare in Ihrem Pythoncode aktiv zu nutzen:

- Als Erinnerung für später, was der Effekt einer Zeile ist.
- Als Erklärung für die Dozierenden, wenn die Zeile fehlerhaft programmiert ist, aber die Idee richtig war.
- Um während des Programmierens den Überblick zu wahren: Ein Kommentar ist schnell geschrieben und hilft Ihnen, sich zu merken, was an welcher Stelle passieren soll.

Software für diesen Kurs

Wir verwenden Jupyter-Notebooks (mehr Infos (http://jupyter.org/)), um die Vorlesungsinhalte zu präsentieren und kleine Übungen zwischendurch auszuführen. Für die Übungsaufgaben in den Mittwochssitzungen verwenden wir stattdessen den Texteditor **Visual Studio Code (VSCode)** (Link zur Webseite von VSCode (https://code.visualstudio.com/)).

Python-Interpreter installieren

Sowohl Jupyter als auch VSCode müssen auf einen Python-Interpreter zugreifen können, der auf Ihrem System installiert ist, damit Pythoncode ausgeführt werden kann. Auf den Rechnern im PC-Pool ist Python bereits installiert. Für Ihre eigenen Geräte können Sie Python von der Python-Webseite (https://www.python.org/downloads/) herunterladen und installieren. Für Fortgeschrittene empfehlen wir die Python-Installation Anaconda (Link zum Download (https://www.anaconda.com/download/)), mit der eine Menge zusätzliche Pakete eingerichtet werden, mit denen Sie komplexere Programmieraufgaben bearbeiten können. Diese Pakete brauchen wir in diesem Kurs noch nicht.

Es gibt zur Zeit eine stabile Version von Python 2 und mehrere stabile Versionen von Python 3. Letzteres wird noch aktiv entwickelt, während Python 2 nur noch aus historischen Gründen existiert: Alte Server oder laufende Systeme, die mit Python 2 gebaut wurden, sind teilweise schwer auf Python 3 umzustellen. Neue Projekte sollten aber immer von Anfang an in Python 3 gebaut werden. Auch in diesem Kurs benutzen wir Python 3.

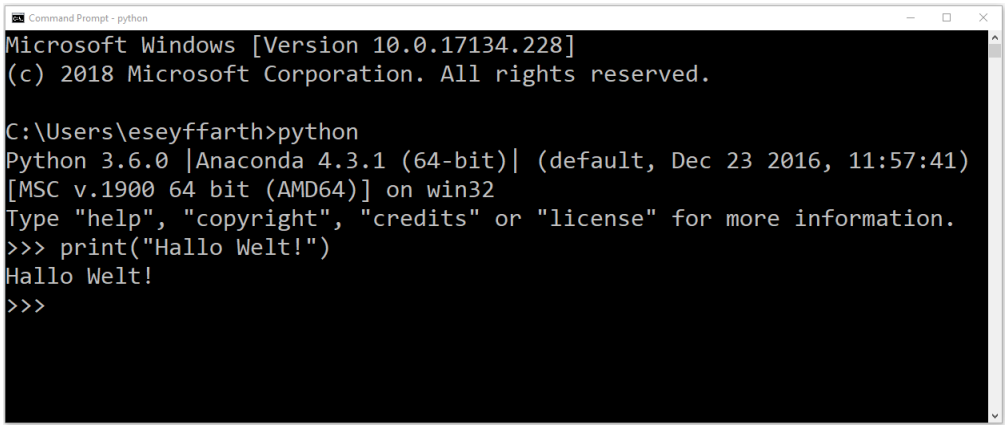
Wenn Sie sich Python zuhause installieren, achten Sie darauf, eine Python-3-Version zu wählen.

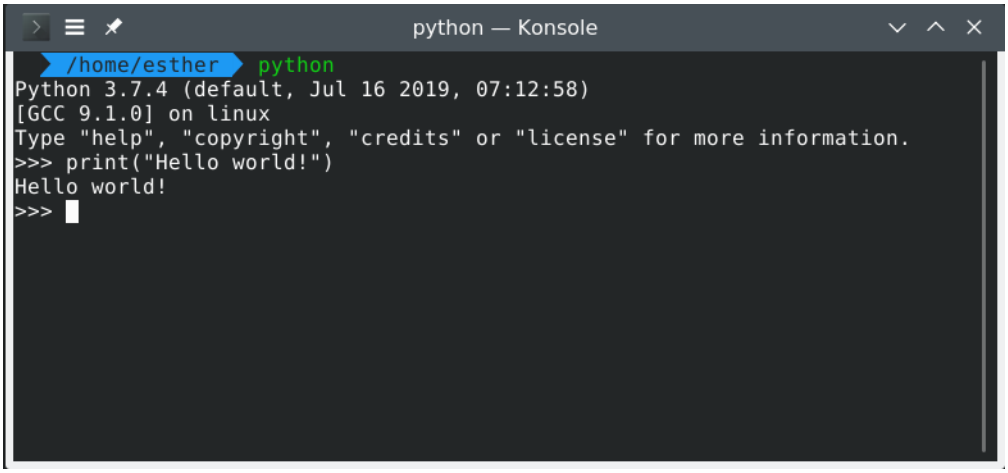
Weitere Infos zu Python 2 vs. Python 3 finden Sie hier (https://wiki.python.org/moin/Python2orPython3) (englisch).

Terminal, Command Line, Command Prompt, Konsole, Kommandozeile, Shell

Das Terminal (https://de.wikipedia.org/wiki/Kommandozeile) (auch unter all den oben aufgelisteten Namen bekannt) ist ein Teil Ihres Betriebssystems und ermöglicht das Ausführen systemnaher Befehle. Typische Beispiele sind das Navigieren durch Ordner auf der Festplatte, das Erstellen, Bearbeiten und Löschen von Dateien oder das Ausführen von bestimmten Programmen.

Auch Python kann vom Terminal aus gestartet werden. Allerdings ist das nicht sehr komfortabel:

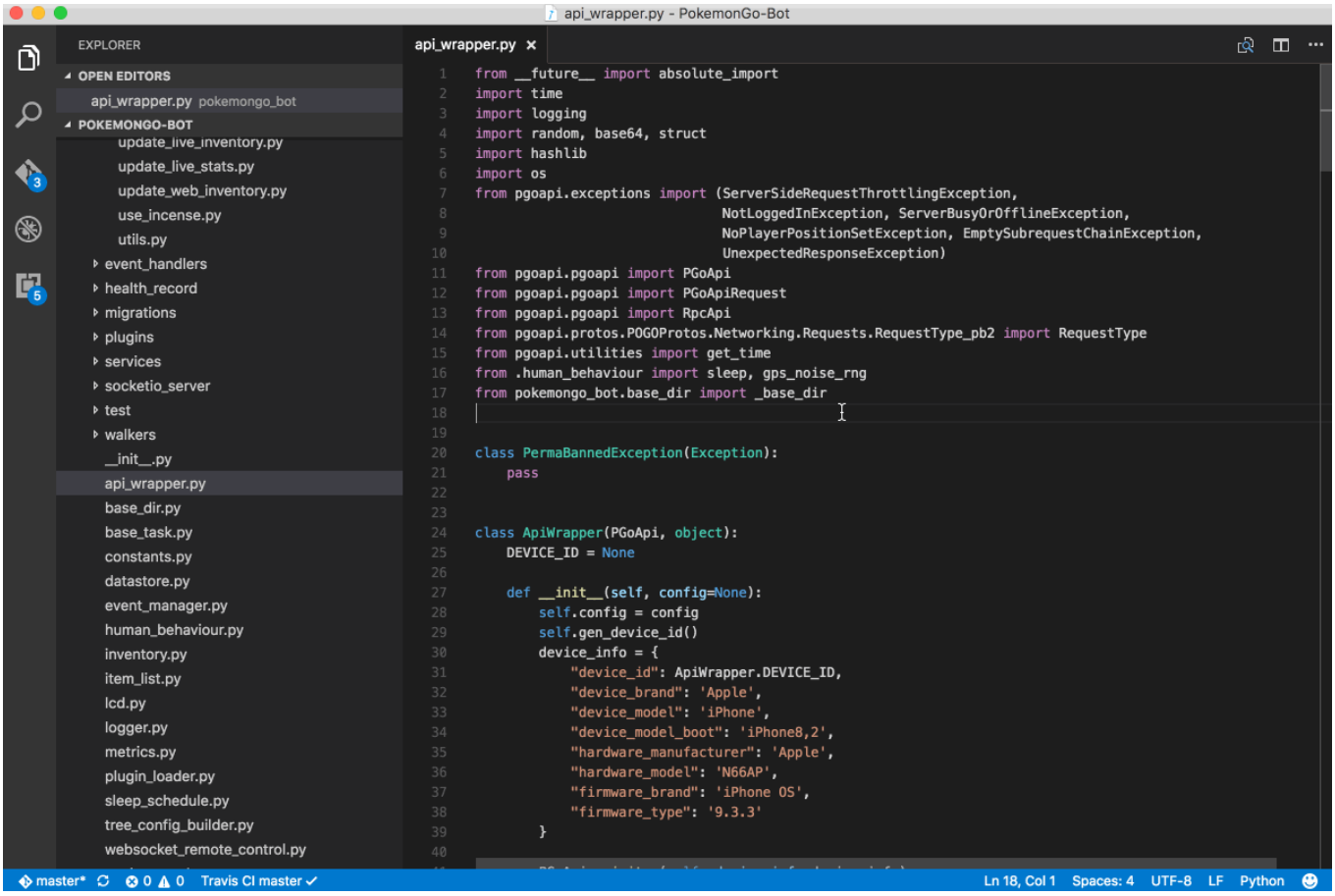




Wir haben zwar Zugriff auf alle Pythonbefehle, aber es fehlen komfortable Möglichkeiten zum Editieren von Dateien, der Code wird nicht eingefärbt und so weiter.

Zum Programmieren benutzen wir deshalb stattdessen Visual Studio Code. Das ist ein Editor mit eingebauter Terminal-Einbindung: Wir können den Komfort des Editors nutzen und bei Bedarf auf Funktionen des Terminals zugreifen.

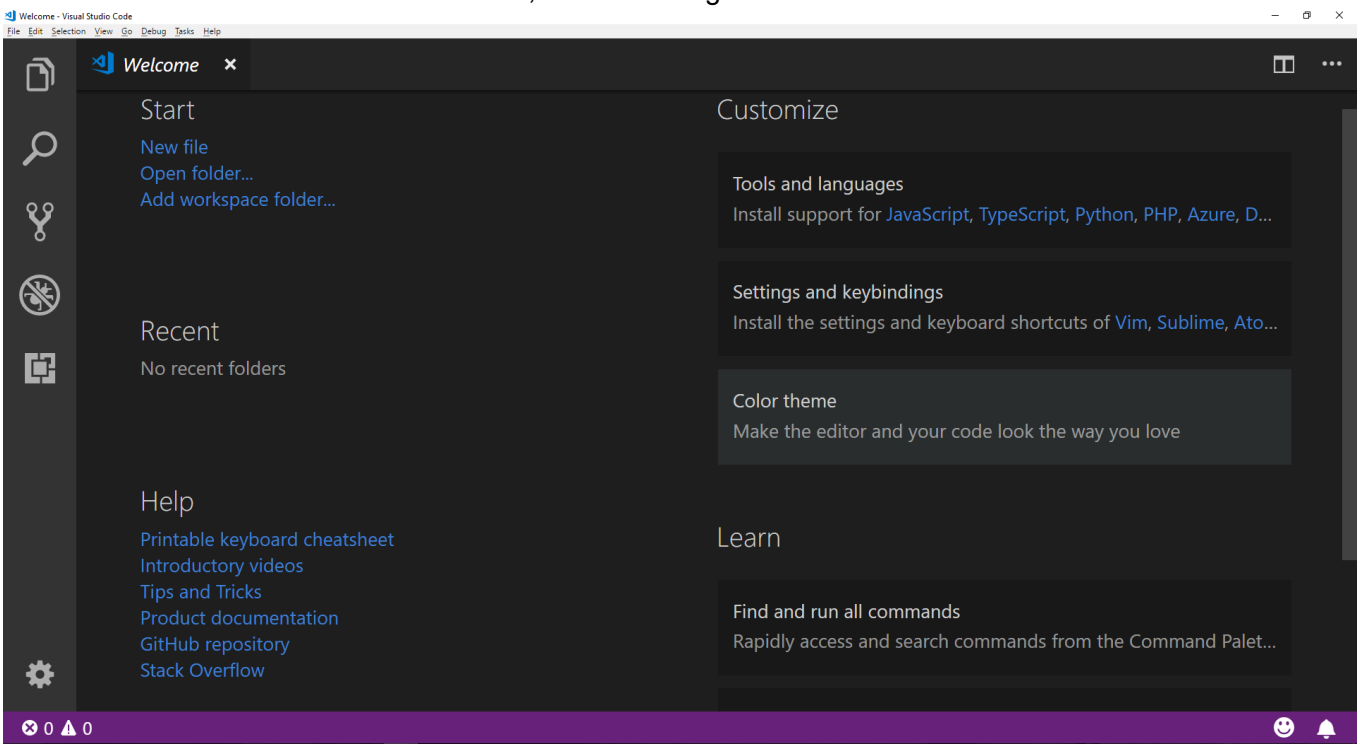
Editor: VSCode



Pythonprogramme werden in Form von Text geschrieben und interpretiert. Theoretisch können Sie also jeden Texteditor verwenden, z.B. Notepad++, TextEdit, gedit, Atom, Sublime Text, ...

VSCode eignet sich für diesen Kurs, weil die Verknüpfung zwischen Editor und Python-Interpreter komfortabel und unproblematisch ist. So können wir im Editor unseren Code schreiben und ihn auch im Editor direkt ausführen. Der Code wird als Datei mit der Endung `.py` abgespeichert.

Wenn Sie VSCode zum ersten Mal starten, sehen Sie folgendes:



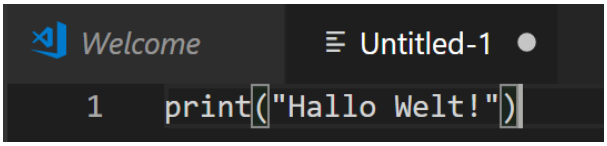
Wenn Sie auf Ihrem eigenen Gerät VSCode installieren, können Sie von der Welcome-Page aus den Support für Python aktivieren. Sobald das Fenster nach der Installation der Python-Extension neu geladen ist, können wir loslegen.

Fangen wir an mit einem unserer Hallo-Welt-Programme von oben. Dazu öffnen wir in VSCode eine neue Datei (`File -> New File` oder `Ctrl + N`).

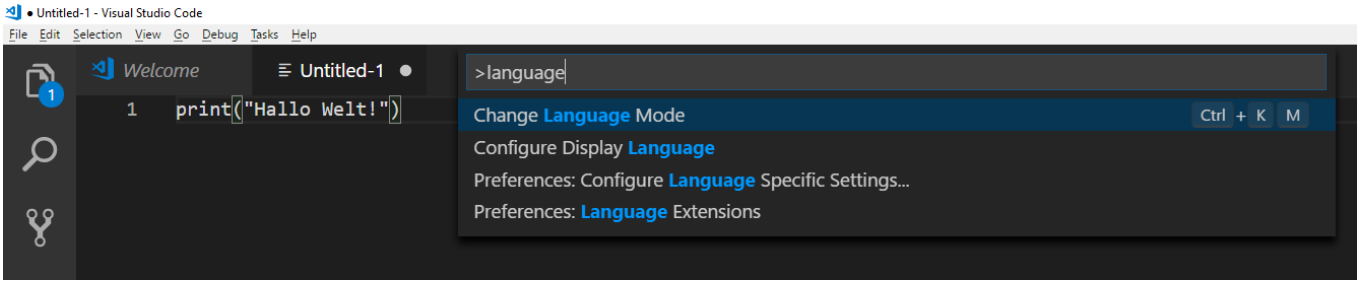
Schreiben Sie im Editor Ihr eigenes Hallo-Welt-Programm:

```
print("Hallo Welt!")
```

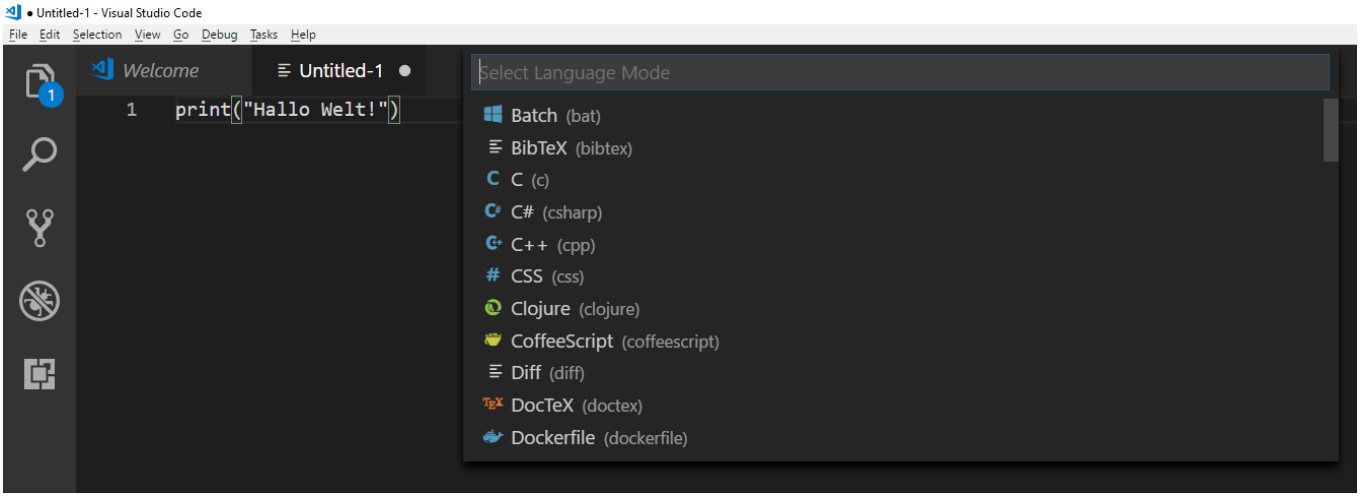
Editoren für Programmierung zeigen Befehle in der jeweiligen Sprache typischerweise in bestimmten Farben an, so wie hier im Jupyter Notebook. In VSCode ist der Code zunächst noch einfarbig:



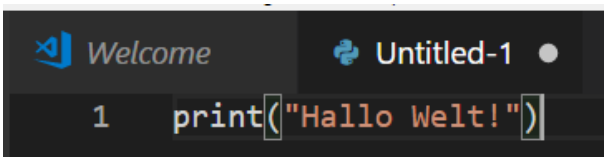
Hier kommt zum ersten Mal Editor-Magie ins Spiel! Drücken Sie **`Ctrl + Shift + P`** , um in VSCode die sogenannte **`Command Palette`** zu öffnen. Hier können wir nach Einstellungen suchen und sie direkt interaktiv ändern. Zum Beispiel wollen wir, dass die Einfärbung des Codes den Regeln für Python-Code folgt. Wir öffnen die Command Palette und geben ein:



Bestätigen wir die Auswahl des obersten Befehls (`Change Language Mode`) mit Enter, bekommen wir eine Reihe von Sprachen zur Auswahl.



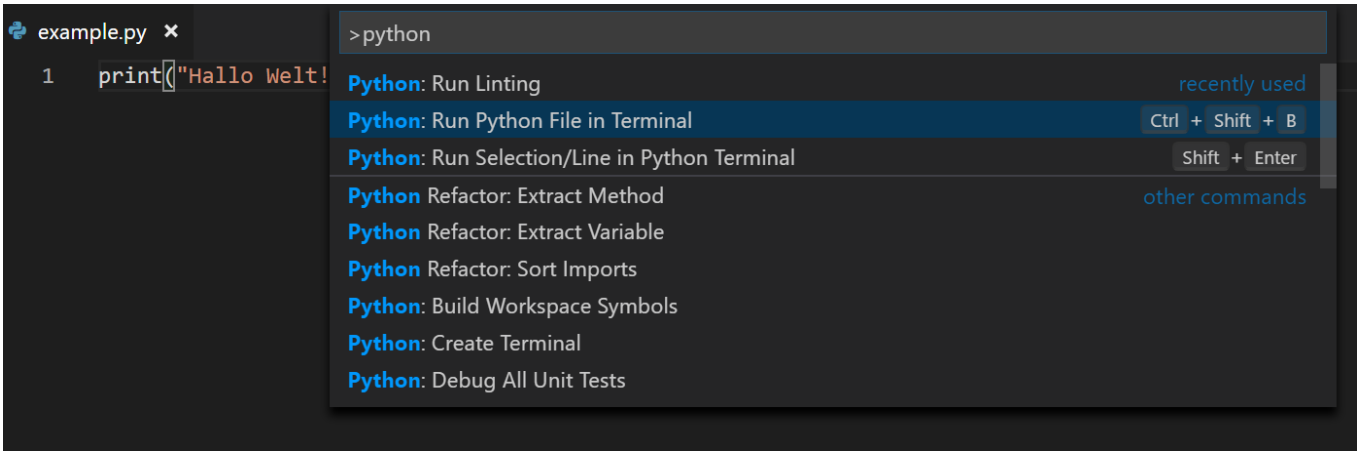
Mit den Pfeiltasten können wir zu `Python` navigieren. Noch schneller geht es, wenn wir `Python` eintippen und die Auswahl mit Enter bestätigen. Sofort wird der Code eingefärbt. Ihre Datei sieht jetzt so aus:



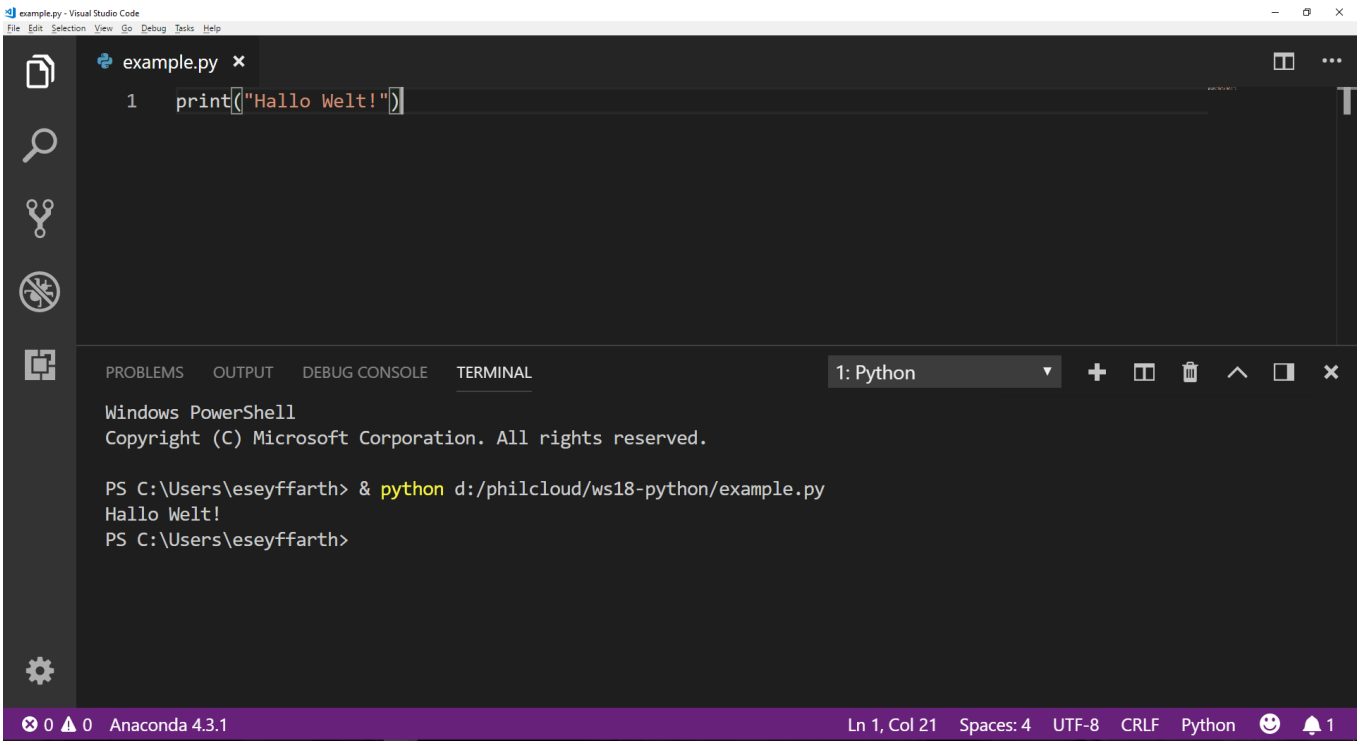
Den gleichen Effekt erzielen wir übrigens, wenn wir die Datei unter einem Namen wie `2018-10-09_Beispiel-01.py` speichern. Dann müssen wir aber die Endung `.py` manuell eingeben. Wenn wir in der Command Palette den Sprachmodus für Python einstellen und danach die Datei speichern, wird die Endung `.py` automatisch beim ersten Speichern ausgewählt.

Der Python-Interpreter kann übrigens nur gespeicherte Dateien ausführen. Wenn die Datei bereits einen Namen hat, speichert der Interpreter die letzten Änderungen automatisch, sobald der Code ausgeführt wird.

Um den Code zum Interpreter zu schicken, öffnen wir wieder die Command Palette und geben `python` ein, um alle Aktionen zu sehen, die für Python zur Verfügung zu stehen:



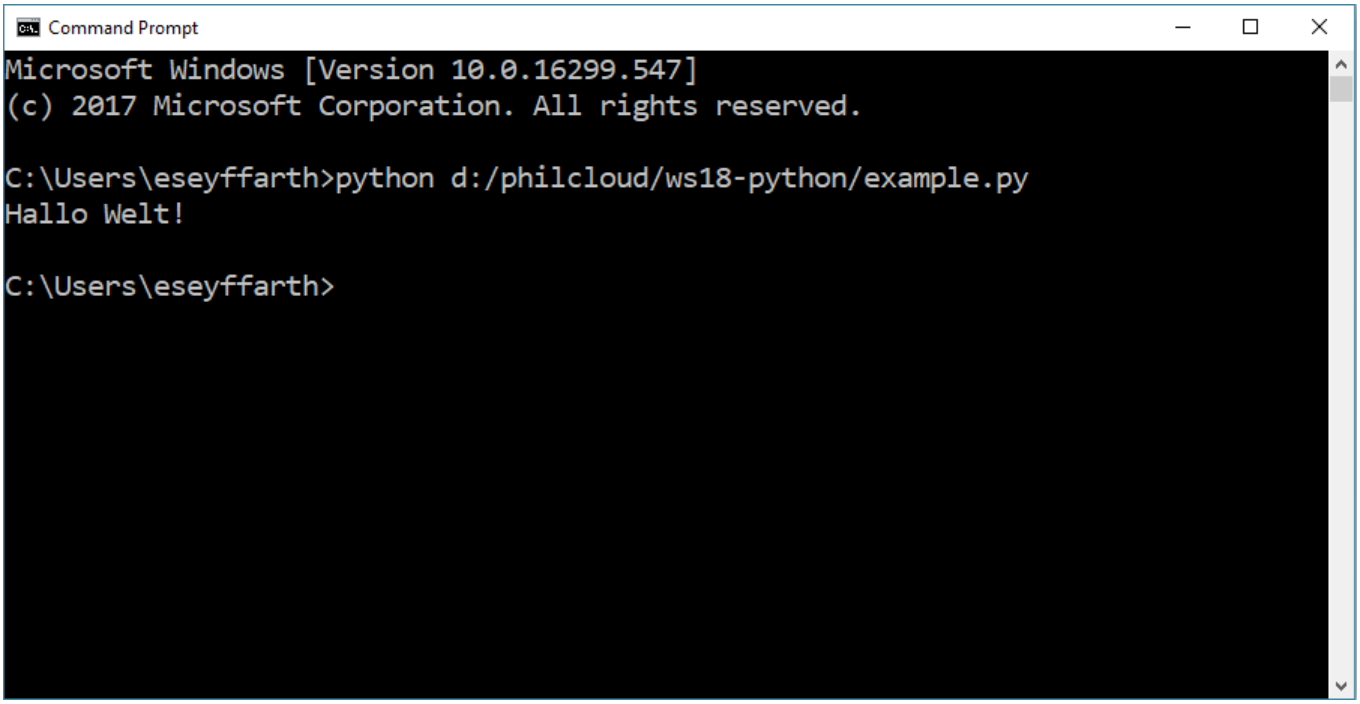
Der im Bild markierte Befehl `Python: Run Python File in Terminal` führt den Code der aktuellen Datei aus. Sie sehen, dass sich ein Terminal unten im Editor öffnet, das Ihre Ausgabe anzeigt:



Im Screenshot oben sehen wir, dass VSCode bei mir auf die Windows PowerShell zugreift (eins der verfügbaren Terminals in Windows).

Um eine Kommandozeile außerhalb des Editors zu öffnen, können Sie z.B. `[Windows Key] + R` drücken, im erscheinenden Fenster `cmd` eingeben und den Befehl mit Enter bestätigen.

Im Terminal können Sie auch gespeicherte Pythonprogramme ausführen. Das ist natürlich viel weniger komfortabel als in einem Editor wie VSCode. Technisch gesehen ist das Terminal im Editor aber nur ein **Wrapper** für die Konsole des Betriebssystems - es sieht ein bisschen anders aus, basiert aber intern auf der gleichen Software.



Ab jetzt verwenden wir zum Schreiben und Ausführen von Python-Code VSCode (oder das Jupyter Notebook).

Aufgaben:

1. Finden Sie heraus, wie Sie in VSCode einen anderen Farbstil einstellen können. Wählen Sie einen Stil aus, der Ihnen gefällt. Tipp: Farbstile in VSCode heißen `Color Themes`.
2. Stellen Sie in VSCode einen Keyboard Shortcut ein, mit dem Sie Ihr Hallo-Welt-Programm ausführen können. (Im Screenshot weiter oben sieht man, dass mein Shortcut `Ctrl + Shift + B` ist.)
3. Eventuell erscheint es auf den ersten Blick unübersichtlich, dass wir so viele unterschiedliche Programme verwenden, um Pythoncode zu schreiben. Zeichnen Sie für sich selbst ein Schaubild, das die Zusammenhänge zwischen Python-Interpreter, VSCode, Jupyter Notebook und Terminal visualisiert.
4. In VSCode können Sie alle Elemente größer oder kleiner anzeigen lassen, indem Sie das ganze Fenster hinein- oder herauszoomen. Dazu können Sie `Ctrl + -/+` drücken. Stellen Sie eine Schriftgröße ein, die für Sie angenehm ist.

Achtung: Einstellungen, die Sie in der VSCode-Installation im PC-Pool ändern, werden beim Neustart der Computer jedes Mal zurückgesetzt. Sie können am Anfang jeder Übungssitzung Ihre Konfiguration wieder einstellen. Dazu können Sie sich in einer separaten Datei ein "Rezept" notieren, damit Sie sich später erinnern. Arbeiten Sie insgesamt viel mit der **Command Palette**, um von Konfigurationsänderungen so wenig wie möglich betroffen zu sein.

Über diesen Kurs (Organisatorisches)

Herzlichen Glückwunsch, Sie haben Ihr erstes Pythonprogramm erfolgreich geschrieben und ausgeführt! Das restliche Semester werden wir damit verbringen, Ihre Programmierfähigkeiten weiter zu schulen, sodass Sie zum Schluss mit einigen typischen Anwendungsfällen vertraut sind und auch mit neuen Herausforderungen umgehen können.

In Ihrem Studium erbringen Sie zwei Arten von Leistungen: Beteiligungsnachweise (BNs) und Abschlussprüfungen (APs). Zu APs müssen Sie sich jeweils explizit anmelden. BNs für einen Kurs erhalten Sie, wenn Sie am Kurs erfolgreich teilgenommen haben.

In diesem Kurs erwerben Sie einen BN. Der BN ist an bestimmte Bedingungen geknüpft. Der Kurs gehört zum Modul P (Propädeutik der CL). Sie können den BN erwerben, wenn...

- Sie noch keinen anderen BN und keine andere AP in einem Kurs erworben haben, in dem die Programmierung mit Python im Mittelpunkt stand.
- Sie im Laufe des Semesters ausreichend viele Übungsaufgaben korrekt lösen.

Das Ziel dieses Kurses ist es, dass Sie eigene praktische Erfahrungen mit Python sammeln. Dafür ist es **nicht** erforderlich, dass Sie an jeder Vorlesung teilnehmen; es gibt keine Anwesenheitspflicht. Wir erwarten natürlich, dass Sie sich die Inhalte der Vorlesungen in dem Maß aneignen, das nötig ist, um die Übungsaufgaben zu lösen.

Dienstags von 16:30 bis 18:00 finden die Vorlesungen statt. Hier werden neue Inhalte präsentiert und im Rahmen kleinerer Übungen ausprobiert. Die Übungen in den Dienstagsterminen sind nur als Unterstützung zum Lernen gedacht und werden nicht überprüft.

Mittwochs von 16:30 bis 18:00 finden die Übungssitzungen statt. Hier bearbeiten Sie die Übungen der aktuellen Woche. Die Dozierenden notieren sich, wer welche Übungen erfolgreich gelöst und vorgezeigt hat.

Im Normalfall müssen Sie die Hausaufgaben nicht abgeben. Es reicht aus, uns in der Übungssitzung die Lösung am Computer zu zeigen.

Der Kurs ist so angelegt, dass Sie die Übungsaufgaben gut bearbeiten können, wenn Sie an der dazugehörigen Vorlesung teilgenommen haben oder die Vorlesungsunterlagen (Jupyter Notebooks) aufmerksam durchlesen.

Den BN bekommen Sie automatisch, wenn Sie am Ende des Semesters mindestens 50% der praktischen Übungen aus den Mittwochssitzungen erfolgreich gelöst haben. Wir empfehlen, in jeder Übung die Hälfte der aktuellen Aufgaben zu lösen. Wenn Sie ein Thema noch intensiver üben möchten, können Sie auch alle Aufgaben einer Woche bearbeiten.

Wenn Sie die Aufgaben abschreiben, bekommen Sie am Ende den BN, haben aber nicht Programmieren gelernt. Es ist deutlich hilfreicher für Sie, wenn Sie selbst versuchen, die Aufgaben zu lösen, und bei Schwierigkeiten Fragen an die Dozierenden oder an Ihre Kommiliton_innen stellen.

Und hier noch einige Tipps...

- Benennen Sie Ihre Pythondateien mit sprechenden Namen! Sie können beispielsweise das Datum im Dateinamen angeben, Stichworte aus der Aufgabenstellung verwenden oder die Übungen durchnummerieren. Sie werden das ganze Semester immer wieder auf alte Dateien zurückgreifen. Je übersichtlicher Ihre Dateien sortiert sind, umso einfacher wird das.

- Sammeln Sie Ihre Pythondateien an einem zentralen Ort! Die PC-Pool-Rechner sind dafür ungeeignet, weil gespeicherte Dateien dort bei jedem Neustart gelöscht werden. Sie können einen USB-Stick verwenden, um Dateien immer mitzunehmen, oder Services wie Roundcube, Dropbox, Google Drive etc. verwenden, um Ihre Dateien zwischen Geräten zu synchronisieren. Im Zweifelsfall können Sie sich die Dateien selbst per Mail zuschicken. Es steht Ihnen auch frei, einen eigenen Laptop mitzubringen, wenn Ihnen das möglich ist.

- Tauschen Sie sich aus! Sie sollen die Übungen zwar eigenständig lösen, aber oft hilft es, wenn man mit jemandem darüber spricht, wie die Lösung aussehen muss. Wenn Sie mögen, suchen Sie sich eine_n Partner_in für regelmäßiges Rubber Duck Debugging (https://en.wikipedia.org/wiki/Rubber_duck_debugging).

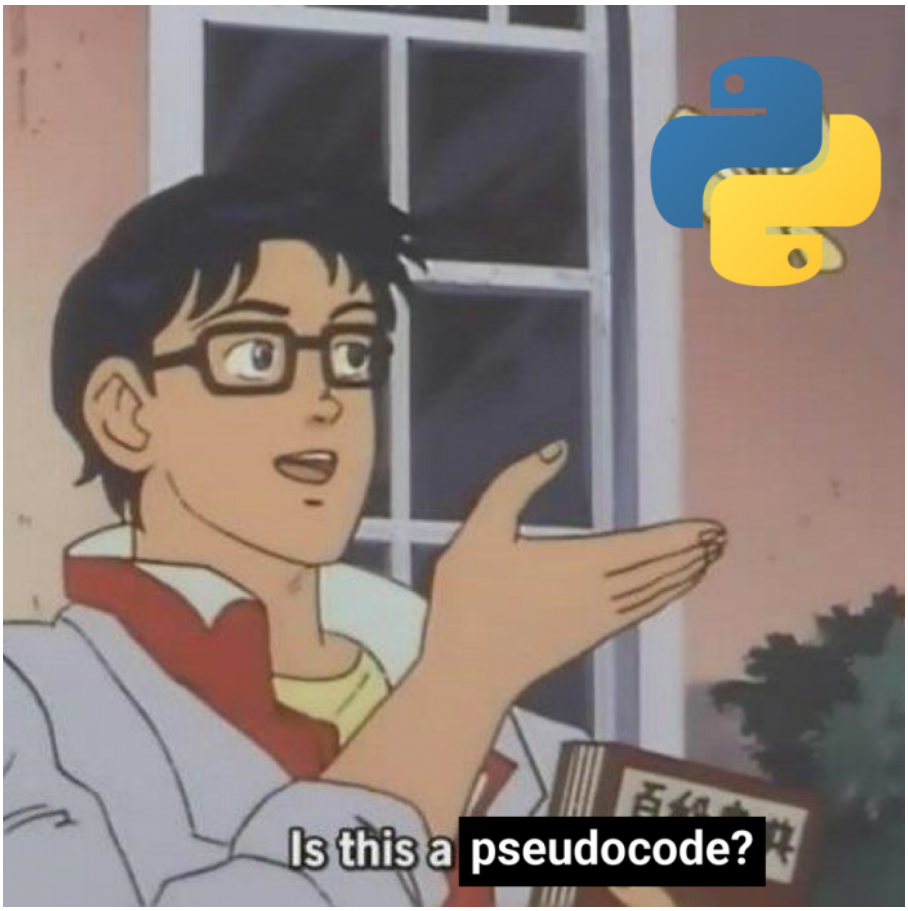
- Ihr_e Partner_in kann z.B. in jeder Übungssitzung die Aufgaben bearbeiten, die Sie *nicht* bearbeiten. So haben Sie zum Schluss alle Aufgaben abgedeckt und können sich gegenseitig berichten, welche Schwierigkeiten Ihnen begegnet sind oder welche Strategien Sie angewendet haben.

- Stellen Sie Fragen! Der Großteil von Ihnen fängt gerade erst an zu programmieren und es ist sehr wahrscheinlich, dass die Dinge, die Sie nicht vollständig verstehen, auch für die anderen schwierig sind. Wenn Sie die Fragen nicht vor allen stellen möchten, können Sie sie zuerst mit Ihrer Rubber-Duck-Partner_in besprechen. Alternativ können Sie mit diesem Formular (https://forms.gle/sw7gN1PMmEG6GX1WA) anonyme Fragen an die Dozierenden schicken. Die Fragen werden dann in der aktuellen Sitzung oder in der folgenden Sitzung im Plenum besprochen.

Leitfaden zum Verhalten im Kurs

Dieser Kurs folgt dem Leitfaden von Esther Seyffarth, in dem das erwünschte Verhalten bei Lehrveranstaltungen beschrieben wird. Sie finden den Leitfaden hier auf deutsch (https://user.phil.hhu.de/~seyffarth/classes/leitfaden.pdf) und hier auf englisch (https://user.phil.hhu.de/~seyffarth/classes/guidelines.pdf). Falls Sie nach der Lektüre des Leitfadens Fragen haben, sprechen Sie uns gerne an.

Variablen in Python


Is this a pseudocode?

Als nächstes beschäftigen wir uns damit, wie Informationen in Pythonprogrammen gespeichert und während der Laufzeit verändert werden können. Dazu erzeugt man im Programmcode **Variablen**, die bestimmte Werte annehmen können und die man mit bestimmten Befehlen verändern kann.

Wird eine Variable im Programmcode aufgerufen, beispielsweise mit `print()`, bezieht der Aufruf sich immer auf den Wert, der in der Variable gerade gespeichert ist. Der Interpreter bewegt sich von oben nach unten durch das Programm und führt nacheinander alle Befehle aus.

Wir erzeugen zuerst eine Variable, der wir einen Namen geben:

```
aktuelle_zahl = 0
```

Wenn wir jetzt `print(aktuelle_zahl)` aufrufen, wird der aktuelle Wert der Variable ausgegeben.

Sie können den Wert im Kästchen unten verändern und den Code dann mit `Ctrl + Enter` ausführen, um zu sehen, dass immer der aktuelle Wert der Variable angezeigt wird.

In [ ]:

```
aktuelle_zahl = 0
print(aktuelle_zahl)
```

Wir können den Wert der Variable beliebig oft überschreiben, indem wir mehrere Wertzuweisungen im Code einfügen. Nach jeder Änderung wird beim Aufruf von `print()` der zuletzt zugewiesene Wert angezeigt:

In [ ]:

```
aktuelle_zahl = 0
print(aktuelle_zahl)

aktuelle_zahl = 1
print(aktuelle_zahl)

aktuelle_zahl = "das ist gar keine Zahl!"
print(aktuelle_zahl)

aktuelle_zahl = 5/2
print(aktuelle_zahl)
```

Wie Sie sehen, können wir der Variable `aktuelle_zahl` ganz unterschiedliche Werte zuweisen. Welchen Typ die Variable gerade hat, können wir mit dem Befehl `type(aktuelle_zahl)` ermitteln. Das Ergebnis dieses Aufrufs wird dann noch in einen Print-Befehl verschachtelt. Probieren Sie es aus und ergänzen Sie im Kästchen oben nach jeder Änderung der Variable einen Aufruf von `print(type(aktuelle_zahl))`!

Sie sehen, dass die Variable zunächst zum Typ `<class 'int'>` gehört, später dann zu den Typen `<class 'str'>` und `<class 'float'>`. Es gibt noch mehr grundlegende Datentypen in Python -- hier eine Auswahl:

Kurzbezeichnung	Bedeutung	Beispiel
`int`	Integer (ganze Zahl)	`zahl = 5`
`str`	String (Zeichenkette)	`wort = "Python"`
`float`	Float (Kommazahl)	`kommazahl = 3.1415`
`list`	Liste (geordnete Sequenz von Werten)	`liste = [1,2,3]`
`dict`	Dictionary (Sequenz von Wertpaaren mit eindeutiger Zuordnung)	`lexikon = {"dog": "Hund"}`
`bool`	Wahrheitswert/Boolean	`True` , `False` (Großschreibung beachten!)

Aufgabe

1. Können Sie mit Hilfe der Tabelle vorhersagen, welche Ausgabe jede Zeile im Code haben wird? Denken Sie über jedes Beispiel zuerst nach und führen Sie es dann aus, um zu prüfen, ob Sie richtig liegen.

In [ ]:
```
print(type("Nie ohne Seife waschen"))
```

In [ ]:
```
print(type(6))
```

In [ ]:
```
print(type(["Wer", "Summen", "kürzt", "der", "ist", "ein", "Schaf"]))
```

In [ ]:
```
print(type(4-9))
```

In [ ]:
```
print(type("Hallo" + " " + "Welt"))
```

In [ ]:
```
print(type({"S": "NP VP"}))
```

Zusammenfassung

Sie haben heute gelernt,

- Wie Sie mit Python "Hallo Welt" ausgeben können
- Wie Sie in VSCode Einstellungen ändern und Programme schreiben
- Wie Sie in VSCode Programme ausführen
- Wie Sie in Jupyter Notebooks Codebeispiele bearbeiten und ausführen können
- Dass bei verschachtelten Befehlen in Pythoncode immer zuerst der innere Befehl interpretiert wird, dessen Ergebnis dann für die Interpretation der äußeren Befehle verwendet wird (vgl. `print(3 + 5)`)
- Wie Kommentare in Pythoncode markiert werden
- Wie der Python-Interpreter, VSCode und das Terminal miteinander zusammenhängen
- Wie Sie in Python Variablen erzeugen und Werte in Variablen speichern können
- Wie Sie den Datentyp eines Werts ermitteln können

Und morgen...

Am 09.10. findet die erste Übungssitzung statt. Wenn Sie an Ihrem eigenen Gerät arbeiten möchten, versuchen Sie bitte bereits vor der Übung, die Software zu installieren, die wir benötigen (Python, VSCode). Eventuelle Installationsschwierigkeiten können wir dann in der Übungssitzung zusammen klären.

In der ersten Übungssitzung lernen Sie den Editor näher kennen und schreiben einige erste Pythonprogramme.

Die Übungssitzung findet in den Räumen **24.21.03.62-64 und 24.21.03.61** statt. Es gibt in beiden Räumen PC-Plätze.