Einführung in Python

Dieser Kurs ist eine Einführung ins Programmieren für Leute, die bisher wenig bis keine Erfahrung damit haben. In vier Workshops gibt es einen Überblick über Python. Die Teilnehmenden programmieren ein Spiel (Pong).

Steckbrief

Vier aufeinander aufbauende Workshops, je 90 Minuten.

Zielgruppe

fünf bis 15 Lernende ohne Programmiererfahrung; 12 bis 16 Jahre

Kursleitung

ein bis drei Betreuende
Plattform: BigBlueButton, anpassbar auf Präsenzworkshops

Lizenz

Kategorie

Coding

Online-Version mit Links zu weiteren zugehörigen Dokumenten

https://jugendhackt.org/oer/projekte/python-einfuehrung/

Übersicht für die Kursbetreuung

Der Kurs ist in vier 90-minütige Workshops aufgeteilt, die beispielsweise im Wochenabstand gehalten werden können.

Die Kursbetreuung besteht aus einer, besser aber aus zwei oder drei Personen, die die Konzepte erklären und für Fragen im Chat zur Verfügung stehen. Wenn es mehrere Betreuer*innen gibt, können auch kleinere Gruppen in Break-Out-Rooms aufgeteilt werden.

Für die Hausaufgaben und mögliche Kommunikation zwischen den Workshops ist eine Lernplattform sinnvoll. Das kann aber auch eine Seite auf Gitlab sein; es geht primär darum, einen Ort zu haben, an dem eins die Aufgaben und Codedateien nachschlagen kann. Die Hausaufgaben sind nicht zwingend erforderlich, sondern eher Übungsaufgaben, die tiefere Einblicke oder weitergehende Ideen ermöglichen. Bis auf die Hausaufgabe vom dritten zum vierten Workshop können sie ignoriert werden. Die Hausaufgabe vom dritten Workshop solltet ihr auf jeden Fall besprechen, weil sie Elemente für das Spiel liefert.

Findet der Kurs in Präsenz statt, könnt ihr einige Dinge konkreter veranschaulichen, zum Beispiel das Steuern der Turtle, die als Schildkröten-Figur über ein Spielfeld geschoben – oder sogar in Form einer Person durch den Raum gesteuert – werden kann. Den Inhalt müsst ihr nicht anpassen.

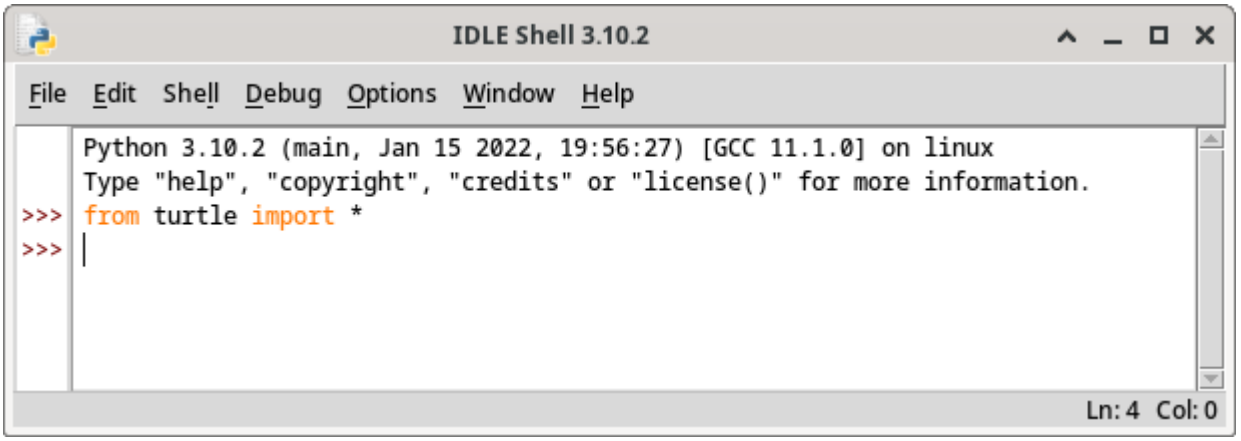
Kursablauf

Vorbereitung

Die Lernenden sollten auf ihren Rechnern Python 3 installiert haben.

> Dafür finden sich aktuelle Anleitungen und der Download-Link für die jeweiligen Betriebssysteme auf der Python-Website (in englischer Sprache). Bei einer Python-Installation ist die entsprechende Version der IDLE automatisch dabei. Um sicherzustellen, dass die Installation funktioniert hat, können die Lernenden die IDLE öffnen und dort (hinter die Eingabeaufforderung „>>>“) folgenden Text eingeben:
>
> ```
> from turtle import *
> ```
>
> Wenn dieses Kommando mit Enter abgeschickt wird und daraufhin lediglich in der nächsten Zeile eine weitere Eingabeaufforderung erscheint, hat alles funktioniert. Wenn hingegen eine Fehlermeldung erscheint, hat etwas nicht funktioniert. Das Problem könnt ihr eventuell beheben, indem ihr die Fehlermeldung und das Betriebssystem im Internet nachschlagt.

Andernfalls muss die Kursbetreuung noch vor dem Kursstart helfen können. Es wäre ungünstig, während des Kurses einzelne Lernende durch die Installation zu leiten. Deswegen solltet ihr die Teilnehmenden im Vorfeld an die Installation erinnern und technische Hilfestellung anbieten. Im Internet finden sich Python-IDLEs, die im Notfall verwendet werden können; diese sollten aber vorher darauf überprüft werden, ob sie mit dem Modul Turtle funktionieren.



Die IDLE („Integrated Development and Learning Environment“) ist ein Programm, das den Einstieg in Python erleichtert. Dort kann eins direkt und interaktiv kurze Kommandos ausprobieren, ohne gleich eine Programmdatei erstellen zu müssen.

Im Zusatzmaterial finden sich Erläuterungen zu den einzelnen Funktionen.

Workshop 1: Taschenrechner und Turtle

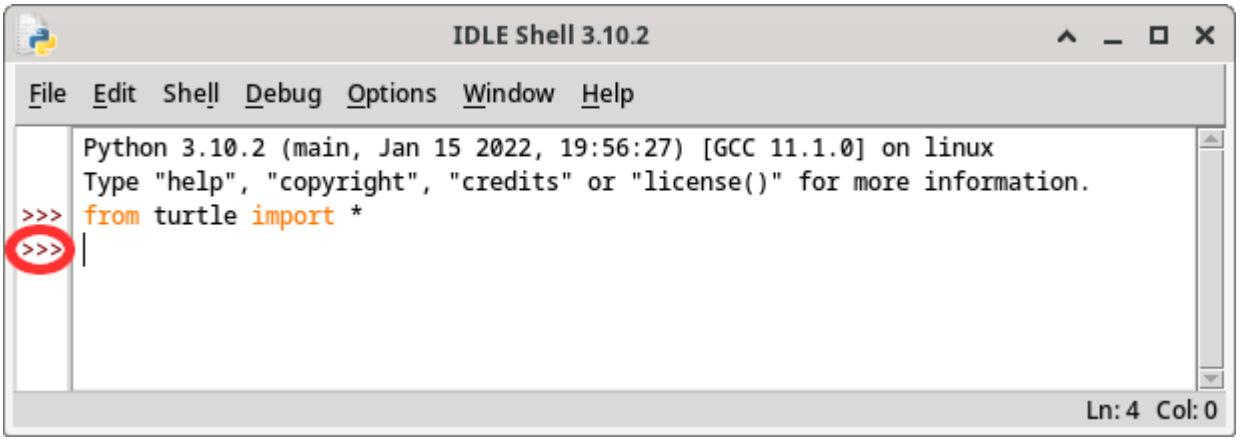
Einleitung

Die Betreuenden sollten vor Beginn des Workshops für technische Problemlösung ansprechbar sein. Bei der Begrüßung solltet ihr direkt überprüfen, ob die IDLE wirklich für alle funktioniert.

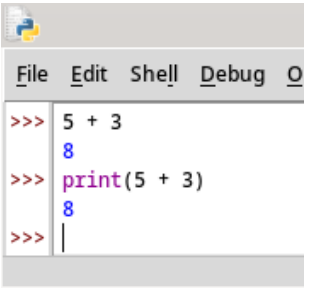
Taschenrechner

Nach der Vorstellungsrunde starten alle die IDLE und verwenden sie als Taschenrechner. Die Kursleitung teilt am besten den Bildschirm mit ihrer IDLE.

- kurz die Funktionsweise der IDLE erklären, besonders den Prompt:



Der Prompt (oder auch „Eingabeaufforderung“ sind die drei “>” am Anfang der Zeile. Er teilt uns mit, dass wir hier neue Dinge eingeben können und unterteilt unsere Befehle.



- Als Taschenrechner verwenden; erst Beispiele, dann kleine Aufgaben:

```
+, -,  *, /,
print(5 + 3)
```

- Zahlen in Variablen speichern, mit print ausgeben lassen, damit rechnen. Evtl. ohne vorher zu erklären ein paar Beispiele zum Experimentieren geben, zB:

```
a = 5
a + 3
print(a)
print(a + 3)
print(7 + (a*4))
```

- An dieser Stelle könnt ihr das erste Mal auf die Auswertungsreihenfolge hinweisen: Es wird zuerst das ausgewertet, was ganz innen ist, und dann immer weiter nach außen.
- Zuweisungen; Zuweisungsreihenfolge erklären, Kontrast zu Mathematischem „=“ betonen.

```
b = a + 2
print(b)

b = b + 2
print(b)
```

Turtle

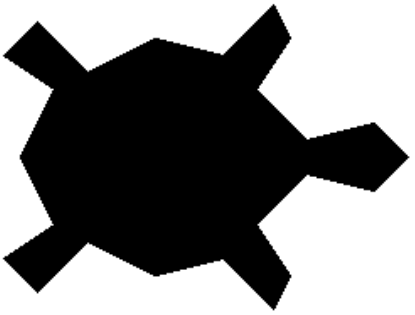
```
from turtle import *
```

Module werden im dritten Workshop genauer erklärt; jetzt solltet ihr nur erwähnen, dass diese Zeile für das Folgende notwendig ist. (Falls beim Import nun Fehlermeldungen auftreten, die sich nicht direkt klären lassen, solltet ihr diese in der Pause lösen.)

Dann folgende Kommandos anzeigen und einige Minuten lang ausprobieren lassen:

```
forward(100)
right(45)
left(90)
circle(50)
pencolor("blue")
pensize(5)
```

Als nächstes können die Kommandos mit Variablen und Ausdrücken (wie a + 5) verwendet werden.



Die Turtle ist ein Grafik-Modul zum Programmieren lernen. Manchmal sieht sie sogar auch aus wie eine Schildkröte!

Dann die offene Frage stellen, was das Kommando goto macht:

```
goto()
```

Hier soll die Fehlermeldung genau gelesen werden.

```
goto(100)
```

Auch hier sollen die Teilnehmenden die Fehlermeldung ansehen und kurz kommentieren.

```
goto(100, 30)
```

Eventuell ist es dann nötig, kurz auf das Konzept von Koordinaten einzugehen.

Nach dieser kurzen Knobelaufgabe folgende Funktionen erklären:

```
penup()
pendown()
clear()
undo()
```

Abschluss

Die Kursleitung fragt nach, ob es Fragen gibt. Falls keine auftauchen, könnt ihr per Bildschirmteilung kurz zeigen, wie eins bei Unklarheiten im Netz nach Antworten suchen kann.

Danach solltet ihr eure Lernplattform zeigen und mit den Teilnehmenden die Aufgaben bis zur nächsten Stunde kurz besprechen.

Nach der Stunde solltet ihr den Link zur Lernplattform und zu den aktuellen Aufgaben an die Lernenden schicken. Die Aufgaben und die Funktionenliste müssen eventuell an den Verlauf der Stunde angepasst werden.

Konzepte, die nach der Stunde klar sein sollten:

- Rechnen
- Variablen
- Turtle
- Dokumentation
- Fehlermeldungen

Workshop 2: Funktionen

Aufgaben besprechen

Die erste Aufgabe sollte nur kurz besprochen werden; für die Ergebnisse der zweiten Aufgabe findet sich im Optimalfall eine Möglichkeit für die Lernenden, die Bilder hochzuladen und zusammenzustellen, beispielsweise in einem Pad wie Hedgedoc.

Programmdateien

Der Übergang von der IDLE in die Programmdateien sollte Hands-on beginnen:

Neue Datei öffnen, dort Code eingeben und ausführen, beispielsweise:

```
print(5 + 2)
```

Dann den selben Code in der IDLE.

Erst danach sollte genauer erklärt werden, was es damit auf sich hat.

strings und print

Hier könnt ihr direkt erklären, was der neue Inhalt ist, dann sollten aber Beispiele folgen.

```
wort1 = "Hello"
wort2 = "world"
text = wort1 + wort2
print(text)
```

und

```
wort1 = "python"
print(wort1 * 3)
wort1 = wort1 * 4
wort1 = wort1 + "!!!"
```

Weitere Beispiele finden sich in der Datei **Workshop-Code.py** im Ordner zur zweiten Woche. Die Lernenden können auch versuchen, diese Beispiele selbst zu erklären oder zumindest Vermutungen darüber anstellen – dies wird sie zum einen aktivieren und mehr beteiligen, zum Anderen ist „Dinge sich selbst erklären“ eine wichtige Fähigkeit beim Programmieren lernen. Wenn wenig Beteiligung kommt, könnt ihr Quizzes oder Umfragen per BigBlueButton machen.

Wichtig ist, auf Zuweisungsreihenfolge (erst wird das, was auf der rechten Seite des „=“ steht, ausgeführt) und Auswertungsabfolge der Ausdrücke (von innen nach außen; dies wird später noch einmal angesprochen) hinzuweisen.

Funktionen

Was Funktionen sind, lernen die Teilnehmenden an den Beispielen int, str und len. Hier könnt ihr auch auf bisher verwendete Funktionen, beispielsweise von der Turtle, verweisen.

```
print("45" * 3)
```

vs.

```
print(45 * 3)
int("45") * 3
```

Weitere Beispiele finden sich in der Datei **Workshop-Code.py** im Ordner zur zweiten Woche, wo auch die Thematik der Auswertungsabfolge genauer in den Blick nehmen kann. Eines der Beispiele wird auch in WS2_Auswertungsreihenfolge.pdf genauer analysiert:

```
Auswertungsreihenfolge

print("Hallo " + name + "! In zwei Jahren wirst Du " + str(alter + 2) + " sein.")
                                                          15

print("Hallo " + name + "! In zwei Jahren wirst Du " + str(  15  + 2) + " sein.")
                                                            17

print("Hallo " + name + "! In zwei Jahren wirst Du " + str(   17    )  + " sein.")
                                                         "17"

print("Hallo " + name +  "! In zwei Jahren wirst Du " +   "17"         + " sein.")
               "Robin"

print("Hallo " +"Robin"+ "! In zwei Jahren wirst Du " +   "17"          + " sein.")
                    "Hallo Robin! In zwei Jahren wirst Du 17 sein."

print("Hallo Robin! In zwei Jahren wirst Du 17 sein.")
```

In der Datei WS2_Auswertungsreihenfolge.pdf zeigt sich, wie Python einen Ausdruck Stück für Stück, von innen nach außen, auswertet.

input

In **Workshop-Code.py** findet sich ein Beispiel, das alle zunächst ohne größere Erklärung ausführen können. Die auskommentierte Zeile, die zu einer Fehlermeldung führt, sollte – mit der Ankündigung einer Fehlermeldung – ausgeführt und die Fehlermeldung dann genau besprochen werden, sodass die Lernenden von selbst auf die Lösung kommen:

```
# print(alter2 + 2) # Fehlermeldung => int()
```

Verzweigungen

„Ok, aber was können wir jetzt mit dem Input anfangen? Wie könnten wir denn das Alter verwenden?“ Bestenfalls schlagen die Lernenden selbst eine Alterseinteilung vor, falls nicht, findet sich ein Beispiel in der Code-Datei.

Zufallszahlen

Mit der Ankündigung, dass wir nun ein Zahlen-Ratespiel bauen, kann die Kursleitung Module einführen: Sie stellen Funktionen bereit, die eins nur manchmal braucht.

```
import random
zahl = random.randint(0, 6)
print(zahl)
```

Hier können die Lernenden ein paar Zahlen ausgeben lassen. Wenn dann viel Zeit übrig ist, könnten die Lernenden selbst versuchen, das Spiel zusammenzubauen; andernfalls könnt ihr durch den Code (der sich in der Code-Datei findet) führen.

Abschluss

Die Lernplattform per Screenshare zeigen und mit den Teilnehmenden die Aufgaben bis zur nächsten Stunde kurz besprechen.

Nach der Stunde den Link zur Lernplattform und zu den aktuellen Aufgaben an die Lernenden schicken. Die Aufgaben und die Funktionenliste müssen eventuell an den Verlauf der Stunde angepasst werden.

Konzepte, die nach der Stunde klar sein sollten:

- Unterscheidung zwischen IDLE und Programmdateien
- strings vs. integers
- Syntax von Funktionsaufrufen
- Umgang mit Fehlermeldungen
- input()
- if, elif, else; „==“ vs. „=“
- Module

Anmerkung: for-Schleifen und anderes kamen bisher nicht vor. Wenn es in der Stunde aufkommt, solltet ihr es auch in den Materialien auf der Lernplattform ergänzen, damit die Lernenden es nachschlagen können.

Workshop 3: Anfangen mit Pong

Aufgaben besprechen

Bei der ersten Aufgabe ist die Logik hinter der Lösung wichtig. Die Lösungen beider Aufgaben sollten die Verwendung einer for-Schleife vermeiden.

turtle.Screen und turtle.Turtle

Die Lernenden experimentieren mit turtle.Screen und können verschiedene Farben, Größen, etc. ausprobieren (siehe Workshop-Code.py).

Dann erklärt ihr Objekte und Methoden anhand von turtle.Turtle(). Die Lernenden können mit mehreren Turtles herumlaufen.

Nach kurzem Ausprobieren sollen die Lernenden mit den Turtles Dreiecke malen. Dann kann die Kursleitung Funktionsdefinitionen einführen. Beispiele finden sich in der Code-Datei.

onkeypress

„Wäre doch schön, wenn wir die Turtle mit der Tastatur direkt steuern könnten. Dafür müssen wir die Funktionen an Tasten binden.“

Auch hierfür findet sich Code in Workshop-Code.py. Zuerst könnt ihr die Funktion *kleines_dreieck* aus der letzten Einheit an die Taste <d> binden und es ausprobieren, dann *vorwaerts()* und *links()* als Funktionen definieren und damit die Turtle steuern.

„Ok, aber wo ist denn die Turtle jetzt?“ Mit dieser Frage könnt ihr die Funktionen *xcor()* und *ycor()* ansprechen.

Schlägerbewegung

Mit diesem Wissen können die Teilnehmenden mit dem Spiel beginnen:

> **Pong_Code_1.py**: Zunächst wird das Screenobjekt eingerichtet, dann wird der Schläger gebaut und konfiguriert.
>
> **Pong_Code_2.py**: Mit der Definition der Funktion für die Bewegung und der Bindung an eine Taste kann der Schläger über den Bildschirm bewegt werden.

Ihr solltet die Lernenden dazu anhalten, die Programmdatei regelmäßig auszuführen, um Fehler schnell zu entdecken.

Punktestand-Funktion

„Jetzt brauchen wir noch eine Turtle, die uns den Punktestand in das Fenster schreibt. Die soll folgende Eigenschaften haben…“

Pong_Code_3.py: Wenn genügend Zeit ist, sollen die Lernenden die Punktestand-Turtle anhand der Beschreibung der Kursleitung selbst konfigurieren. Ihr könnt dazu die Funktion *punktestand* erklären.

Abschluss

Die Lernplattform per Screenshare zeigen und mit den Teilnehmenden die Aufgaben bis zur nächsten Stunde kurz besprechen. Dabei die Code-Kommentare kurz erläutern.

Nach der Stunde schickt ihr den Link zur Lernplattform und zu den aktuellen Aufgaben an die Lernenden. Die Aufgaben und die Funktionenliste müssen eventuell an den Verlauf der Stunde angepasst werden.

Konzepte, die nach der Stunde klar sein sollten:

- Funktionen definieren
- turtle.Screen(), turtle.Turtle()
- onkeypress
- Kommentare

Workshop 4: Pong

Aufgaben besprechen

Für die erste Aufgabe eine Minimal-Lösung mit den Teilnehmenden teilen und auf eventuelle Fehlermeldungen eingehen.

Bei der zweiten Aufgabe bei der Besprechung die beiden folgenden Zeilen hinzufügen und erklären:

```
ball.xbewegung = 2
ball.ybewegung = 2
```

Für die Besprechung der dritten Aufgabe ist es sinnvoll, einmal mit Screenshare in Erinnerung zu rufen, was passiert (nämlich, dass sich ein Eingabefenster öffnet), und dann die Fragen von den Lernenden beantworten zu lassen.

Struktur für Pong

Im Gespräch sollen die Teilnehmenden eine Liste/Mindmap erarbeiten, in der sichtbar wird, welche Programmteile Pong braucht und welche schon fertig sind.

Die fertigen Teile werden in einer Datei abgespeichert, die ungefähr **Pong_Code_3.py** entsprechen sollte.

Schlägerbewegung

Pong_Code_4.py: Noch einmal den Code zur Schlägerbewegung aus der letzten Stunde durchgehen. Dies kann wieder mit Umfragen bzw. Quizzes passieren. Dann sollen die Lernenden selbstständig die anderen Funktionen fertigstellen.

Ballbewegung

Dies ist der wichtigste Teil der Stunde und kann, falls noch Unklarheiten bestehen, auch soweit ausgedehnt werden, dass für den Punktezähler keine Zeit mehr bleibt.

Zunächst sollten alle die vier Bewegungsarten des Balles sammeln: *„Was passiert jetzt eigentlich während des Spiels mit dem Ball? Was gibt es für Fälle?“.*

- Der Ball fliegt geradeaus.
- Der Ball prallt an den Seiten ab.
- Der Ball fliegt aus dem Spielfeld, weil einer der Spielenden zu langsam war.
- Der Ball prallt von einem Schläger ab und flieg zum anderen Schläger zurück.

Pong_Code_5.py: Danach solltet ihr die while-Schleifen erklären und dann die einzelnen Fälle vorprogrammieren. Zur Auflockerung und Aktivierung könnt ihr immer wieder Quizzes oder offene Fragen verwenden.

Wenn das Programm steht, können die Lernenden kurz spielen.

Punktezähler

„Wir haben jetzt ein funktionierendes Spiel, aber was fehlt jetzt noch?“

Die Kursleitung kann mit den Lernenden zusammen nachdenken, wie die Punkte am besten gezählt und ausgegeben werden – beispielsweise, indem ihr Fragen an die Gruppe stellt. Dazu

gehört die Anzeige, also wo und in welcher Form die Punkte angezeigt werden, und die Art und Weise, wie der Punktestand im Code selbst hochgezählt wird. Danach können die Lernenden noch einmal das fertige Spiel spielen.

Weiterführende Wünsche sammeln

Zum Abschluss können alle gemeinsam weitere Features sammeln, beispielsweise eine zufällige Startkoordinate des Balles. Diese Ideen kann die Kursleitung mit Tipps in der Lernumgebung sammeln.