Inhaltsverzeichnis

Vorwort

Willkommen in dem Kurs des zweiten Trimesters des Talentkolleg Ruhr im Jahr 2022. Wie bereits besprochen schicke ich euch immer mal wieder ein paar Dateien und Informationen zu den im Kurs behandelten Themen zu. Ihr könnt mir natürlich jederzeit bescheid sagen, dass ihr das nicht mehr wollt. Auch ist noch zu erwähnen, dass ihr mir jederzeit eine Mail schreiben könnt solltet ihr eine Frage zu dem Kurs, den Themen im Kurs oder natürlich auch anderen Themen haben.

Wie gehe ich mit dem Dokument um?

Das Dokument dient als Dokumentation für das Projekte. Es reicht normalerweise, wenn ihr euch die relevanten Sachen raussucht und euch nur die anguckt, aber natürlich könnt ihr euch auch das komplette Dokument durchlesen.

Feedback

Wenn ihr Feedback oder einen Fehler zu dem Dokument gefunden habt bitte unter meiner Email melden und mir bescheid geben.

Github

Wir haben auch ein eigenes Repository, damit ihr den Code und die Doku immer beisammen habt: https://github.com/Talentkolleg-Herne/trimester2-2022-montagskurs-dominik

Aufteilung des Projektes

Da das Projekt als ganzes recht umfangreich ist würde ich es gerne aufteilen, indem wir aus dem gesamten Projekt einzelne Phasen machen, die wir dann bei bedarf immer wieder abschließen können.

Phase 1 - Grundlagen legen

In dieser Phase legen wir die Grundlagen der Sprache einmal fest und legen zusätzlich alle benötigten Ressourcen für unser Projekt an (Programme usw.). Zu guter letzt legen wir genau fest was das Projekt

beinhaltet

Phase 2 - Das Spielfeld

In dieser Phase soll es darum gehen, das Spielfeld zu zeichnen. ## Phase 3 - Die Symbole zeichnen Hier werden die Symbole für den Spieler, den Schlüssel, die Tür und die Schlange auf das Spielfeld gezeichnet. * Spieler (Player) = P * Schlange (Snake) = S * Schlüssel (Key) = K * Tür (Door) = D ## Phase 4 - Die Bewegung Hier wird es um die Bewegung des Spielers gehen um ihn mit den Tasten WASD zu bewegen. ## Phase 5 - Die Schlange bewegen Hier wird die Schlange auf den Spieler zu bewegt. Wie wir das genau machen anhand von welchen Regeln entscheiden wir in der dafür vorgesehenen Stunde. ## Phase 6 - Kollision Hier muss die Kollision bestimmt werden, da wir ja auch gewinnen und verlieren wollen. Somit müssen wir dann quasi den Spieler auch Gewinnen oder Verlieren lassen wenn er die Siegbedingung erfüllt oder die Schlange berührt. # Python installieren Hier wird beschrieben wie man PyCharm installiert.

Was ist überhaupt Python?

Python ist eine sehr bekannte und mittlerweile weit verbreitete Scriptsprache, welche für automatisierungen oder aber auch für Künstliche Intelligenz verwendet wird. Natürlich kann die Sprache aber auch für quasi alle anderen Sachen herhalten und benutzt werden.

Installation

Die installation teile ich mal in drei Stufen ein. - Download - Installation - Testen

Download

Python kann von der offiziellen Webseite https://www.python.org/ heruntergeladen werden oder aber von diesem direktlink, der auf den Download der Seite zeigt: https://www.python.org/ftp/python/3.9.10/python-3.9.10-amd64.exe

Benutzen tun wir hier die Version für Windows in der 3.9.10er Version. Auf dieser Seite kann die gleiche Version von Python auch für andere Betriebssysteme heruntergeladen werden auf die ich jetzt aber nicht weiter eingehen werde (https://www.python.org/downloads/release/python-3910/).

Die eigentliche Installation

Um die Installation zu starten, navigieren wir zu dem Download-Ordner und doppelklicken auf die eben gedownloadete Datei. Sie sollte ungefähr wie folgt bennant sein wobei die X für die Versionsnummer stehen: `python-x.x.x-amd64.exe`

Sollte nun ein Fenster scheinen wählen wir nochmal Ausführen.

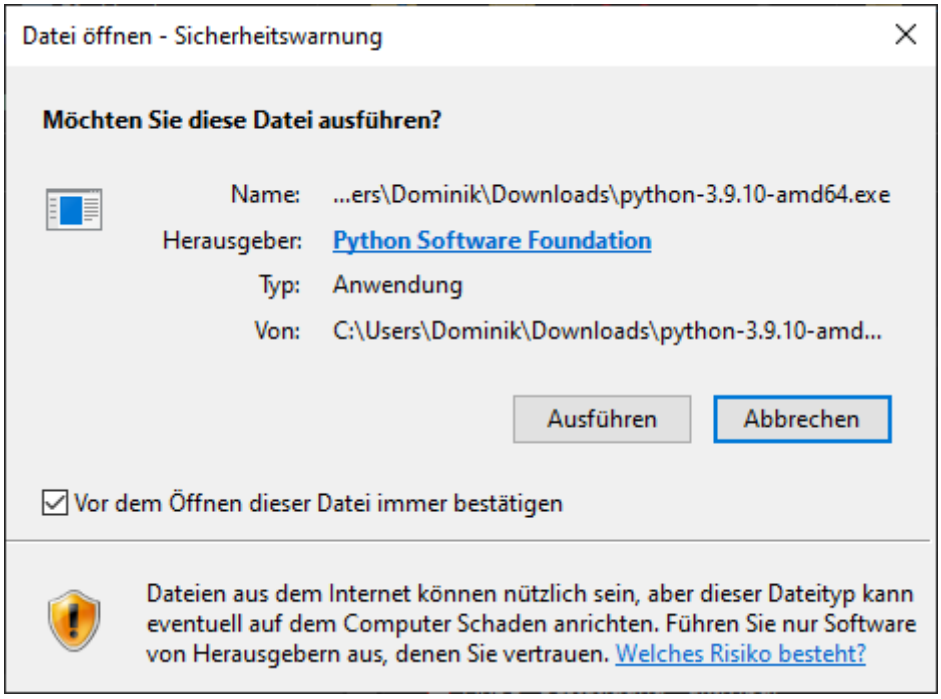

Abbildung 1: Administrator Berechtigung

Nun erscheint ein weiteres Fenster, indem wir anklicken sollen was wir genau wollen. Da klicken wir auf das rot Umrandete `Install Now`

Nun bekommen wir als letztes das Fenster, das alles geklappt hat.

Testen

Um die Installation zu testen, können wir recht einfach ein Terminal aufmachen und dort `python --version` eingeben um zu sehen ob es den Befehl nun gibt.

Um das Terminal aufzumachen drücken wir die `Windows`-Taste und die Taste `R`. Damit rufen wir ein Fenster auf (siehe Abbildung 1), indem wir Programme ausführen können. In dieses Fenster schreiben wir nun cmd rein und drücken Enter. Damit öffnen wir die Eingabeauffroderung (siehe Abbildung 2) in der wir nun ebenfalls wieder Befehle ausführen können.

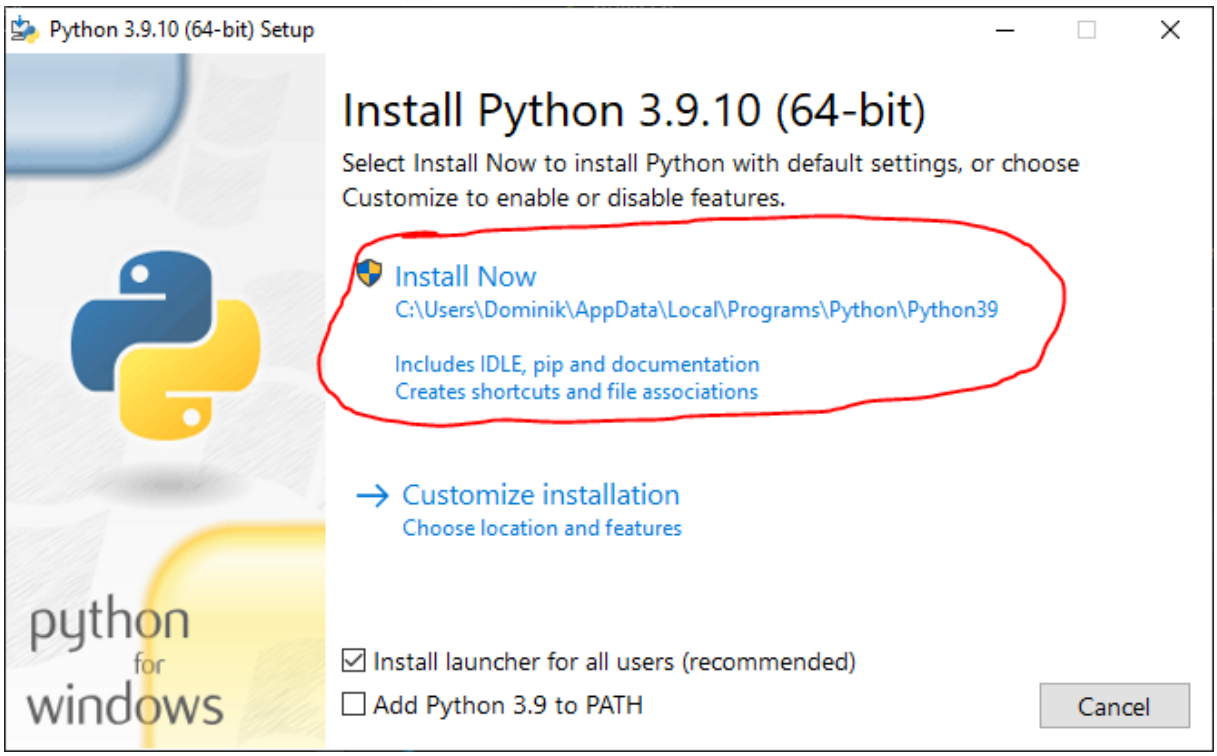

Abbildung 2: Installieren

Nun können wir in diesem Fenster den Befehl `python --version` eingeben um zu schauen ob wir eine Ausgabe mit dr Versionsnummer bekommen:

```
1 Z:>python --version
2 Python 3.9.10
```

PyCharm installieren

Hier wird beschrieben wie man PyCharm installiert. Dabei wird das Betriebssystem Windows amausführlichsten behandelt und lediglich die Download Links für andere Systeme hier angegeben.

Was ist überhaupt PyCharm?

PyCharm ist eine IDE (Integrated Development Environment) und bietet quasi neben einem Editor wie ihr ihnkennt auch noch zusätzlich Features genau für die Programmiersprache Python angepasst an. So kann manzum Beispiel in pyCharm automatisch eine Auflistung von verfügbaren Variablen bekommen.

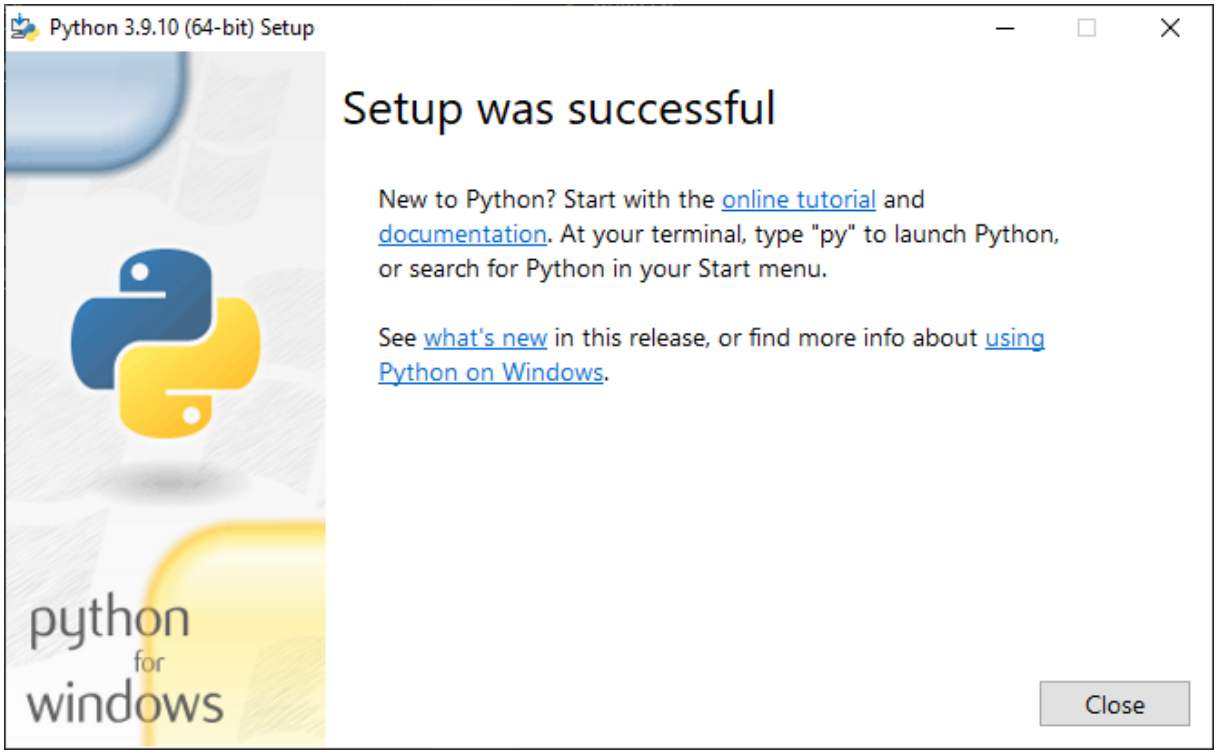

Abbildung 3: Erfolgreich installiert

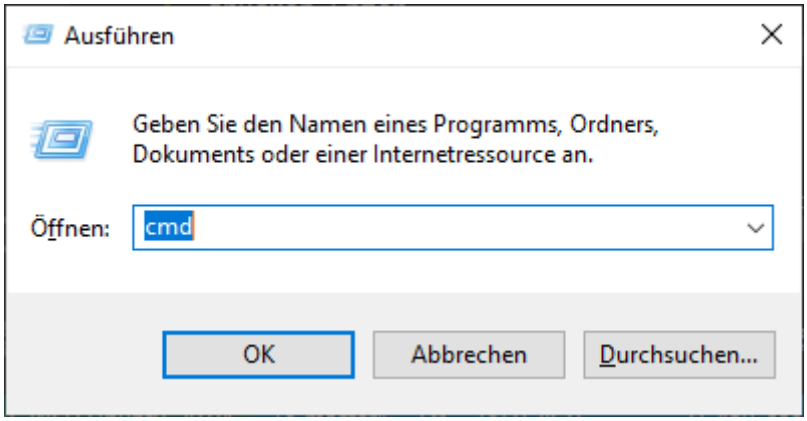

Abbildung 4: In dem Ausführen Fenster können wir Programme oder Befehle ausführen

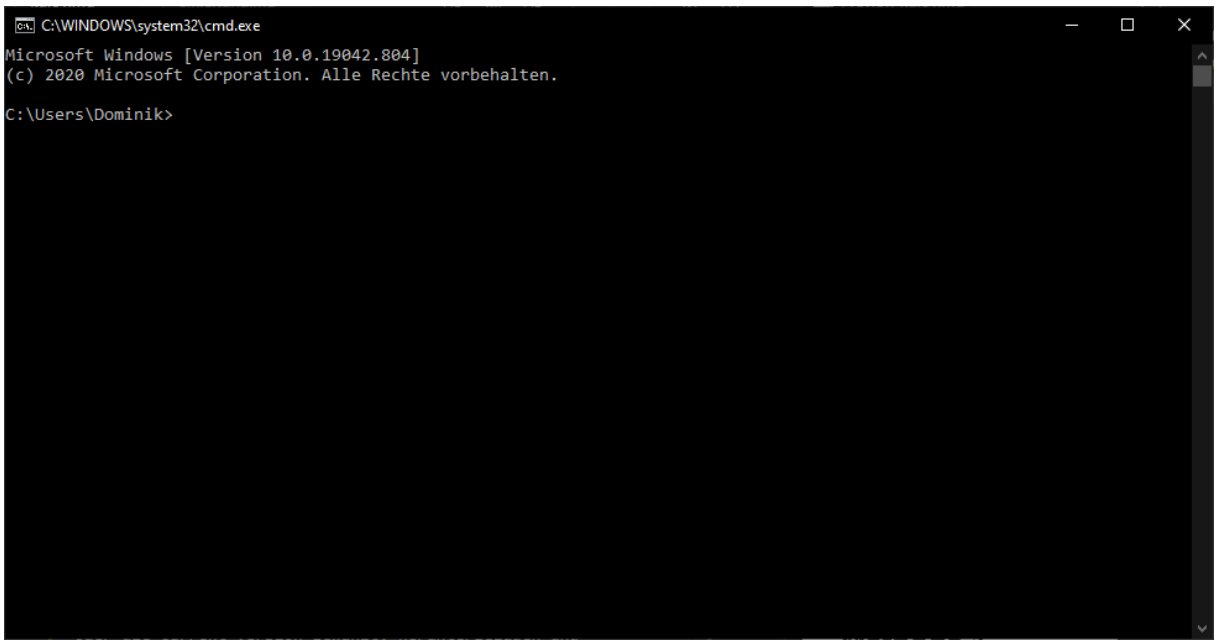

Abbildung 5: Unsere Eingabeaufforderung für Befehle unter Windows

PyCharm auf Windows

Die installation teile ich mal in drei Stufen ein.

- Download
- Installation
- Testen

Download

PyCharm kann von der offiziellen Webseite https://www.jetbrains.com/de-de/pycharm/download/#section=windows heruntergeladen werden oder aber von diesem Direktlink, der aufden Download der Seite zeigt: https://www.jetbrains.com/de-de/pycharm/download/download-thanks.html?platform=windows&code=PCC

Die eigentliche Installation

Um die Installation zu starten, navigieren wir zu dem Download-Ordner und doppelklicken auf die ebengedownloadete Datei. Sie sollte ungefähr wie folgt bennant sein wobei die X für die Versionsnummer stehen: `pycharm-community-X.X.X.exe`

Sollte nun ein Fenster scheinen wählen wir nochmal Ausführen.

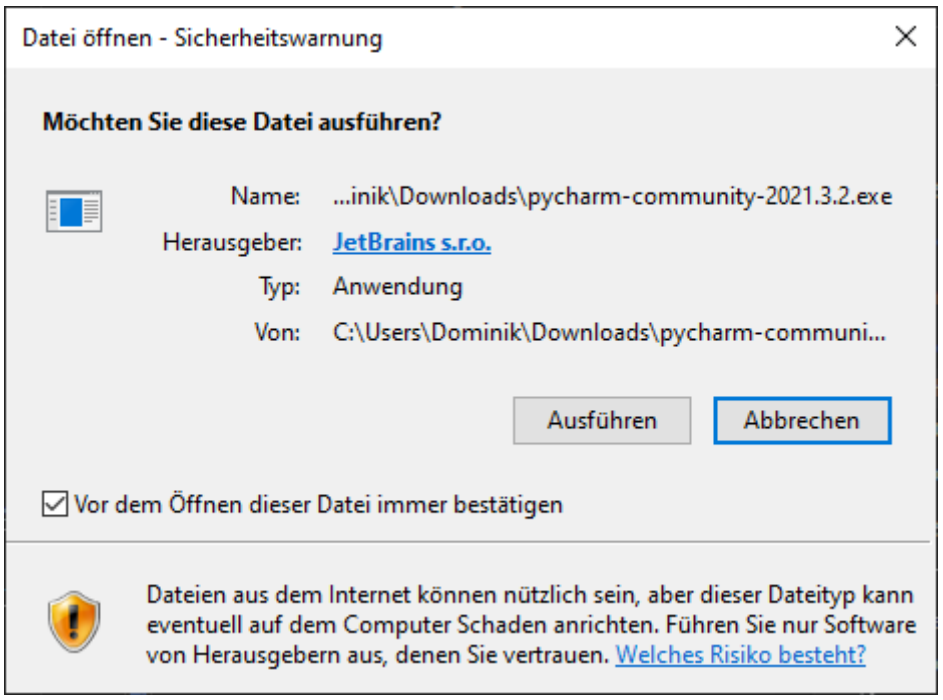

Abbildung 6: Administrator Berechtigung

Danach erscheint das folgende Fenster, welches wir mit Next bestätigen.

Jetzt wählen wir den Pfad auf, wo wir das Programm installieren wollen.

Das nächste Fenster ist jetzt dafür da um auszuwählen mit welchen Dateien wir das Programm verknüpfenund ob wir z.B. bei einem Rechtsklick die Option haben wollen, die Datei direkt mit der PyCharm IDE zuöffnen.

Das nächste Fenster können wir auch einfach mit Next bestätigen. Danach folgt nun die Installation.

Bei dem allerletzten Fenster dieser Installation, können wir uns noch aussuchen ob wir sofort unserenComputer neustarten möchten oder das zu einem späteren Zeitpunkt machen wollen. Dabei ist es auch völligin Ordnung, dass zu einem späteren Zeitpunkt zu machen.

Testen

Das testen ist ganz einfach, indem man das Programm einfach öffnet. Das kann durch Start -> PyCharm seinoder aber durch das Desktop Icon, wenn ihr euch eins angelegt habt.

PyCharm auf Debian Seite: https://www.jetbrains.com/de-de/pycharm/download/#section=linux https://www.jetbrains.com/de-de/pycharm/download/download-thanks.html?platform=linux&code=PCC

PyCharm auf macOSDownload Seite: https://www.jetbrains.com/de-de/pycharm/download/#section=macM1

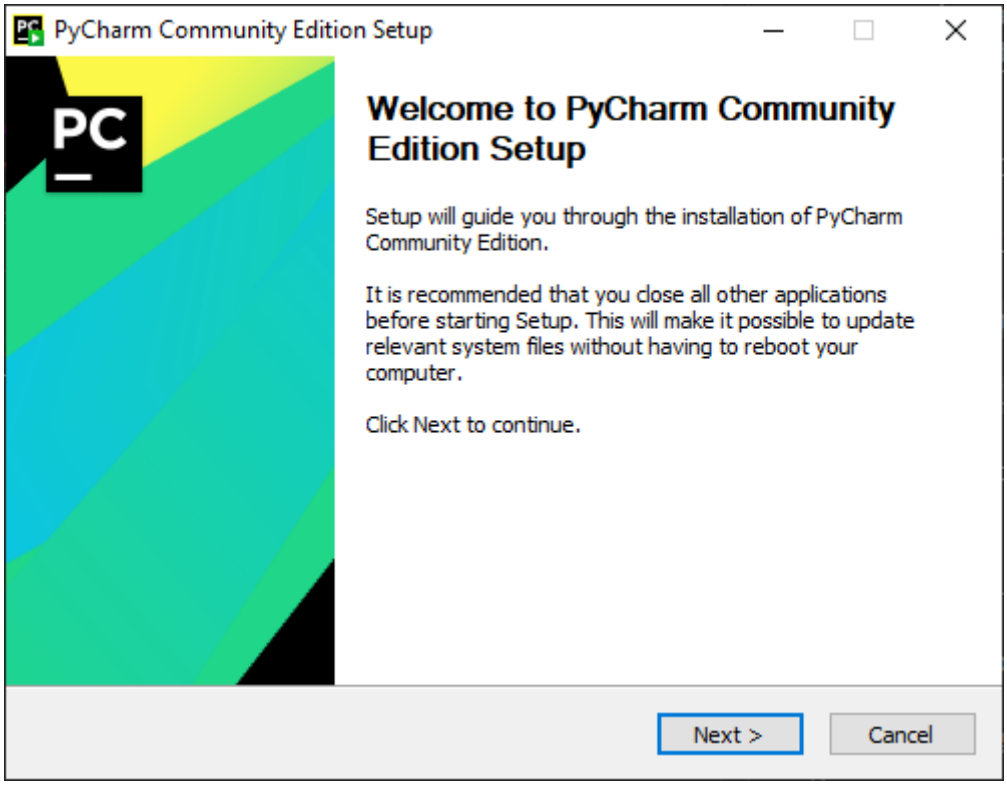

Abbildung 7: Administrator Berechtigung

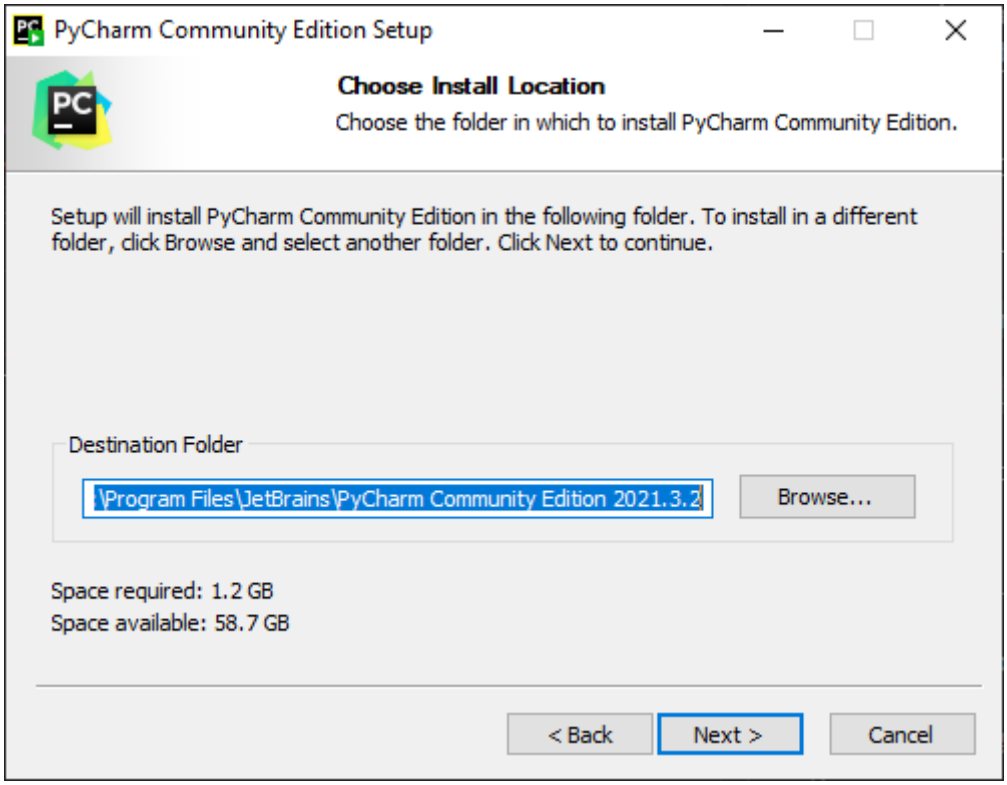

Abbildung 8: Administrator Berechtigung

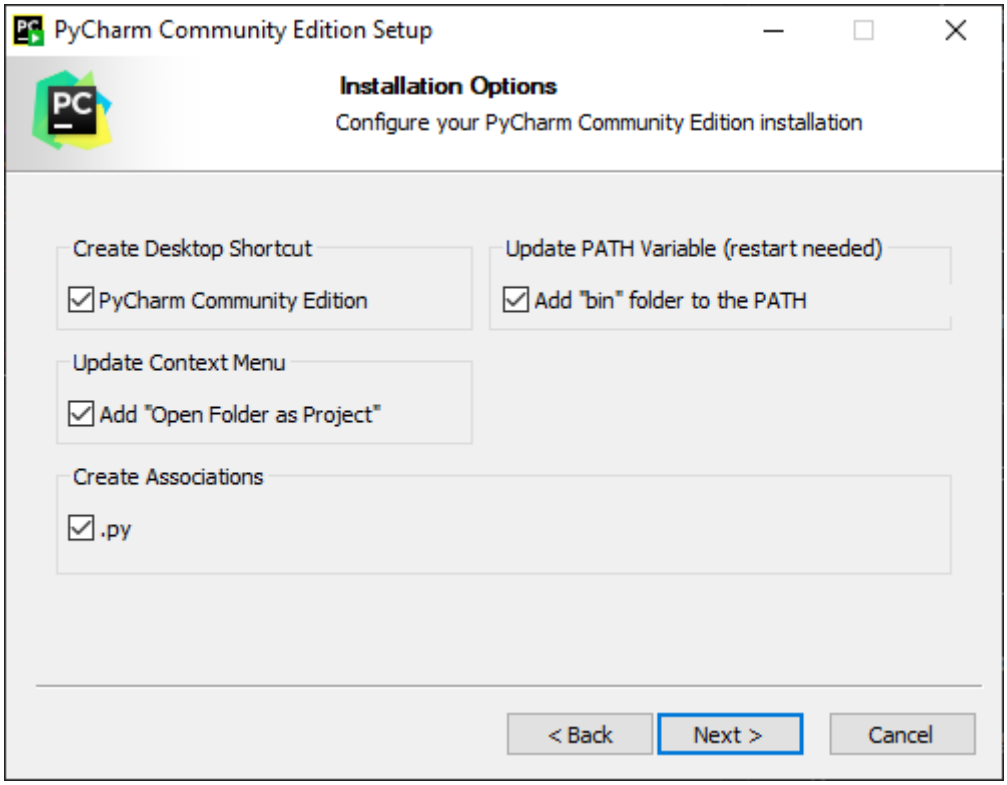

Abbildung 9: Administrator Berechtigung

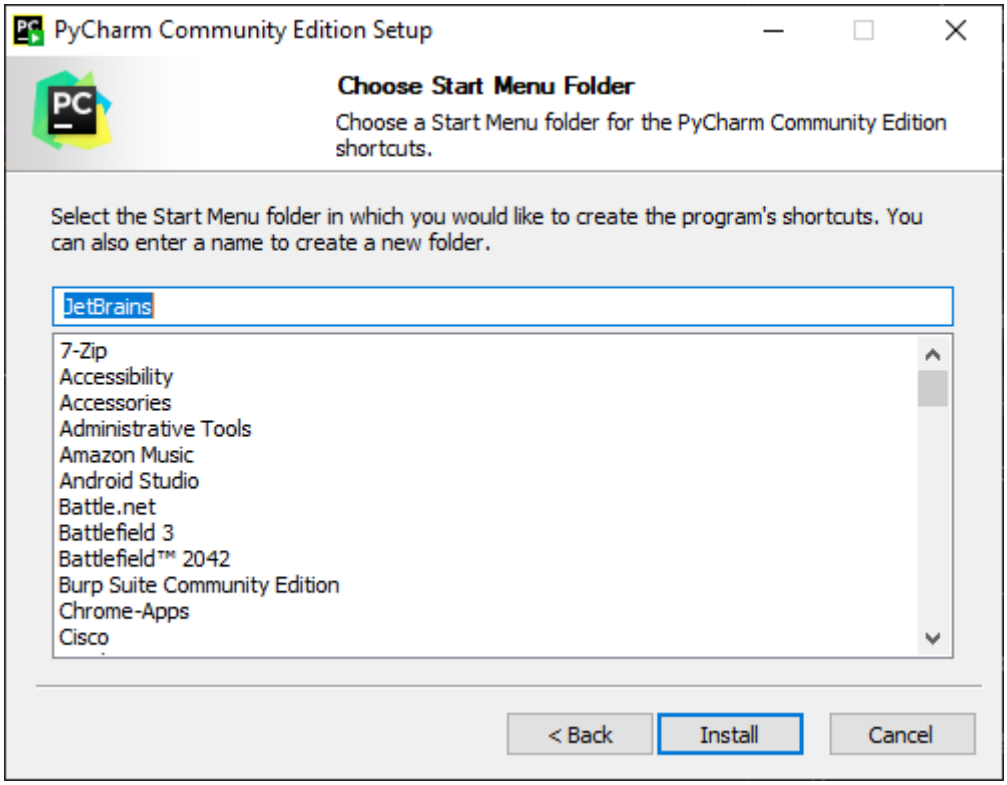

Abbildung 10: Administrator Berechtigung

Download: https://www.jetbrains.com/de-de/pycharm/download/download-thanks.html?platform=mac&code=PCC

Intel Download: https://www.jetbrains.com/de-de/pycharm/download/download-thanks.html?platform=macM1&co

Python - Grundlagen

Python ist eine Skriptsprache in der die Ausführung immer von oben nach unten erfolgt. Das heißt der Befehl in Zeile 1 wird normalerweise vor dem in Zeile 2 gemacht.

Ausgaben

Die zwei Möglichkeiten von Ausgaben sind: Die Ausgabe auf der Konsole oder im Terminal und die andere Möglichkeit wäre die Ausgabe innerhalb einer GUI (Grafische Oberfläche engl. Graphical User Interface) zu machen. Da wir allerdings aktuell nichts in einer GUI machen wollen arbeiten wir mit einer Konsole und den Ausgaben innerhalb dieser.

Eine Ausgabe kann in Python mit der Funktion `print()` erfolgen. Innerhalb der Klammern von print kommt der Text, die Zahlen oder aber die Variable die wir auf der Konsole ausgeben wollen. Hier ein paar Beispiele:

```
1 print("Hallo")
2 print("Welt")
3 print("Hallo TKR")
4 print("Dies ist mal ein etwas längerer Satz")
5 print("Auch Zahlen können in einem Text stehen")
6 print(10)
7 print(meineVariable)
```

Datentypen

Mittels eines Datentyps einer Variablen kann bestimmt werden, welche Werte diese Variable annehmen kann oder welche Operationen wir auf Sie anwenden können. So kann man auf Text die + Operation anwenden um Texte miteinander zu kombinieren und auf Zahlen +,-,/ und *. Außerdem kann noch unterschieden Werden zwischen einem Float (Fließkommazahl) und einem Int (Ganzzahl)

Kommentare

Kommentare sind in einem Programm Zeilen, die vom Computer nicht beachtet werden. Das heißt Sie werden bei der Ausführung einfach ignoriert. In Python kann man einen Kommentar schreiben, indem man eine Raute (#) vor einer Zeile packt.

```
1 # Ich bin ein Kommentar
```

Aufgaben

Hier sind ein paar Aufgaben, die du machen kannst um noch etwas weiter zu üben:

Name Ausgeben

Erstelle ein Programm, das deinen Vorname, Nachname und dein Geburtsdatum auf der Konsole ausgibt.

> Zusatzaufgabe: Erweitere das Programm, so dass es zusätzlich noch deine Adresse und deine Telefonnummer auf der Konsole ausgibt.

Rechnen

Computer können nichts besser als rechnen, daher kann man auch in allen Programmiersprachen mit den Zahlen berechnungen anstellen. Python stellt uns dafür verschiedene Operatoren zur Verfügung:

- + (Pluszeichen) dient zur Addition
- - (Minuszeichen) dient zur Subtraktion
- * (Asteriks) dient zur Multiplikation
- / (Slash oder Schrägstrich) dient zur Division

Dabei muss man aber aufpassen mit welchen Datentyp man zu tun hat, denn je nachdem welcher Datentyp es ist stehen nur wenige oder alle Operatoren zur Verfügung. Es ist allerdings auch verstädnlich, dass man bei einem Text keine Division durchführen kann und das nur bei Zahlen geht.

Aufgaben

Hier sind ein paar Aufgaben, die du machen kannst um noch etwas weiter zu üben. Nächste Woche bekommt ihr dann die Lösung dafür:

Weitere Operatoren

Probiere die unten stehenden Operatoren mit verschiedenen Zahlen aus und notiere dir, was sie tun. Setze für x und y verschiedene Zahlen ein, bis du herausgefunden hast, was die Operatoren tun.

```
x + y
x - y
x * y
x / y
x ** y
x // y
x % y
```

> In der nächsten PDF werden diese Operatoren natürlich oben ergänzt, so dass ihr auch eine Erklärung habt.

Was gibt das Programm aus?

Was ist der Rückgabewert des unten stehenden Programms? Gehe es zuerst im Kopf durch und probiere es anschliessend aus, in dem du das Programm mit PyCharm in einer neuen Datei abspeicherst.

```
x = 2
y = 3
z = x + y
x = z
y = x
print(x)
print(y)
print(z)
```

Zeit umrechnen

Schreibe ein Programm, welches eine Zeitangabe in Stunden, Minuten und Sekunden in die Anzahl Sekunden insgesamt umrechnet

Werkstatt kosten

Eine Werkstatt verlangt für die Benützung einer Maschine eine Grundgebühr von 60 Franken sowie 35 Fanken pro Stunde. Man überlege sich, welche Daten einzugeben und wel che Daten zu berechnen sind und erstelle ein passendes Programm.

Leihwagen kosten

Ein Kleinunternehmen stellt seinen Kunden für Tran sporte einen Lieferwagen und einen Lastwagen zur Verfügung. Für Transporte mit dem Lieferwagen werden Fr. 1.60 pro Kilometer verrechnet, für den

Lastwagen beträgt der Tarif Fr. 2.80 pro Kilometer. Welche Gebühr hat ein Kunde zu bezahlen, wenn der Lieferwagen 85 km und der Lastwagen 120 km zurückgelegt hat?

Der Weinhändler

Ein Weinhändler verkauft Rotwein zu 18 Franken pro Flasche, Roséwein zu 13 und Weisswein zu 12 Franken pro Flasche. Ein Kunde bestellt (beispielsweise) 12 Flaschen Rotwein, 6 Flaschen Rosé und 24 Flaschen Weisswein. Schreibe ein Programm, welches den Gesamtpreis berechnet.

Temperatur umrechnen

Schreibe ein Programm, das Temperaturen in verschiedene Skalensystemen ineinander umwandelt. Das Programm soll zu Beginn eine Auswahl mit den verschiedenen Möglichkeiten anbieten:

- Umrechnung von Celsius nach Kelvin
- Umrechnung von Celsius nach Fahrenheit
- Umrechnung von Kelvin nach Celsius
- Umrechnung von Kelvin nach Fahrenheit
- Umrechnung von Fahrenheit nach Celsius
- Umrechnung von Fahrenheit nach Kelvin

> Dies kann dir sicher helfen. - Celsius = 5/9 * (Fahrenheit - 32). - Celsius = Kelvin - 273.15. - Die tiefste mögliche Temperatur ist den absoluten Nullpunkt 0K.

Fallunterscheidung

Eine Fallunterscheidung dient dazu zwischen zwei Zuständen zu unterscheiden und verschiedenen Code für die Zustände auszuführen. Als Beispiel möchte man z.B. zusätzlichen Code ausführen wenn jemand über 18 ist (z.B. Zugang zu einer Webseite). Auch in der realen Welt gibt es natürlich auch Fallunterscheidungen die wir intuitiv fällen. So gibt es z.B. die Möglichkeit sich wenn es kalt ist eine Jacke anzuziehen.

In der Programmierung ist es allerdings nicht ganz so einfach mit den Bedingungen. Hier muss jede Bedingung nach WAHR oder FALSCH beurteilt werden können. Diese Werte WAHR oder FALSCH werden auch als boolsche Werte bezeichnet. Für diese Aussagen (boolsche Bedingung => Eine Bedingung die nach einer boolschen Wert auswertbar ist) stehen uns folgende Operatoren zur Verfügung:

Operator	Wort	Beispiel
>	größer	5 > 10 5 ist kleiner als 10 (WAHR)
<	kleiner	10 < 5 10 ist kleiner als 5 (FALSCH)
>=	größer gleich	16 >= 17 16 ist größer oder gleich 17 (FALSCH)
<=	kleiner gleich	16 <= 16 16 ist kleiner oder gleich 16 (WAHR)
!=	nicht gleich	0 != 1 0 ist nicht gleich 1 (WAHR)
==	gleich	1 == 1 1 ist gleich 1 (WAHR)

Mithilfe dieser Operatoren lassen sich nun die Bedingungen schreiben. Um so eine Bedingung zu schreiben, müssen wir vorher das Schlüsselwort für eine Fallunterscheidung voran schreiben **`if`** um in Klammern dahinter die Bedingung zu schreiben. Sollte nun die Bedingung wahr sein, wird alle was in den geschweiften Klammern dahinter kommt ausgeführt. Wenn man möchte kann man hinter der **`if`** noch ein **`else`** schreiben um einen block nur Auszuführen, wenn die Bedingung in der **`if`** Anweisung den Wert FALSCH liefert.

Als Beispiele:

```
1  # definiere eine Variable die mein Alter speichert
2  meinAlter = 23
3
4  if meinAlter >= 18:
5    print('Du bist volljährig')
```

> Programm welches das Alter speichert und „Du bist volljährig“ auf der Konsole schreibt wenn man älter oder gleich 18 Jahre alt ist.

```
1  meinAlter = 23
2
3  if meinAlter >= 18:
4    print('Du bist volljährig')
5  else:
6    print('Du bist noch minderjährig')
```

> Programm welches das Alter speichert und „Du bist volljährig“ auf der Konsole schreibt wenn man älter oder gleich 18 Jahre alt ist. Sollte man jedoch jünger als 18 Jahre alt sein wird „Du bist noch minderjährig“ auf die Konsole geschrieben

```
1  meinAlter = 23
2
```

```
3 if meinAlter >= 18:
4   print('Du bist volljährig')
5 elif meinAlter >= 16:
6   print('noch nicht volljährig, aber nah dran')
7 else:
8   print('Du bist noch minderjährig')
```

> Dieses Programm gibt ein „Du bist volljährig“ wenn der Benutzer älter oder gleich 18 Jahre alt ist. Zudem wird noch ein anderer Text ausgebenen wenn das alter größer oder gleich 16 ist. Als alternative wenn nichts anderes zutrifft wird „Du bist noch minderjährig“ ausgegeben. In dem Beispiel wird das `elif`-Konstrukt verwendet, welches nur die erste Bedingung ausführt. Das ist hier besonders Sinnvoll, da ansonsten immer beide Texte ausgebeben werden würden (da jemand der über 18 Jahre ist auch immer über 16 Jahre alt ist) und wir mit dem `elif`-Konstrukt nun nach der ersten Bedingung den rest des Konstrukts überspringen können, sollte jemand über oder gleich 18 Jahre alt sein.

Aufgaben

Wie viel kostet das Paket? Schreibe ein Programm welches berechnet wie teuer ein Paket ist, wenn folgende Annahmen gelten: - Das Gewicht des Pakets darfst du dir aussuchen - Der Grundpreis beträgt 3,50 € - Für Pakete bis 5kg gibt es eine Gebühr von zusätzlich 1€ - Für Pakete bis 7,5 kg gibt es eine Gebühr von zusätzlich 2€ - Für Pakete bis 10 kg gibt es eine Gebühr von zusätzlich 3€ - Für Pakete ab 10kg gibt es eine Gebühr von zusätzlich 3€ und pro KG nochmal 0,50 € dazu

Arrays

Ein Array ist eine Möglichkeit unter einer Variable mehrere Werte zu addressieren. So brauchen wir aktuell um zum Beispiel die Werte für die Temperaturen der letzten Woche zu speichern insgesamt 7 Variablen. Leider können wir mit diesen Variablen nicht sehr dynamisch umgehen, da bei einem Zugriff auf eine Variable der Name der Variablen immer angegeben werden muss. Um die Werte also zum Beispiel alle zu addieren müssten wir die in unserem Fall die 7 Variablen zusammen addieren. Sobald wir jetzt aber ein weiteren Wert speichern wollen, müssen wir uns wieder eine neue Variable anlegen und das Programm ändern, so dass wir die neue Variable mit in die Berechnungen mit einbeziehen.

Um das einfacher und dynamischer zu halten kommen nun Arrays in spiel. Hiermit ist es möglich wie gesagt mehrere Werte über einen gemeinsamen Namen und einen Index zu definieren. So können wir uns zum Beispel die ersten 5 Ungerade Zahlen in einem Array so definieren:

```
1 meineZahlen = [1,3,5,7,9]
```

> Ein Array wird wie eine Variable definiert, nur statt einen Wert anzugeben, können wir hier in Eckgen-Klammern mehrere Werte desselben Typs angeben.

Die länge eines Arrays kann man sich in Python ebenfalls dynamisch Anzeigen lassen, indem man die in Python integrierte Funktion `len()` benutzt.

```
1 meineZahlen = [1,2,3,5,7,9]
2 print(len(meineZahlen))
```

Schleifen

Schleifen dienen in der Programmierung dazu Aufgaben zu bewältigen die öfters getan werden müssen. So ist bei einem Spiel immer so eine Schleife dabei die z.B. unseren Bildschirm immer neu lädt um so z.B. Dinge bewegen zu können. So kann man mit dem Schlüsselwort **`while`** so eine Schleife einleiten und dann in runden Klammern dahinter sofort die Bedingung definieren.

Die while-Schleife

```
1 while(x < 1000):
2   # Wird solange ausgeführt wie die Bedingung (in den Klammern) wahr
      ist.
```

Also um ein kleines Beispiel zu machen, wollen wir nun die Zahlen von 1 bis 1000 ausgeben. Mit der alten weise müssten wir leider 1000x `print()` schreiben. Wir wollen das aber natürlich etwas vereinfachen und nehmen daher eine Schleife.

```
1 x = 1; # wir wollen bei 1 anfangen
2
3 while(x <= 1000): # solange x kleiner oder gleich ist als 1000 gehen
    wir in die Schleife
4   # wird solange ausgeführt wie die Variable x kleiner oder gleich 1000
      ist.
5   # Ausgabe der Variablen x
6   print(x)
7
8   # erhöhe x um eins um die Schleife auch irgendwann beenden zu können.
9   x = x + 1
```

Die for-Schleife

Die for-Schleife dient dazu über eine Sequenz zu iterieren. Das heißt bei jeden Schleifendurchlauf nimmt die Zählvariable jedes Element in der Datenstruktur einmal als Wert an.

```python
temp = [9, 11, 14, 16, 16, 15, 12, 15]

for i in temp:
    print(i)
```

> Ausgabe: 9 11 14 16 16 15 12 15

Aufgaben

Summe Berechnen

Schreibe ein Programm, welches die Summe einer Zahlenfolge berechnet. Als Beispiel von 50 bis 200

Fibonacci Zahlen

Gebe die Fibonacci Zahlen aus bis zu einer bestimmten Stelle. Die Fibonacci Zahlen werden wie folgt berechnet:

1 1 2 3 5 8 11

Man nimmt die erstren beiden Zahlen 1 + 1 wobei dann 2 rauskommt. Als nächstes rechnet man 1 + 2 wo dann drei rauskommt usw.

Fakultät berechnen

Schreibe ein Programm, welches die Fakultät einer Zahl berechnet. Die Fakultät ist das Produkt einer Zahlenfolge. Beispiel Fakultät von 5 1 * 2 * 3 * 4 * 5 = 120

Quersumme

Lasse die Quersumme von einer bestimmten Zahl berechnen

Multiplizieren

Versuche eine Multiplikation (* oder mal nehmen) ohne das * Zeichen zu schreiben

Dividieren

Versuche eine Division (/ oder geteilt nehmen) ohne das / Zeichen zu schreiben

Modulo

Versuche eine Modulo (% oder mal nehmen) ohne das % Zeichen zu schreiben

Funktionen

Mittels Funktionen können Codeteile ausgelagert werden und zentral an einer Stelle zur Verfügung gestellt werden. Das ist von Vorteil, wenn man Code in einem Programm oft verwendet und daher den Code nur an einer Stelle updaten möchte. Desweiteren ist es mit Funktionen einfacher funktionalitäten zu teilen und anderen Bereit zustellen oder aber in diese Funktionen selber in anderen Projekten wiederzuverwenden.

Um in Python eine Funktion zu erstellen, wird das Schlüsselwort `def` vor dem Namen der Funktion geschrieben. Danach kommen Klammern `()`. In diesen Klammern können nun Parameter definiert werden um bestimmte Variablen auch innerhalb der Funktion zur Verfügung zu haben. Die Funktion kann nach der definition aufgerufen werden, indem man den Namen mit den beiden Klammern schreibt.

```
1  # definition der Funktion
2  def meineFunktion():
3    print('Ich bin eine Funktion')
4
5
6  # Aufruf der Funktion.
7  meineFunktion()
8
9  # definition der Funktion mit einem Parameter
10 def meineFunktion2(meinParameter):
11   print(meinParameter)
12
13
14 # Aufruf der Funktion mit Parameter
15 # Hier wird in den Klammern der Wert eingetragen und kann in der
       Funktion benutzt werden
16 meineFunktion2('hello')
```

Die Phase1 umsetzen

Um Phase 1 umzusetzen würde ich mir zwei Schleifen anlegen, die die X- und die Y Richtung abdecken. Dafür kann ich entweder eine For-Schleife oder eine While-Schleife nehmen um das wie gewünscht umzusetzen. Es macht normalerweise kaum einen Unterschied welche Schleife ich nehme, wichtig dabei ist nur, dass ich bei beiden auf verschiedene Sachen achten muss.

Beispiel 1 - FOR-Schleife

Zuerst lege ich mir zwei verschiedene Variablen an, die einmal für die Spielfeld Breite und die Höhe stehen.

```
1 SPIELFELD_BREITE = 40
2 SPIELFELD_HOEHE = 25
```

> definition der beiden Variablen. Die Werte können natürlich auch andere sein.

Jetzt muss ich mit der for-Schleife von 0 bis zu der Spielfeld höhe gehen und danach nochmal mit einer weiteren Schleife den Wert bis zur Spielfeld Breite. > Ich muss zuerst den Y-Wert bzw. die Y Richtung machen, da in der Konsole leider kein Rückweg eingebaut ist. Für uns heißt das, dass ich immer eine komplette Zeile mache und anschließend eine Zeile weiter springe. Die Reihenfolge wäre damit wie folgt: Zeichnen der kompletten Zeile und anschließend ein Weiter springen in die nächste und das gleiche nochmal tun. Da ich damit im inneren immer die Zeile machen, die hier Stellvertretend für X steht ist bei unseren zwei Schleifen auch X die innere Schleife.

```
1 for y in range(SPIELFELD_HOEHE):
2     for x in range(SPIELFELD_BREITE):
3         print("_", end="")
4     print()
```

Das `in range` in Python ist eine eingebaute Funktion von Python, die mir die Möglichkeit gibt von einer Zahl bis zu einer anderen Zahl zu laufen innerhalb meiner Schleife. Da ich nur einen Parameter habe (meine Breite bzw. meine Höhe) läuft diese Funktion immer von 0 bis zu dem Wert. Alternativ kann ich auch den Startwert als erstes mitgeben und mit der zweiten Zahl den Endwert.

In der Python Funktion print ist Standardmäßig ein NewLine drin (das heißt nach jedem Print-Statement geht Python in eine neue Zeile. Das kann ich unterbinden, indem ich Python das Zeichen für das Ende mitgebe und zum Beispiel auf ein leeren Text ändere)

Beispiel 2 - WHILE-Schleife

Die WHILE-Schleife hat erstmal einen ähnlichen Aufbau wie die FOR-Schleife, nur das wir uns hier auch um die Variablen kümmern müssen, die wir benötigen um zu Zählen und den richtigen Punkt zu haben. Dafür kann ich mir vorher die zwei Variablen x und y erstellen und mit 0 belegen.

```
1 x = 0
2 y = 0
```

Anschließend kann ich meinen Aufbau genauso halten wie bei der FOR-Schleife, nur das ich am Anfang der ersten Scvhleife meinen X-Wert wieder auf 0 setzen muss, da diese Variable außerhalb der Schleife definiert wurde und damit überall auch gültigkeit hat und ihren Wert somit nicht verliert.

```
1 while(y < SPIELFELD_HOEHE):
2     x = 0
3     while(x < SPIELFELD_BREITE):
4         x = x + 1
5         print("_", end="")
6     print()
7     y = y + 1
```