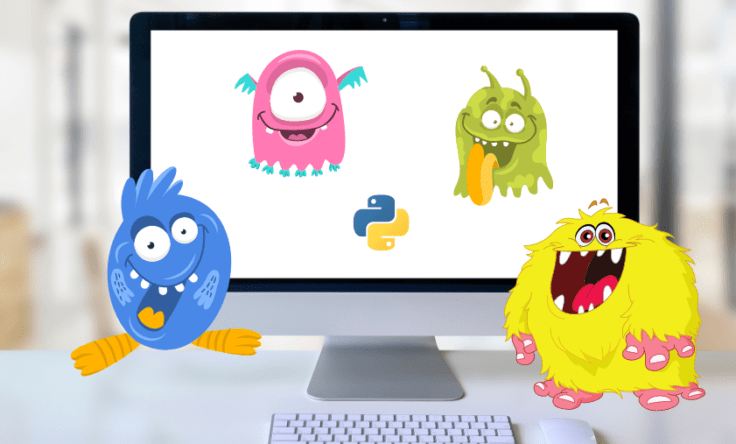
Pymonster



[1 Begrüßung](#_h9jqs8egpgef)

[1.1.Vorstellung des 2-tägigen Kurses](#_9yk0dks98vnv)

[1.2 Ziele und Inhalt](#_g0g8d0l2u7ty)

[1.3 Vorstellung der Inspirer](#_el6692thbdjk)

[2 Ablauf](#_9r2643xw2te2)

[2.1 Thematische Einführung](#_u625kv2s118v)

[2.2 Technische Einführung](#_9gfkf64xcqad)

[2.2.1 IDE](#_mk1ko0d5y2aa)

[2.2.2 Basics in Python via Live Coding](#_f8n2shursytv)

[2.2.2.1 Hallo, Welt!](#_iqi8we9zqke2)

[2.2.2.2 Einführung von Variablen](#_o5zcgajjgklh)

[2.2.2.3 Datentypen](#_2fgpzei259xm)

[2.2.2.4 Fallunterscheidung](#_xw96jpdg94r7)

[2.2.2.5 input-Funktion](#_19jy7ue8z0r)

[2.2.2.6 while-Schleife](#_a22rgijt4onf)

[2.2.2.7 for-Schleife](#_9m34w2i8rkad)

[2.2.2.8 Liste in Python](#_fio9qht2ixo5)

[2.2.3 Methoden schreiben in Python](#_2xoc2w6ztiwg)

[2.2.5 Live-Coding - erste Einführung objektorientierte Programmierung, Methoden und Klassen](#_g90f3tyslql8)

[2.2.6 warm-up nach der Mittagspause](#_bjby5pwhaj8y)

[2.2.7 objektorientierte Programmierung](#_ii4c32fsvqgr)

[2.2.8 Spielablauf konstruieren](#_18r9cwkkueht)

[3. Zweiter Tag](#_cztzdu79wzl3)

[3.1 Rekapitulation vom Vortag](#_e60zom7qo90e)

[3.2 Gameloop erklären](#_hiivq97vc239)

[3.3 Gruppenphase](#_wvz6gutbesad)

[4 Code ausführen](#_59zyaaukvoue)

[5. Gegenseitiges vorstellen des Codes](#_dm3s3rbsvck)

[6. Verabschiedung](#_9x781rj8ek5s)

[Musterlösung](#_vrxnwypeytts)

[Quelle](#_3x6wbjop28ts)

## Zeitplan

|  |  |  |  |
| --- | --- | --- | --- |
| Ordnung | Zeit | Thema | Anmerkung |
| Einführung, Rahmenbedingungen | 10 min  10:00 - 10:10 | Vorstellung Kurs + Ziele | Benni |
|  | 10 min  10:10 - 10:20 | Vorstellung Inspirer | Benni |
|  | 20 min  10:20 - 10:40 | Vorstellung Teilnehmer | Benni |
| Einteilung | 10 min  10:40 - 10:50 | thematische Einführung | Benni |
|  | 10 min  10:50 - 11:00 | Vorstellung IDE | Jan |
| Techn. Einführung | 10 min  11:00 - 11:10 | Hallo, Welt | Jan |
|  | 10 min  11:10 - 11:20 | Einführung von Variablen | Jan |
|  | 10 min  11:20 - 11:30 | Datentypen | Jan - print von anderen Datentypen |
|  | 10 min  11:30 - 11:40 | Fallunterscheidung | Jan |
|  | 10 min  11:40 - 11:50 | input | Arne |
|  | 10 min  11:50 - 12:00 | while-Schleife | Arne |
|  | 10 min  12:00 - 12:10 | Liste | Arne |
|  | 15 min  12:10 - 12:25 | Pause | Benni |
|  | 5 min  12:25 - 12:30 | Gruppeneinteilung in Breakoutrooms | Yannick |
|  | 15 min  12:30 - 12:45 | Methoden schreiben | Yannick |
|  | 20 min  12:45 - 13:05 | Klassen erstellen, Objektorientierte Programmierung | Yannick |
|  | 20 min  13:05 - 13:25 | Attacken-Klasse erstellen | Yannick |
|  | 20 min  13:25 - 13:45 | Gameloop-Skizze erstellen | Benni |
|  |  | Rückfragen |  |
| **Beginn Tag 2** | 5 min | Begrüßung |  |
|  | 5-10 min | Feedback/Fragen |  |
| Erklärung Gameloop | 10 min | SuS stellen ihre eigenen Loops vor wir gehen gemeinsam die Lösung durch |  |
| Wiederholung | 10-15 min | Wiederholung |  |
| Gruppenbildung | 5 min | Teams bilden für Pairprogramming |  |
| Programmierung | 30-45 min | charakter erstellen | Breakoutrooms, Pairprogramming (Erfahren/Unefahren) |
|  | 30 min | Code Ausführen | Klassen verbinden |
| Pause | 20 min | pinkeln |  |
|  | x min | Code optimieren und erweitern | Aufgaben werden gestellt oder von SuS selbst überlegt |
| Vorstellen der Ergebnisse der Teilnehmer | 20-30 min | Ergebnisse präsentieren | je nach Meldung |
| Verabschiedung | 5 min | Feedback von Teilnehmern |  |

# 1 Begrüßung

## 1.1.Vorstellung des 2-tägigen Kurses

Am ersten Tag des Kurses wird sich nach einer Einführungsphase mit den Grundlagen von Python befasst. Es werden Datentypen und Operationen kennengelernt sowie objektorientierte Programmierung. Am zweiten Tag werden, angeleitet von den Inspirern, mit Hilfe der Programmiersprache und -umgebung eigene Monster in einem Spiel erstellt.

[Checkliste](https://docs.google.com/document/d/1Xnr6U-0yQlVeoZwS3spFreA2mAT9o2nM-GsPbLv-VyA/edit)

## 1.2 Ziele und Inhalt

Interessierte lernen in dem 2-Tageskurs die Grundlagen der Programmiersprache Python und Ihre Funktion kennen. Darüber hinaus soll den Teilnehmern insbesondere die technischen Möglichkeiten der Programmierung aufgezeigt werden. Hier soll insbesondere durch eine zielgruppennahe Projektgestaltung am Beispiel eines bekannten 90er Jahre Kinderspiels das Interesse an dem Fachgebiet der allgemeinen Programmierung geschaffen werden.

Der Kurs ist ausgelegt für Teilnehmer, die gar keine, wenige oder in anderen Programmiersprachen fernab von Python Vorkenntnisse haben.

## 1.3 Vorstellung der Inspirer

Die Inspirer stellen sich den Teilnehmern vor. Berichte zu Ihrer Person und ihrem Bezug zur Programmierung sowie der Motivation.

1.4 Vorstellungsrunde der Teilnehmer

Die Teilnehmer stellen sich mit Vornamen vor und berichten jeweils von Ihren Vorkenntnisse. Sofern vorhanden ggf. auch den beruflichen Hintergrund der Teilnehmer und die Intention der Teilnahme an dem Kurs.

[Steckbrief](https://docs.google.com/document/d/1T1XP98WXTTpAfASp2Tr1MUl1hYjA460h6n2BnAM_J-o/edit)

# 2 Ablauf

Ablauf der beiden Tage des Python-Kurses. Nach der Vorstellung wird die Gruppe bezogen auf ihre Pythonkenntnisse in erfahren und unerfahren eingeteilt. Für die erfahrenen wird es eine Einführung in die Basics von Python geben. Die Erfahrenen, kriegen einen Aufgabenzettel, der durch eben diese zu bearbeiten ist. Die Aufgaben sind in replit hinterlegt.

Am zweiten Tag arbeiten die Teilnehmer in Pairs an dem Hauptcode “Pymon”. Die Teilnehmer werden als Paar zusammengesetzt (Erfahren+unerfahren).

## 2.1 Thematische Einführung

Die Programmiersprache Python ist vielseitig einsetzbar. Die Einsatzgebiete gehen von der Datenanalyse über die Homepageentwicklung sowie Künstlicher Intelligenz bis hin zur Spieleentwicklung. Ebendiese Vielfältigkeit macht diese Programmiersprache unter anderem so interessant.

Hinzu kommt, dass Python als eine sehr nutzerfreundliche Programmiersprache angesehen wird, da sie im vergleich zu anderen Programmiersprachen verkürzte Befehle verwendet. Das Ergebnis ist ein komprimierter, übersichtlicher Code.

Diese Vorteile, die Einfachheit und gleichzeitig Vielseitigkeit, machen Python so populär. Folgedessen wird Sie viel verwendet und stetig weiterentwickelt.

Für Berufsschüler interessant, da Schnittstellen zu techn. Berufen sowie kaufmännischen Berufen besteht.

## 2.2 Technische Einführung

Python-Programmierung basiert, wie viele andere Programmiersprachen auf einer Textquelle. Die Entwicklungsumgebung erkennt nach den Regeln der Programmiersprache den Text und wandelt die entsprechenden Syntax um.

### 2.2.1 IDE

Wir nutzen die Entwicklungsumgebung von replit.com.

Der Inspirer führt transparent aus der Checkliste den Pfad bis hin zur IDE aus, sodass den Teilnehmern dies visuell verdeutlicht wird.

Als nächstes zeigt der Inspirer wie ein Projekt innerhalb der IDE erstellt wird.

Dazu sollten die grundsätzlichen der IDE aufgezeigt werden.

-Projekt benennen

-Console

-editor (Eingabe)

[Präsentation Python Pymon Einführung](https://docs.google.com/presentation/d/1ZjN0Tpg_R0CywDGNLc8GzKClVJ7QBs-QzuxnqRtHTxo/edit#slide=id.p)

### 2.2.2 Basics in Python via Live Coding

#### 2.2.2.1 Hallo, Welt!

Zeigen der Entwicklungsumgebung, starten mit “Hallo, Welt!”, danach einfach mathematische Rechnungen, um die Operatoren und Datentypen zu zeigen.

Befehle zeigen: print (hierbei ggf. auf templates hinweisen). Das Programm wird in HelloWorld.py genannt. Erstes Programm!

print(“Hallo, Welt!”) #Console: Hallo Welt!

Ziele

* Die Konsole soll erkundet werden
* String als erster Datentyp
* print-Funktion soll kennengelernt werden
* erste Schritte in der IDE

Rechenbeispiele mit Basisoperatoren:

*print(10+5)#console: 15*

*print(10-5)#console: 5*

*print(10/5)#console: 2*

*print(10\*5)#console: 50*

*print(10%5)#console: 0*

*print(12%5)#console: 2*

*print(10 \*\* 3)#console: 1000*

// Operatoren:

**+** (plus), **-** (minus), **\*** (mal), **/** (dividiert durch), **\*\*** (hoch), **%** (Modulo, Rest einer Division)

16/3 liefert 5 (Ganzzahldivision), 16.0/3 liefert 5.333333333333333(Gleitkommazahldivision)

Ziel:

* andere Operatoren und deren Darstellung in Python werden kennengelernt

#### 2.2.2.2 Einführung von Variablen

Eine Variable kann man sich sinnbildlich als Gefäß vorstellen, dessen Inhalt gefüllt werden kann. Es ist eine der wichtigsten Werkzeuge der Programmierung. Innerhalb eines Programms gibt es viele Variablen, die sich dynamisch ändern.

*x=4*

*y=10*

*print(x+y) #console: 14*

*print(y-x) #console:6*

*print(y\*x) #console:40*

*i=9*

*i=5*

*print(i) #console: 5*

*f=1.1 #Punkt als Komma*

*pymon= “Pymonster” #durch Anführungszeichen oder einfache ‘ zu kennzeichnen*

*b= True #boolean sieht zwei Werte vor: wahr(True) und falsch(False)*

pymon = “*Pymonster*”

print( “Ein wildes” + pymon + ”taucht auf!”) #Die Variable pymon wird durch den String *Pymonster* gefüllt und im nächsten Schritt ausgegeben, wenn die Variable abgefragt wird.

Auf das Thema Datentypen wird erst einmal eingeschränkt eingegangen. Wichtig ist, dass grundsätzlich Zahlenwert und Strings verstanden werden.

Ziele:

* Definition einer Variable als “Speicherort” soll klar sein
* Dynamik von Variablen
* überschreiben von Variablen
* Variablen entsprechen einem Datentyp
* EInführung in die Syntax

#### 2.2.2.3 Datentypen

Integer (Ganzzahl), String(Zeichenkette), Boolean(Logik)

print(y/x) #console:2.5

*x=true*

*print(x)#console:True*

*x=4, y=5*

*print(x<y) #console: True*

*print(x>y) #console: False*

*print(x==y) #console: False*

*print(x<=y) #console: True*

*print(x>=y) #console: True*

*x=”hallo”, y=”Hallo”*

*print(x==y) #console: False*

*print(x+y) #console: hallo Hallo*

*x= 17 , y=”Ich bin ”, z=”Jahre alt ”*

*print(y+x+z) #console: Ich bin 17 Jahre alt*

*x=14*

*x= 25*

*print (x) #console: 25*

Ziel:

* weitere Operatoren werden kennengelernt

//Operatoren

**==** (gleich), **!=** (ungleich), > (größer), < (kleiner), >= (größer oder gleich), <= (kleiner oder gleich)

#### 2.2.2.4 Fallunterscheidung

Mit dem Keyword if- kann eine Fallunterscheidung zwischen Werten. Sofern die Bedingung zutrifft, kann eine Folge definiert werden.

*if(Bedingung):*

*Folge*

*else: #im anderen Fall zeige dies an*

*Folge im anderen Fall*

*Wenn, dann:*

*Ansonsten:*

#### 2.2.2.5 input-Funktion

Python sieht zudem eine input-Funktion vor. Die Input-Funktion ermöglicht es uns einen Wert auf der Konsole einzugeben. Dazu kann man die Eigenschaft der Variablen nutzen und diesen Wert abspeichern.

alter=input(“Wie alt bist du?”) #console: Wie alt bist du?

Eingabe durch user: 18

print(“Du bist” + alter + “Jahre alt”) #console: Du bist 18 Jahre alt.

#### 2.2.2.6 while-Schleife

Mit dem Signalwort while wird in Python eine Schleife/Wiederholung ausgeführt, die den Schleifenrumpf solange ausführt wie es in den Bedingungen im Schleifenkopf.

while Bedingung:

Schleifenrumpf ausführen

Beispiel:

i = 1

while i < 6:

print(i)

i += 1

#### 

#### 

#### 2.2.2.7 for-Schleife

Mit der For-Schleife kann man über eine Liste iterieren

attackenliste = [“ohrfeige”, “karatekick”, “schwerthieb”]

Beispiel:

for attacke in attackenliste

print(attacke)

for attacke in attackenliste

counter +=1

#### 2.2.2.8 Liste in Python

Eine Liste in Python wird wie folgt erstellt. Hierbei ist es notwendig die Grundlage der Variablen zu kennen. Die Variable wird initialisiert mit

meinePymon = [“Pytschu”,”Pyrak”,”Pysasam”]

Attacken= [“Tackle”, “Donnergrollen”, “Heilung”, “Kick” ]

In einer Liste sind diese Objekte gespeichert, dies können ganz einfache Zahlen sein oder auch String/ Objekte (Hinweis auf folgende Einführung ooP geben).

Die Liste selber ist aber auch ein Objekt. Sie hat verschiedene Funktionen, die sie ausführen kann. Grundsätzlich wird dies mit einer Punktnotation ausgeführt, danach folgt ein Signalwort / Methode.

Beispielsweise:

einen Wert hinzufügen: Attacken.append(“Kopfnuss”)

einen Wert entfernen: meinePymon.remove(“Pyrak2)

Zahl der Listeneinträge: meinePymon.count()

Länge der Liste: len(meinePymon)

Einfügen an Stelle 2: meinePymon.insert(1, “Pykachu”)

Du kannst Listen auch addieren: meinePymon = [Pytschu,Pyrak,Pysasam]

deinePymon = [Pyto, Muhrak]

meinePymon= meinePymon + deinePymon

!!! Beachte Der Index einer jeden Liste fängt bei 0 an !!!

### 2.2.3 Methoden schreiben in Python

Wie die Abläufe in den tasks ausgeführt worden sind, ist es auch möglich diese zu speichern und diese zu benennen.

Grundsätzlich sind Methoden schon aus der Einführung in die Liste bekannt.

Dies macht Sinn, da man Funktionen, die komplexer sind benennen will und wieder verwenden will. Hier ein Beispiel:

*def Anprobe(self):*

*if( x!= “gelb”)*

*print(x + “passt perfekt zu deinem outfit!”)*

*else:*

*print(“Du solltest nochmal was anderes aussuchen”)*

Methoden werden mit def als Signalwort markiert und benannt.

### 2.2.5 Live-Coding - erste Einführung objektorientierte Programmierung, Methoden und Klassen

Eine Klasse bezeichnet eine Schablone/Muster/Plan für ein Objekt.

Das Wort, das in Python eine Klasse erzeugt, lautet *class*.

*class Katze*

Beispiel zum Verhältnis Klasse-Objekt: Klasse(Name, Farbe, Alter) ist Katze, Objekt davon ist Vincent (Vincent, grau, 2), anderes Objekt ist Carlo(Carlo, gelb,1).

Die Klasse Katze ist ein Muster für jede Katze, die Klasse gibt grundlegende Eigenschaften vor, die jede Katze haben muss. In diesem Fall Name, Farbe und Alter.

Diese Schablone der Klasse enthält auch Methoden, die dann von jedem Objekt übernommen werden. Eine Methode ist immer eine Funktion der Klasse bzw. des individuellen Objektes.

Der sogenannte Konstruktor konstruiert eine Klasse. Das heißt auch, dass jede Klasse einen Konstruktor hat. Dieser ist in Python wie folgt zu erstellen:

*class Katze: //zuerst die Klasse*

*def \_\_init\_\_ (self, name, farbe, alter) // dann der Hinweis auf den Konstruktor def \_\_init\_\_*

*//in Klammern die Parameter die zum Erzeugen //benötigt werden sowie das Signalwort self*

*self.Name= name //Name wird aus der Eingabe(name) übernommen*

*self.Farbe=farbe //Farbe wird aus der Eingabe(farbe) übernommen*

*self.Alter=alter //Alter wird aus der Eingabe(alter) übernommen*

*self.entwurmt= false //grundsätzlich bei Geburt hat die Katze kein Wurmkur*

*def Miauen(self): //darunter folgen die Methoden der Klasse*

*print(self.Name+ “:”+“Miau”) //Codierung der Methode ->eingerückt*

*def hatGeburtstag(self):*

*self.Alter= self.Alter+1 // das Alter wird um eins erhöht*

*def ändereFarbe(self,farbe): //diese Methode übergibt einen Parameter farbe*

*self.Farbe=farbe //die neu eingegebene Farbe überschreibt die alte*

*der gibEntwurmungszustand(self): //ist die Katze entwurmt ?*

*return self.entwurmt //true oder false , ob entwurmt*

Diese Schablone der Klasse enthält auch Methoden, die dann von jedem Objekt übernommen werden. Eine Methode ist immer eine Funktion der Klasse bzw. des individuellen Objektes.

Das Signalwort *return* wird als “Wiedergabe” interpretiert. Wird also die Methode aufgerufen, wird als Konsequenz der Werte oder Variable nach dem return-Begriff wiedergegeben.

Ziel

* der Teilnehmer lernt Objekte und Klassen zu unterscheiden
* die return-Funktion wird eingeführt
* Exemplarvariablen werden nochmal verdeutlicht
* Konstruktor erläutern -> besondere Methode
* self ist bekannt und anwendbar

[Objektorientierte Programmierung Beispiel - Katze-](https://www.python-lernen.de/oop-objektorientierte-programmierung-grundlagen.htm)

### 2.2.6 warm-up nach der Mittagspause

Kurze Reflektion des am Vormittag gelernten mit einem niedrigschwelligem Quiz <https://kahoot.it/> . Antworten werden mit 2-3 Sätzen durch die Inspirer erläutert.

// Was ist ein Integer? Wie ist der Operator für das Multiplizieren? a=”hallo” - was ist das?

Ziel:

* Teilnehmer wieder niedrigschwellig auf die Thematik fokussieren

### 2.2.7 objektorientierte Programmierung

## 2.2.8 Spielablauf konstruieren

Als letzte Aufgabe von Tag 1 ist es vorgesehen, dass die Teilnehmer in dem Zoom-Call zusammenkommen.

Die Aufgabenstellung lautet dann einen Gameloop darzustellen. Die Aufgabe ist es nicht diesen schon zu programmieren. Vielmehr sollen die SuS sich Gedanken machen, welche Mechanismen nötig sind.

Die Ideenfindung wird in 4er Gruppen in BreakOut Sessions durchgeführt.

# 3. Zweiter Tag

## 3.1 Rekapitulation vom Vortag

Als Einstieg in den 2. Tag wird, angeleitet von den Inspirern das am Vortag erworbene Wissen zusammengetragen. Im Plenum sollen wesentliche Vokabeln zusammengetragen und erläutert werden.

Es ist Zeit für Rückfragen der SuS.

## 3.2 Gameloop erklären

Die Teilnehmer präsentieren ihre Gameloops vom Vortag und es gibt eine gemeinsame Diskussion. Zum Ende gehen wir gemeinsam den vorgefertigten Gameloop durch und erklären diesen. Dabei sehen wir das was am ersten Tag gelernt wurde einmal praktisch.

## 3.3 Gruppenphase

<https://replit.com/@b2bless/Pymonsters-Tag2#testbox.py>

Aufgabe: Die Teilnehmer bekommen ein Link zu einem replit gestellt und sollen die Charakter-Klasse erstellen.

Die SuS werden in 2er Gruppen eingeteilt. Es soll immer ein erfahrener mit einem unerfahrenen Schüler in einer Gruppe zusammen sein. Bei ungeraden Zahlen sind 3er Gruppen möglich.

Die SuS gehen in BreakOut Sessions

Die Charakterklasse soll mit dem gewonnenen Wissen befüllt werden.

Methodenköpfe sind vorhanden

Task:

2 Objekte der Klasse Charakter erstellen.

Charakter hat eine Liste mit Objekten der Klasse Attacke

mindestens 1 Attacke in der Attackenliste vorhanden

Die beiden erstellten Charakter kämpfen gegeneinander.

* die Teilnehmer sollen im Team kommunizieren
* Austausch im Sinne der Pair Programming-Methode, Stärken ergänzen
* weitere Erfahrung in der IDE sammeln, Erfolge durch sichere Nutzung
* Zusatz: Lösungsansatz ohne strikten vorgaben
* gelernte Funktionen verknüpfen

Charakter-Klasse erstellen

Bei der objektorientierten Programmierung werden Inhalte und Methoden in Objekten zusammengefasst, die diese unter Umständen auch für andere Objekte verfügbar machen.

Im Gegenteil zur prozeduralen Programmierung eignet sich für umfangreiche Projekte.

Hier werden Daten und Funktionen in Objekten zusammengefasst, die mit Hilfe von Klassenmustern erstellt werden können.

Vererbung wird aus Vereinfachungsgründen erst einmal weggelassen.

|  |  |
| --- | --- |
|  |  |
|  | class Charakter |
|  | def \_\_init\_\_(self, name, attacken, lebenspunkte, istAmLeben, typen, statusEffekt): |
|  | self.name = name |
|  | self.attacken = attacken |
|  | self.lebenspunkte = lebenspunkte |
|  | self.istAmLeben = istAmLeben |
|  | self.typen = typen |
|  | self.statusEffekt = statusEffekt |

charakter-Klasse wird erstellt. Der Konstruktor und das Prinzip der Exemplarvariablen wird den Teilnehmer aufgezeigt. Hier muss den Teilnehmer Zeit und Raum für Nachfragen gegeben werden.

Danach werden Methoden implementiert.

|  |
| --- |
| def veraendereLebenspunkte(self, veraenderung): |
| self.lebenspunkte -= veraenderung #positive werte sind Schaden, negative sind Heilung |
| if self.lebenspunkte <= 0: |
| self.istAmLeben = False |
| else: |
| self.istAmLeben = True |

Zuletzt wird ein Objekt eines Charakter erstellt. An dieser Stelle wird nochmal sichergestellt, dass der Unterschied zwischen Objekt und Klasse verinnerlicht ist.

spielereins = Charakter("Yannick", 200000, "Wasser")

# 

## 4 Code ausführen

Die Teilnehmer sehen die Ausführung ihres Codes und sind Motiviert diesen zu optimieren.

Im Prozess auftretende Probleme kommen bei der Ausführung ans Licht und können mit den Inspirern diskutiert werden.

Zusatzaufgaben für schnelle:

* Implementation eines Typ (Attackentyp, Charaktertyp)
* Heilung implementieren
* Gegner KI implementieren (Spieler 2 reagiert von alleine)
* CP System: eine Attacke darf nur x-mal ausgeführt werden
* etc…

## 5. Gegenseitiges vorstellen des Codes

Gegen Ende können die SuS ihren Code den anderen SuS präsentieren.

Die SuS können dem Rest der Gruppe von ihren Schwierigkeiten und Erfolgen berichten.

## 6. Verabschiedung

Wir verabschieden uns von den SuS und weisen noch einmal auf das Padlet für Feedback hin und wünschen schöne Ferien. Ggf. sagen wie sie weiter lernen können.

Feedback

Die Teilnehmer werden nach einem anonymen Feedback über Padlet gefragt.

## Quelle

https://www.python-kurs.eu/klassen.php

<https://www.python-lernen.de/oop-objektorientierte-programmierung-grundlagen.htm>

<https://www.python-forum.de/viewtopic.php?t=40224>

https://www.w3schools.com/python/python\_while\_loops.asp

## Teilnehmerliste

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Teilnehmer** | **Geschlecht** | **Alter** | **Erfahrungsstand** | **Anwesend** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Gruppenliste

|  |  |  |  |
| --- | --- | --- | --- |
| Gruppenname | Teilnehmer 1 | Teilnehmer 2 | Anmerkungen |
| Gruppe 1 |  |  |  |

## Erstellt durch

Arne Jurr, Jan Erik Neelsen, Yannick Proegler, Benjamin Blessing

## Lizenz

This license lets others distribute, remix, adapt, and build upon your work, even commercially, as long as they credit you for the original creation. This is the most accommodating of licenses offered. Recommended for maximum dissemination and use of licensed materials.

## 