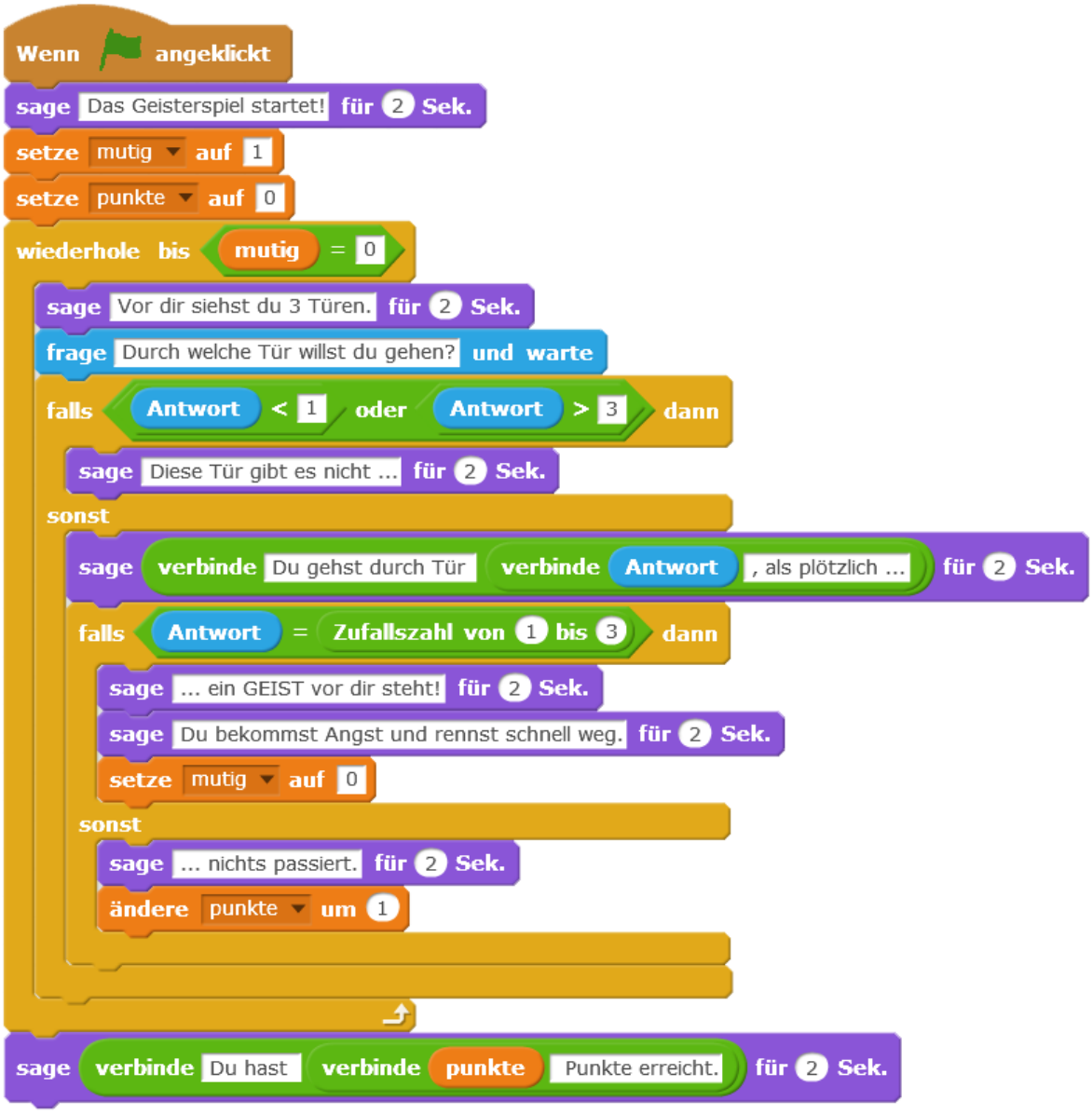
Das Geisterspiel

Zum Einstieg in Python, wollen wir zunächst ein kleines Spiel programmieren. Dieses Spiel handelt von mehreren Türen und einem Geist. Damit du weißt, was genau auf dich zu kommt, kannst du dir das folgende Scratch-Skript anschauen. Hier wurde das Spiel bereits einmal programmiert.



Aufgabe 1

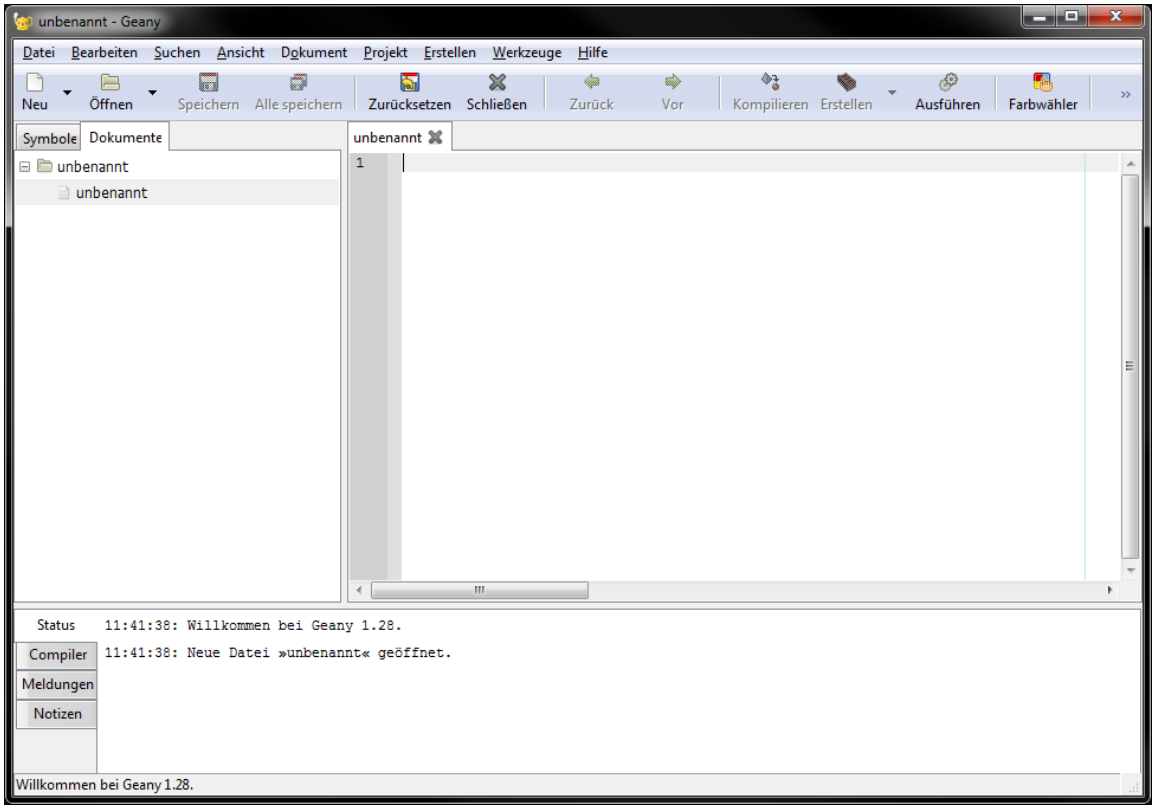
Schau dir das obige Skript genau an und versuche die einzelnen Abschnitte zu verstehen. Beschreibe kurz für dich den Ablauf des Spiels.

Aufgabe 2

Arbeite die folgenden Schritte sorgfälltig durch.

Schritt 1: Die Entwicklungsumgebung starten

Zunächst einmal musst du die Entwicklungsumgebung für Python starten. Sofern dir dein Lehrer nichts anderes gesagt hat, startest du das Programm *Geany*. Du findest es entweder auf dem *Desktop* oder im *Startmenü* und *Alle Programme*. Das Fenster sieht dann ungefähr so aus:



Schritt 2: Die Programmierung

Wir werden nun die Bausteine aus dem Scratch-Skript Baustein für Baustein von oben nach unten durchgehen. Dabei kannst du immer wieder vergleichen, welcher Befehl in Python zu dem Baustein in Scratch passt.

Den ersten Baustein *Wenn grüne Fahne angeklickt* können wir ignorieren, daher starten wir erst einmal mit den weiteren drei Bausteinen. Schreibe folgendes (**ohne** die Zeilennummern) in *Geany* ab:

```
01  print("Das Geisterspiel startet!")
02  mutig = 1
03  punkte = 0
```

Anmerkung: Der Befehl *print* sorgt dafür, dass Ausgaben auf dem Bildschirm angezeigt werden. Das setzen von Varianten ist in Python auch ganz einfach. Du kannst einfach die Variable gleich ihrem Wert setzen.

Bei dem nächsten Baustein handelt es sich um eine Schleife, die solange durchlaufen wird, wie der Spieler mutig ist. Anschließend soll dem Spieler gesagt werden, dass er vor drei Türen steht. In Python sieht dies dann so aus:

```
04  while(mutig == 1):
05      print("Vor dir siehst du 3 Türen.")
```

Anmerkung: Bei vergleichenden Aussagen (wie zum Beispiel ist mutig gleich 1), wird immer ein doppeltes Gleichheitszeichen verwendet. Damit unterscheidet man Zuweisungen von Werten (einfaches Gleichheitszeichen) von Vergleichungen (doppeltes Gleichheitszeichen).
Sehr wichtig ist, dass in Python die Zeilen innerhalb einer Schleife oder Bedingung immer eingerückt werden. Sehen kannst du dies auch in Scratch, wo die weiteren Bausteine auch etwas nach rechts rücken. Drücke in *Geany* dazu einfach die *Tabulator* Taste.

Nun soll der Spieler aufgefordert werden, sich für eine Tür zu entscheiden. In Python verwenden wir hierfür:

```
06    Antwort = input("Durch welche Tür willst du gehen?")
```

Anmerkung: Eingabeaufforderungen werden in Python mit dem Befehl *input* eingeleitet. Vor dem Befehl muss die erwartete Eingabe einer Variable zugewiesen werden. Dabei muss diese Variable nicht zwingend Antwort heißen.

Nun erfolgt zum ersten Mal etwas typisches für Python. Bei Scratch musstest du dir keine Gedanken um Datentypen von Variablen machen. Du konntest ganz einfach die Zahl 1 mit der Eingabe 1 bzw. dem Text bestehend aus der Ziffer 1 vergleichen. In Python ist dies nicht so einfach möglich. Eingaben werden immer als Strings bzw. Texte behandelt. Das bedeutet, dass bei einem Vergleich die Eingabe nicht direkt mit der Zahl 1 verglichen werden kann. Um dies doch zu tun, müssen wir die Eingabe in einen Integer bzw. ganze Zahl umwandeln.

Der folgende Quellcode zeigt dir, wie ein *falls-dann*-Baustein in Python aussehen muss:

```
07    if((int(Antwort) < 1) or (int(Antwort) > 3)):
08        print("Diese Tür gibt es nicht ...")
09    else:
10        print("Du gehst durch Tür " + Antwort + ", als plötzlich ...")
```

Anmerkung: Mit dem *int* Befehl kann man zum Beispiel aus einem String einen Integer machen.
Um im *print* Befehl mehrere Strings (Texte, Variablen) zu verketten, kann man einfach das + verwenden.
Bei Python ist immer auf den Datentyp von Variablen zu achten. Wäre zum Beispiel die Variable *Antwort* kein String, so wäre das Verketten nicht so direkt möglich. (Dazu später mehr.)

Weiter geht es mit der Frage, ob hinter der ausgewählten Tür sich nun der Geist befindet. Es gibt also wieder einen *falls-dann*-Baustein mit entsprechenden Ausgaben und Variablenzuweisungen, der in Python dann so aussieht:

```
11        if((int(Antwort) == randint(1, 3))):
12            print("... ein GEIST vor dir steht!")
13            print("Du bekommst Angst und rennst schnell weg.")
14            mutig = 0
15        else:
16            print("... nichts passiert.")
17            punkte = punkte + 1
```

Anmerkung: In Python werden *falls-dann*-Bausteine zu *if-else* Anweisungen.
Der Befehl *randint* ermittelt eine ganzstellige Zufallszahl. Bei dem obigen Beispiel wird eine Zufallszahl zwischen 1 und 3 ermittelt.
Mit dem *int* Befehl kann man zum Beispiel aus einem String einen Integer machen.
Um im *print* Befehl mehrere Strings (Texte, Variablen) zu verketten, kann man einfach

das + verwenden.
Bei Python ist immer auf den Datentyp von Variablen zu achten. Wäre zum Beispiel die Variable *Antwort* kein String, so wäre das Verketten nicht so direkt möglich. (Dazu später mehr.)

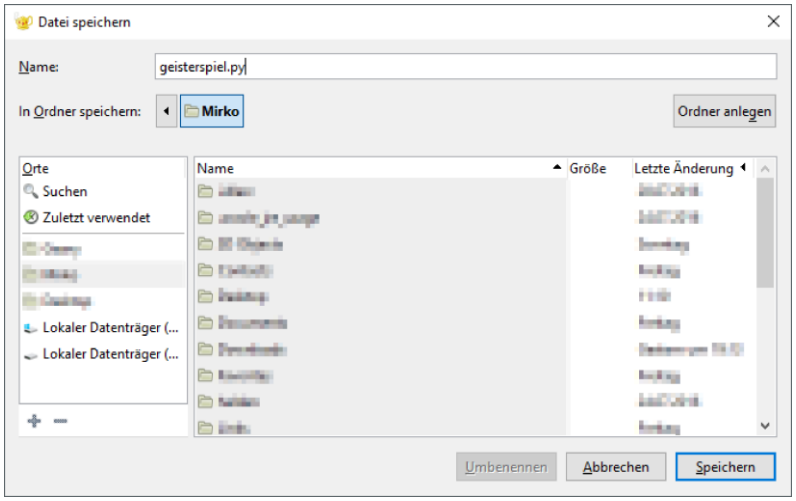
Am Ende des Programms, sobald der Spieler nicht mehr mutig ist, sollen noch die Punkte angezeigt werden:

```
18  print("Du hast " + str(punkte) + " Punkte erreicht.")
```

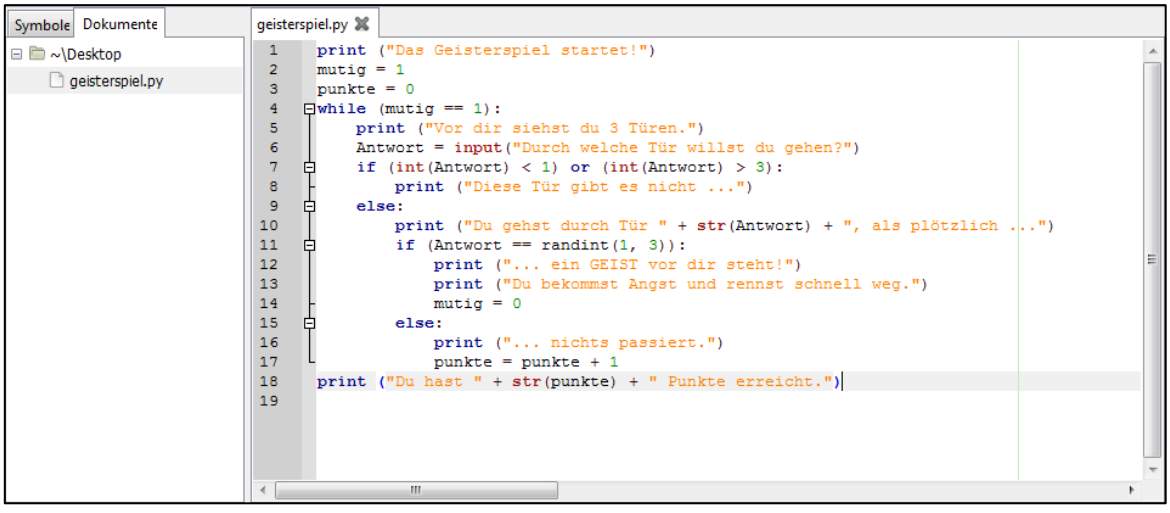
Anmerkung: Auch hier ist auf den Datentyp der Variable punkte zu achten. Da die Variable ein Integer ist (durch die Zuweisung von 0 am Anfang), müssen wir diese für die Ausgabe kurzzeitig zu einem String machen. Dies geht mittels *str* Befehl.

Schritt 3: Erste Test mit vorprogrammiertem Crash

Speichere zunächst dein Programm ab. Klicke dazu auf Speichern, wähle einen passenden Speicherort und achte beim Dateinamen auch auf das richtige Dateiformat für Python. Als Dateinamen empfehlen wir geisterspiel.py.



Sobald du das Programm abgespeichert hast, sollte dir auffallen, dass deine Befehle im Programm farbig sind. *Geany* markiert dir so bestimmte Befehle aus Python, damit du dich besser im Quellcode orientieren kannst. Wenn du fertig bist, sollte das ganze so aussehen:



Nun wollen wir das ganze aber auch einmal testen. Du musst auf Ausführen klicken, um dein Programm zu starten.

Beim Testen solltest du merken, dass dein Programm direkt nach der Eingabe nicht mehr funktionieren sollte. Die Fehlermeldung sollte wie folgt lauten:

```
Traceback (most recent call last):
  File "geisterspiel.py", line 11, in <module>
    if (Antwort == randint(1, 3)):
NameError: name 'randint' is not defined
```

Ganz unten steht dabei die eigentliche Fehlermeldung. Der Befehl randint ist nicht definiert. Das heißt, dass Python damit gar nichts anfangen kann. (Die Lösung dafür siehst du gleich.) Die Zeilen darüber helfen dir, den Ursprung des Fehlers herauszufinden. In der zweiten Zeile steht, wo der Fehler aufgetreten ist: In der Datei geisterspiel.py, in der Zeile 11. Darunter wird dir in der dritten Zeile der Quellcode der fehlerhaften Zeile angezeigt.

Fehler treten beim Programmieren immer wieder auf und sind generell auch nichts schlimmes. Wichtig ist nur, sich die Fehlermeldung genau anzuschauen und den Fehler so ausfindig zu machen. Sollte dir das mal nicht direkt gelingen, kann es auch helfen, eine Suchmaschiene zu fragen.

Schritt 4: Fehler beheben

Wie bereits erwähnt, kennt Python den Befehl *randint* noch nicht. Das lässt sich schnell ändern und ist auch nicht ungewöhnlich. Python beinhaltet nicht immer alle Befehle, da sonst die Programme zu groß wären, wenn für jede Eventualität jeder Befehl eingebunden wäre. Daher, müssen manche Befehle erst noch importiert bzw. bekannt gemacht werden.

In unserem Fall heißt das, dass wir folgenden Quellcode an den Anfang unseres Programms hinzufügen müssen:

```
from random import randint
```

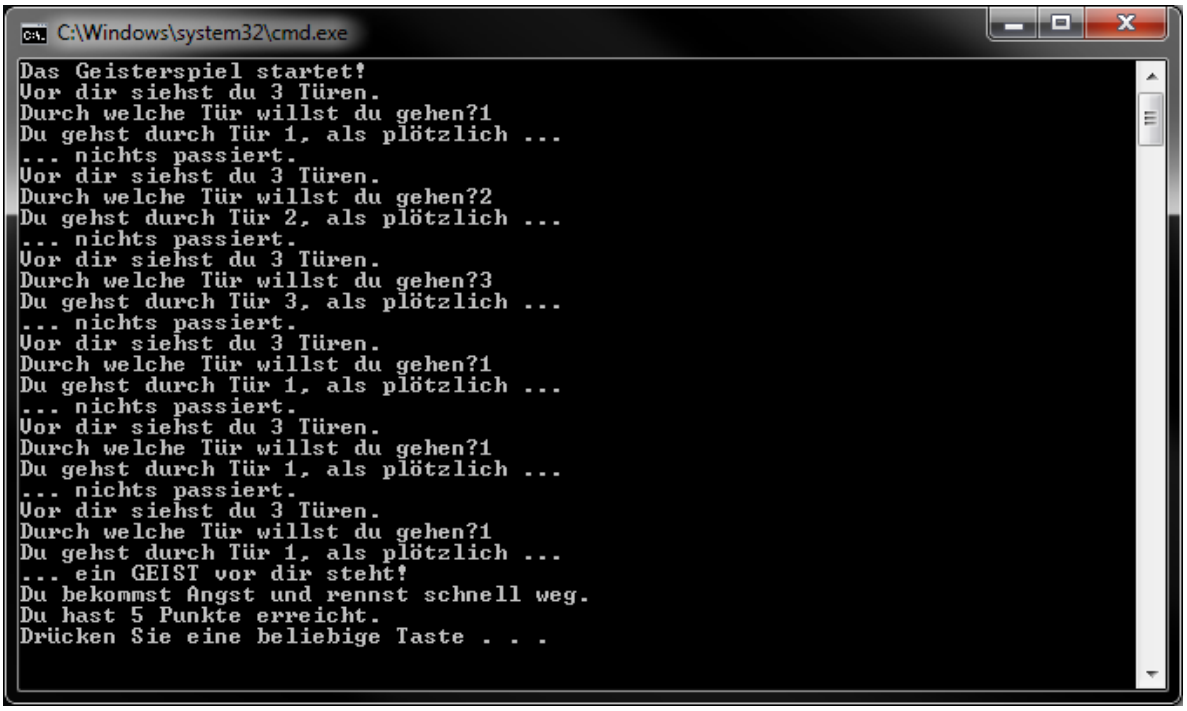
Das ganze Programm sollte nun wie folgt aussehen:

```
01  from random import randint
02
03  print("Das Geisterspiel startet!")
04  mutig = 1
05  punkte = 0
06  while(mutig == 1):
07      print("Vor dir siehst du 3 Türen.")
08      Antwort = input("Durch welche Tür willst du gehen?")
09      if((int(Antwort) < 1) or (int(Antwort) > 3)):
10          print("Diese Tür gibt es nicht ...")
11      else:
12          print("Du gehst durch Tür " + Antwort + ", als plötzlich ...")
```

```
13          if((int(Antwort) == randint(1, 3))):
14              print("... ein GEIST vor dir steht!")
15              print("Du bekommst Angst und rennst schnell weg.")
16              mutig = 0
17          else:
18              print("... nichts passiert.")
19      punkte = punkte + 1
20  print("Du hast " + str(punkte) + " Punkte erreicht.")
```

Schritt 5: Spiel testen

Du kannst nun dein Spiel testen. Das Ergegniss könnte ungefähr so aussehen:



Aufgabe 3

Mache nun selbst kleinere Änderungen am Spiel. Füge zum Beispiel weitere Türen hinzu oder sorge dafür, dass der Text etwas leserlicher ist.

Anmerkung: Möchtest du im *print* Befehl einen Zeilenumbruch machen, dann musst du an der entsprechenden Stelle ein \n einfügen, wie im folgenden Beispiel gezeigt:

```
print("Zeile 1\nZeile2")
```

Aufgabe 4

In der folgenden Tabelle siehst du links, die Bausteine aus Scratch. Schreibe in die rechten Spalten den entsprechenden Befehl in Python.

Scratch	Python