TicTacToe

Hier wirst du Schritt für Schritt lernen wie man TicTacToe progrmmiert zu zweit in der Konsole spielen spielen kann. An manchen Stellen werden in den Codestücken `??` stehen. Diese Fragezeichen musst du durch den korrekten Wert ersetzten. Damit du am Ende ein fertiges Programm hast, musst du die einzelnen Teile mitschreiben und richtig zusammenfügen.

1 Vorbereitung

I. Das Ganze wird ein zwei Spieler-Spiel. Als Erstes fragen wir also nach den Namen der Spieler.

```
1  s1 = input("Spieler 1, wie heißt du? ")
2  s2 = ??
```

Spieler 1 und Spieler 2 können nun ihren Namen eintippen. Die Namen werden in den Variablen `s1` und `s2` gespeichert.

Deine Aufgabe: Vervollständige die zweite Zeile.

II. Um die beiden Spieler auf dem Spielfeld unterscheiden, benötigt jeder Spieler ein Zeichen, z.B. 'X' und 'O'. Dieses Zeichen wird später auf dem Spielfeld angezeigt.

```
1  z1 = input("Spieler 1, gib bitte dein Zeichen (ein Buchstabe)
   ↪ ein? ")
2  z2 = ??
```

Deine Aufgabe: Vervollständige die zweite Zeile.

III. Damit die Spieler Bescheid wissen, wir mitspielt und wer welches Zeichen hat sollten wir die Infos nochmal zusammengefasst ausgeben! Z.B.:

```
Es spielen Mark: M und Ricarda: R. Mark beginnt.
```

Deine Aufgabe: Erzeuge diese Ausgabe.

```
1  print(??)
```

Bubber

2 Das Spielfeld

I. Das Spielfeld besteht aus 3 auf 3 Kästchen. Nummeriern wir diese reihenweise durch erhalten wir folgendes Spielfeld.

```
0 | 1 | 2
3 | 4 | 5
6 | 7 | 8
```

Wir müssen speichern, welcher Spieler welches Feld markiert hat. Dafür werden wir eine Liste verwenden. Unsere Liste hat also 9 Einträge. Die Einträge sind am Anfang 0 und werden nach und nach durch die Zeichen der Spieler ersetzt! Du kannst dir die Liste so vorstellen, dass dort die drei Zeilen des Spielfelds gespeichert werden. Wenn du dir die Darstellung der Liste anschaust, wird dir hoffentlich klar wie das Ganze aussieht.

```
1  felder = [0, 0, 0,
2            0, 0, 0,
3            0, 0, 0]
```

II. Hat z.B. Mark das Feld 4 belegt und Ricarda Feld 9:

```
1  0 | 1 | 2   entspricht: felder =  [0 , 0,  0,
2  M | 4 | 5                          'M', 0,  0,
3  6 | 7 | R                           0 , 0, 'R']
```

Dir ist hoffentlich aufgefallen, dass `'M'` bzw. `'R'` ein Text ist und in Anführungzeichen steht, die 0 hingegen nicht, diese ist eine Zahl. Dies ist eine Absicherung, damit kein kleverer Spieler die 0 als sein Zeichen angibt und damit sofort gewinnt. Wird die 0 als Zeichen gewählt so es der Text `'0'` und nicht die Zahl! Dies macht für den Computer einen Unterschied.

III. Wir müssen nun das Spielfeld anzeigen, indem wir es auf der Konsole mit `print(..)` ausgeben. Dafür schreiben wir eine Funktion! Eine Funktion sammelt einfach eine Reihe von Anweisungen, die in einem Block zusammengefasst werden. Dieser Block kann beliebig oft ausgeführt werden. Funktionen sind also sehr praktisch, wenn man Dinge wiederholt tun will.

```
1  def spielfeld_anzeigen():
2
3    for zahl in range(0, 9):
4      # Das Zeichen kann 0, z1 oder z2 sein
5      zeichen = felder[zahl]
6
7      # Entweder die Feldnummer, oder das Zeichen ausgeben
8      if zeichen == 0:
9        print(zahl, end=" ")
```

```
10        else:
11          print(zeichen, end=" ")
12
13        # print erzeugt normalerweise einen Zeilenumbruch
14        # mit , end=" " haben wir dies verhindert.
15        # Wir wollen nur nach jedem dritten Zeichen einen Umbruch
16        if zahl % 3 == 2:
17          print("")
```

Unsere Liste hat 9 Elemente, wir beginnen bei 0 zu zählen. **Wichtig:** Wir beginnen beim Programmieren immer bei 0 zu zählen. Danach geben wir entweder die Feldnummer aus, falls ein Feld unbesetzt ist, ansonsten das Zeichen des Spielers. Die Ausgabe ist etwas komplizierter. Wir wollen immer 3 Zeichen in einer Reihe anzeigen. D.h. wir müssen verhindern, dass `print(..)` automatisch einen Zeilenumbruch einfügt. Diesen müssen wir selbst bei jeder dritten Zahl erzeugen.

IV. Eine Funktion wird erst ausgeführt, wenn man sie über ihren Namen aufruft. Wir können das Spielfeld also jederzeit so ausgeben lassen:

```
1   spielfeld_anzeigen()
```

3 Spiellogik

I. **Aktiver Spieler:** Während des Spiels ist immer ein Spieler aktiv und darf einen Zug machen. Nachdem der Zug abgeschlossen ist, wird der andere Spieler der aktive Spieler. Wir benötigen also Variablen um den aktiven Spieler und desssen Zeichen zu speichern.

```
1   # Spieler 1 darf beginnen
2   aktiver_spieler = ??
3   aktives_zeichen = ??
```

II. **Die Spielschleife:** Das Spiel geht solange, bis ein Spieler gewonnen hat oder alle Felder belegt sind und das Spiel mit Unentschieden endet.

Frage: Wieviele mögliche Züge gibt es?

```
1   zuege = 0
2   while zuege < ??:
3     # Inhalt kommt noch...
4
5     # Ein Zug erfolgreich abgeschlossen
6     zuege = zuege + 1
```

III. **Einen Zug machen:** Da die Felder praktischerweise in einer durchnummerierten Liste stehen, können wir die Nummerierung ausnutzen. Um also anzugeben, dass man das Feld 5 besetzen will, muss der Spieler einen 5 eingeben.

```
print("Du bist dran", aktiver_spieler)
eingabe = input("Welches Feld möchtest du besetzen?")
# In eingabe steht ein Text => in eine Zahl umwandeln
feld_nummer = ??
```

Wichtig: Der Code aus diesem und den nächsten Schritten muss innerhalb der Spielschleife stehen.

IV. **Korrekter Zug:** Überlege dir kurz, bevor du weiter liest, wann ein Zug erlaubt ist und welche Werte der Benutzer eingeben darf? Ein Zug ist erlaubt, wenn das Feld frei ist. Der Benutzer muss eine Zahl zwischen 0-8 eingegeben und das Feld, also der Eintrag in der Liste an dieser Stelle, muss noch unbesetzt sein.

```
# Ist die Eingabe korrekt?
if feld_nummer < 0 or feld_nummer > 8:
  continue

# Ist das Feld schon besetzt?
if felder[feld_nummer] != 0:
  continue
```

Wird eine ungültige Nummer eingeben oder ist das Feld schon besetzt, führen wir die Spielschleife nicht normal aus. Stattdessen springen wir mit `continue` wieder an den Schleifenanfang. Dadurch wird wird kein Feld gesetzt und stattedessen wird der Spieler aufgefordert erneut eine Feldnummer einzugeben.

V. **Zug durchführen:** Nun müssen wir das Spielfeld aktualisieren und das gewählte Feld mit dem Zeichen des aktiven Spielers belegen.

```
felder[feld_nummer] = aktives_zeichen
spielfeld_anzeigen()
```

4 Gewonnen?

Wir benötigen nun eine Funktion um feststellen zu können, ob ein Spieler (der aktive Spieler) durch seinen letzten Zug gewonnen hat.

I. Zunächst müssen wir uns überlegen auf wie viele Arten ein Spieler gewinnen kann.

Frage: Wie viele Reihen, Zeilen, Diagonalen gibt es?

Wir müssen lediglich jede dieser Möglichkeiten überprüfen, um festzustellen ob der aktive Spieler gewonnen hat. Dies erfordert ein bisschen Schreibarbeit:

```
def hat_gewonnen():
  # erste Reihe überprüfen
  if felder[0] == aktives_zeichen and felder[1] ==
↪ aktives_zeichen and felder[1] == aktives_zeichen:
    return True
  # Weitere Zeilen ...

  # Erste Spalte überprüfen
  if felder[0] == aktives_zeichen and felder[3] ==
↪ aktives_zeichen and felder[6] == aktives_zeichen:
    return True

  # Weitere Spalten ...

  # Erste Diagonale überprüfen
  if felder[0] == aktives_zeichen and felder[4] ==
↪ aktives_zeichen and felder[8] == aktives_zeichen:
    return True

  # Weitere Diagonalen ...

  # Falls nicht gewonnen => False
  return False
```

Die Funktion überprüft, ob die Liste eine Reihe, Spalte oder Diagonale mit drei gleichen Zeichen enthält.
Wichtig: Diese Funktion muss vor der Spielschleife stehen.

II. Nun müssen wir nur noch in der Spielschleife überprüfen, ob der aktive Spieler gewonnen hat und falls nicht, den aktiven Spieler wechseln.

```
# Hat der aktive Spieler gewonnen
if hat_gewonnen():
  print(aktiver_spieler, "hat gewonnen!")
  break # Die Schleife verlassen

# Spieler wechseln
if aktiver_spieler == s1:
  aktiver_spieler = s2
  aktives_zeichen = z2
else:
  aktiver_spieler = s1
  aktives_zeichen = z1
```

5 Was ist mit Unentschieden?

Wir haben nun die Spiellogik und Spielschleife progrmmiert, wenn der aktive Spieler gewinnt, wird dies erkannt. Allerdings könnte es vorkommen, dass das Spiel Unentschieden ausgeht. Wann ist die Spielschleife zu Ende? Nachdem alle möglichen Felder besetzt sind. In diesem Fall ist es ein unentschieden. Folglich müssen wir nur nach der Spielschleife abfragen, wieviele Züge vergangen sind.

```
1 if zuege == ??:
2   print("Unentschieden!")
```

6 Das wars!

Du solltest jetzt, wenn du alle Teile richtig zusammenfügst ein fertiges TicTacToespiel haben.