Blatt 4

Vorkurs Bauingenieurwesen - Informatik - 19.04.24

Heute spielen wir Tic-Tac-Toe im lokalen Mehrspielermodus. Die Aufgaben 1 bis 4 sind die nötigen Bausteine, die dann in Aufgabe 5 zu dem fertigen Spiel zusammengesetzt werden.

Aufgabe 1

Das Feld soll aus einer Liste mit neun Einträgen bestehen, wobei jeder Eintrag eine Zeichenkette (`str`) ist. Die Zeichenketten sollen später die Werte `"_"`, `"X"` und `"O"` annehmen können, zu Beginn aber alle den Wert `"_"`. Schreibe eine Funktion `baue_spielfeld`, welche den Wert `["_", "_", "_", "_", "_", "_", "_", "_", "_"]` zurückgibt.

```
def baue_spielfeld() -> list[str]:
```

Aufgabe 2

Schreibe eine Funktion `zeige_spielfeld`, die den aktuellen Spielstand in der Konsole ausgibt. Eine mögliche Formatierung ist etwa

```
Feld:   Züge:
_ X _   1 2 3
O O _   4 5 6
_ X _   7 8 9
```

für das Spielfeld `["_", "X", "_", "O", "O", "_", "_", "X", "_"]`. Die Kopfzeile ist genau wie der rechte Block optional. Tipp: `print("a", "b", "c")` gibt `a b c` auf der Konsole aus.

```
def zeige_spielfeld(feld: list[str]):
```

Aufgabe 3

Schreibe eine Funktion `naechster_zug`, die eine Eingabe eines noch freien Feldes erwartet. Umgesetzt kann die Funktion wie folgt werden. Innerhalb einer Endlosschleife (`while True:`) soll

1. eine Eingabe als Zeichenkette entgegengenommen werden (`input("lelele")` gibt `"lelele"` auf der Konsole aus und erwartet dann eine Eingabe, die mit Enter abgeschlossen wird.)
2. getestet werden ob diese zu einer Zahl umgewandelt werden kann (`"abc".isdigit()` gibt zurück, ob `"abc"` zu einer Zahl umgewandelt werden kann.)
3. wenn ja: erzeue eine neue Variable mit der Eingabe umgewandelt als Zahl (`int("123")` gibt `123` zurück.)
4. gucke ob die Zahl minus 1 ein gültiger Index in `feld` ist
5. wenn ja: gucke ob an dieser Stelle `"_"` steht.
6. wenn ja: gebe die Zahl minus 1 zurück.

```
def naechster_zug(feld: list[str]) -> int:
```

Aufgabe 4

Um zu testen ob ein Spieler gewonnen hat, muss für jede Zeile, für jede Spalte und für die beiden Diagonalen geguckt werden, ob eine solche komplett mit dem Symbol des Spielers (also `"X"` oder `"O"`) gefüllt ist. Schreibe eine Funktion `gewonnen`, die guckt, ob die Einträge einer Zeile, Spalte oder Diagonale von `feld` komplett dem Symbol `spieler` entsprechen. Umgesetzt werden kann die Fnktion etwa mit acht `if`-Bedingungen, eine pro Zeile, Spalte und Diagonale. Ist eine Bedingung erfüllt, wird `True` zurückgegeben, sonst ganz am Ende `False`.

```
def gewonnen(feld: list[str],
    spieler: str) -> bool:
```

Aufgabe 5

Schreibe eine Funktion `spielen`, die den Spielablauf implementiert. Erzeuge zuerst eine Variable `feld`, in der das Spielfeld gespeichert ist. Der Rest passiert in einer Endlosschleife:

1. für jedes Spielersymbol in `["X", "O"]`:
2. zeige das Spielfeld auf dem Bildschirm
3. erwarte die Eingabe des Spielers
4. setzte den Wert in `feld`
5. teste ob der Spieler gewonnen hat
6. teste ob noch Zugoptionen über sind (`"_" in feld`)
7. wenn gewonnen oder keine Zugoptionen mehr: `return`

```
def spielen():
```