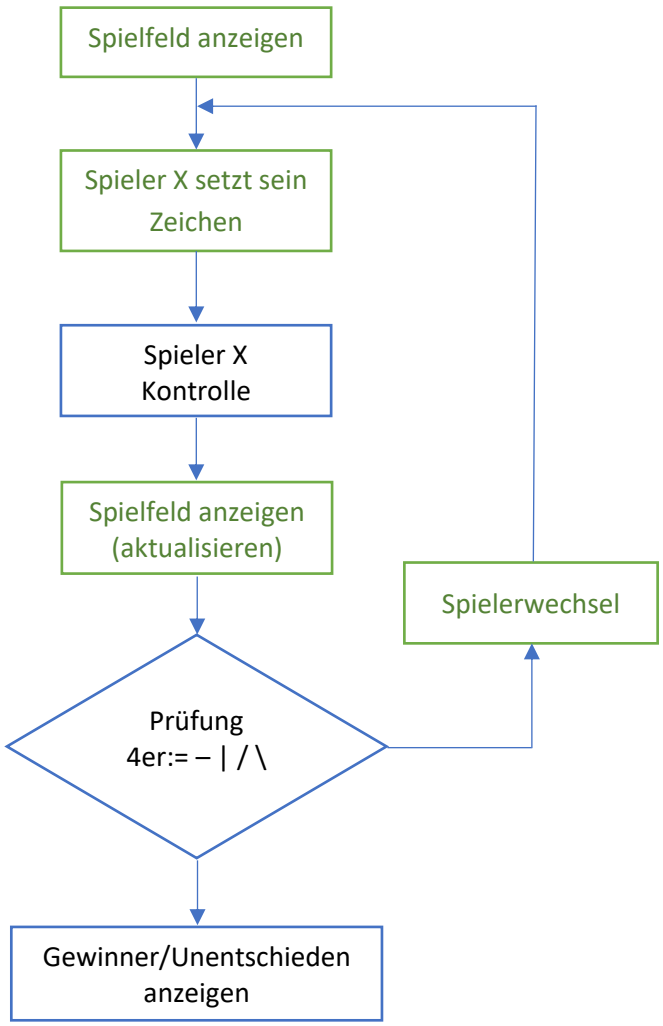
Aufgabe 5 - (Teil 2 von 4 Gewinnt)

Situation: Laut einer aktuellen VuMA-Umfrage spielen mehr als 40 Prozent der Deutschen Computer- und Videospiele. In der Altersgruppe der 14- bis 29-Jährigen beträgt der Anteil der Videospieler sogar über 70 Prozent. Die meisten Gamer interessieren sich für Action-Spiele/ Ego-Shooter, Abenteuer-Spiele und Geschicklichkeitsspiele.

Es sollte das bekannte Spiel **4 GEWINNT** als Python-Programm erstellt werden.

Zuerst betrachten wir den **Spielverlauf** und überlegen uns, welche Prozesse wir als **Funktionen** umsetzen können.

Spielverlauf



Folgende **Funktionen** leiten wir vom Spielverlauf ab:

Spielfeld anzeigen	spielfeld_ausgeben()
Spieler Eingabe	spieler_eingabe()
Kontrolle der Spielereingabe	spieler_eingabekontrolle()

Spielerwechsel	spieler_wechsel()
Prüfung 4er	spieler_gewinnt()
Prüfung unentschieden	spiel_unentschieden()

In der heutigen Aufgabe gilt es zwei Aufgaben zu lösen: Die Eingabe des Spielers zu verarbeiten und den Spielerwechsel zu vollziehen.

b.) Spielerwechsel

Das Spielfeld ist nun bereit für das Spiel. Die Spieler geben abwechselnd ein Stein nach dem anderen ein. Wir führen am Anfang des Programms eine Variable aktive_spieler="X" ein. Im Hauptprogramm wird dann mit jedem Spielstein ein Spielerwechsel, mit der Funktion spieler_wechsel(), bewirkt.

WENN aktive_spieler=="X" DANN aktive_spieler="O" SONST aktive_spieler="X"

c.) Spielereingabe

Nach der Eingabe gilt es zu prüfen, ob in der Spalte bereits ein Stein vorhanden ist, so dass der Stein oberhalb des Steins platziert wird. Die Eingabe der Spalte erfolgt als integer Zahl.

```
  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
5 |   |   |   |   |   |   |   |
4 |   |   |   |   |   |   |   |
3 |   |   |   |   |   |   |   |
2 |   |   |   |   |   |   |   |
1 |   |   | X |   |   |   |   |
0 |   |   | X |   |   |   |   |
```

Hinweis:
die Zahl der Eingabe (z.B. 2) entspricht der Nummer der Spalte, also auch beginnend mit z[2],z[9],z[16],z[23], z[30], z[37]

Also falls z[2] mit X oder O belegt, dann erfolgt die Platzierung des Steins oberhalb in z[9]. Falls aber auch z[9] belegt ist, wird oberhalb in z[16] der Stein platziert. Usw.
Zu prüfen ist dann die Spalte 2: → 2+7=**9**, 9+7=**16**, 16+7=**23**, 23+7=**30** und 30+7=**37**
Lösen Sie es auf mit Hilfe einer for-Schleife und prüfen Sie ob das Feld bereits belegt ist, ansonsten kann die for-Schleife mit break unterbrochen werden. Vergessen Sie nicht vorher den Stein zu platzierten.

Erstellen Sie hierzu die Funktion: spieler_eingabe()

Ihr Programm… **afg5_spielereingabe.py**

```
z=[" "," "," "," ",]                    ← " ", etc. 42 mal
anzSteine=1
aktive_spieler="X"

def spieler_wechsel():
        b.)

def spieler_eingabe():
        c.)

def spielfeld_anzeigen():
        print("  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |")
        print("5 | "+z[35]+" | "+z[36]+" | "+z[37]+" | "+z[38]+" | "+z[39]+" | "+z[40]+" | "+z[41]+" |")
        print("4 | "+z[28]+" | "+z[29]+" | "+z[30]+" | "+z[31]+" | "+z[32]+" | "+z[33]+" | "+z[34]+" |")
        print("3 | "+z[21]+" | "+z[22]+" | "+z[23]+" | "+z[24]+" | "+z[25]+" | "+z[26]+" | "+z[27]+" |")
        print("2 | "+z[14]+" | "+z[15]+" | "+z[16]+" | "+z[17]+" | "+z[18]+" | "+z[19]+" | "+z[20]+" |")
        print("1 | "+z[7]+" | "+z[8]+" | "+z[9]+" | "+z[10]+" | "+z[11]+" | "+z[12]+" | "+z[13]+" |")
        print("0 | "+z[0]+" | "+z[1]+" | "+z[2]+" | "+z[3]+" | "+z[4]+" | "+z[5]+" | "+z[6]+" |")


#Hauptprogramm
spielfeld_anzeigen()
while anzSteine<43:
        spieler_eingabe()
        spielfeld_anzeigen()
        anzSteine=anzSteine+1
        spieler_wechsel()
```