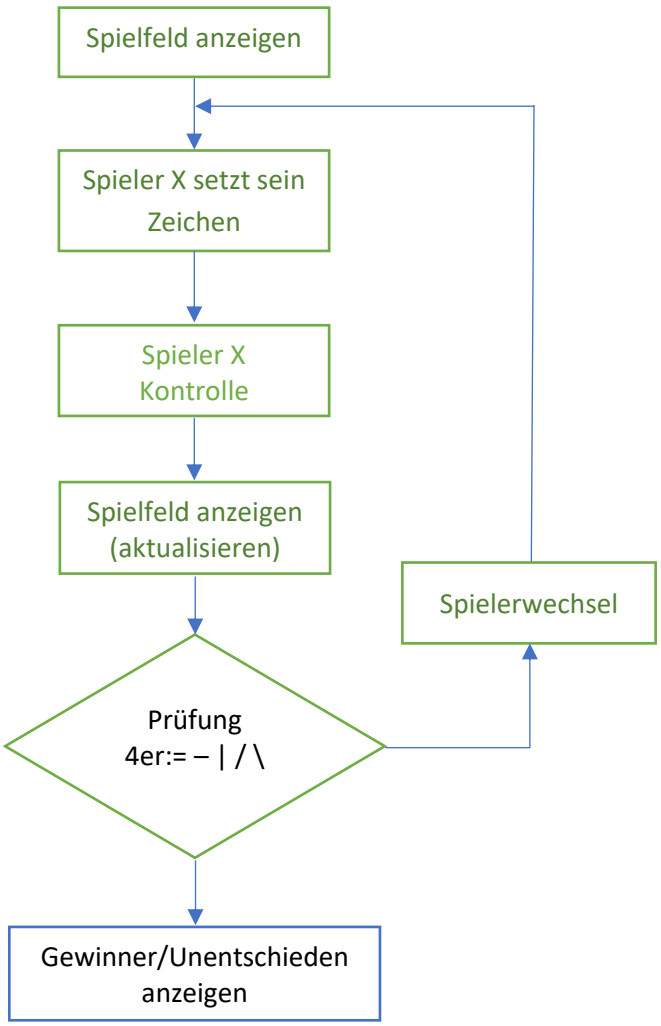
Aufgabe 6 - (Teil 3 von 4 Gewinnt)

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 zu lösen: Hat der Spieler mit seinem Stein ein 4er erzeigt?

d.) Spieler hat gewonnen

Nach der Eingabe gilt es zu prüfen, nachdem sein Stein auf dem Spielfeld angezeigt wird, ob der Spieler ein 4er erzeugt hat. Geprüft wird Horizontal, Vertikal und Diagonal.

Vertikale

```
  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
5 |   |   |   |   |   |   |   |
4 |   |   |   |   |   |   |   |
3 |   |   | X |   |   |   |   |   z[23]
2 |   |   | X |   |   |   |   |   z[16]
1 |   |   | X |   |   |   |   |   z[9]
0 |   |   | X |   |   |   |   |   z[2]
```

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]

z.B. der letzte Stein **X** wurde gesetzt – **z[23]** in der vierten Zeile. Zu überprüfen sind die darunter liegenden Steine ob alle drei X. REGEL: Die Vertikale ist erst ab **Zeile 3** zu prüfen, da vorher keine 4 entstehen kann.

Horizontale

```
  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
5 |   |   |   |   |   |   |   |
4 |   |   |   |   |   |   |   |
3 |   |   |   |   |   |   |   |
2 |   |   |   |   |   |   |   |
1 | X | X | X | X |   |   |   |   z[7] z[8] z[9] z[10]
0 | O | O | X | O |   |   |   |   z[10] z[11] z[12] z[13]
```

Hinweis:
die Zahl der Eingabe (z.B. 3) entspricht der Nummer der Spalte, die Zeile (links->rechts) z[7],z[8],z[9],z[10], z[11], z[12]

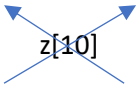
z.B. der letzte Stein **X** wurde gesetzt – **z[10]** in der zweiten Zeile. Zu überprüfen sind drei Steine ob alle drei X sind links **und** rechts. REGEL: Die Horizontale kann links oder rechts mit 4 gebildet werden, falls Spalte=3, ansonsten Spalte <3: nur nach rechts prüfen bzw. Spalte >3

Diagonale

```
  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
5 |   |   |   |   |   |   |   |
4 |   |   |   |   |   |   |   |
3 |   | X |   |   |   |   |   |   z[21] z[22]
2 |   | O | X |   |   |   |   |   z[14] z[15] z[16]
1 | X | O | X | X | O |   |   |   z[07] z[08] z[09] z[10]
0 | O | O | X | O | X |   |   |   z[00] z[01] z[02] z[03] z[04]
```

Hinweis:
die Zahl der Eingabe (z.B. 3)
Es gibt zwei mögliche Diagonale nach links oder nach rechts

Dies ist die Schwierigste Prüfung von allen.
z.B. der letzte Stein **X** wurde gesetzt – **z[10]** in der zweiten Zeile. Zu überprüfen sind drei Steine in der Diagonalen (nach links oder nach rechts), ob alle drei X vorhanden sind. Hier ist auch die darunter liegende Zeile zu beachten. Zuerst prüft man: darüber oder darunter ein X ist 3er X
ob diagonal– z[04] **z[10]** z[16], dann kann auch z[22] geprüft werden, bzw. **z[10]** z[16] z[22] und z[28] oder
ob diagonal- z[02] **z[10]** z[18] , dann kann auch z[26] geprüft werden, bzw. **z[10]** z[18] z[26] und z[34]
es gilt immer +Zeile =+7, nach links -1 bzw. nach rechts +1 um die Diagonale zu erfassen.



Eine harte Nuss, die Diagonalen!
Erstellen Sie hierzu die Funktion: spieler_gewinnt()

e.) Spielerkontrolle

Wir kontrollieren, ob der Spieler eine korrekte Eingabe gemacht hat. Der Spieler darf nur eine ganze Zahl zwischen 0 und 6 eingeben. Ebenso darf die Eingabe die Zahl nicht 41 überschreiten!

Erstellen Sie hierzu die Funktion: spieler_eingabekontrolle()

Ihr Programm… **afg6_teil3.py**

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

def spieler_gewinnt():
        d.)

def spieler_eingabekontrolle():
        e.)

def spieler_wechsel():
        global aktive_spieler
         if aktive_spieler=="X":
            aktive_spieler="O"
        else:
           aktive_spieler="X"

def spieler_eingabe():
        global aktive_spieler
        eingabe=int(input("Bitte die Spalte angeben: "))
        → e.) spieler_eingabekontrolle()
        while z[eingabe]!=" ":
                eingabe=eingabe+7
                if eingabe>41:
                        break
                if z[eingabe]==" ":
                        z[eingabe]=aktive_spieler
                        break
        else:
                z[eingabe]=aktive_spieler

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()
```