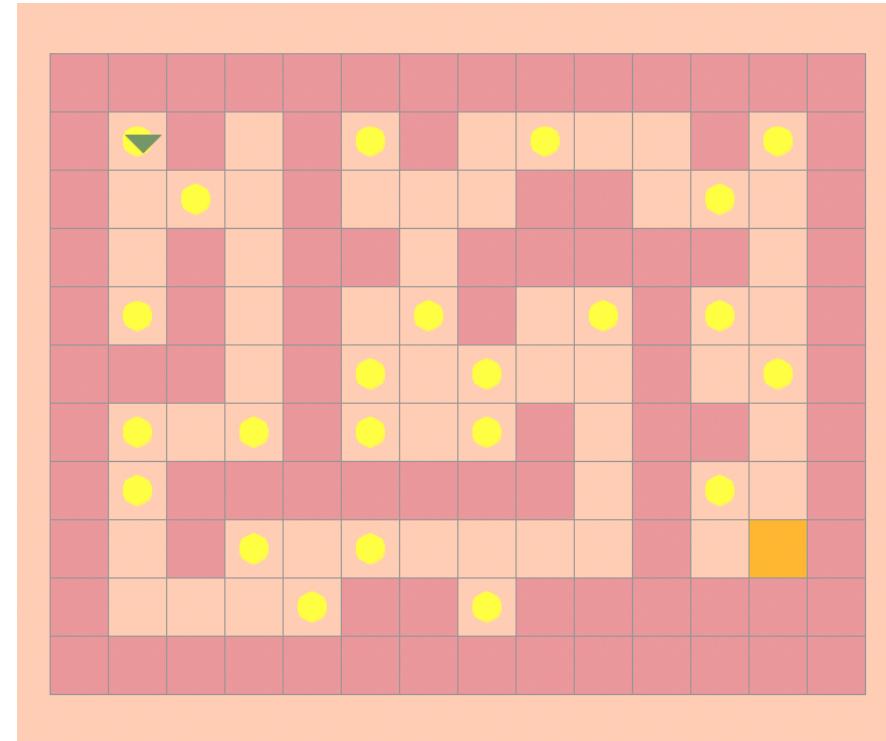


Starcode: Python

Programmierkurs für Schülerinnen

Von: Helena Schulz und Furat Hamdan



Über Uns

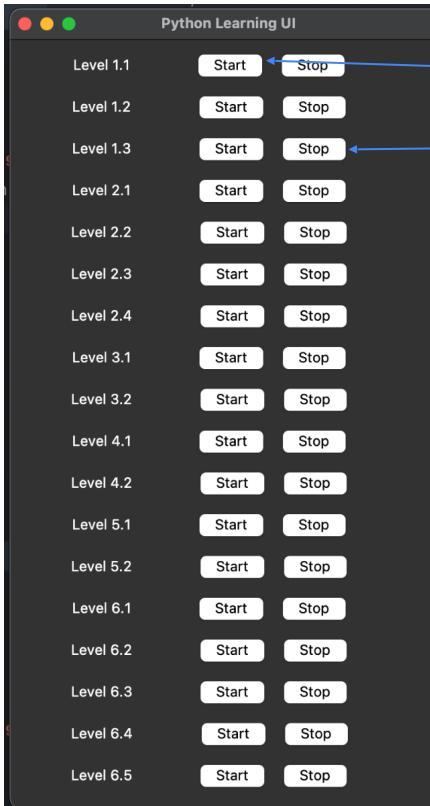


Helena Schulz
M.Sc. Informatik
23 Jahre



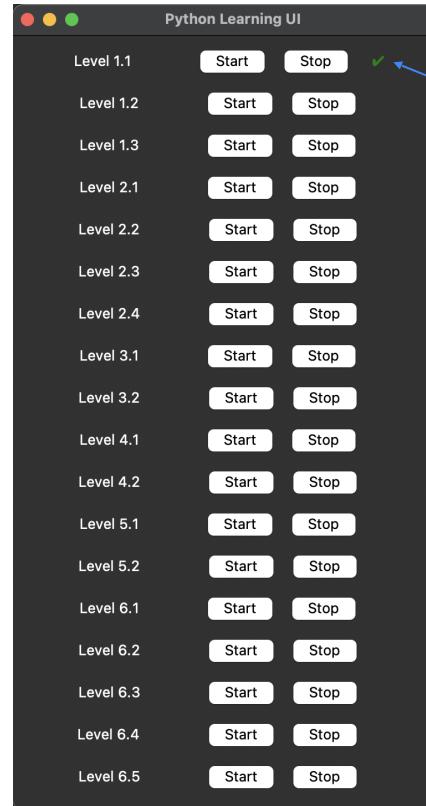
Furat Hamdan
M.Sc. Wirtschaftsinformatik
23 Jahre

Aufbau des Kurses



Start von Levels

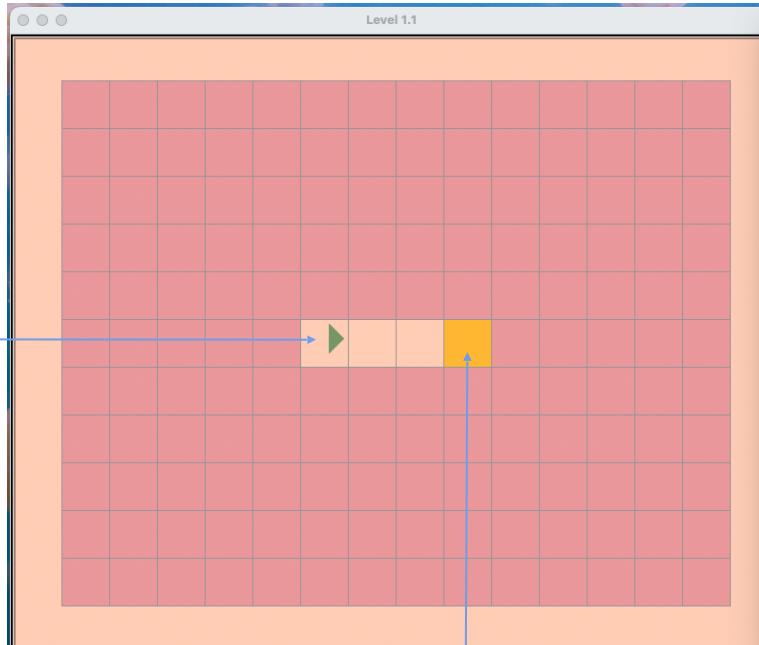
Stopp von Levels



Erfolgreich
abgeschlossene
Levels

Aufbau des Kurses

Spieler



Ziel

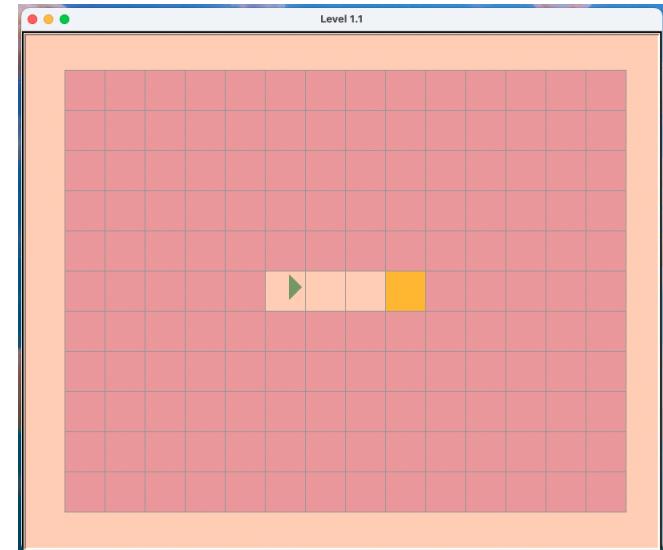
Code Ausführung
(Code wird nach jeder Ausführung zwischengespeichert)

Code
Eingabe

Aufbau des Kurses

Ziel muss erreicht werden, und zwar:

- Mit einer Ausführung des Codes
- Ohne gegen die Wand zu laufen
- Teilweise mit einer bestimmten Bedingung, je nach Level



Kontrollbefehle

`move()`

Bewegt den Spieler in Pfeilrichtung um einen Schritt nach vorne

`rotate_left()`

Dreht den Spieler um 90 Grad nach links

`rotate_right()`

Dreht den Spieler um 90 Grad nach rechts

`goal_reached()`

prüft ob das Ziel erreicht wurde und gibt **True** oder **False** zurück

`can_move_forward()`

Prüft ob der Spieler sich in Pfeilrichtung bewegen kann und gibt

True oder **False** zurück

`is_on_coin()`

Prüft ob der Spieler sich auf einer Münze befindet und gibt **True** oder **False** zurück

`collect_coin()`

Sammelt Münze wenn sich der Spieler auf ihr befindet

`pick_up_coin()`

Sammelt Münze wenn sich der Spieler auf ihr befindet und gibt die ID der Münze zurück

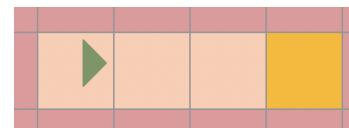
`get_position()`

Gibt die aktuelle Position des Spielers in der Form [x, y] zurück

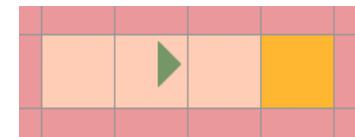
Level 1 - Kontrollbefehle

move()

Bewegt den Spieler in Pfeilrichtung um einen Schritt nach vorne

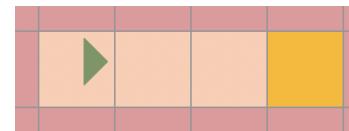


->

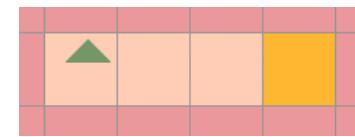


rotate_left()

Dreht den Spieler um 90 Grad nach links

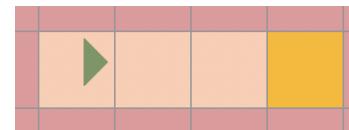


->

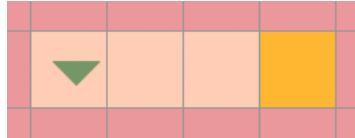


rotate_right()

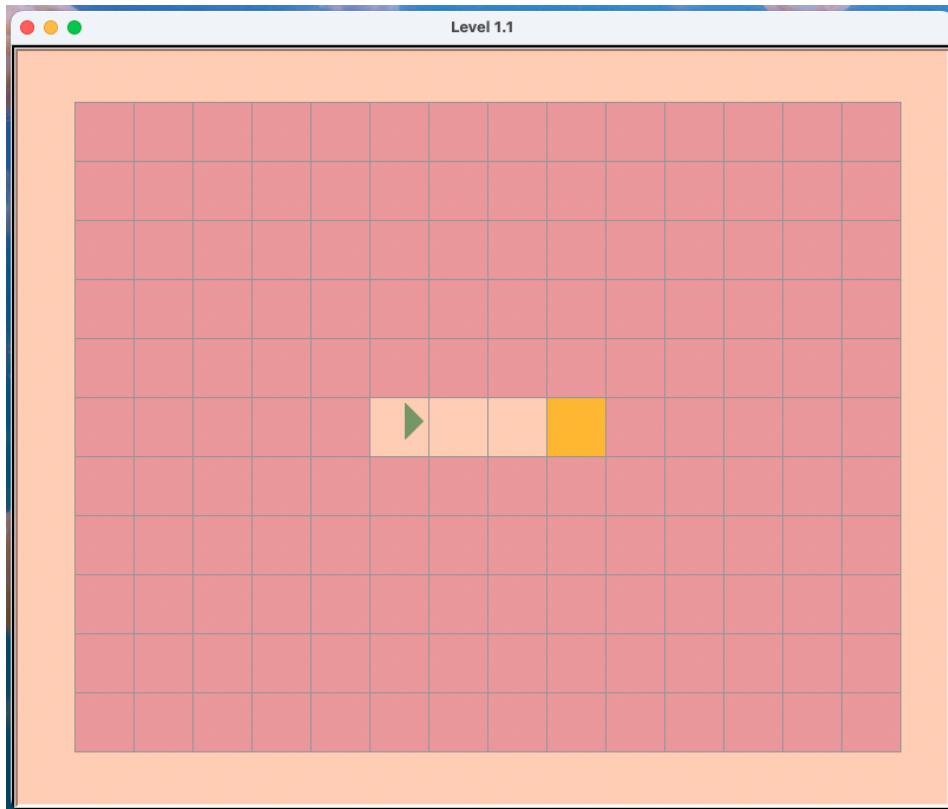
Dreht den Spieler um 90 Grad nach rechts



->



Level 1.1

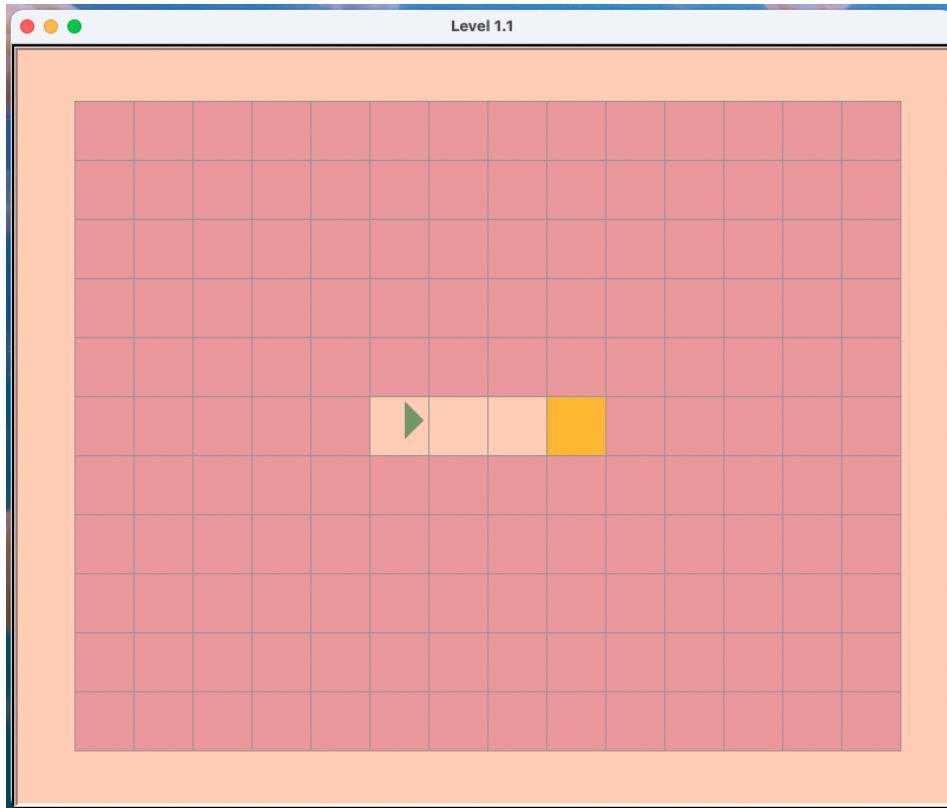


Notwendige Befehle:

move ()

Bewegt den Spieler in Pfeilrichtung um einen Schritt nach vorne

Level 1.1 - Lösung

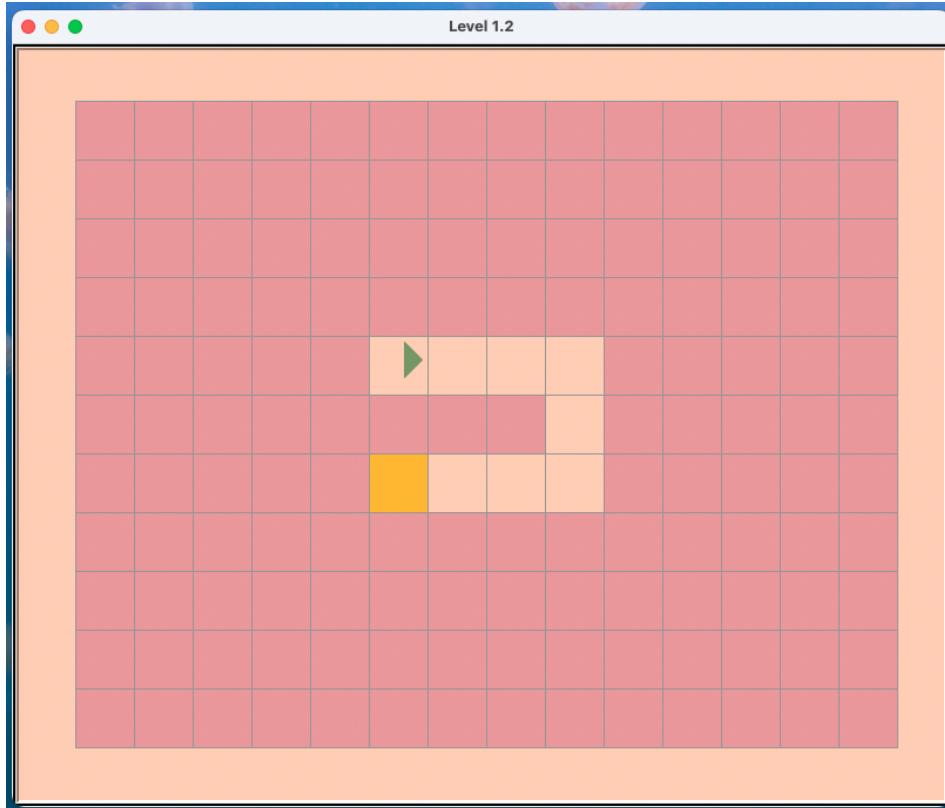


`move ()`

`move ()`

`move ()`

Level 1.2



Notwendige Befehle:

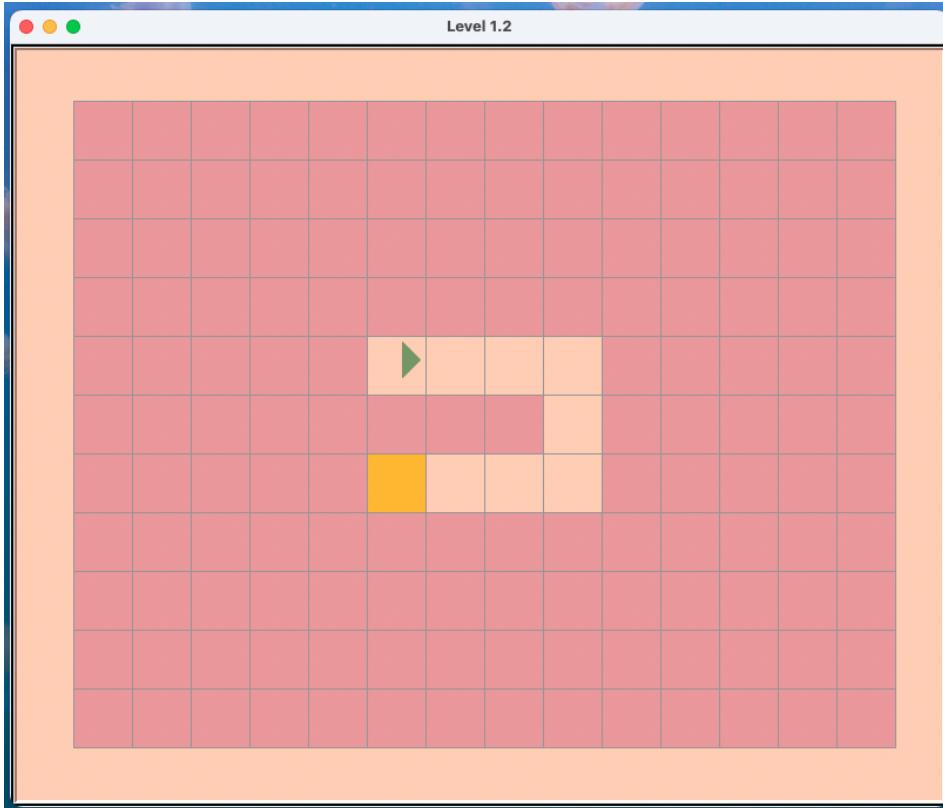
move()

Bewegt den Spieler in Pfeilrichtung um einen Schritt nach vorne

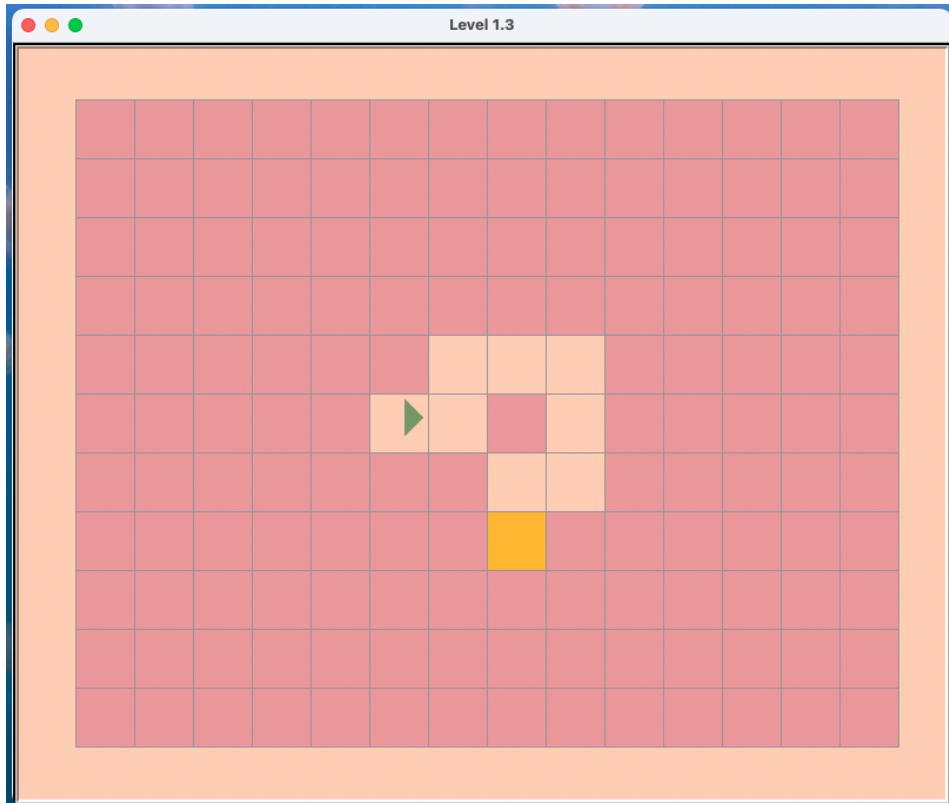
rotate_right()

Dreht den Spieler um 90 Grad nach rechts

Level 1.2 - Lösung



Level 1.3



Notwendige Befehle:

`move()`

Bewegt den Spieler in Pfeilrichtung um einen Schritt nach vorne

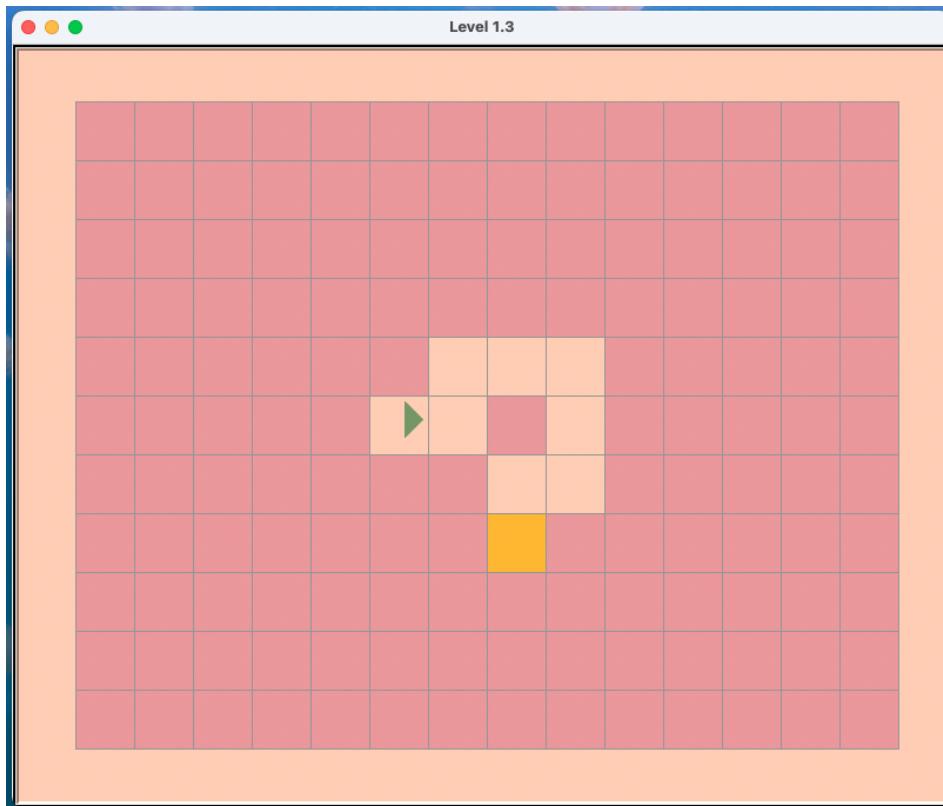
`rotate_right()`

Dreht den Spieler um 90 Grad nach rechts

`rotate_left()`

Dreht den Spieler um 90 Grad nach links

Level 1.3 - Lösung

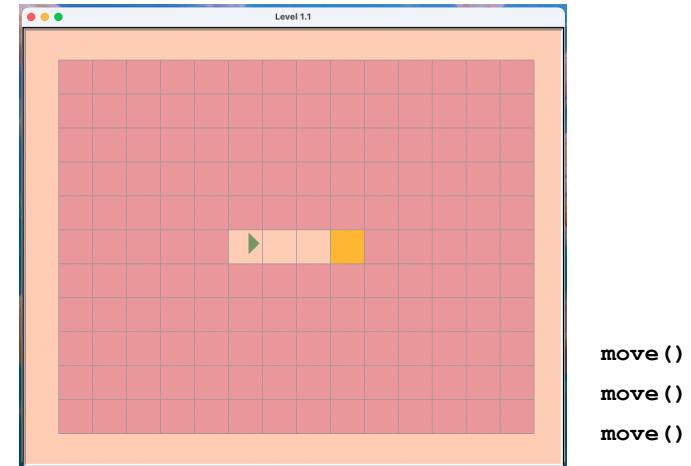


```
move ()  
rotate_left()  
move ()  
rotate_right()  
move ()  
move ()  
rotate_right()  
move ()  
move ()  
rotate_right()  
move ()  
rotate_left()  
move ()
```

Level 2 - For-Schleifen

Was, wenn die Level größer werden und bestimmte Bewegungen oft wiederholt werden müssen?

- Befehle einzeln aufzuschreiben wäre nicht sinnvoll
- Mit For-Schleifen können wir:
 - Ein Bewegungsmuster festlegen
 - Festlegen, wie oft das Muster wiederholt werden soll



Level 2 - For-Schleifen

```
for var in sequenz:  
    Anweisung
```

For-Schleifen führen für jedes Objekt in der Sequenz den Code in der Schleife einmal aus

```
for i in range(5):  
    print("Hallo")
```

Hallo
Hallo
Hallo
Hallo
Hallo

Level 2 - For-Schleifen

Welche Code-Ausführung ist die richtige?

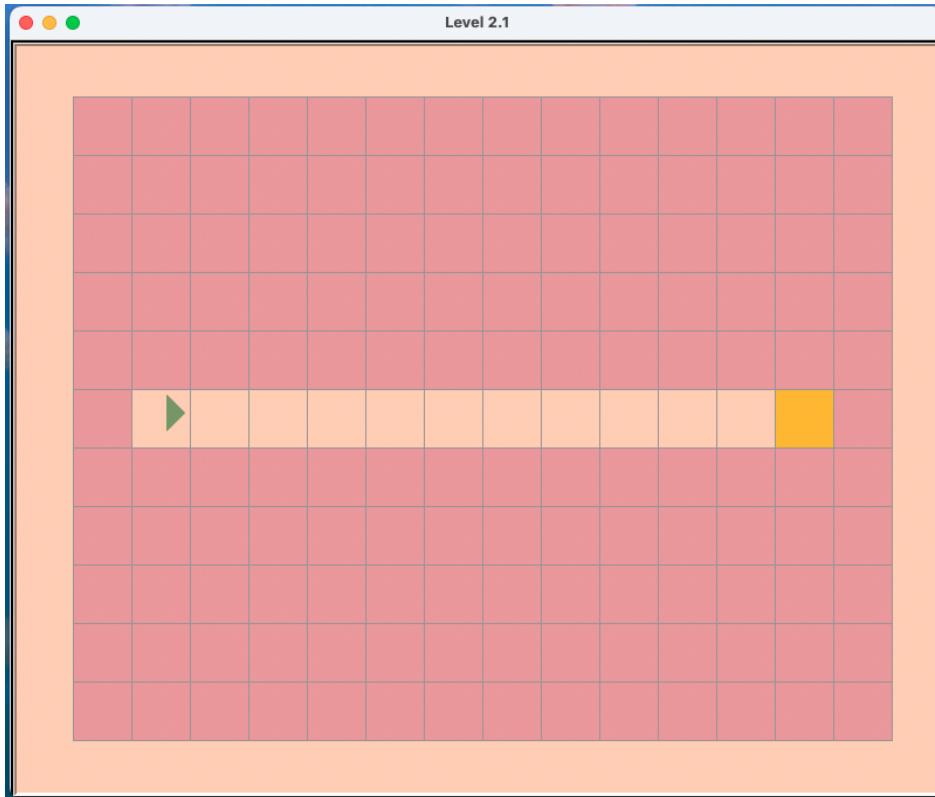
```
for i in range(3):  
    print("Hallo")  
    print("Welt")
```

Hallo
Welt
Hallo
Welt
Hallo
Welt

```
for i in range(3):  
    print("Hallo")  
    print("Welt")
```

Hallo
Hallo
Hallo
Welt
Welt
Welt

Level 2.1



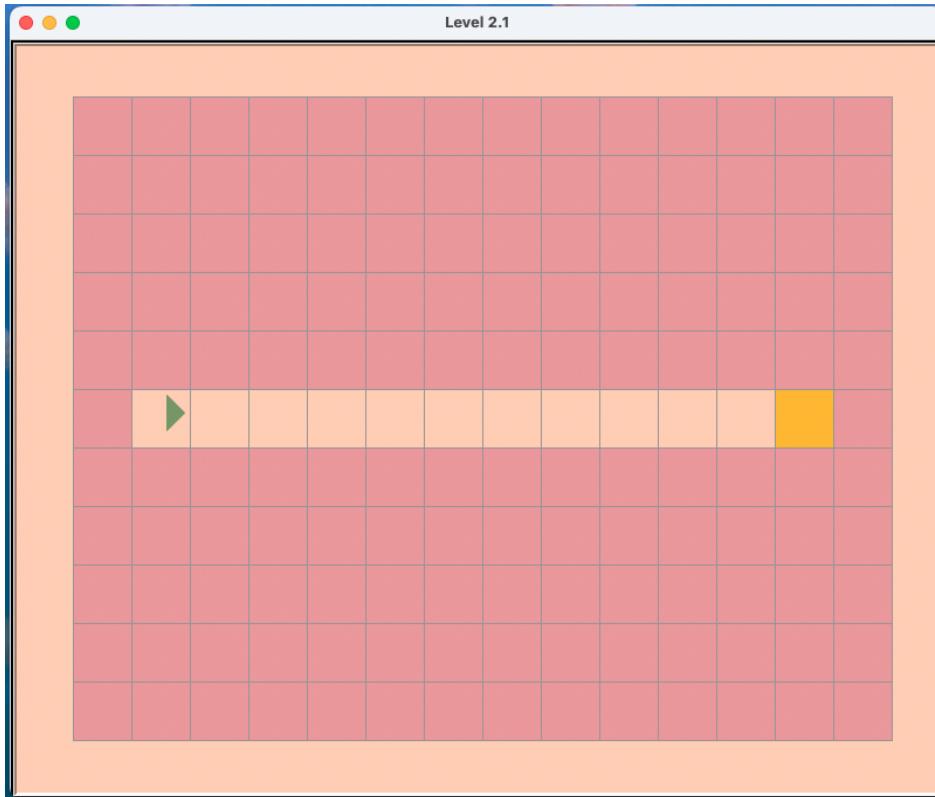
Notwendige Befehle:

move()

Bewegt den Spieler in Pfeilrichtung um einen Schritt nach vorne

for var in range(x):
Anweisung

Level 2.1 - Lösung



```
for i in range(11):  
    move()
```

Level 2.2



Notwendige Befehle:

move()

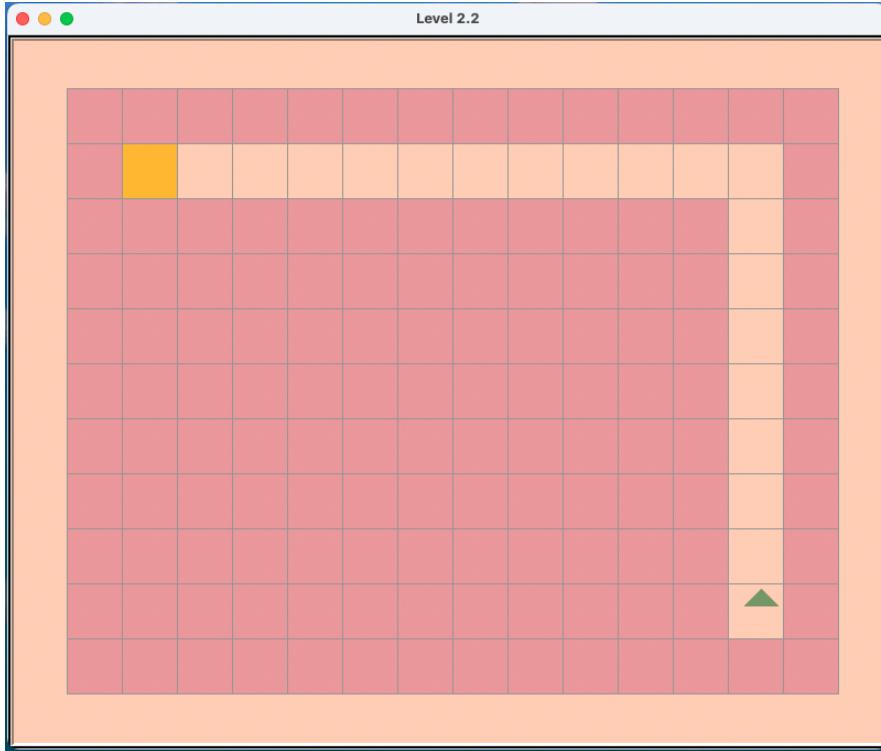
Bewegt den Spieler in Pfeilrichtung um einen Schritt nach vorne

rotate_left()

Dreht den Spieler um 90 Grad nach links

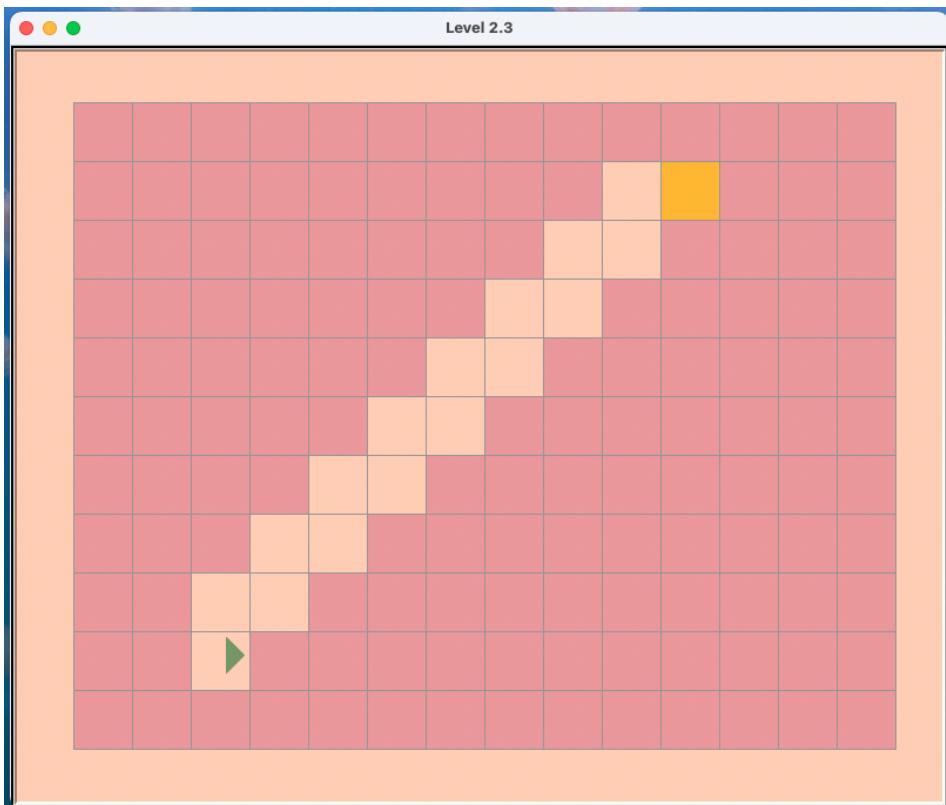
for var in range(x):
Anweisung

Level 2.2 - Lösung



```
for i in range(8):  
    move()  
rotate_left()  
for i in range(11):  
    move()
```

Level 2.3



Notwendige Befehle:

move()

Bewegt den Spieler in Pfeilrichtung um einen Schritt nach vorne

rotate_left()

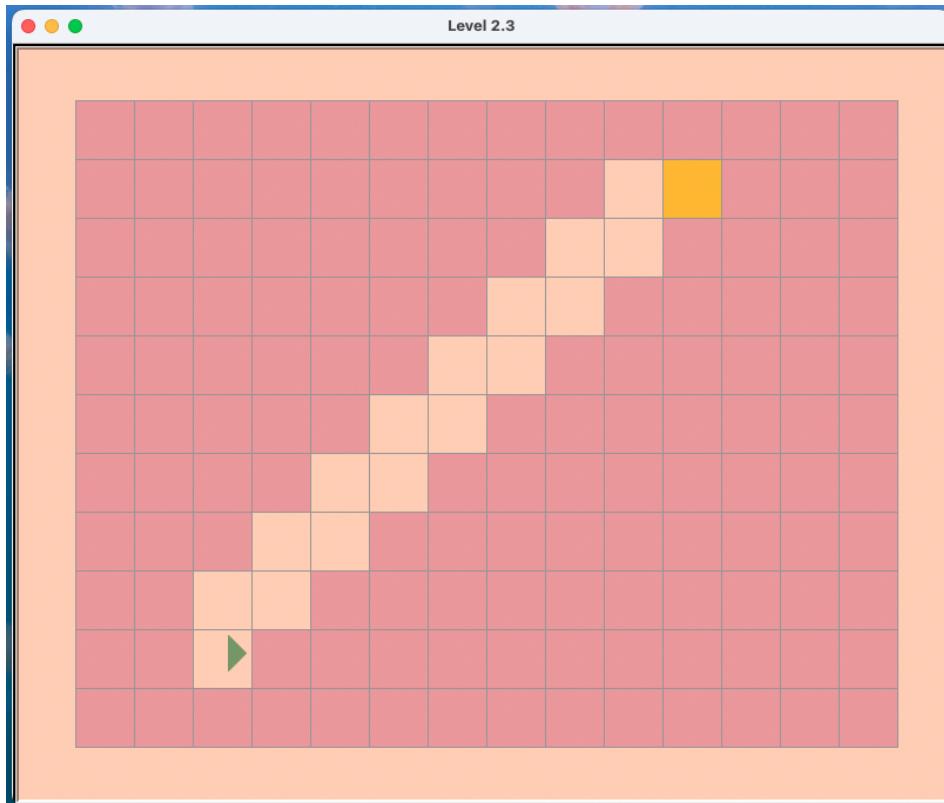
Dreht den Spieler um 90 Grad nach links

rotate_right()

Dreht den Spieler um 90 Grad nach rechts

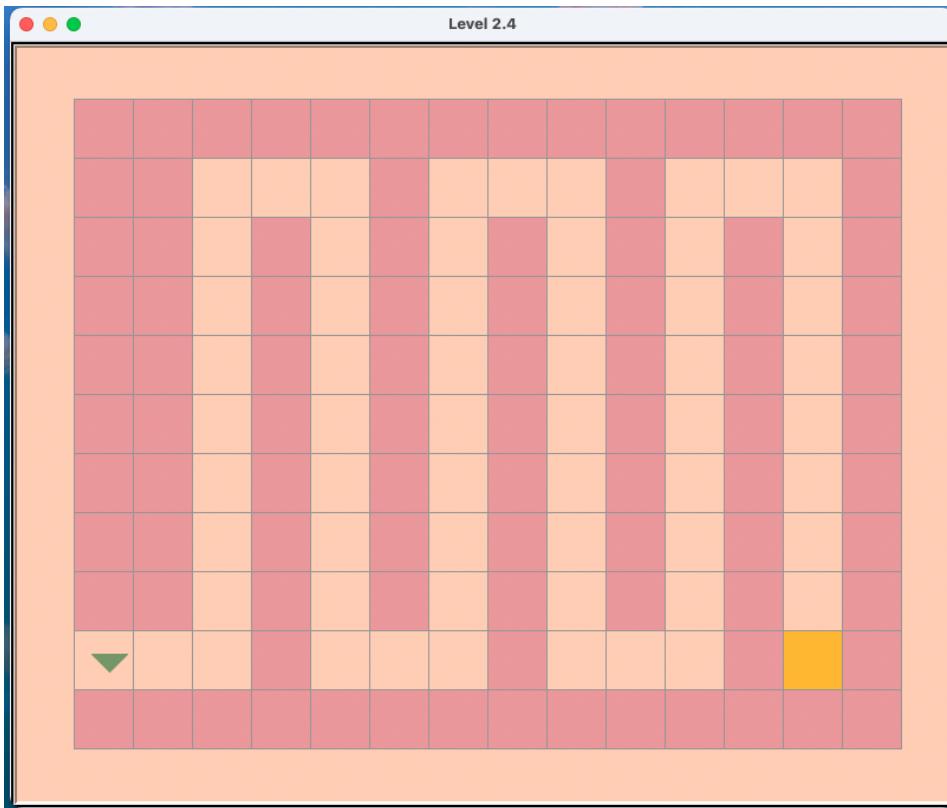
for var in range(x):
Anweisung

Level 2.3 - Lösung



```
for i in range(8):
    rotate_left()
    move()
    rotate_right()
    move()
```

Level 2.4



Notwendige Befehle:

move()

Bewegt den Spieler in Pfeilrichtung um einen Schritt nach vorne

rotate_right()

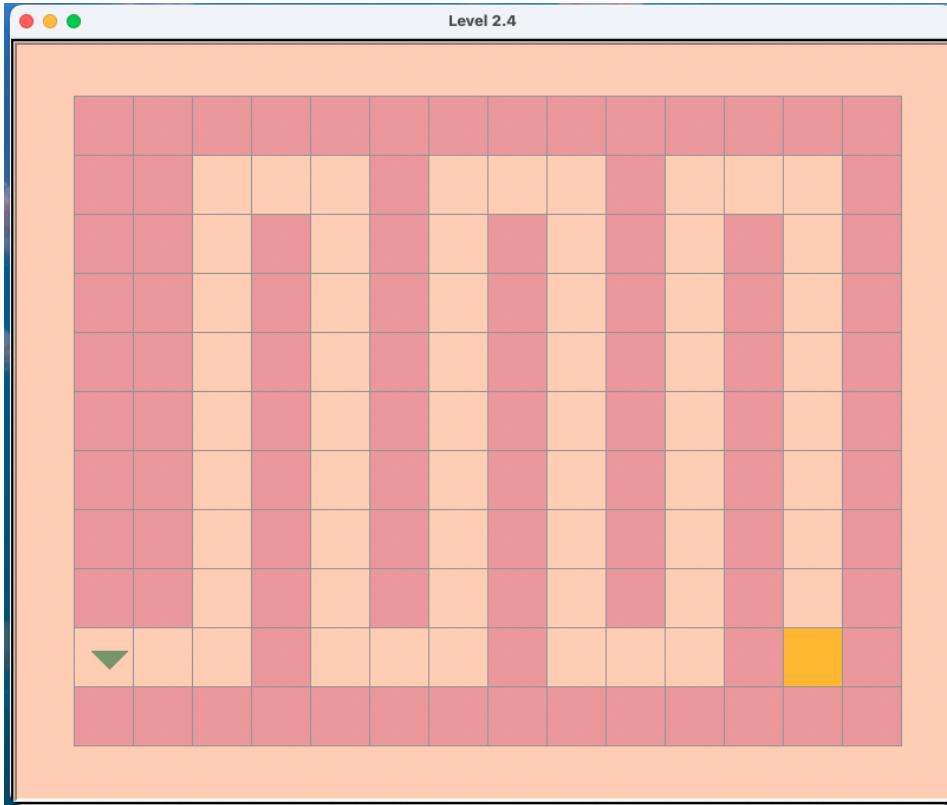
Dreht den Spieler um 90 Grad nach rechts

rotate_left()

Dreht den Spieler um 90 Grad nach links

for var in range(x):
Anweisung

Level 2.4 - Lösung



```
for i in range(3):
    rotate_left()
    move()
    move()
    rotate_left()
for i in range(8):
    move()
rotate_right()
move()
move()
rotate_right()
for i in range(8):
    move()
```

Level 3 - While-Schleifen

Es gibt noch eine andere Art von Schleifen

- Bei For-Schleifen müssen die Wiederholungen vorher festgelegt werden
- While-Schleifen knüpfen die Code-Ausführung an eine Bedingung

```
while Bedingung :  
    Anweisung
```

Eine Bedingung wertet immer zu **True** oder **False** aus

Eine While-Schleife läuft solange die Bedingung True ist

Level 3 - While-Schleifen

while Bedingung :
Anweisung

Unser erster Bedingungs-Befehl: **goal_reached()**
prüft ob das Ziel erreicht wurde und gibt **True** oder **False** zurück

Movement-Befehle:
(move(), etc.)
bewegen den Spieler

VS.

Bedingungs-Befehle:
(goal_reached(), etc.)
geben True / False zurück

Level 3 - While-Schleifen

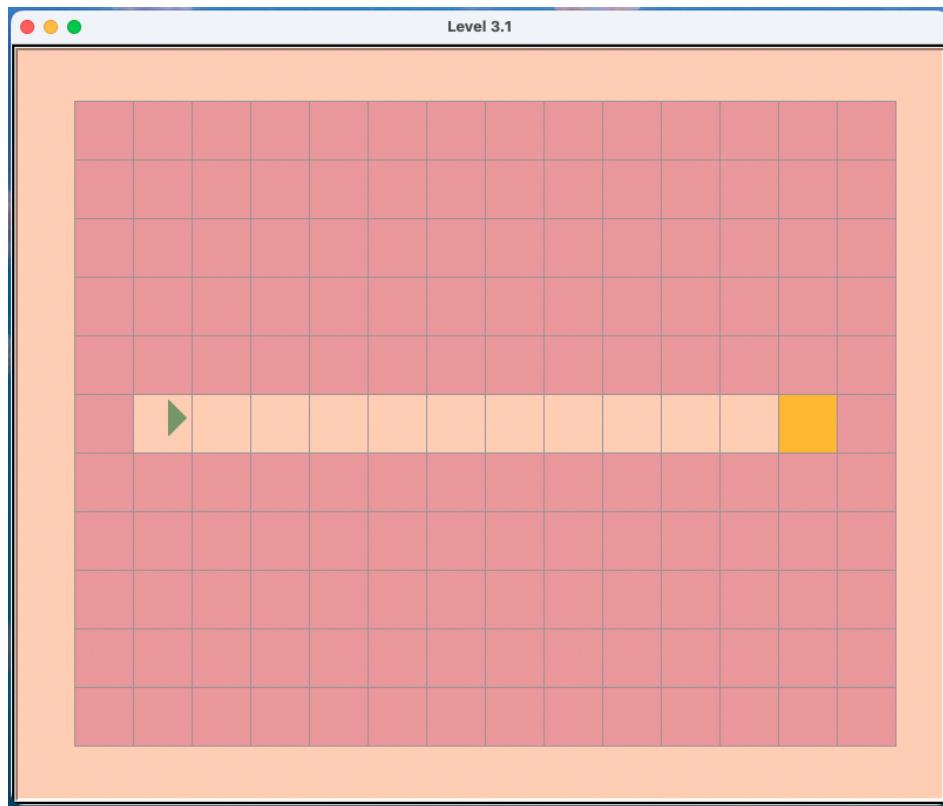
```
while not Bedingung :  
    Anweisung
```

not dreht den Wert einer Bedingung um:

- not True = False
- not False = True

Was ist not not False ?

Level 3.1



Notwendige Befehle:

move()

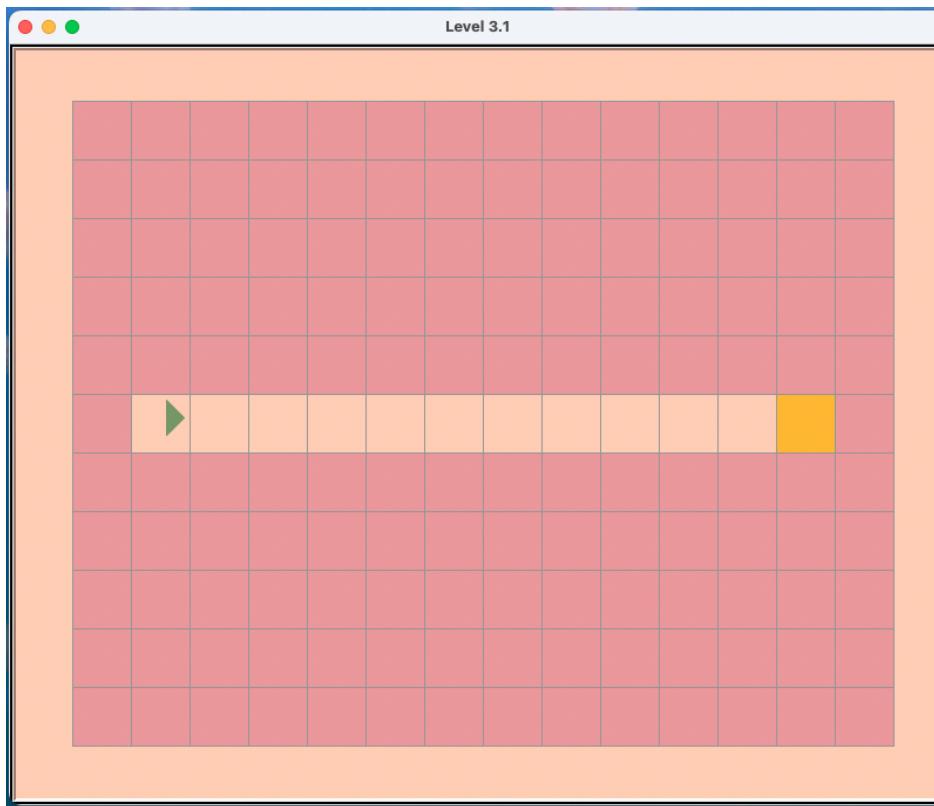
Bewegt den Spieler in Pfeilrichtung um einen Schritt nach vorne

goal_reached()

prüft ob das Ziel erreicht wurde und gibt **True** oder **False** zurück

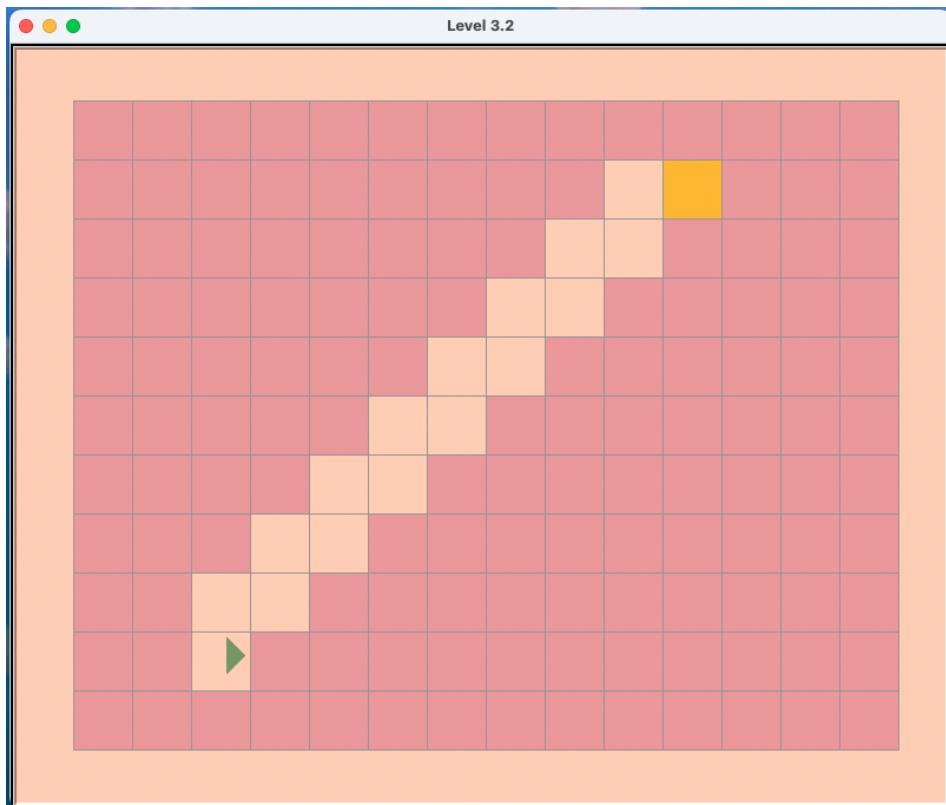
while Bedingung:
Anweisung

Level 3.1 - Lösung



```
while not goal_reached():
    move()
```

Level 3.2



Notwendige Befehle:

move()

Bewegt den Spieler in Pfeilrichtung um einen Schritt nach vorne

rotate_left()

Dreht den Spieler um 90 Grad nach links

rotate_right()

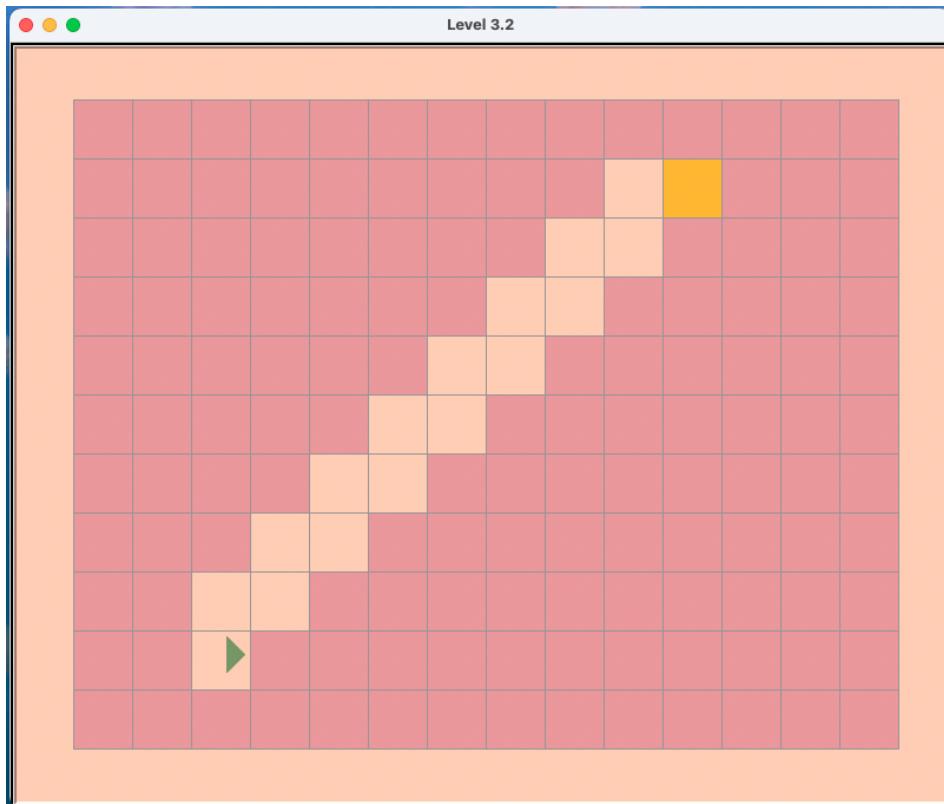
Dreht den Spieler um 90 Grad nach rechts

goal_reached()

prüft ob das Ziel erreicht wurde und gibt **True** oder **False** zurück

while Bedingung:
Anweisung

Level 3.2 - Lösung



```
while not goal_reached():

    rotate_left()

    move()

    rotate_right()

    move()
```

Level 4 - If-Anweisungen

Operatoren

if Bedingung:
Anweisung

Anweisung wird ausgeführt nur wenn die Bedingung zu True ausgewertet wird

```
if 1 < 10:  
    print("1 ist kleiner als 10")
```

==	gleich
!=	ungleich
<	kleiner
>	größer
<=	kleiner oder gleich
>=	größer oder gleich

Level 4 - Else

```
if Bedingung:  
    Anweisung
```

```
else:  
    Anweisung
```

Else Block wird ausgeführt wenn die Bedingung zu False ausgewertet wird

```
if 1 < 10:  
    print("1 ist kleiner als 10")  
else:  
    print("1 ist größer als 10")
```

Operatoren

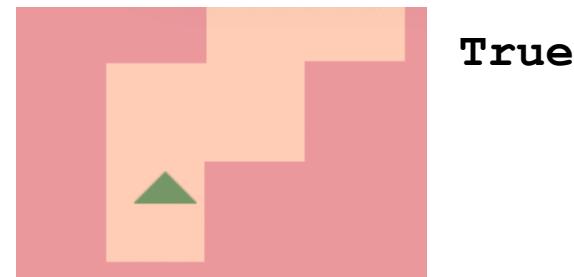
==	gleich
!=	ungleich
<	kleiner
>	größer
<=	kleiner oder gleich
>=	größer oder gleich

Level 4 - If-Else

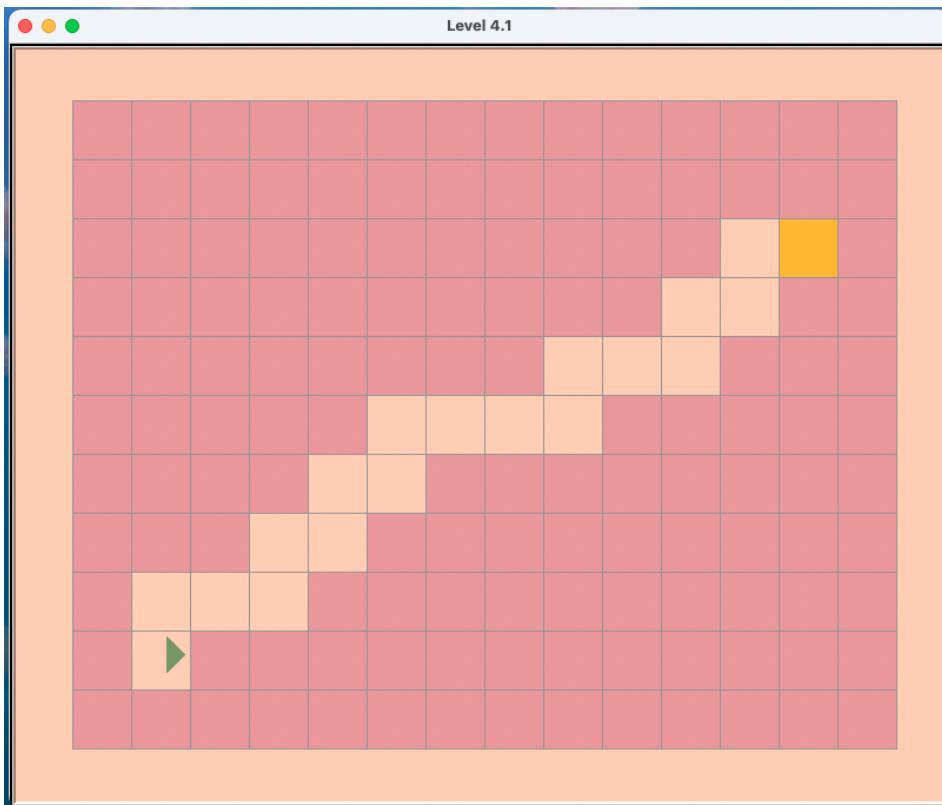
```
if Bedingung:  
    Anweisung  
else:  
    Anweisung
```

Ein weiterer Bedingungs-Befehl: **can_move_forward()**

Prüft ob der Spieler sich in Pfeilrichtung bewegen kann und gibt **True** oder **False** zurück



Level 4.1



Notwendige Befehle:

move()

Bewegt den Spieler in Pfeilrichtung um einen Schritt nach vorne

can_move_forward()

Prüft ob der Spieler sich in Pfeilrichtung bewegen kann und gibt **True** oder **False** zurück

rotate_left()

Dreht den Spieler um 90 Grad nach links

rotate_right()

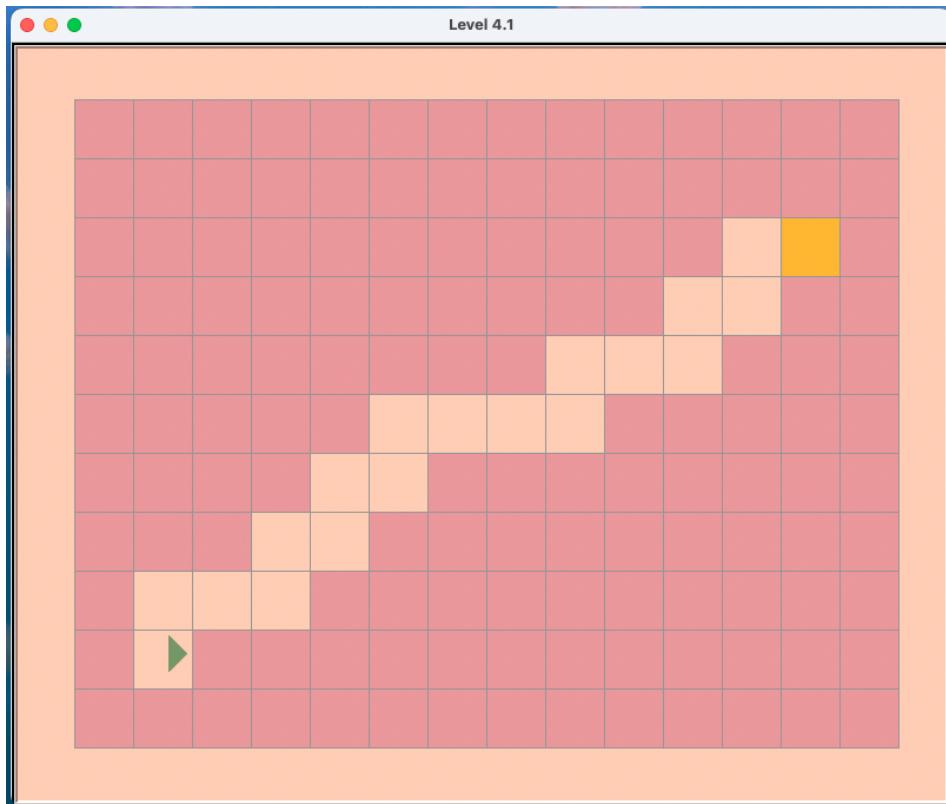
Dreht den Spieler um 90 Grad nach rechts

goal_reached()

prüft ob das Ziel erreicht wurde und gibt **True** oder **False** zurück

if Bedingung:
Anweisung
else:
Anweisung

Level 4.1 - Lösung



```
while not goal_reached():

    if can_move_forward():

        move()

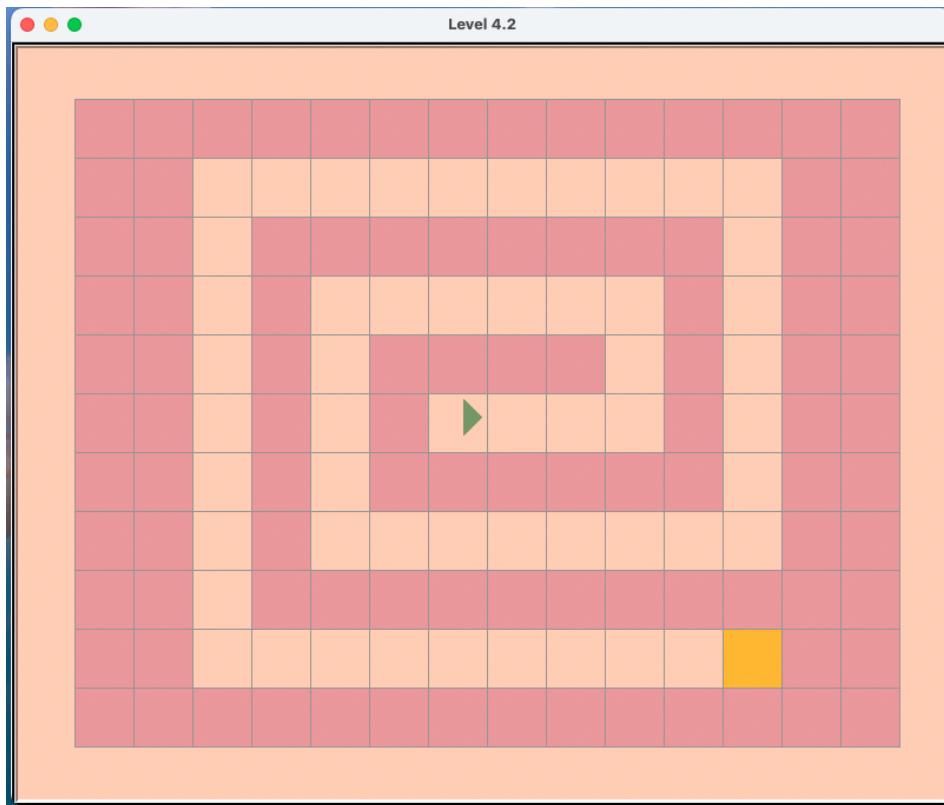
    else:

        rotate_left()

        move()

        rotate_right()
```

Level 4.2



Notwendige Befehle:

move()

Bewegt den Spieler in Pfeilrichtung um einen Schritt nach vorne

can_move_forward()

Prüft ob der Spieler sich in Pfeilrichtung bewegen kann und gibt **True** oder **False** zurück

rotate_left()

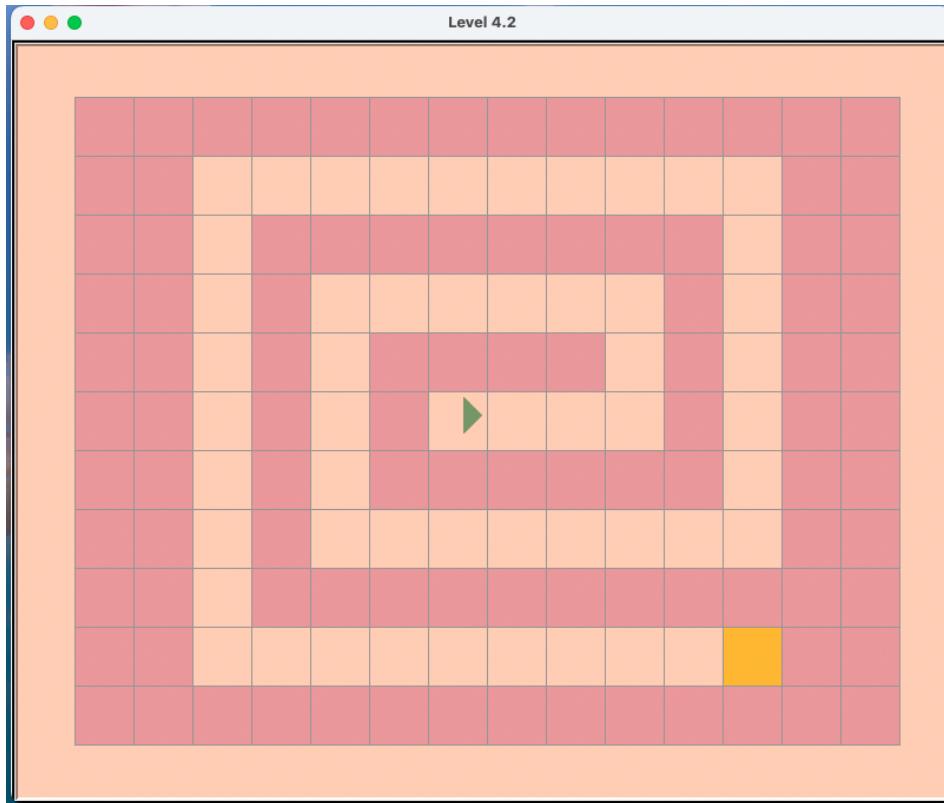
Dreht den Spieler um 90 Grad nach links

goal_reached()

prüft ob das Ziel erreicht wurde und gibt **True** oder **False** zurück

if Bedingung:
Anweisung
else:
Anweisung

Level 4.2 - Lösung



```
while not goal_reached():
    if can_move_forward():
        move()
    else:
        rotate_left()
```

Level 5 - Variablen

Behälter (Container) zur Aufbewahrung von bestimmten Werten: Zeichenketten, Wahrheitswerte und Zahlen

```
var = "Ich bin eine Variable"
print(var) // "Ich bin eine Variable"

n1 = 10

n2 = 15
n2 = 5
n2 = n2 + 5
print(n2) // 10

n2 += 5 // Dasselbe wie n2 = n2 +5

zaehler = 0
while not goal_reached():
    move()
    zaehler += 1
```

Level 4/5 - If-Else

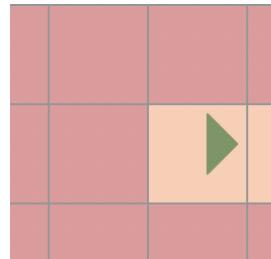
```
if Bedingung:  
    Anweisung  
else:  
    Anweisung
```

collect_coin()

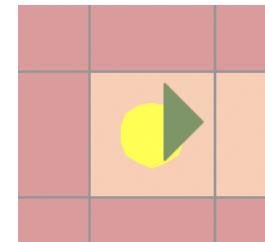
Sammelt Münze wenn sich der Spieler auf ihr befindet

Ein weiterer Bedingungs-Befehl: **is_on_coin()**

Prüft ob der Spieler sich auf einer Münze befindet und gibt **True** oder **False** zurück



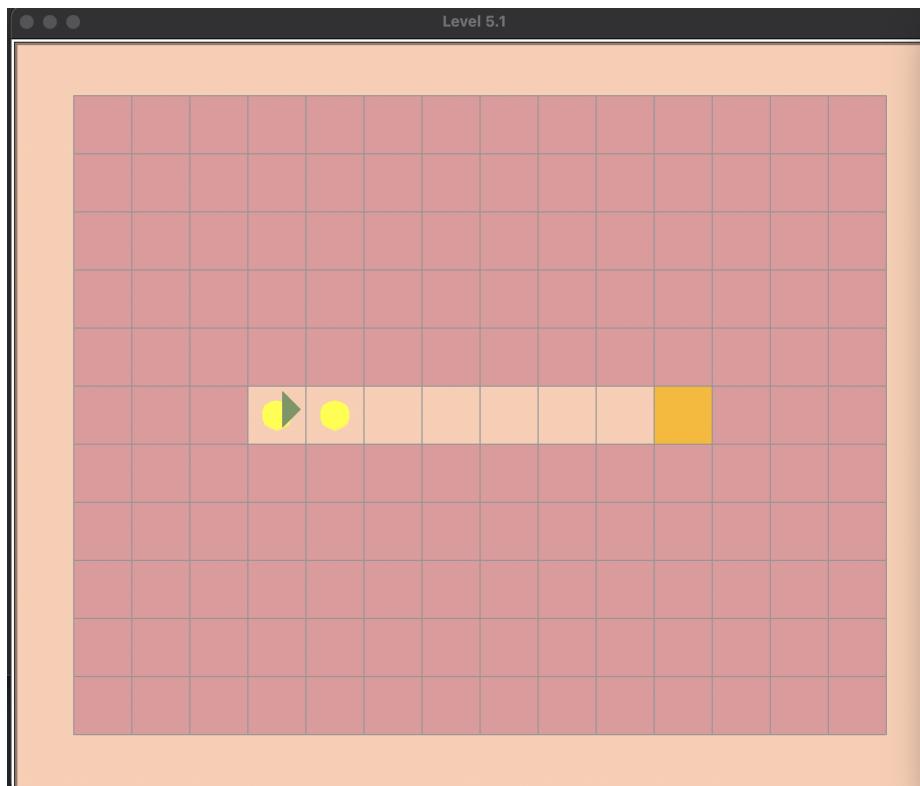
False



True

Level 5.1

Sammle 2 Münzen ein! Das Zielfeld muss nicht erreicht werden.



Notwendige Befehle:

`move()`

Bewegt den Spieler in Pfeilrichtung um einen Schritt nach vorne

`is_on_coin()`

Prüft ob der Spieler auf einer Münze steht und gibt True oder False zurück

`collect_coin()`

Sammelt die Münze auf

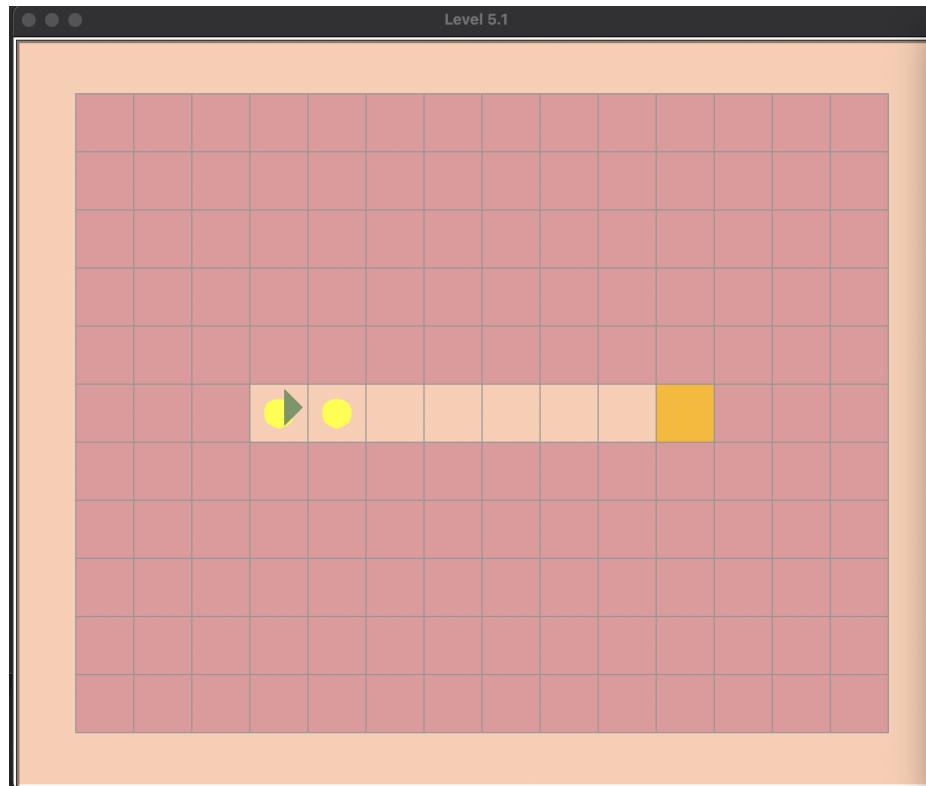
`zaehler = 0`

```
while not goal_reached():
    move()
```

```
    zahler += 1
```

<code>==</code>	gleich
<code>!=</code>	ungleich
<code><</code>	kleiner
<code>></code>	größer
<code><=</code>	kleiner oder gleich
<code>>=</code>	größer oder gleich

Level 5.1



Lösung

```
coins = 0
while coins != 2:
    if(is_on_coin()):
        collect_coin()
        coins += 1
    move()
```

Level 6 - Listen

Listen können mehrere Werte auf einmal speichern

```
list = [1, 2, 3, 4, 5]  
another_list = [8, 1, 9]
```

Einen Eintrag in einer Liste nennt man **Element**

Level 6 - Listen

Jedes Element hat einen **Index**

- Der Index dient der Durchnummerierung der Listenelemente
- Vorsicht! Die Indizes starten immer bei **0**!

```
list = [0, 1, 2, 3, 4, 5]
```

- Durch den Index kann man auf ein bestimmtes Element der Liste zugreifen

```
elem = list[2]
```

Welches Element haben wir in elem gespeichert?

Level 6 - Listen

Listen können verändert werden

- Wir können neue Elemente hinten an die Liste anhängen

```
list = [1, 2, 3, 4, 5]
list.append(1)
list.append(8)
```

Wie sieht list jetzt aus?

Level 6 - Listen

Es gibt mehrere Operationen, die Listen verändern oder Informationen zu Listen ausgeben können

Länge (wie viele Elemente sich in der Liste befinden):

```
length = len(list)
```

Was ist der Wert von length?

```
list = [5, 2, 7]
```

Aufsteigend sortieren:

```
new_list = list.sort()
```

Wie sieht new_list aus?

Level 6 - Listen

Neuer Befehl: **pick_up_coin()**

Sammelt Münze wenn sich der Spieler auf ihr befindet und gibt die ID der Münze zurück

- Hat einen Rückgabewert **und**
- Lässt die Münze verschwinden

```
coins = []
```

```
coins.append(25)
```

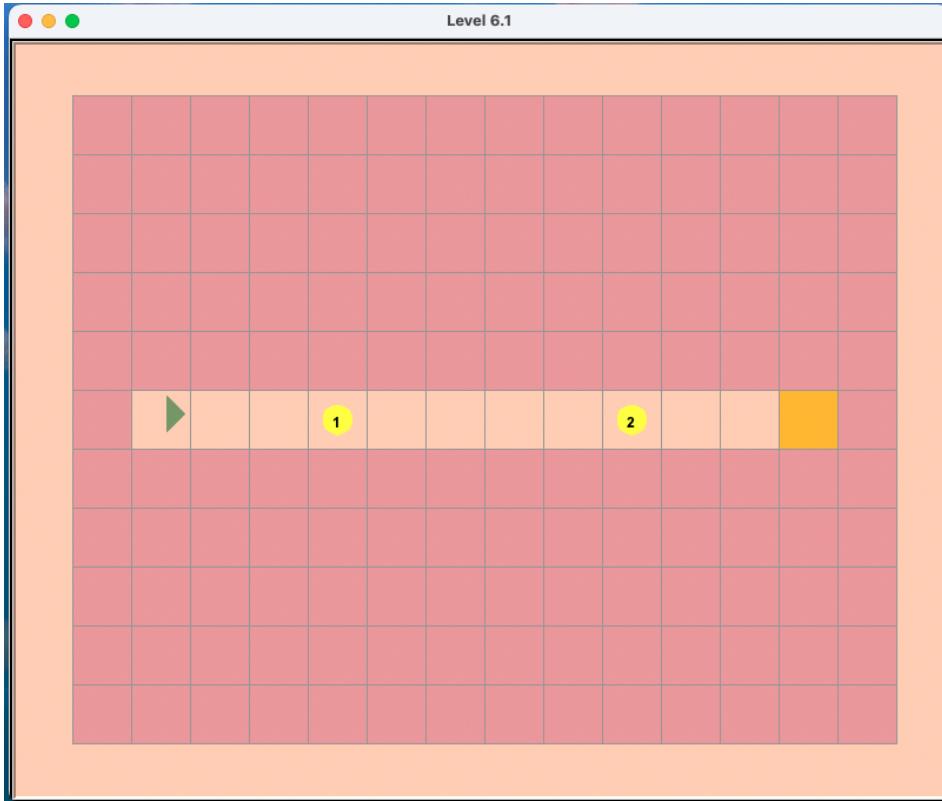
```
Coin = [25]
```

```
Coin.append(8)
```

```
Coin = [25, 8]
```

Level 6.1

Füge alle Münzen der Liste coins hinzu, bevor du das Ziel erreichst!



Listenoperationen: `list[x]` , `len(list)` ,
`list.sort()` , `list.append(a)`

Notwendige Befehle:

`move()`

Bewegt den Spieler in Pfeilrichtung um einen Schritt nach vorne

`goal_reached()`

prüft ob das Ziel erreicht wurde und gibt **True** oder **False** zurück

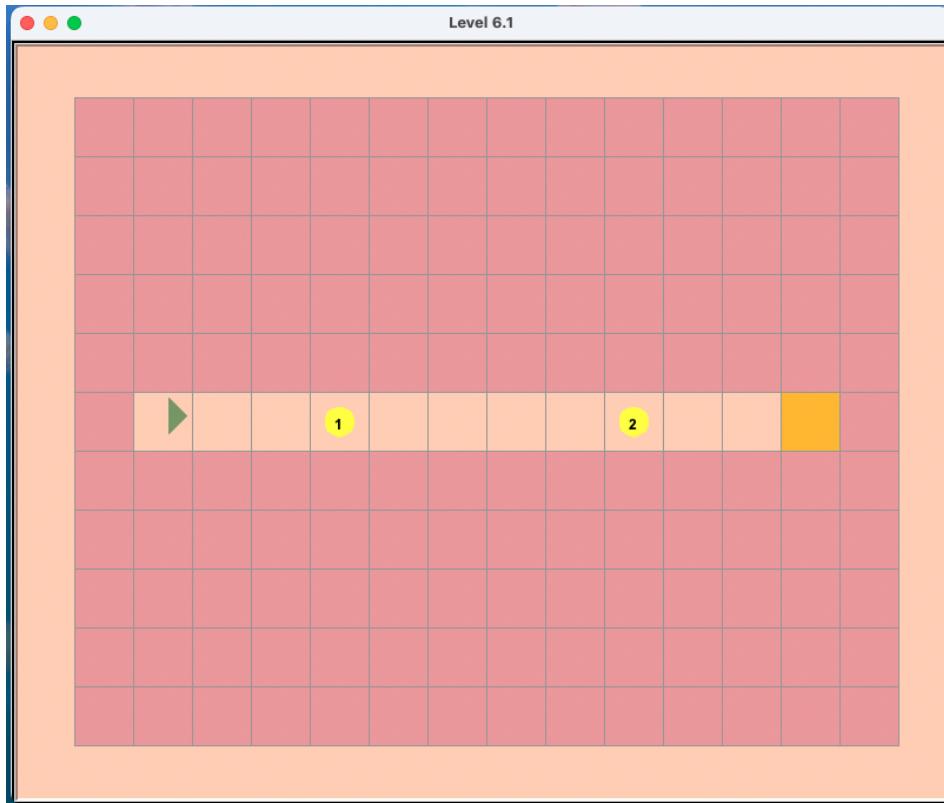
`is_on_coin()`

Prüft ob der Spieler sich auf einer Münze befindet und gibt **True** oder **False** zurück

`pick_up_coin()`

Sammelt Münze wenn sich der Spieler auf ihr befindet und gibt die ID der Münze zurück

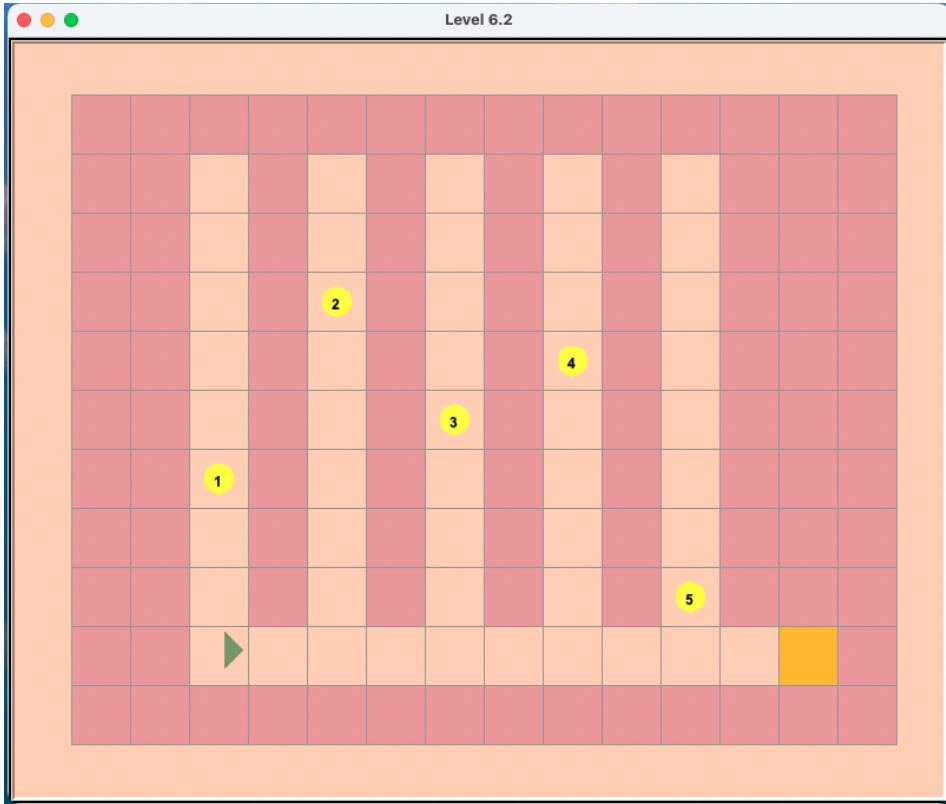
Level 6.1 - Lösung



```
while not goal_reached():
    if is_on_coin():
        coins.append(pick_up_coin())
        move()
```

Level 6.2

Füge alle Münzen der Liste coins hinzu, bevor du das Ziel erreichst!



Listenoperationen: `list[x]` , `len(list)` ,
`list.sort()` , `list.append(a)`

Notwendige Befehle:

`move()`

Bewegt den Spieler in Pfeilrichtung um einen Schritt nach vorne

`rotate_right()`

Dreht den Spieler um 90 Grad nach rechts

`rotate_left()`

Dreht den Spieler um 90 Grad nach links

`is_on_coin()`

Prüft ob der Spieler sich auf einer Münze befindet und gibt `True` oder `False` zurück

`pick_up_coin()`

Sammelt Münze wenn sich der Spieler auf ihr befindet und gibt die ID der Münze zurück

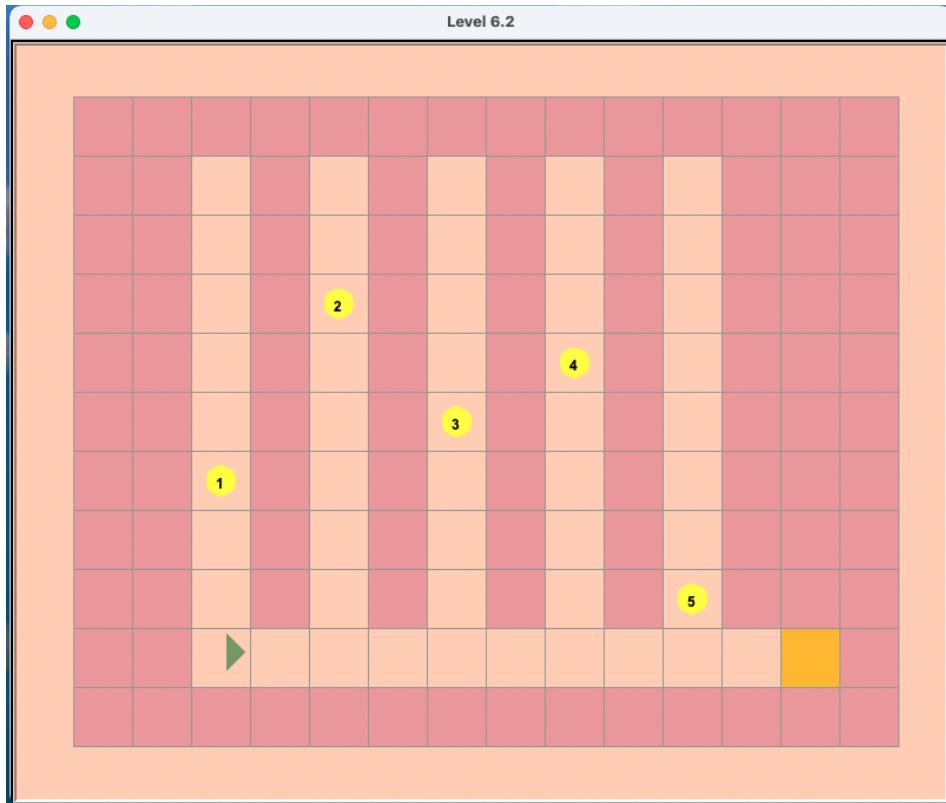
`can_move_forward()`

Prüft ob der Spieler sich in Pfeilrichtung bewegen kann und gibt `True` oder `False` zurück

`goal_reached()`

prüft ob das Ziel erreicht wurde und gibt `True` oder `False` zurück

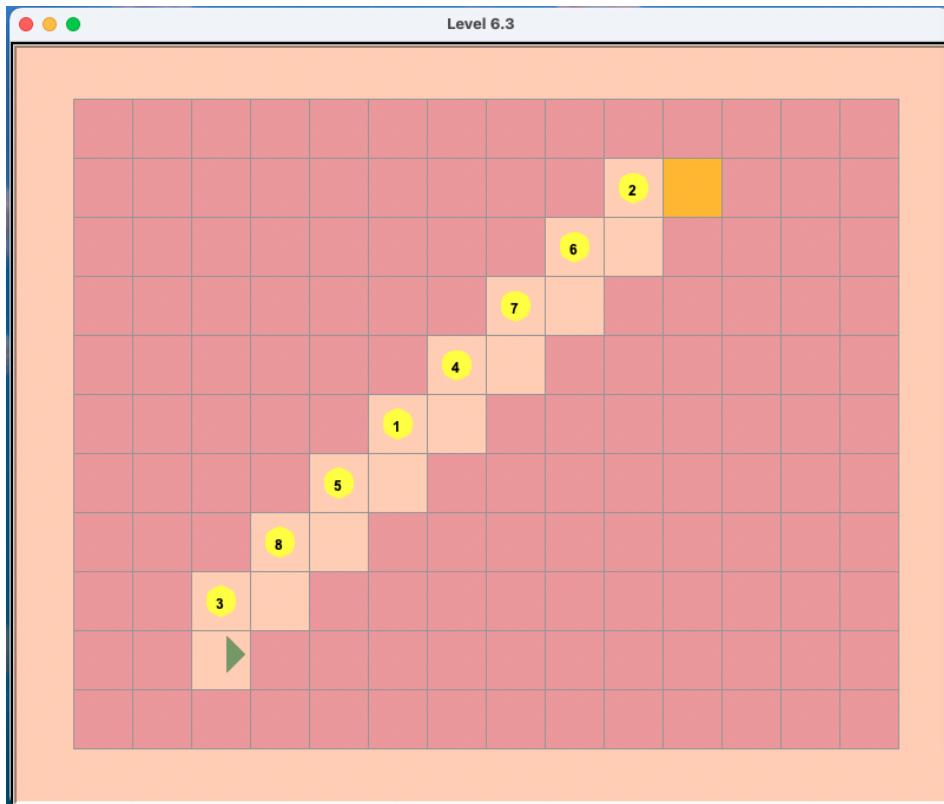
Level 6.2 - Lösung



```
while not goal_reached():
    rotate_left()
    while not is_on_coin():
        move()
    coins.append(pick_up_coin())
    rotate_right()
    rotate_right()
    while can_move_forward():
        move()
    rotate_left()
    move()
    move()
```

Level 6.3

Füge alle Münzen der Liste `coins` hinzu und sortiere sie aufsteigend, bevor du das Ziel erreichst!



Listenoperationen: `list[x]` , `len(list)` ,
`list.sort()` , `list.append(a)`

Notwendige Befehle:

`move()`

Bewegt den Spieler in Pfeilrichtung um einen Schritt nach vorne

`rotate_right()`

Dreht den Spieler um 90 Grad nach rechts

`rotate_left()`

Dreht den Spieler um 90 Grad nach links

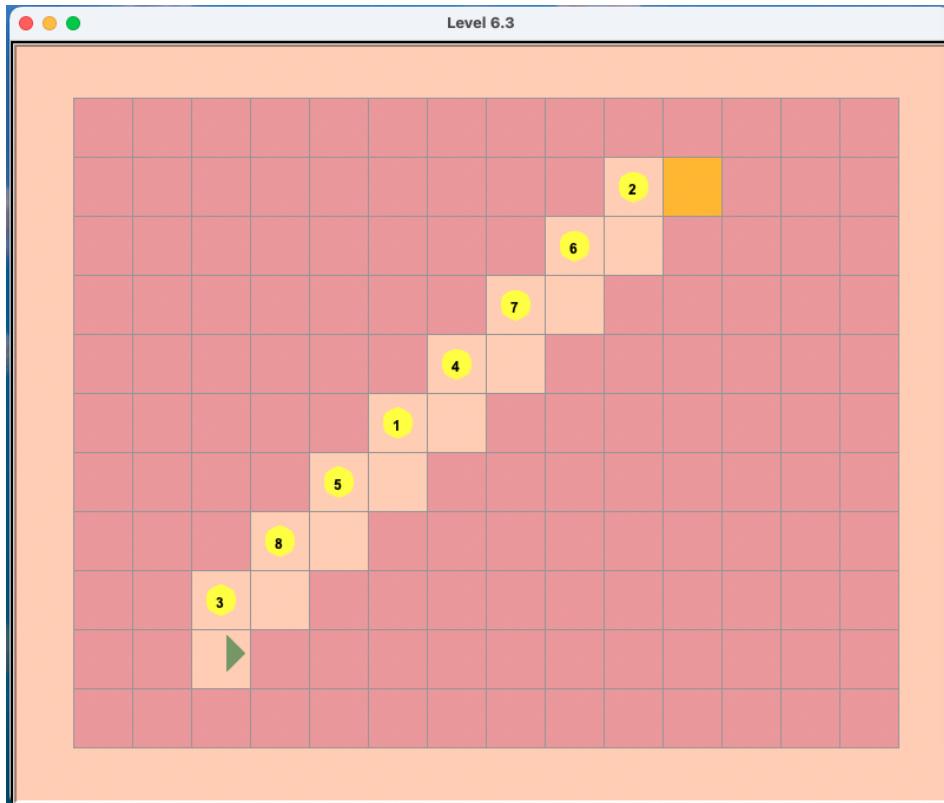
`pick_up_coin()`

Sammelt Münze wenn sich der Spieler auf ihr befindet und gibt die ID der Münze zurück

`goal_reached()`

prüft ob das Ziel erreicht wurde und gibt `True` oder `False` zurück

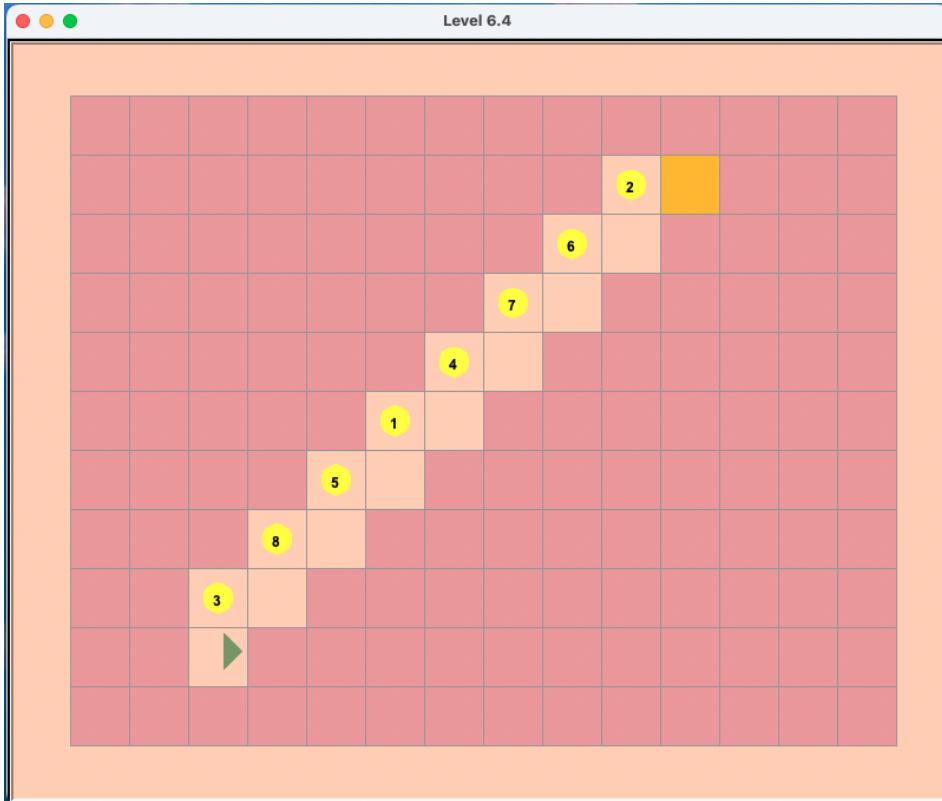
Level 6.3 - Lösung



```
while not goal_reached():
    rotate_left()
    move()
    coins.append(pick_up_coin())
    coins.sort()
    rotate_right()
    move()
```

Level 6.4

Speichere die kleinste Münze in der Variable `smallestCoin`, bevor du das Ziel erreichst!



Listenoperationen: `list[x]` , `len(list)` ,
`list.sort()` , `list.append(a)`

Notwendige Befehle:

`move()`

Bewegt den Spieler in Pfeilrichtung um einen Schritt nach vorne

`rotate_right()`

Dreht den Spieler um 90 Grad nach rechts

`rotate_left()`

Dreht den Spieler um 90 Grad nach links

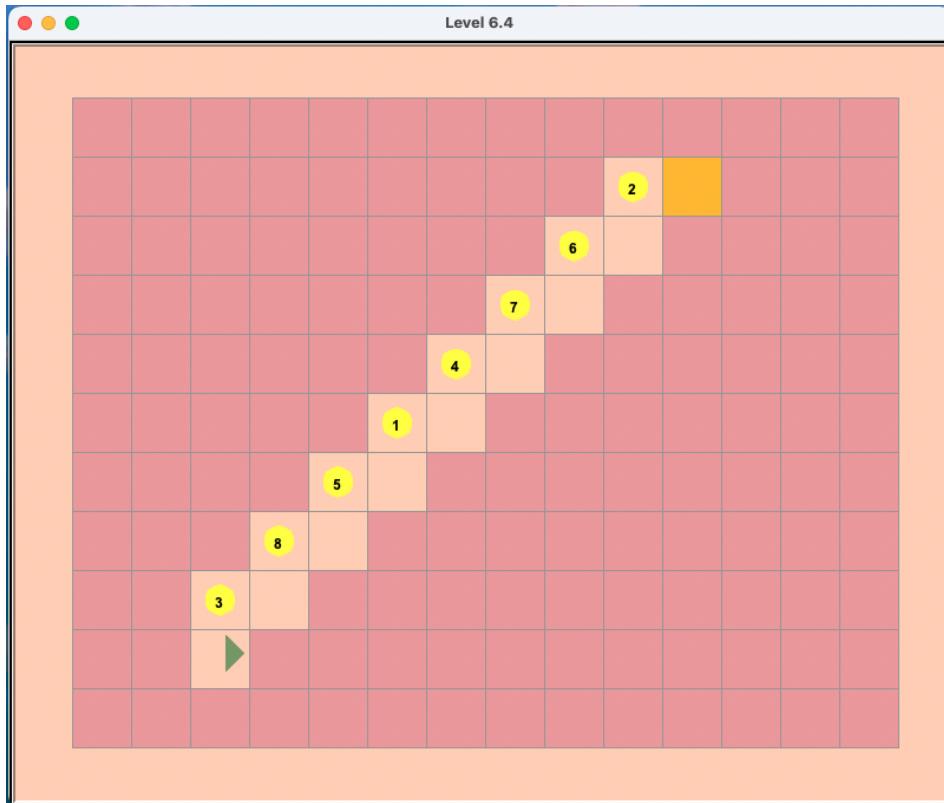
`pick_up_coin()`

Sammelt Münze wenn sich der Spieler auf ihr befindet und gibt die ID der Münze zurück

`goal_reached()`

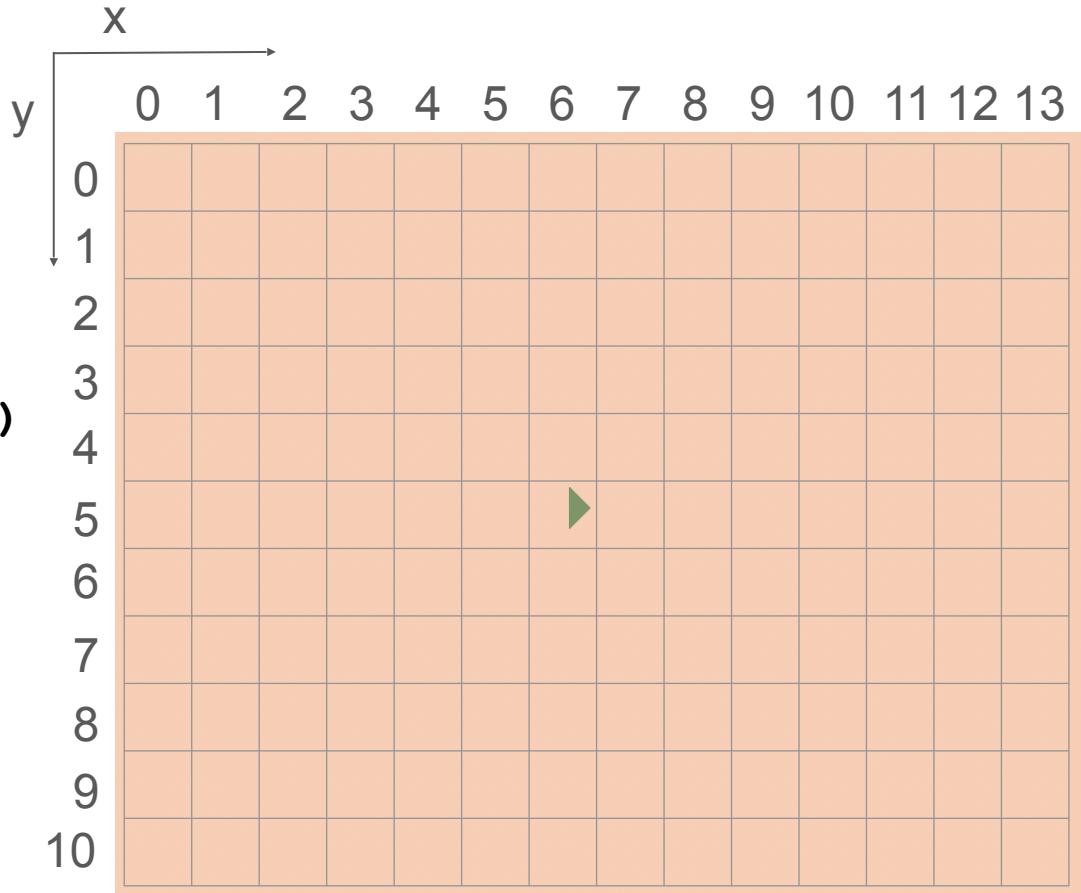
prüft ob das Ziel erreicht wurde und gibt `True` oder `False` zurück

Level 6.4 - Lösung



```
while not goal_reached():
    rotate_left()
    move()
    coins.append(pick_up_coin())
    coins.sort()
    if len(coins) == 8:
        smallestCoin =
coins[0]
    rotate_right()
    move()
```

Koordinaten

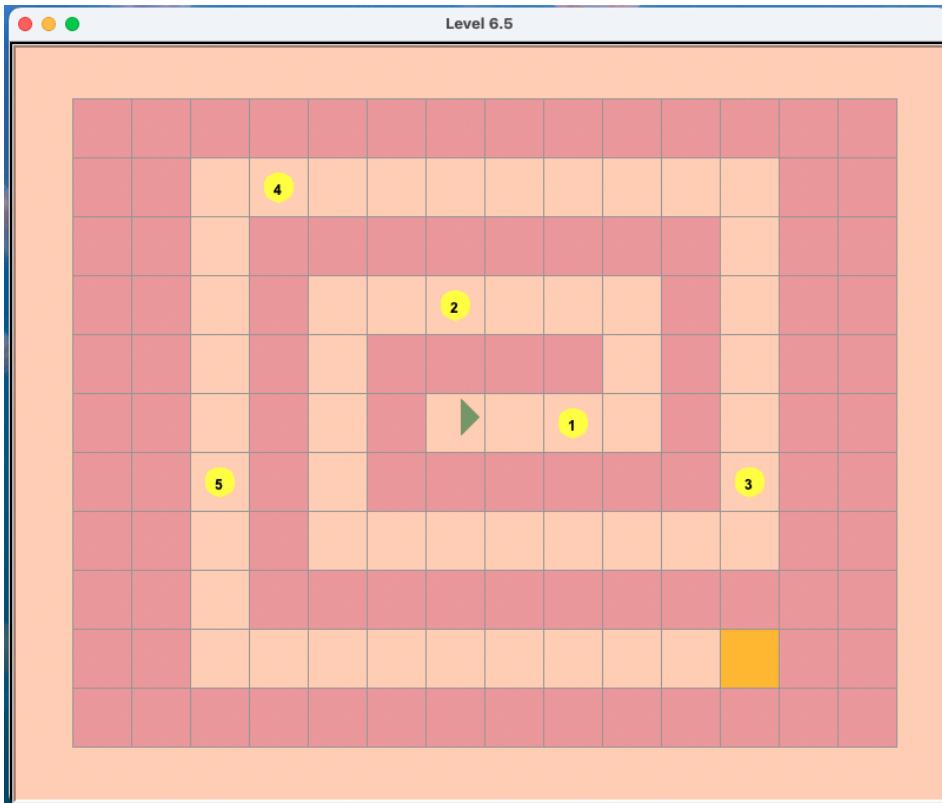


Neuer Befehl: **get_position()**

Gibt die aktuelle Position des Spielers
in der Form [x, y] zurück

Level 6.5

Füge die Summe der Koordinaten jeder der Münzen zur Liste `coord_sums` hinzu und sortiere sie aufsteigend, bevor du das Ziel erreichst!



Listenoperationen: `list[x]` , `len(list)` ,
`list.sort()`, `list.append(a)`

Notwendige Befehle:

`move()`

`rotate_left()`

`get_position()`

Gibt die aktuelle Position des Spielers in der Form
[x, y] zurück

`is_on_coin()`

Prüft ob der Spieler sich auf einer Münze befindet und
gibt `True` oder `False` zurück

`pick_up_coin()`

Sammelt Münze wenn sich der Spieler auf ihr
befindet und gibt die ID der Münze zurück

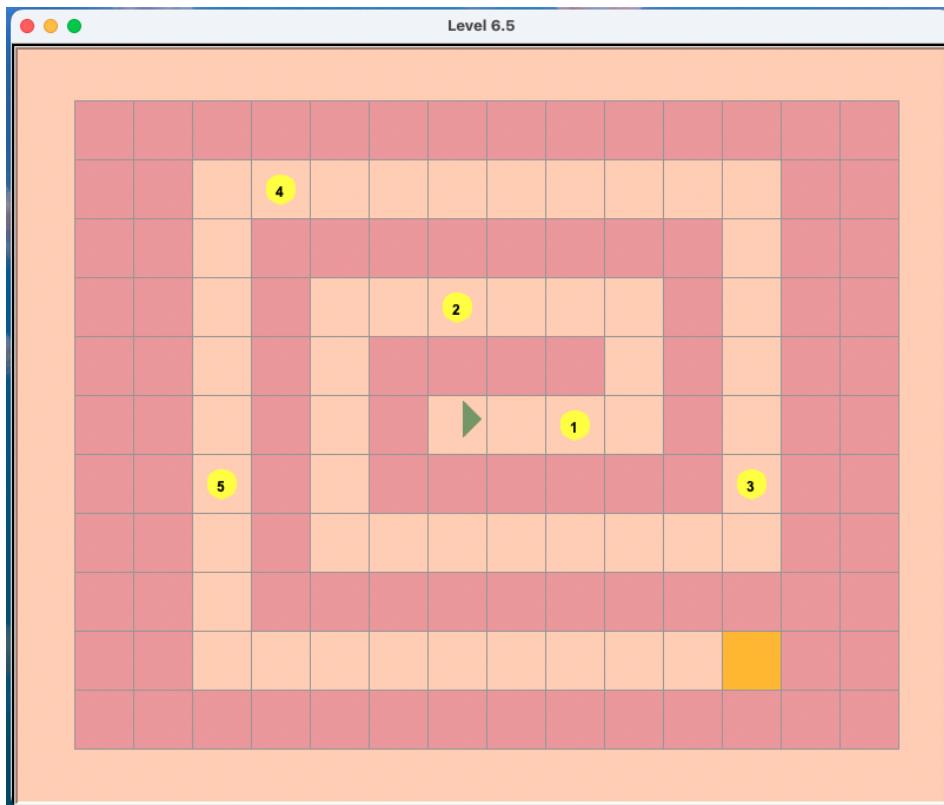
`can_move_forward()`

Prüft ob der Spieler sich in Pfeilrichtung bewegen kann und gibt
`True` oder `False` zurück

`goal_reached()`

prüft ob das Ziel erreicht wurde und gibt `True` oder
`False` zurück

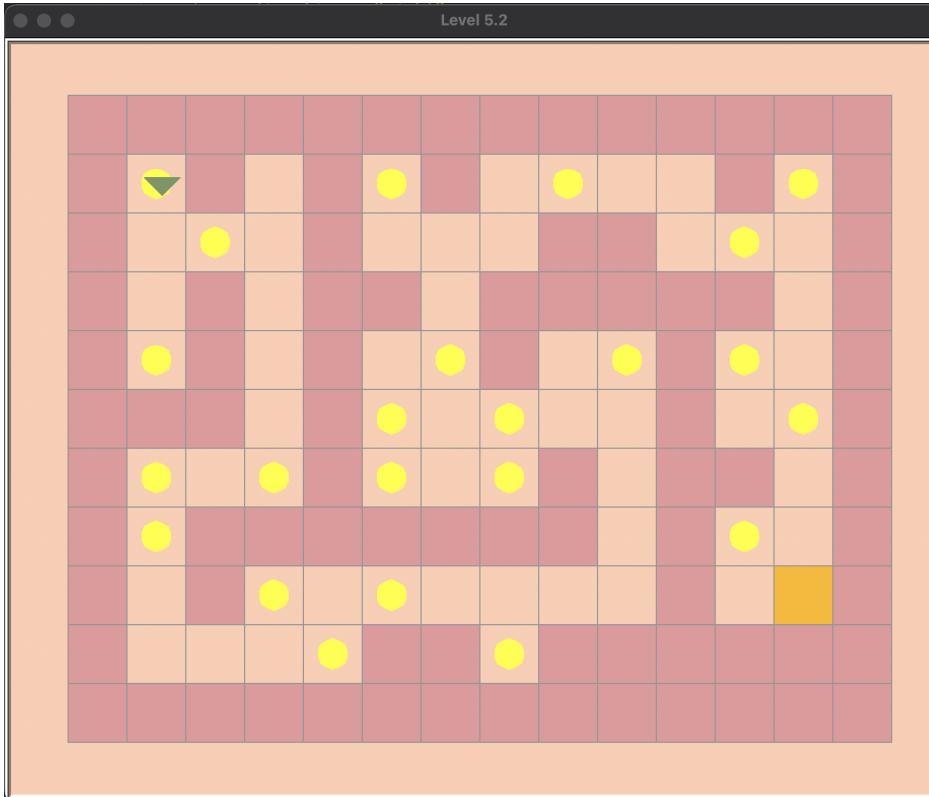
Level 6.5 - Lösung



```
while not goal_reached():
    if not can_move_forward():
        rotate_left()
    else:
        move()
        if is_on_coin():
            coords =
get_position()
sum =
coords[0] + coords[1]
coord_sums.append(sum)
coord_sums.sort()
```

Bonus-Level

Sammle auf deinem Weg zum Ziel so viele Münzen, wie möglich!



Notwendige Befehle:

`move()`

Bewegt den Spieler in Pfeilrichtung um einen Schritt nach vorne

`is_onCoin()`

Prüft ob der Spieler auf einer Münze steht und gibt True oder False zurück

`collect_coin()`

Sammelt die Münze auf

`can_move_forward()`

Prüft ob der Spieler sich in Pfeilrichtung bewegen kann und gibt **True** oder **False** zurück

`rotate_left()`

Dreht den Spieler um 90 Grad nach links

`rotate_right()`

Dreht den Spieler um 90 Grad nach rechts

Bonus-Level



Lösung

```
while not goal_reached():

    collect_coin()

    rotate_right()

    while not can_move_forward():

        rotate_left()

    move()
```