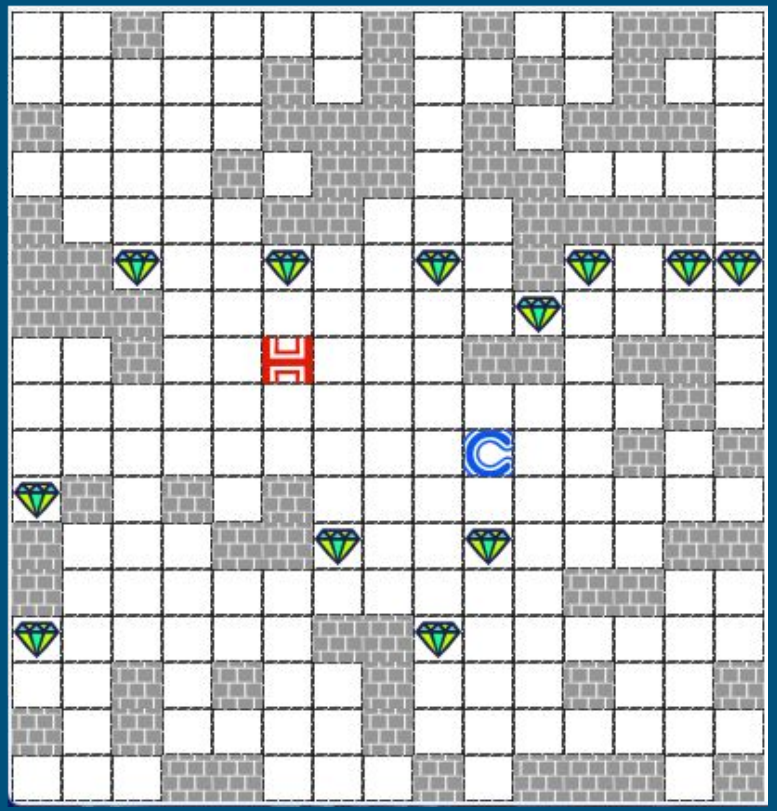
U-16プログラミングコンテスト

講習会

U-16プログラミングコンテスト　ルール



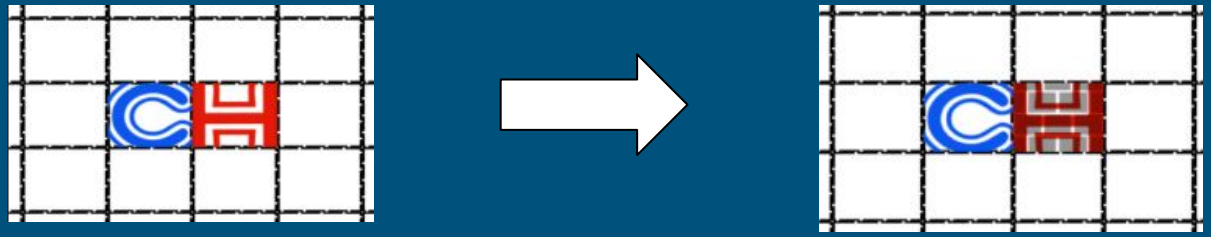
COOL（先行）とHOT（後攻）が交互に動きます。

多くのアイテムを獲得した方が勝ちになります。

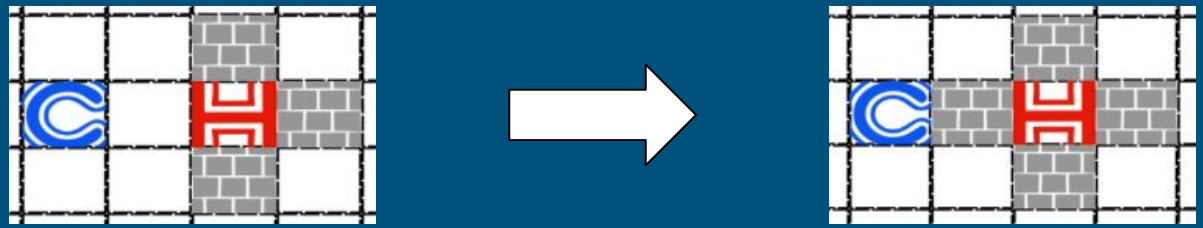
相手をブロックを被せる、またはブロックで身動き取れなくすればPUT勝ちとなります。

PUT勝ち

対戦相手にブロックを被せる

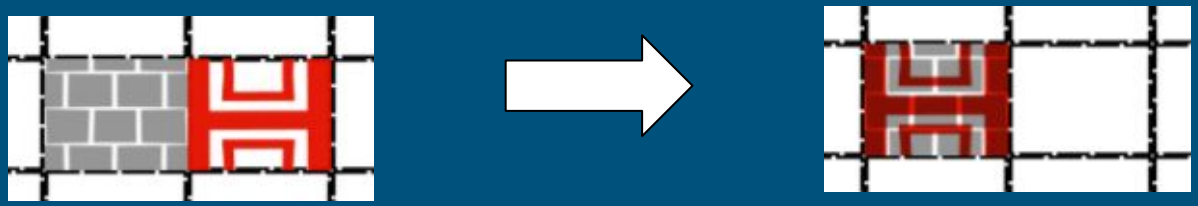


対戦相手を四方をブロックで囲む



PUT勝ち（相手の自滅）

対戦相手がブロックに移動する

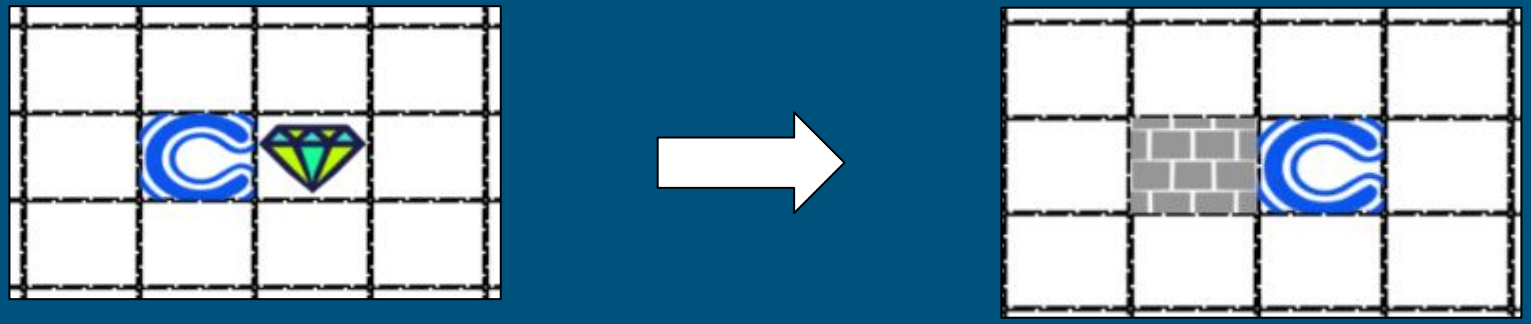


対戦相手が通信エラーで試合を中断させる

COOL WIN!
[HOT 切断により]

アイテムを獲得した時の動き

アイテムを獲得すると、キャラクターがいた
場所にブロックが置かれる

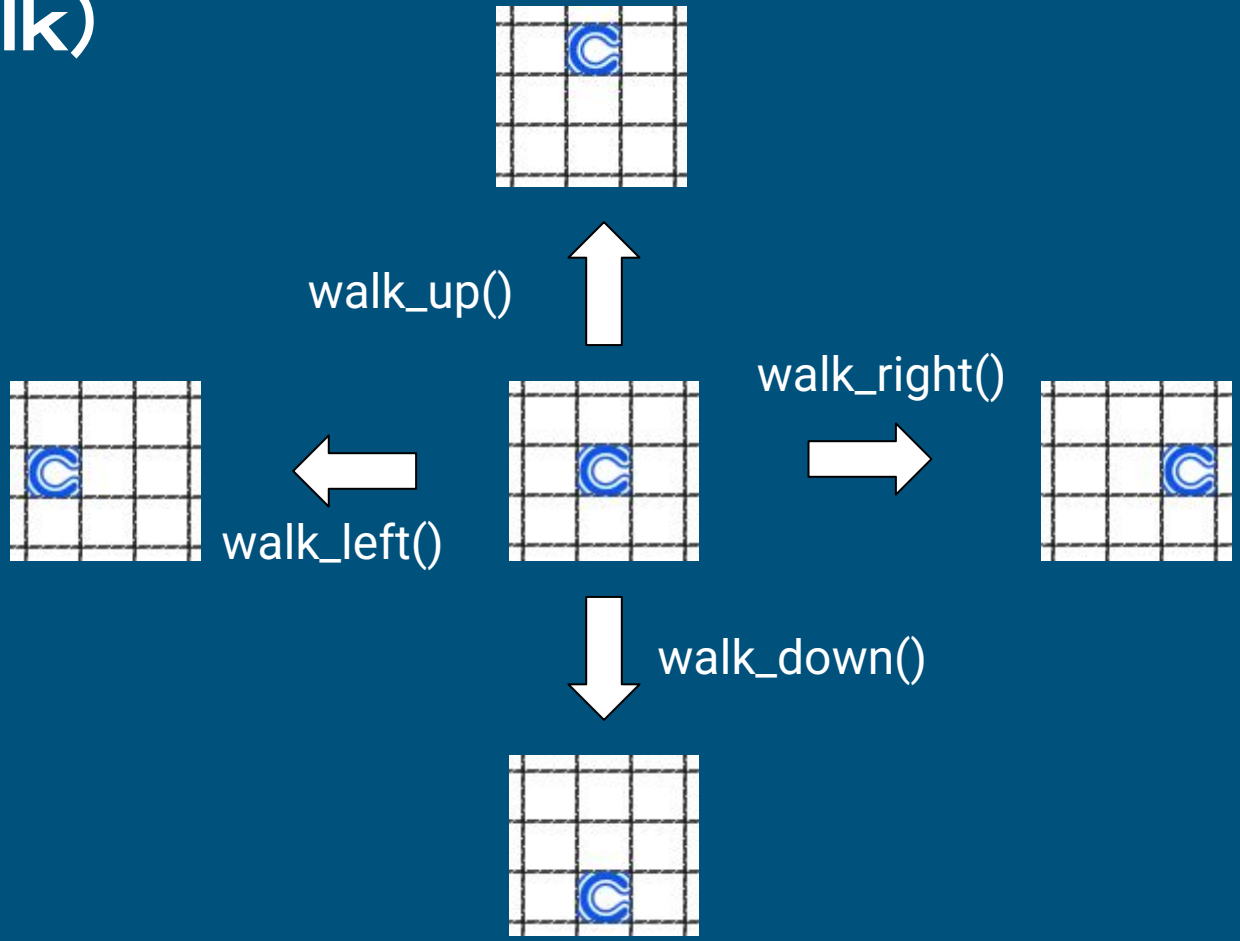


実行できる行動

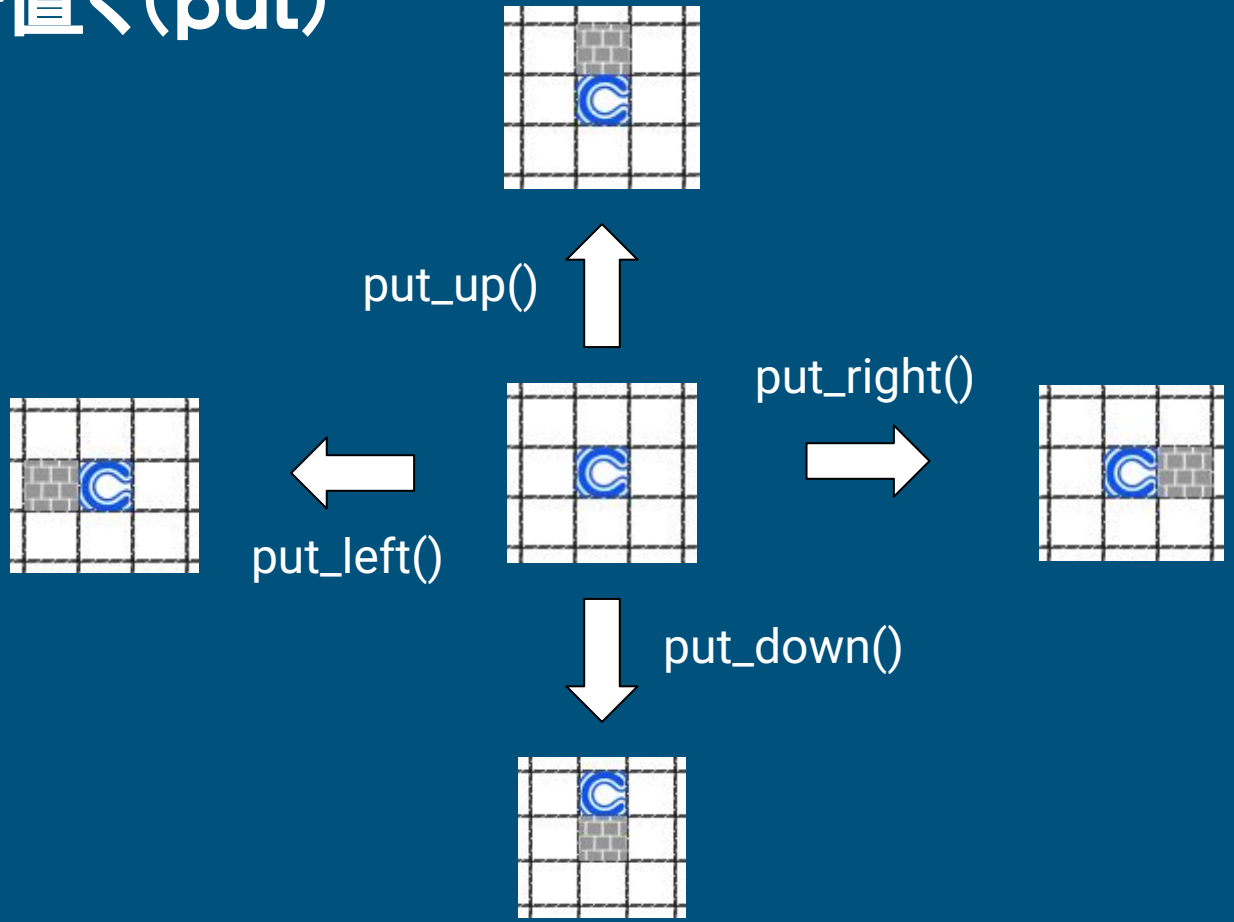
- 歩く（walk）
- ブロックを置く（put）
- 見る（look）
- 調べる（search）

全ての行動に上（up）、下（down）、
右（right）、左（left）の4方向に実行できる。

歩く（walk）



ブロックを置く（put）



配列とマップ情報について

（見る（look）、調べる（search）の前に）

0：何もなし
1：対戦相手
2：ブロック
3：アイテム

0	0	3
0	0	2
2	3	1

{ 0, 0, 3, 0, 0, 2, 2, 3, 1}

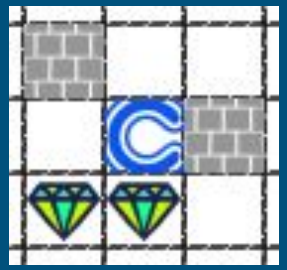
0 1 2 3 4 5 6 7 8

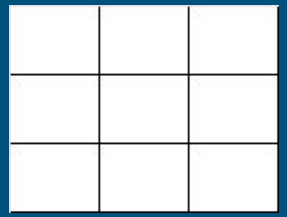
先頭から数える

配列とマップ情報について

練習してみましょう！

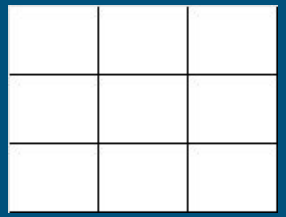
練習1





{ , , , , , , , , }

練習2



{ , , , , , , , , }

0：何もなし
1：対戦相手
2：ブロック
3：アイテム

配列とマップ情報について

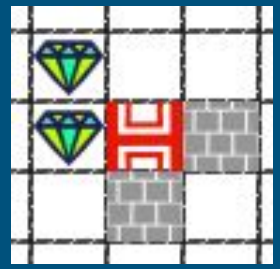
練習してみましょう！　解答

練習1

2	0	0
0	0	2
3	3	0

{ 2, 0, 0, 0, 0, 2, 3, 3, 0 }

練習2



3	0	0
3	0	2
0	2	0

{ 3, 0, 0, 3, 0, 2, 0, 2, 0 }

0：何もなし
1：対戦相手
2：ブロック
3：アイテム

配列とマップ情報について

練習してみましょう！

0	0	3
0	0	2
2	3	1

練習1

キャラクタの右のブロックはどれでしょうか？

{ 0, 0, 3, 0, 0, 2, 2, 3, 1}

練習2

キャラクタの右のブロックの番号は？

{ 0, 0, 3, 0, 0, 2, 2, 3, 1}

0 1 2 3 4 5 6 7 8

配列とマップ情報について

練習してみましょう！　　解答

0	0	3
0	0	2
2	3	1

練習1

キャラクタの右のブロックはどれでしょうか？

{ 0, 0, 3, 0, 0, 2, 2, 3, 1}

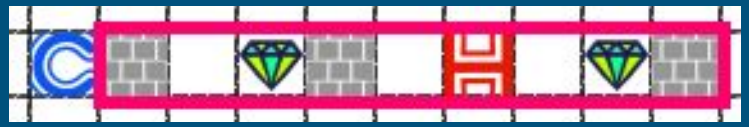
練習2

キャラクタの右のブロックの番号は？

{ 0, 0, 3, 0, 0, 2, 2, 3, 1}

5

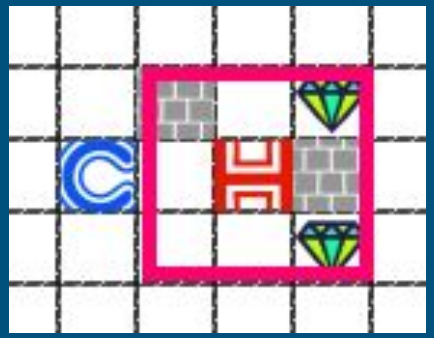
調べる（search）



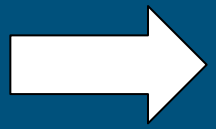
search_right()

{ 2, 0, 3, 2, 0, 1, 0, 3, 2}

見る(look)



look_right()

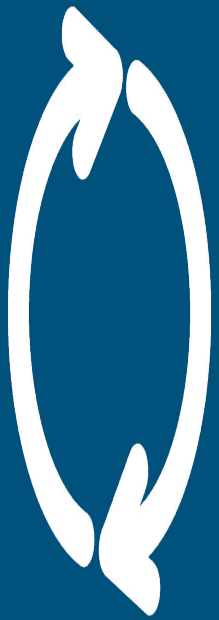


2	0	3
0	1	2
0	0	3

{ 2, 0, 3, 0, 1, 2, 0, 0, 3}

プログラムの基本

while(True):
を繰り返す。



```
while(True):
    value = client.get_ready()
    value = client.search_left()
```
} 動作①

```
    value = client.get_ready()
    value = client.walk_down()
```
} 動作②

```
    value = client.get_ready()
    value = client.look_up()
```
} 動作③

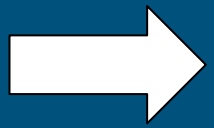
```
    value = client.get_ready()
    value = client.put_right()
```
} 動作④

動作①→相手→動作②→相手→動作③→相手→動作④→相手→動作①→相手→・・・

get_ready() とは

ここからプログラムがスタートします。

get_ready()

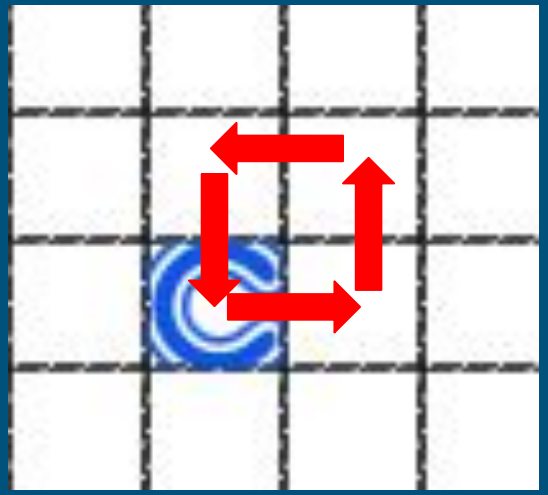


2	2	0
0	0	0
3	0	3

{ 2, 2, 0, 0, 0, 0, 3, 0, 3}

Exercises　1

アイテムを「下 →右→上→左」と動くようにプログラミングします。



```
while(True):
        value = client.get_ready()
        value = client.walk_down()
```

} 下に移動

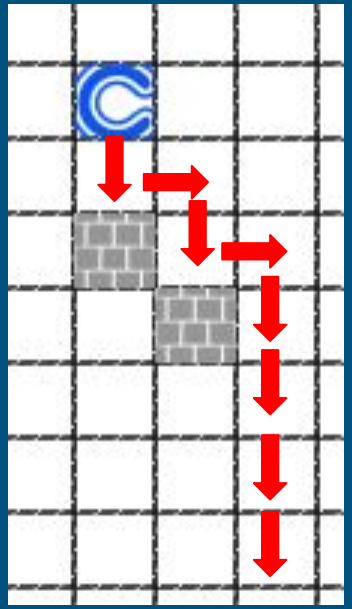
} 右に移動

} 上に移動

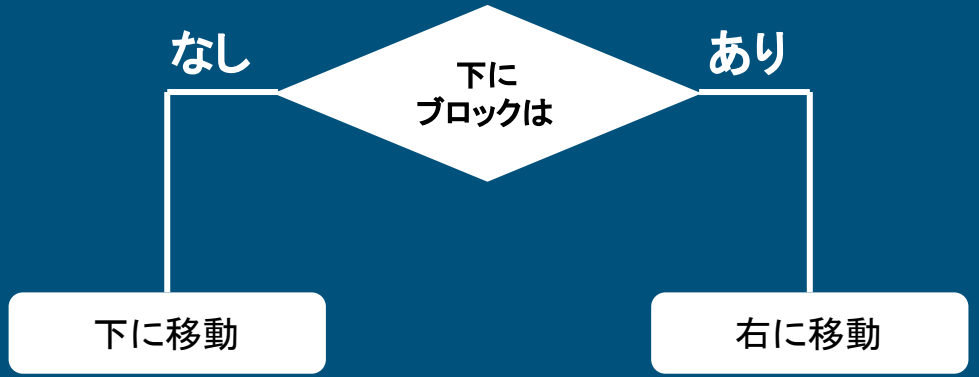
} 左に移動

Exercises　2

ブロックが無いときは「下」へ、ブロックがある時は右へ移動するようにプログラミングします。

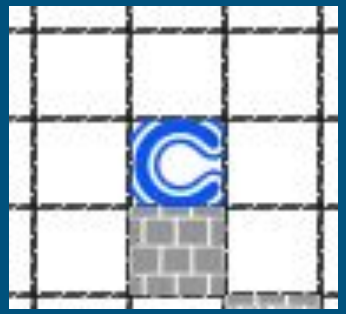


アイテムが下に移動するときにブロックが無いことを確認してから移動します。もし、ブロックがある時は右に移動します。

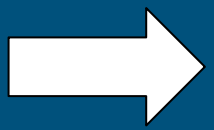


Exercises 2

ブロックがキャラクタの下にあるかどうかを判断する方法は。



get_ready()



0	0	0
0	0	0
0	2	0

{ 0, 0, 0, 0, 0, 0, 0, 2, 0}

7

Exercises　2

プログラミングで「もし」を扱うには　if文を使います。

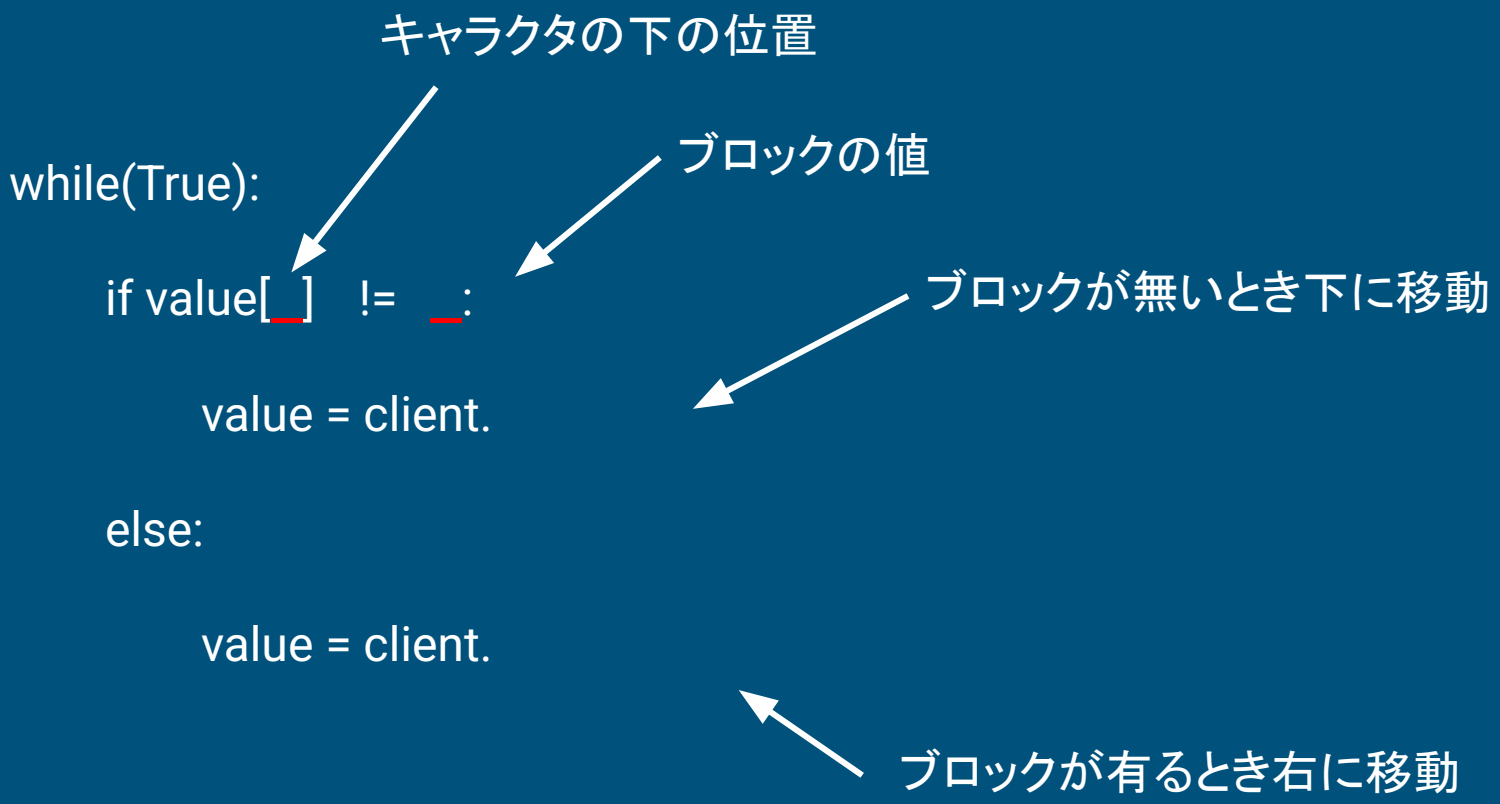
```
if 条件 :
        処理 1
        処理 2
          .
          .
else:
        処理 3
        処理 4
          .
          .
```

条件が正しい

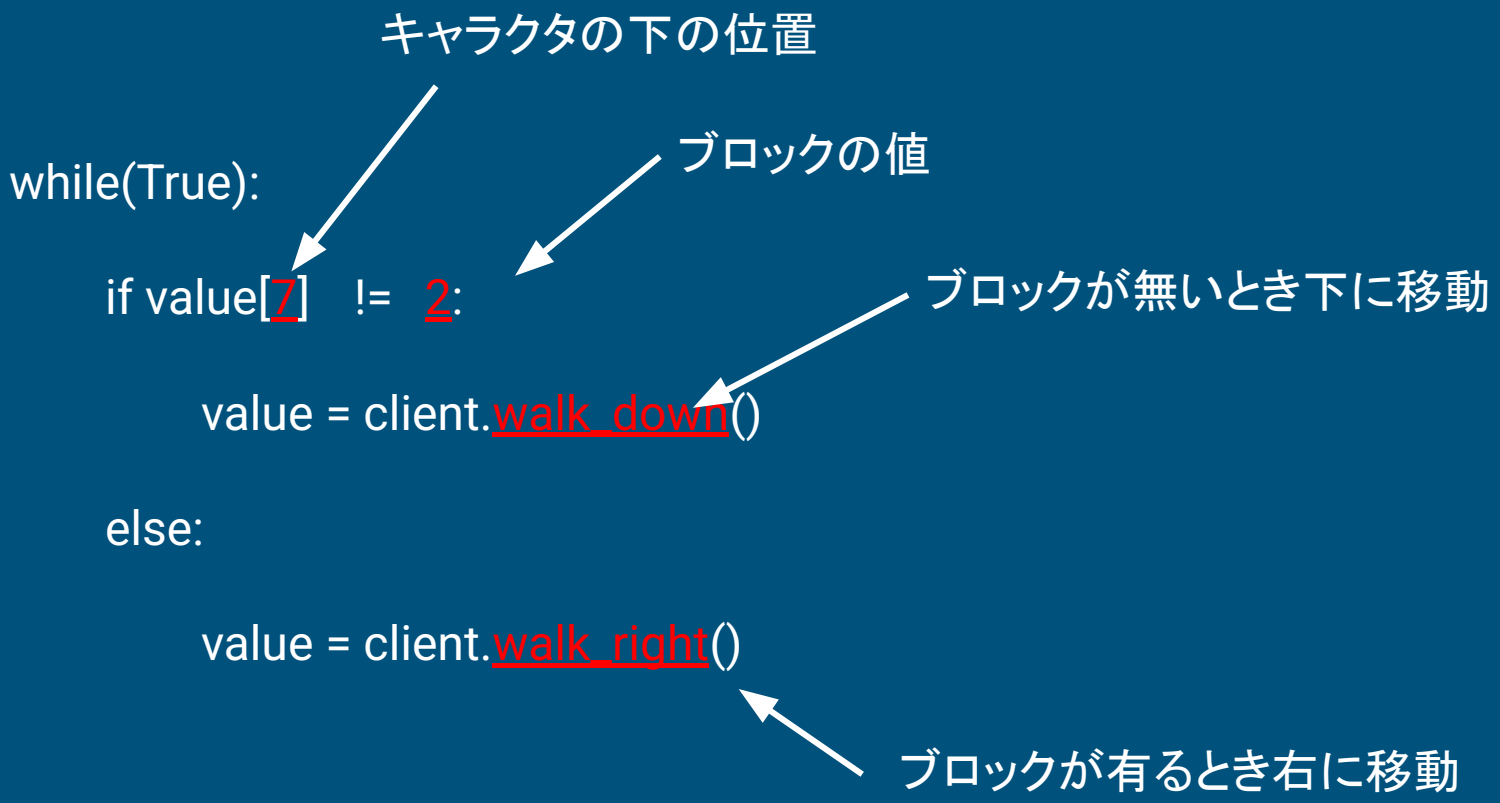
条件が間違い

条件（演算子）	
a == b	a が b と等しい
a != b	a が b と異なる
a < b	a が b よりも小さい
a > b	a が b よりも大きい
a <= b	a が b 以下である
a >= b	a が b 以上である
a is b	a が b と等しい
a is not b	a が b と異なる
a in b	a が b に含まれる
a not in b	a が b に含まれない

Exercises 2

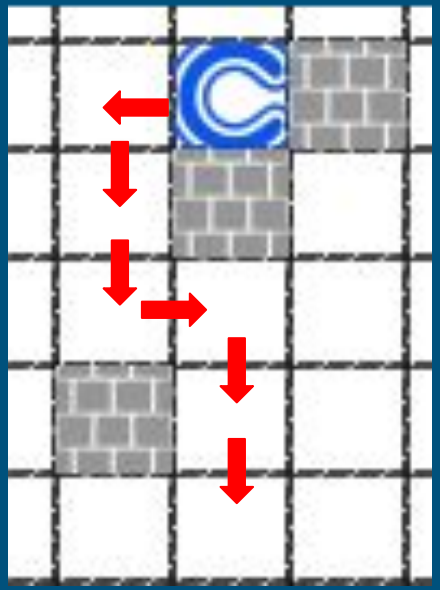

キャラクタの下の位置
ブロックの値
while(True):
if value[__] != __:
ブロックが無いとき下に移動
value = client.
else:
value = client.
ブロックが有るとき右に移動

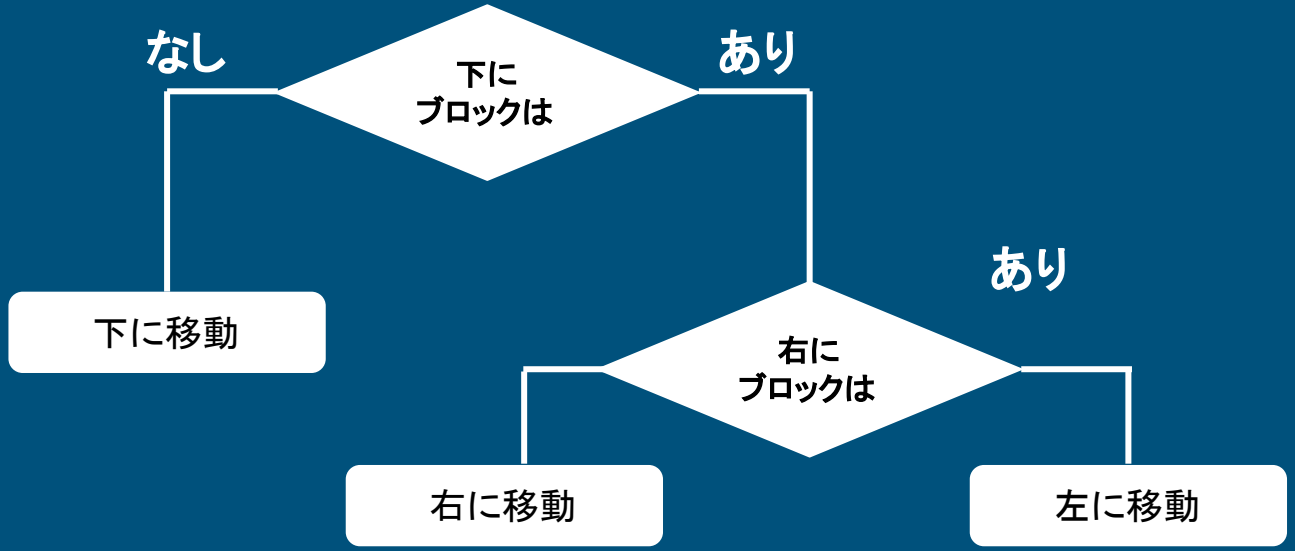
Exercises 2


キャラクタの下の位置
ブロックの値
while(True):
if value[7] != 2:
ブロックが無いとき下に移動
value = client.walk_down()
else:
value = client.walk_right()
ブロックが有るとき右に移動

Exercises　3

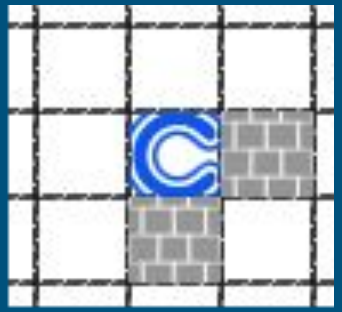
先ほどのプログラムに追加します。ブロックが無いときは「下」へブロックがある時は右へ移動するようにしますが、右にもブロックがあると左に移動するようにプログラミングします。



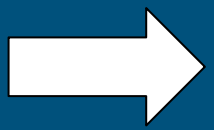


Exercises 3

ブロックがキャラクタの下にあるかどうかを判断する方法は。



get_ready()



0	0	0
0	0	2
0	2	0

{ 0, 0, 0, 0, 0, 2, 0, 2, 0}

5 7

Exercises 3

キャラクタの下の位置

ブロックの値

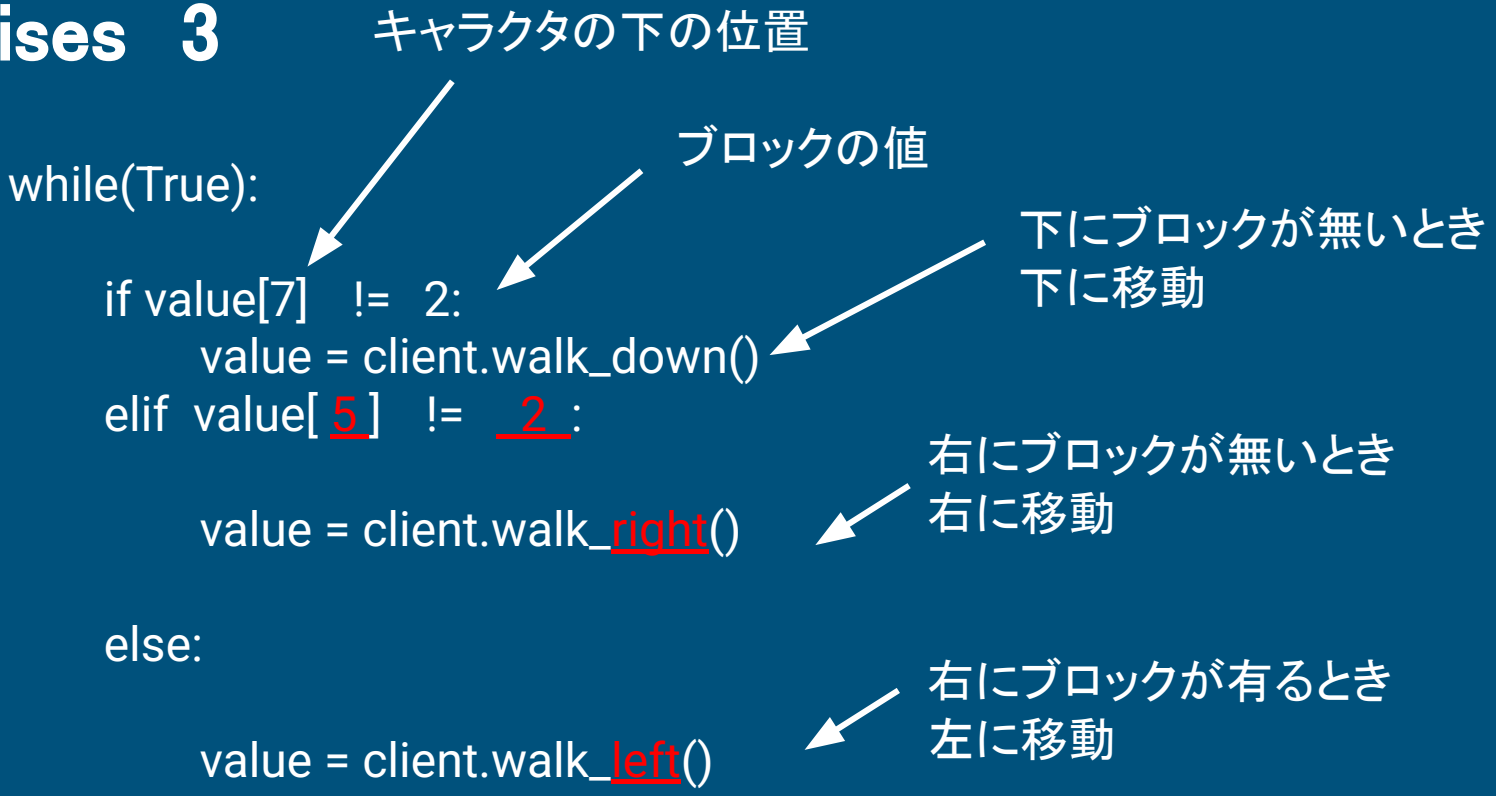
```
while(True):

    if value[7]   !=   2:
        value = client.walk_down()
    elif  value[ __]   !=   ___:

        value = client.walk_

    else:

        value = client.walk_
```

下にブロックが無いとき
下に移動

右にブロックが無いとき
右に移動

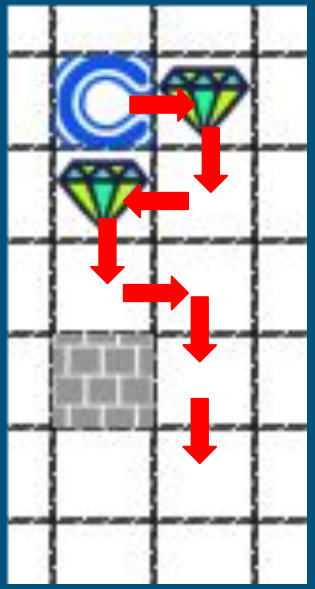
右にブロックが有るとき
左に移動

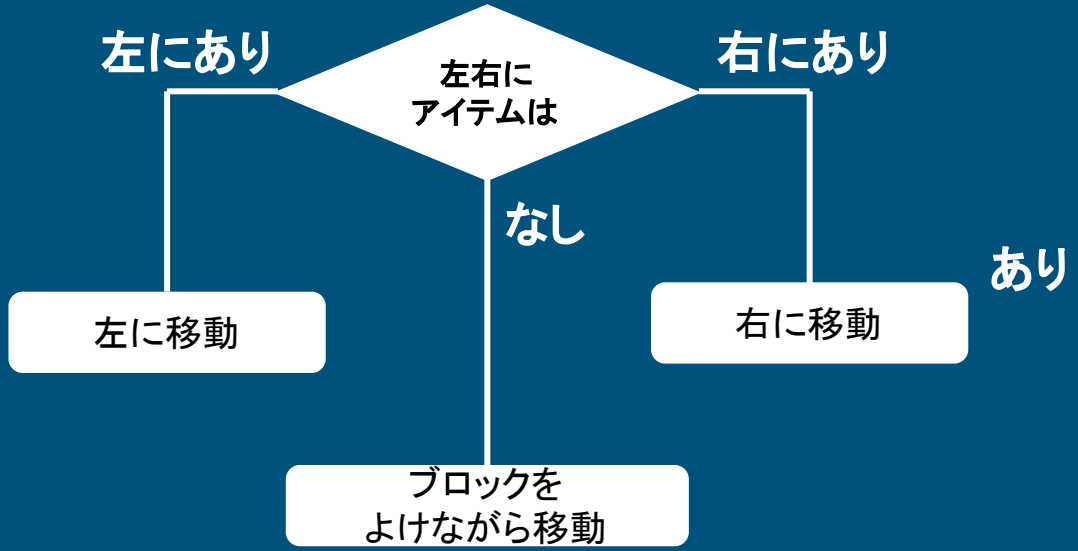
Exercises 3


キャラクタの下の位置
ブロックの値
while(True):
if value[7] != 2:
value = client.walk_down()
elif value[5] != 2 :
value = client.walk_right()
else:
value = client.walk_left()
下にブロックが無いとき
下に移動
右にブロックが無いとき
右に移動
右にブロックが有るとき
左に移動

Exercises　4

先ほどのプログラムを応用して、横にアイテムがある時、アイテムを取るようにしましょう。横にアイテムがない時は先ほど作成したように動きます。

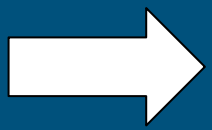




Exercises 4

アイテムがキャラクタの左右にあるかどうかを判断する方法は。

get_ready()



0	0	0
3	0	3
0	0	0

{ 0, 0, 0, 3, 0, 3, 0, 0, 0}

3 5

Exercises 4

```
while(True):

    if value[ __ ]   ==   ___:
        value = client.walk_
    elif  value[ ___]   ==   ___:
        value = client.walk_
    else:
        if value[7]   !=  2:
            value = client.walk_down()
        elif  value[ 5 ]   !=    2  :
            value = client.walk_right()
        else:
            value = client.walk_left()
```

キャラクタの左位置にアイテムがある
左に移動

キャラクタの右位置にアイテムがある
右に移動

アイテムが無い時の動き

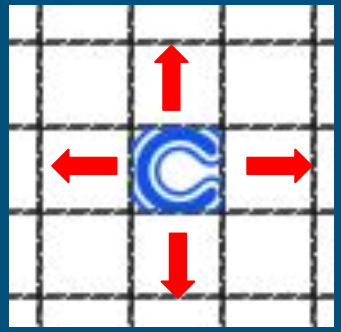
Exercises　5

先ほどのプログラムを応用して、上と下にアイテムがあるときはアイテムを取りに行くようにしましょう

```
while(True):

    if value[ __ ]   ==   ___:
        value = client.walk_
    elif  value[ ___]   ==   ___:
        value = client.walk_
    elif  value[ ___]   ==   ___:
        value = client.walk_
    elif  value[ ___]   ==   ___:
        value = client.walk_
    else:
        壁をよけながら進む
```

Exercises　6

キャラクタをランダムに動かしてみます。



ランダムな数字を発生させて

0 → 上
1 → 左
2 → 下
3 → 右

に移動します。

Exercises　6

```
while(True):

    value = client.get_ready()          # サーーに行動準備が完了したと伝える

    number = random.randint(0, 3)
    if number == 0:
        client.walk_up()
    elif number == 1:
        client.walk_down()
    elif number == 2:
        client.walk_left()
    elif number == 3:
        client.walk_right()
```

Exercises　7

キャラクタをランダムに動かしてみます。

問題点

キャラクタをランダムに動かすと壁に移動することがある。
そこで、壁には移動しないようにする。

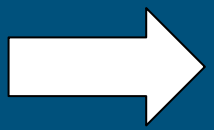
```
while(True):
        value = client.get_ready() # サーバーに行動準備が完了したと伝える
                number = random.randint(0, 3)
               if number == 0:
                  if value[_] !=    _:
                     client.walk_up()
                  else:
                     client.search_down()

                     以下つづく
```

Exercises 8

Exercises 4 復習

アイテムがキャラクタの左右にあるかどうかを判断する方法は。

get_ready()



0	0	0
3	0	3
0	0	0

{ 0, 0, 0, 3, 0, 3, 0, 0, 0}

3 5

Exercises 8

```
while(True):

    if value[ __ ]   ==   ___:
        value = client.walk_
    elif  value[ ___]   ==   ___:
        value = client.walk_
    else:
        number = random.randint(0, 3)
        if number == 0:
            if value[_] !=   _:
                client.walk_up()
            else:
                client.search_down()

                以下つづく
```

キャラクタの左位置にアイテムがある
左に移動

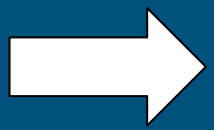
キャラクタの右位置にアイテムがある
右に移動

アイテムが無い時の動き

Exercises 9

キャラクタの上下左右にあるかどうかを判断する。

get_ready()



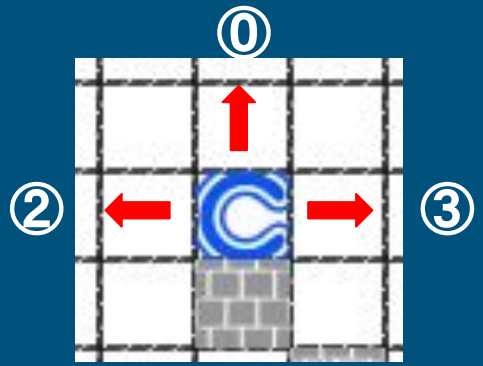
0	3	0
3	0	3
0	3	0

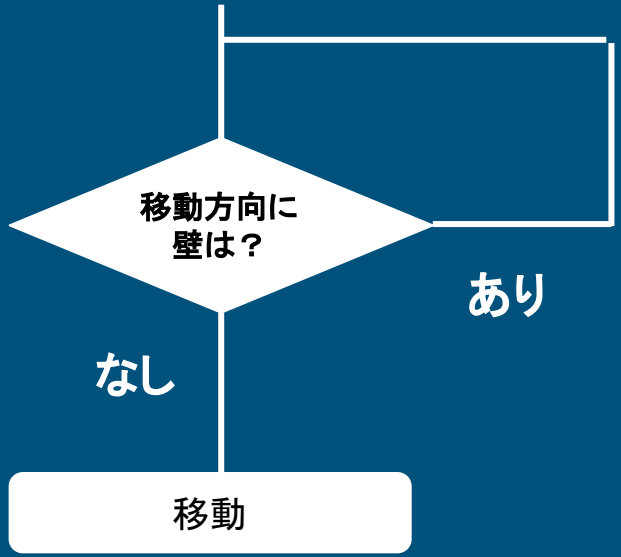
{ 0, 3, 0, 3, 0, 3, 0, 3, 0}

1 3 5 7

Exercises　10

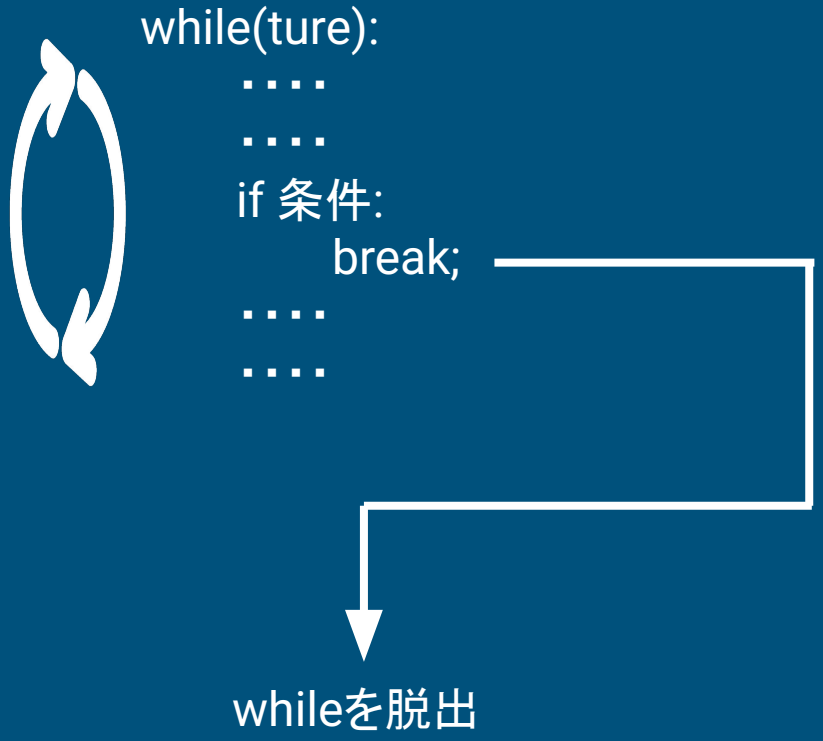
キャラクタをランダムに動かす　Exercises 7　で壁に移動できないので、サーチをして無駄な動きをしている。そこで、 移動できる方向でランダムに移動する。





Exercises　10

while文とbreak

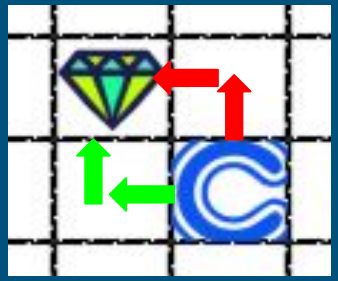


Exercises 10

```
else:
        #ランダムに進む方向に進む（壁があるとやりなおす）
        while(True):
            number = random.randint(0, 3)
            if number == 0:
                if value[1] != 2:
                    client.walk_up()
                    _____;
            elif number == 1:
                if value[7] != 2:
                    client.walk_down()
                    _____;
            elif number == 2:
                if value[3] != 2:
                    client.walk_left()
                    _____;
            elif number == 3:
                if value[5] != 2:
                    client.walk_right()
                    _____;
```

Exercises　11

アイテムが斜め方向にある時に2ターンで取りに行く。



ルート①

ルート②

ルート①でもルート②のどちらでも可能

Exercises　11

ルート①

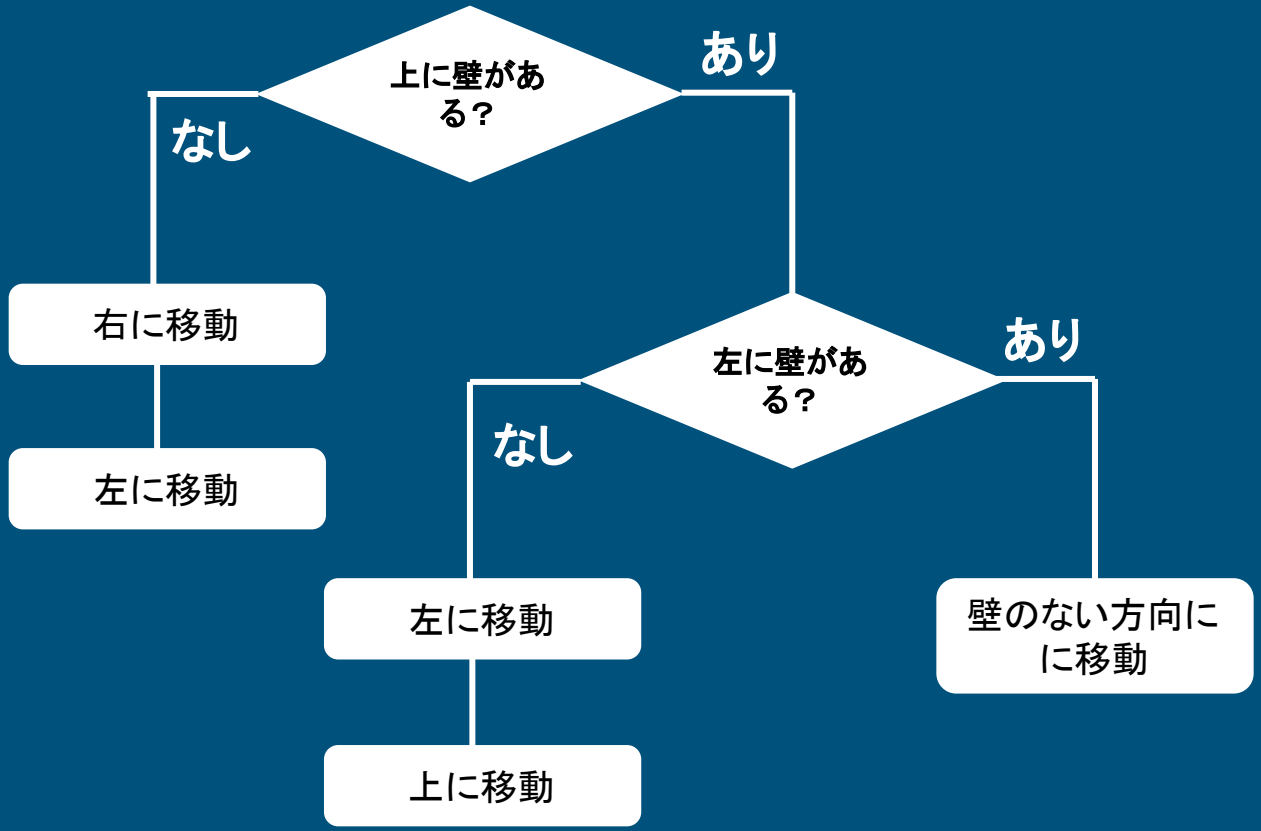
ルート①しか行けない

ルート②しか行けない

ルート②

行けない

Exercises 11

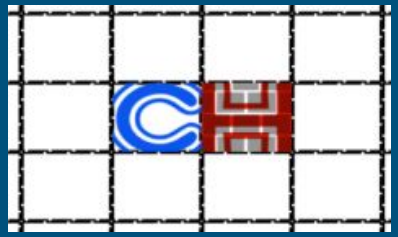


Exercises　11

```
elif value[_] == 3:              #斜め左上にアイテムがある時
    if value[_] != 2:
        value = client.walk_up()
    elif value[_] != 2:
        client.walk_left()
    elif value[_] != 2:
        client.walk_right()
    else:
        client.walk_down()
```

Exercises　12

対戦相手にブロックを被せる



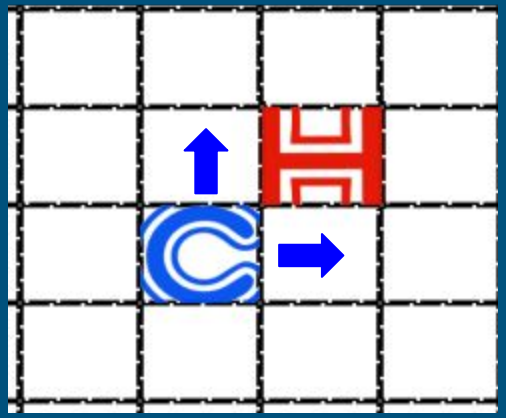
❖　ブロックを置く（put）

Exercises　12

```
#相手にブロックをかぶせる
if value[_] == 1:              #上に相手がいる時
    value = client.put_up()
elif value[_] == 1:            #右に相手がいる時
    value = client.put_right()
elif value[_] == 1:            #下に相手がいる時
    value = client.put_down()
elif value[_] == 1:            #左に相手がいる時
    value = client.put_left()
```

Exercises　13

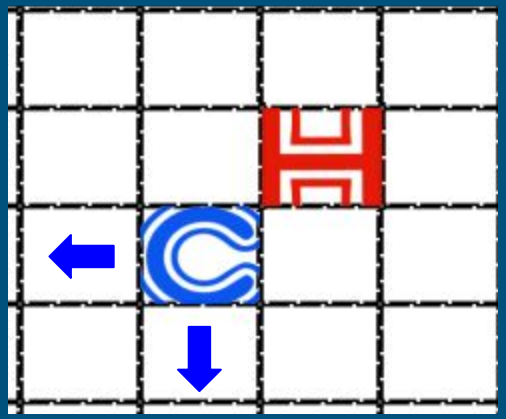
対戦相手から逃げる



青い矢印方向に移動いるとputされて負ける。

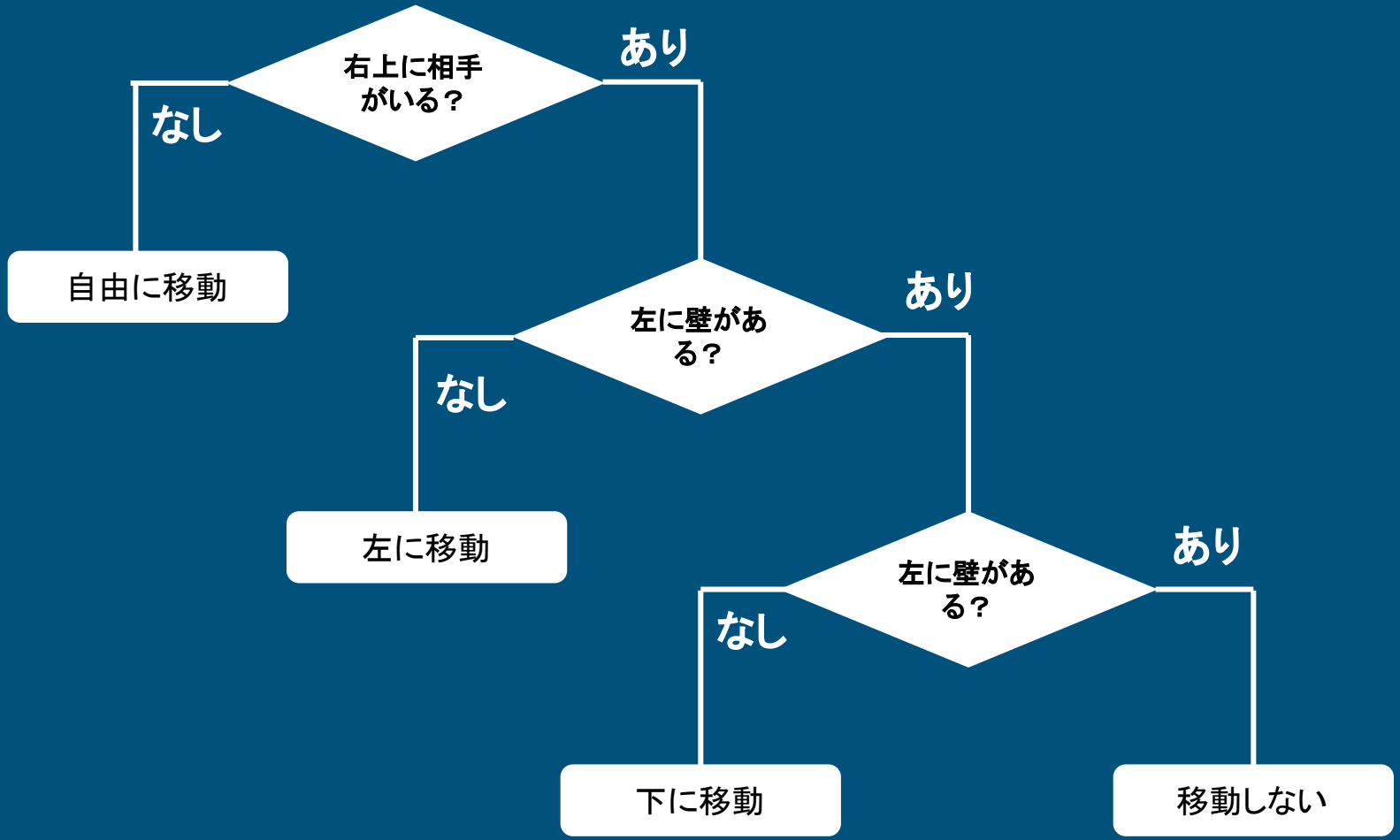
Exercises 13

対戦相手から逃げる



青い矢印方向に逃げる必要がある。

Exercises 13



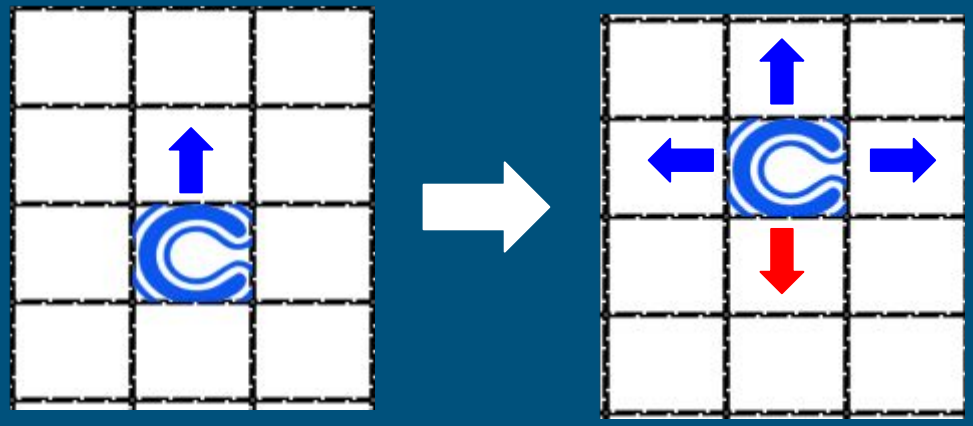
Exercises　13

```
#斜めに相手がいる時にげる
        elif value[__] == 1:                #右上に相手がいるか？
            if value[__] != 2:              #左にブロックがあるか？
                client.walk_right()
            elif  value[__] != 2:
                client.walk_down()          #下にブロックがあるか？
            else:
                client.search_down()        #とりあえず下を見る
```

同様に「右上」「左下」「右下」に相手がいる場合の処理を書く必要がある。

Exercises　14

移動した場所に戻らない



赤い方向に戻ると無駄な動きになるため他の方向に移動する。

Exercises 14

前の情報を関数で覚えておく必要がある。

```
def main():
    number = 5
      .
      .
      .
    print(number)
```

number変数はmain関数のみで使用され、終了すると変数に記憶している内容も無くなってしまう。

ローカル変数

前のターンでどの方向に移動したかを覚えておくにはこのままでは使えない。

Exercises 14

前の情報を関数で覚えておく必要がある。

```
number = 0
def main():
    global number
    print(number)
      .
      .
      .
    number += 1
```

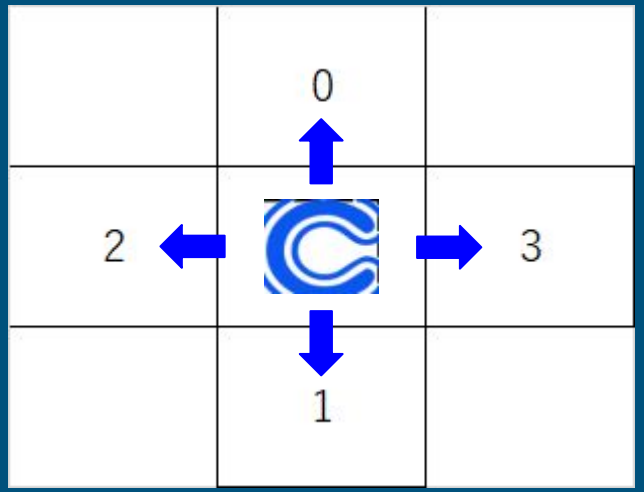
実行結果

0
1
2
.
.
.

グローバル変数

グローバル変数を利用すれば、前のターンでどの方向から移動してきたかがわかる。

Exercises 14



移動方向の番号

前の位置から移動してきた方向	0:上	1:下	2:左	3:右
	↓	↓	↓	↓
移動してはいけない方向	1:下	0:上	3:右	2:左

移動方向してはいけない方向

Exercises　14

グローバル変数を使えるようにする

```
pre_walk = 0　←　グローバル変数　0に初期化

def main():
    value = []
    client = CHaser.Client()
    global pre_walk　←　グローバル変数にアクセスする

    while(True):
            .
            .
            .
            .
```

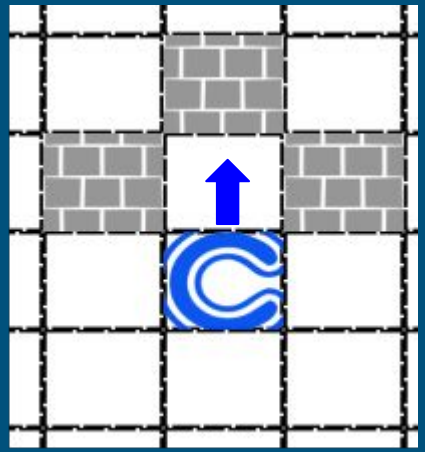
Exercises　14

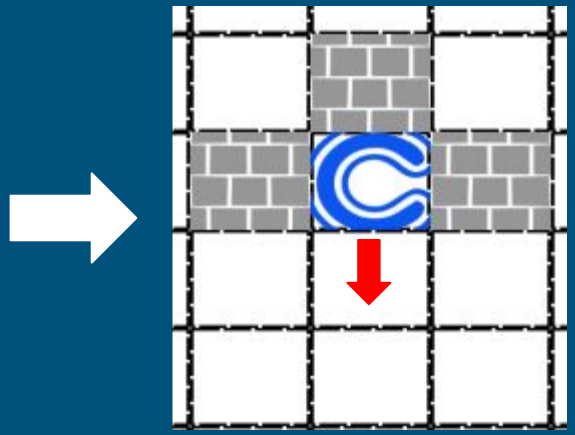
前に移動してきた方向をグローバル変数に記憶する

```
number = random.randint(0, 3)
if number == 0:                    #上に移動する
    if value[1] != 2:              #上にブロックがあるか
        if pre_walk != 1:          #前の位置と同じでないか
            client.walk_up()
            pre_walk = 0           #今移動した方向を記憶しておく
            break;
```

Exercises 14

注意！！

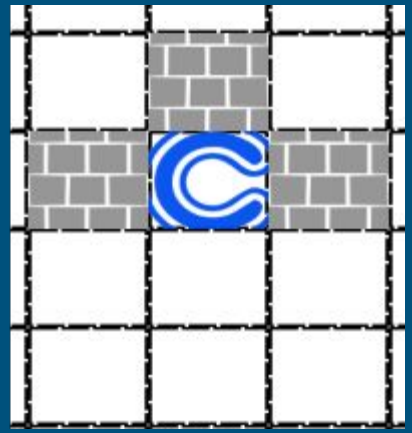




赤い方向は前の位置であるが、移動できる方向はこの方向しかないので移動する。

Exercises　14

この状況を判断する



4方向でブロック数を数える

↓

ブロック数が3個ならばこの状態

↓

進める方向に進む

Exercises 14

```
#動ける場所が何か所あるか
free_walk = 0
if value[__] == 2:
    free_walk = free_walk + 1
if value[__] == 2:
    free_walk = free_walk + 1
if value[__] == 2:
    free_walk = free_walk + 1
if value[__] == 2:
    free_walk = free_walk + 1
```

```
#１か所しか行くことができない
if free_walk == 3:
    if value[__] != 2:
        client.walk_up()
    elif value[__] != 2:
        client.walk_left()
    elif value[__] != 2:
        client.walk_down()
    elif value[__] != 2:
        client.walk_right()
```

Exercises 15

関数化

Exercises 16

モジュール化（オブジェクト指向）