Junioraufgabe 2: Kacheln

Team-ID: #####

Team-Name: ###

Bearbeiter/-innen dieser Aufgabe:
Snocember

23. November 2019

Inhaltsverzeichnis

Lösungsidee

Die Aufgabe ist, die unvollständige Karte zu einer vollständige Landschaft zu ergänzen. Es kommen nur Wasser und Land vor, visualisiert als 0 und 1. Dabei dürfen nur Kacheln ergänzt werden, deren Ränder zu 100% zu dem Rand des Nachbarn übereinstimmen. Die Kacheln bestehen aus vier „Landfeldern“.

Also wird die Karte in das Programm eingelesen. Dann würden alle fehlenden Kacheln, die an andere vollständige Kacheln grenzen, ergänzt nach den Regeln. Somit können große „weiße Löcher“ von außen hin geschlossen werden, damit keine Komplikationen auftreten. Dann werden wieder und wieder alle Kacheln ergänzt, die an andere vollständige Kacheln grenzen, bis die Karte vollständig gefüllt ist. Wenn eine Ergänzung nicht möglich ist, also zum Beispiel die linke obere Ecke der leeren Kachel an Wasser und Land grenzt, wird dies als Fehler ausgegeben.

Umsetzung

Die Lösungsidee wird in Python 3.7.0 implementiert. Es wurde MacOS Mojave mit Terminal.app genutzt.

Die Textdatei wird eingelesen, die Dimensionen-Kommentare oben entfernt und jede Zeile in eine Liste eingetragen in der Liste „karte“, sodass eine „2-dimensionale Liste“ entsteht. Dann wird die Karte in einer dreischachtligen Schleife ausgewertet. Es wird jedes Feld einer Kachel in der Matrix von links oben bis rechts unten begutachtet. Dann geht es schließlich darum, den Wert der leeren Kachel zu berechnen, indem man zuerst mit dem Index der Listen des Feldes herausfindet, ob das Feld in einer Ecke ist, ob es an einem Rand ist, ob es am oberen, unteren, linken oder rechten Rand einer Kachel ist. Dadurch geht das Program jedes Mal anders vor um IndexErrors zu vermeiden. Da bei der Betrachtung bei Feldern von Kachelrändern nur die angrenzenden Felder außerhalb der eigenen Kachel interessieren, werden jedes Mal nur 2 Kacheln betrachtet, in x und y-Richtung. Als Sonderfälle werden in den Kartenecken keine Kacheln betrachtet, bei Rändern nur eine Kachel. Wenn bei einem Feld die anderen angrenzenden Kacheln leer sind, wird diese Kachel übersprungen und später bearbeitet. Zur Berechnung: Wenn ein Feld an 2 Landflächen (1) grenzt, wird es auch in Land umgewandelt, das

selbe gilt bei Wasser (0). Kommt es zu einem Problem, wenn eine Land- und Wasserfläche an eine Kachel angrenzen, wird das leere Feld zu einem roten „X“.

Am Ende wird die gesamte Karte ausgefüllt ausgegeben.

Beispiele

(Die Programmausgabe hat beim Programm einen blauen (0) oder grünen (1) oder roten (x) Felder-Hintergrund. Dies wird aus Format-Gründen hier nicht gemacht.)

Das Programm ‚kacheln.py‘ wird ausgeführt.

$ python3 ./kacheln.py

```
Verfügbare Karten: 0, 1, 2, 3, 4, 5, 6
_ _ _ _ _ _ _ _ _ _ (6 ist eine eigene karte)
Ausgewählte Karte-nr: 0
dimensionen: x2y2
KARTE:
10 **
01 **

** 10
** 11

-----------------------------------
ERGEBNIS: (altes Wasser, altes Land, neu gen. Wasser, neu gen. Land)
10 00
01 10

01 10
01 11

-----------------------------------
(noch eine Ausgabe ohne Farben)
```

```
Ausgewählte Karte-nr: 1
dimensionen: x7y6
KARTE:
10 01 ** ** 11 10 01
01 10 ** ** 01 10 01

01 10 ** ** ** 10 01
11 10 ** ** ** 11 11

** ** 00 00 01 ** **
** ** 10 00 01 ** **

00 01 10 ** 01 11 **
01 11 11 ** 11 10 **

01 11 11 11 11 10 00
11 10 00 00 01 10 01

11 10 ** ** ** 10 01
01 11 ** ** ** 00 01

-----------------------------------
ERGEBNIS: (altes Wasser, altes Land, neu gen. Wasser, neu gen. Land)
10 01 10 01 11 10 01
01 10 00 00 01 10 01

01 10 00 00 01 10 01
11 10 00 00 01 11 11
```

```
11 10 00 00 01 11 11
00 01 10 00 01 11 10

00 01 10 00 01 11 10
01 11 11 11 11 10 00

01 11 11 11 11 10 00
11 10 00 00 01 10 01

11 10 00 00 01 10 01
01 11 10 00 00 00 01

_____________________________________
(noch eine Ausgabe ohne Farben)
```

```
Ausgewählte Karte-nr: 2
dimensionen: x3y4
KARTE:
** ** **
** ** **

** 00 **
** 10 **

** 10 **
** 00 **

** ** **
** ** **

_____________________________________
ERGEBNIS: (altes Wasser, altes Land, neu gen. Wasser, neu gen. Land)
00 00 00
00 00 00

00 00 00
01 10 00

01 10 00
00 00 00

00 00 00
00 00 00

_____________________________________
(noch eine Ausgabe ohne Farben)
```

```
dimensionen: x7y4
KARTE:
10 00 01 11 10 00 01
11 11 10 01 11 10 00

11 11 ** ** ** ** 00
00 01 ** ** ** ** 11

00 01 ** ** ** ** 11
11 10 ** ** ** ** 01

11 10 01 10 00 00 01
11 11 11 10 00 01 11

_____________________________________
ERGEBNIS: (altes Wasser, altes Land, neu gen. Wasser, neu gen. Land)
10 00 01 11 10 00 01
11 11 10 01 11 10 00

11 11 10 01 11 10 00
00 01 10 00 00 01 11

00 01 10 00 00 01 11
11 10 01 10 00 00 01
```

```
11 10 01 10 00 00 01
11 11 11 10 00 01 11

----------------------------------
(noch eine Ausgabe ohne Farben)
```

```
dimensionen: x16y9
KARTE:
11 10 00 00 01 10 01 11 10 01 11 10 00 01 11 11
00 01 10 01 11 10 00 01 11 11 10 00 00 01 10 01

00 01 10 01 11 10 00 01 11 11 10 00 00 01 10 01
01 11 11 11 11 11 10 00 00 00 01 10 01 11 10 00

01 11 11 11 ** ** ** 00 00 00 01 10 01 11 10 00
11 10 00 00 ** ** ** 01 10 01 11 10 01 11 10 00

11 10 00 00 ** ** ** 01 10 01 11 10 01 11 10 00
00 00 00 00 ** ** ** 11 11 11 11 11 10 00 00 00

00 00 00 00 00 01 11 11 11 11 11 11 10 00 00 00
00 00 00 00 01 11 11 11 11 10 00 00 01 11 10 01

00 00 00 00 01 11 11 11 11 10 ** ** 01 11 10 01
00 01 10 01 11 11 10 00 01 10 ** ** 11 10 00 00

00 01 10 01 ** 11 10 00 01 10 ** ** 11 10 00 00
00 01 10 01 ** 00 00 00 00 00 ** ** 11 11 11 11

00 01 10 01 ** ** 00 00 00 00 01 11 11 11 11 11
01 11 10 00 ** ** 00 00 00 01 11 11 10 00 00 00

01 11 10 00 ** ** ** 00 00 01 11 11 10 00 00 00
00 00 00 00 ** ** ** 00 00 00 00 00 00 00 00 00

----------------------------------
ERGEBNIS: (altes Wasser, altes Land, neu gen. Wasser, neu gen. Land)
11 10 00 00 01 10 01 11 10 01 11 10 00 01 11 11
00 01 10 01 11 10 00 01 11 11 10 00 00 01 10 01

00 01 10 01 11 10 00 01 11 11 10 00 00 01 10 01
01 11 11 11 11 11 10 00 00 00 01 10 01 11 10 00

01 11 11 11 11 11 10 00 00 00 01 10 01 11 10 00
11 10 00 00 00 00 00 01 10 01 11 10 01 11 10 00

11 10 00 00 00 00 00 01 10 01 11 10 01 11 10 00
00 00 00 00 00 01 11 11 11 11 11 11 10 00 00 00

00 00 00 00 00 01 11 11 11 11 11 11 10 00 00 00
00 00 00 00 01 11 11 11 11 10 00 00 01 11 10 01

00 00 00 00 01 11 11 11 11 10 00 00 01 11 10 01
00 01 10 01 11 11 10 00 01 10 00 01 11 10 00 00

00 01 10 01 11 11 10 00 01 10 00 01 11 10 00 00
00 01 10 01 10 00 00 00 00 00 01 11 11 11 11 11

00 01 10 01 10 00 00 00 00 00 01 11 11 11 11 11
01 11 10 00 00 00 00 00 00 01 11 11 10 00 00 00

01 11 10 00 00 00 00 00 00 01 11 11 10 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

----------------------------------
(noch eine Ausgabe ohne Farben)
```

```
Ausgewählte Karte-nr: 5
dimensionen: x5y5
KARTE:
11 11 11 11 11
10 00 00 01 11

10 00 00 ** 11
```

```
00 01 10 ** 00

** 01 ** 10 00
** 00 ** 00 00

** 00 01 ** 00
** 11 11 ** 11

01 11 11 11 11
11 10 00 01 11

-----------------------------------
ERGEBNIS: (altes Wasser, altes Land, neu gen. Wasser, neu gen. Land)
11 11 11 11 11
10 00 00 01 11

10 00 00 01 11
00 01 10 x0 00

00 01 1x 10 00
00 00 0x 00 00

00 00 01 x0 00
01 11 11 11 11

01 11 11 11 11
11 10 00 01 11

-----------------------------------
(noch eine Ausgabe ohne Farben)
```

```
Ausgewählte Karte-nr: 6
dimensionen: x6y6
KARTE:
11 11 11 11 11 11
1* ** 0* ** ** *1

1* *1 ** ** ** *1
1* ** ** ** ** *1

1* ** ** ** ** *1
1* ** ** ** ** *1

1* ** ** 11 ** *1
1* ** ** ** ** *1

1* ** ** ** ** *1
1* ** ** ** ** *1

1* ** ** ** ** *1
11 11 11 11 11 11


-----------------------------------
ERGEBNIS: (altes Wasser, altes Land, neu gen. Wasser, neu gen. Land)
11 11 11 11 11 11
10 0x 00 00 00 01

10 01 x0 00 00 01
10 00 00 00 00 01

10 00 00 00 00 01
10 00 01 11 10 01

10 00 01 11 10 01
10 00 00 00 00 01

10 00 00 00 00 01
10 00 00 00 00 01

10 00 00 00 00 01
11 11 11 11 11 11

-----------------------------------
```

Quellcode

```
#!/usr/bin/env python3
# ----#---- Importe, Initialisierungen und Definitionen ----#----
import sys                              # Für Programmabbruch benötigt, falls Datei nicht vorhanden.
# ascii-farben
rst = "\033[0m"
[… weitere ascii-farben (bl, grn, wss, cyn, red, glb, newbl, newgrn)]
xdim = 0
ydim = 0
karte = []                              # Initialisierungen der Karten-Matrixen
kacheln = []
kachelreihe = []
Fkacheln = []                           # für Farben
Fkachelreihe = []                       # für Farben
cKacheln = []                           # später geclonte kacheln-liste

def check(parY,parX,y,x,d):             # Definition um das Programm leichter zu machen. ...
    sf = 0 #Sternfaktor sf               # wenn parY=0, beide y-Richtungen checken; -1 = oben; 1 = unten
    nf = 0 #NullFaktor nf                # wenn parX=0, beide x-Richtungen checken; -1 = links; 1 = rechts
    ef = 0 #EinsFaktor ef                # y und x sind die aktuellen Koordinaten des weißen Feldes
    nummer = 0                           # d ist der Durchgang von „übersprungenen“, ignorierten Feldern
    if parY == -1 or parY == 0:         # Feld darüber testen
        nummer = nummer+1                       # Musterkommentar für die nächsten 3 Blöcke
        if (y-1)%2:                              # wenn Feld außerhalb der Kachel ist:
            if ckacheln[y-1][x] == "0": nf = nf+1    # wenn Feld = 0, Nullfaktor+1
            elif ckacheln[y-1][x] == "1": ef = ef+1  # wenn Feld = 1, Einsfaktor+1
            elif ckacheln[y-1][x] == "*": sf = sf+1  # wenn Feld = *, Sternfaktor+1
        else:                                    # wenn Feld innerhalb der Kachel ist:
            sf = sf+1                            # da Felder innerhalb Kachel unwichtig, Sternfaktor+1
    if parY == 1 or parY == 0:          # Feld darunter testen
        nummer = nummer+1
        if (y)%2:                                # siehe Musterkommentar
            if ckacheln[y+1][x] == "0": nf = nf+1
            elif ckacheln[y+1][x] == "1": ef = ef+1
            elif ckacheln[y+1][x] == "*": sf = sf+1
        else:
            sf = sf+1
    if parX == -1 or parX == 0:         # Feld links testen
        nummer = nummer+1
        if (x-1)%2:                              # siehe Musterkommentar
            if ckacheln[y][x-1] == "0": nf = nf+1
            elif ckacheln[y][x-1] == "1": ef = ef+1
            elif ckacheln[y][x-1] == "*": sf = sf+1
        else:
            sf = sf+1
    if parX == 1 or parX == 0:          # Feld rechts testen
        nummer = nummer+1
        if (x)%2:                                # siehe Musterkommentar
```

```
            if ckacheln[y][x+1] == "0": nf = nf+1
            elif ckacheln[y][x+1] == "1": ef = ef+1
            elif ckacheln[y][x+1] == "*": sf = sf+1
        else:
            sf = sf+1
# Auswertung der Tests
    if d == 3:                                  # wenn Feld bereits zum 3. Mal übersprungen wurde:
        return "0"                                # wird zu Wasser (0)
    if nummer == 2:
        if sf==2:                               # wenn 2 Felder gecheckt wurden und beide *-Felder sind:
          return "*"                              # Feld wird „übersprungen,“ bleibt bei weißem Feld
    elif nummer == 3:                           # wenn 3 Felder gecheckt wurden… “ “
        if sf==3: return "*"
    elif nummer == 4:                           # wenn 4 Felder gecheckt wurden… “ “
        if sf==4: return "*"
    if nf == 0 and ef>nf:                       # wenn nur Einsfaktor positiv ist:
        return "1"                                # – > Feld wird zu Land (1)
    elif ef == 0 and nf>ef:                     # wenn nur Nullfaktor positiv ist:
        return "0"                                # – > Feld wird zu Wasser (0)
    elif nf == 0 and ef == 0:                   # wenn beide Faktoren 0 sind:
        return "*"                                # Feld wird übersprungen, bleibt bei weißem Feld
    else:                                       # wenn alles nicht möglich:
        return "x"                                # Feld wird als Fehler notiert

# ----#----#---- ABFRAGE ----#----
print("Verfügbare Karten: 0, 1, 2, 3, 4, 5, 6")  # verfügbare Kartennummern
input = input("Ausgewählte Karte-nr: ")          # Abfrage des Programms welche Karte benutzt wird
try:
    # […]  Check, dass nur Zahlen von 0 - 6 eingegeben werden.
    if input == "0":
        file = open("./map_spacy.txt")
    else:
        file = open("./map"+input+"_spacy.txt")  # Einlesen der ausgewählten Karte als Datei
    content = file.read()                        # Einlesen der Zeilen der datei
except FileNotFoundError:
    [...]
    sys.exit(0)                                  # Falls Datei nicht vorhanden: sys.exit(0)

# ----#----#---- EINLESEN ----#----
karte = content.split()                          # Karte wird nach Leerzeichen aufgeteilt.
ydim = int(karte[0])                             # x-Dimensionen und ...
xdim = int(karte[1])                             # y-Dimensionen werden eingelesen ...
karte.pop(0)                                     # und entfernt aus der Karte.
karte.pop(0)
print("dimensionen: x"+str(xdim)+"y"+str(ydim))
for y in range(0, ydim*2):                       # Eine Forschleife um Liste in Karte umzuwandeln
    kachelreihe = []
    Fkachelreihe = []
    for x in range(0, xdim*2):
        kachelreihe.append(karte[0])             # Alle Reihen in die Kartenreihen eingetragen.
```

```
        if karte[0] == "0": Fkachelreihe.append(bl+karte[0]+rst)          # Eintragung in eine farbige Karte
        if karte[0] == "1": Fkachelreihe.append(grn+karte[0]+rst)
        if karte[0] == "*": Fkachelreihe.append(wss+karte[0]+rst)
        karte.pop(0)
    kacheln.append(kachelreihe)                    # Kartenreihen in Kartenmatrix eintragen
    Fkacheln.append(Fkachelreihe)
ckacheln = kacheln                                 # Kacheln werden geclont
# ----#----#---- anschaulichere Farb-vor-ausgabe   # Karte vor Bearbeiten wird gezeigt
anzahl1 = len(Fkacheln)
anzahl2 = len(Fkachelreihe)
print(red+"KARTE"+rst+":")
for k in range(0, anzahl1):
    for i in range(0, anzahl2):
        print(Fkacheln[k][i], end="")
        if i % 2: print("", end=" ")               # Um nach 2 Feldern (1 Kachel) Leerzeichen zu machen
    print()
    if k % 2: print()                              # Um nach 2 Feldern (1 Kachel) Leerzeichen zu machen

# ----#----#---- AUSWERTEN----#----
for d in range(0,3):
    us = 0 #übrige sternfelder
    for y in range(0, ydim*2):                     # 2fache For-Schleife, die die Kartenmatrix durchläuft
        for x in range(0, xdim*2):
            if kacheln[y][x] == "*":               # checkt ob das Feld * ist
                us = us+1
                if y == 0:                         # checkt ob das Feld in der ersten reihe ist
                    if x == 0:                     # checkt ob das Feld bei 0,0 sich befindet
                        ckacheln[y][x] = "0"
                        Fkacheln[y][x] = newbl+"0"+rst
                    elif x == anzahl2-1:           # checkt ob das Feld am rechten ende der reihe ist
                        ckacheln[y][x] = "0"
                        Fkacheln[y][x] = newbl+"0"+rst
                    else:                          # checkt Felder, die in der Mitte der ersten Reihe sind
                        check1 = check(1,0,y,x,d)  # Definition check() wird ausgeführt
                        if check1 == "0":
                            ckacheln[y][x] = "0"               # Musterkommentar 2
                            Fkacheln[y][x] = newbl+"0"+rst     # Je nach Rückgabewert, wird
                        elif check1 == "1":                    # das weiße Feld verändert.
                            ckacheln[y][x] = "1"                 # Bei 0 zu Wasser (0), bei 1 zu Land (1)
                            Fkacheln[y][x] = newgrn+"1"+rst      # Bei * bleibt es so,
                        elif check1 == "x":                      # Bei x wird es rot gekennzeichnet
                            ckacheln[y][x] = "x"
                            Fkacheln[y][x] = red+"x"+rst
                elif y == anzahl1-1:               # checkt, ob das Feld in der untersten reihe ist
                    if x == 0:                     # checkt ob das Feld in der linken unteren Ecke ist
                        ckacheln[y][x] = "0"
                        Fkacheln[y][x] = newbl+"0"+rst
                    elif x == anzahl2-1:           # checkt ob das Feld in der rechten unteren Ecke ist
                        ckacheln[y][x] = "0"
                        Fkacheln[y][x] = newbl+"0"+rst
```

```
            else:                                   # Alle anderen Felder in der unteren Reihe
                check1 = check(-1,0,y,x,d)          # Definition check() wird ausgeführt
                if check1 == "0":
                    ckacheln[y][x] = "0"
                    Fkacheln[y][x] = newbl+"0"+rst  # siehe Musterkommentar 2
                elif check1 == "1":
                    ckacheln[y][x] = "1"
                    Fkacheln[y][x] = newgrn+"1"+rst
                elif check1 == "x":
                    ckacheln[y][x] = "x"
                    Fkacheln[y][x] = red+"x"+rst
        else:                                       # alle anderen Möglichkeiten:
            if x == 0:                              # checkt ob das Feld am linken Rand ist
                check1 = check(0,1,y,x,d)           # Definition check() wird ausgeführt
                if check1 == "0":
                    ckacheln[y][x] = "0"
                    Fkacheln[y][x] = newbl+"0"+rst  # siehe Musterkommentar 2
                elif check1 == "1":
                    ckacheln[y][x] = "1"
                    Fkacheln[y][x] = newgrn+"1"+rst
                elif check1 == "x":
                    ckacheln[y][x] = "x"
                    Fkacheln[y][x] = red+"x"+rst
            elif x == anzahl2-1:                    # checkt ob das Feld am rechten Rand ist
                check1 = check(0,-1,y,x,d)          # Definition check() wird ausgeführt
                if check1 == "0":
                    ckacheln[y][x] = "0"
                    Fkacheln[y][x] = newbl+"0"+rst  # siehe Musterkommentar 2
                elif check1 == "1":
                    ckacheln[y][x] = "1"
                    Fkacheln[y][x] = newgrn+"1"+rst
                elif check1 == "x":
                    ckacheln[y][x] = "x"
                    Fkacheln[y][x] = red+"x"+rst
            else:                                   # alle Felder in der Mitte:
                check1 = check(0,0,y,x,d)           # Definition check() wird ausgeführt
                if check1 == "0":
                    ckacheln[y][x] = "0"
                    Fkacheln[y][x] = newbl+"0"+rst  # siehe Musterkommentar 2
                elif check1 == "1":
                    ckacheln[y][x] = "1"
                    Fkacheln[y][x] = newgrn+"1"+rst
                elif check1 == "x":
                    ckacheln[y][x] = "x"
                    Fkacheln[y][x] = red+"x"+rst

# ----#----#---- ERGEBNIS ----#----
for k in range(0, anzahl1):                         # Zum Ende wird die ergänzte Karte mit 2 For-Schleifen
    for i in range(0, anzahl2):                     # … ausgegeben
        print(Fkacheln[k][i], end="")
```

```
        if i % 2: print(" ", end="")                # Um nach 2 Feldern (1 Kachel) Leerzeichen zu machen
    print()
    if k % 2: print()                               # Um nach 2 Feldern (1 Kachel) Leerzeichen zu machen
for k in range(0, anzahl1):
    for i in range(0, anzahl2):
        print(ckacheln[k][i], end="")
        if i % 2: print(" ", end="")                # s. letzten Kommentar
    print()
    if k % 2: print()                               # s. letzten Kommentar
```