Uitwerking opdracht

Opdracht :	*dronken mier*
Weeknummer :	*4*
Studentnummer :	*s1060679*
Naam student :	*Dion van den Berg*
Specialisatie :	*MEDT*
Pogingnummer :	*1*

1. Vraagstelling

De mier wil terug komen naar zijn nest. Hoe komt hij daar ?

2. Specificatie

Invoer

De gebruiker voert een grid in met richtingen

Uitvoer

De gebruiker krijgt te zien in welke stappen hij daar kan komen.

Verband tussen in- en uitvoer

De uitvoer wordt berekend door de ingevoerde richtingen.

Beperkingen

Als er een foute string aan richtingen ingevoerd word kan het programma niets en krijg je een error message.

Voorbeelden (testscenario's)

Test 1

invoer:

```
vierkant = rooster(4, '>>>>^<^v^v^^>>v>')
print(tekst(vierkant))
```

Uitvoer:

```
vierkant
[['>', '>', '>', '>'], ['^', '<', '^', 'v'], ['^', 'v', '^', '^'], ['>', '>', 'v', '>']]

> > > >
^ < ^ v
^ v ^ ^
> > v >
```

3. Ontwerp

Hoe maak ik het rooster ?
Door het richting tekentjes mee te geven

Hoe ga ik om met foutieve strings ?
Door de assertion error af te vangen

Hoe laat ik alles uiteindelijk zien ?
in regels netjes onderverdeeld tot een vierkant ter grote van de opgegeven grote.

4. Pseudocode

```
Input een grote en een string aan richtingen ( 4, < ^ > V etc. )

Print het lijstje (lijst)
        Loop rij in lijst
                Loop cell in rij
                        Output += cell + " "

Print stap ( lijst, coordinaat )
        Vindt tekentje in de lijst via de coordinaat.
        Is tekentje gelijk aan < of > of ^ of V
                Zet stap
                Draai tekentje 90graden naar de volgende

Print stappen ( lijst )
        Begin coordinaat ingeven
        Loop tot coordinaat gelijk is aan nest ( 0, 3 )
                Stap ( lijst, coordinaat )
                Voeg coordinaat toe aan stappenlijst
```

5. Code

```
def rooster(getal: int, spoor: str):

    # check of de opgegeven string correct is
    # maak het rooster aan
    if len(spoor) % getal == 0:
        t = 0
        lijst2 = []
        for x in range(getal):
            lijst = []
            for y in range(getal):
                lijst.append(spoor[t])
                t += 1
            lijst2.append(lijst)
            x += 1
        return lijst2
    # vang de Assertion Error af
    else:
        raise AssertionError("ongeldige argumenten")
        # return print("Assertion Error: ongeldige argumenten")


def tekst(lijst: [[]]) -> str:
    # zet de lijst om naar tekst om te laten zien op meerdere regels
    output = ""
    for row in lijst:
        for cell in row:
            output += cell + " "
        output += "\n"
    return output


def stap(lijst: [[]], coordinaat: tuple):

    coX = coordinaat[0]
    coY = coordinaat[1]
    lijstX = lijst[coordinaat[0]]
    tekentje = lijstX[coordinaat[1]]
    array = ['v', '<', '^', '>']
    index = 0
    for x in range(len(array)):
        if tekentje == array[x]:
            index = x
    newCoordinaat = (0, 0)

    # ga 1 stap verder
    # en verander icoontje 90 graden met klok mee
    if tekentje == 'v' and coX < len(lijst)-1:
        newCoordinaat = (coX + 1, coY)
        lijstX[coordinaat[1]] = '<'
    elif tekentje == '^'and coX > 0:
        newCoordinaat = (coX - 1, coY)
        lijstX[coordinaat[1]] = '>'
    elif tekentje == '>' and coY < len(lijst)-1:
        newCoordinaat = (coX, coY + 1)
        lijstX[coordinaat[1]] = 'v'
    elif tekentje == '<' and coY > 0:
        newCoordinaat = (coX, coY - 1)
        lijstX[coordinaat[1]] = '^'
```

```
 # als hij out of bounds wil gaan blijf op dezelfde plek
    else:
        newCoordinaat = (coX, coY)
        if index < 3:
            lijstX[coordinaat[1]] = array[index + 1]
        else:
            lijstX[coordinaat[1]] = array[0]

    return newCoordinaat


def stappen(lijst: [[]]):
    # probeer naar het nest op 3,0 te komen.
    coordinaat = stap(lijst, (3, 0))
    stappenlijst = [(3, 0)]
    while coordinaat != (0, 3):
        stappenlijst.append(coordinaat)
        coordinaat = stap(lijst, coordinaat)
    stappenlijst.append(coordinaat)
    return stappenlijst


def main() -> None:
    # zoals het voorbeeld
    vierkant = rooster(4, '>>>>^<^v^v^^>>v>')
    print(tekst(vierkant))
    print(stap(vierkant, (3, 0)))
    print(tekst(vierkant))
    print(stap(vierkant, (3, 1)))
    print(tekst(vierkant))

    vierkant = rooster(4, '>>>>^<^v^v^^>>v>')
    print(tekst(vierkant))
    print(stappen(vierkant))
    print(tekst(vierkant))

    rooster(4, '>>>>^<^v^v^>>v>')

try:
    main()
except AssertionError:
    print("Ongeldige argumenten")
```

6. Test

Test 1

> > > >
^ < ^ v
^ v ^ ^
> > v >

(3, 1)
> > > >
^ < ^ v
^ v ^ ^
v > v >

(3, 2)
> > > >
^ < ^ v
^ v ^ ^
v v v >

> > > >
^ < ^ v
^ v ^ ^
> > v >

[(3, 0), (3, 1), (3, 2), (3, 2), (3, 1), (3, 1), (3, 0), (3, 0), (3, 0), (2, 0), (1, 0), (0, 0), (0, 1), (0, 2), (0, 3)]
v v v >
> < ^ v
> v ^ ^
> ^ ^ >

Assertion Error: ongeldige argumenten