Inleiding Programmeren

Examen Python

1B Informatica,
Academiejaar 2024-2025

Tom Hofkens
tom.hofkens@uantwerpen.be

Tim Apers
tim.apers@uantwerpen.be

15 januari 2025

Belangrijke informatie

- Maak deze opdracht individueel. Het gebruik van communicatiemiddelen zoals smartphones en smartwatches is niet toegestaan. Internet mag enkel gebruikt worden om informatie zoals Python documentatie op te zoeken. Kopiëren van online beschikbare code is strikt niet toegestaan. Dit omvat ook het gebruik van tools die op basis van een beschrijving code genereren, bestaande code uitbreiden, of commentaren toevoegen.
- Je hebt **2u** tijd om het examen op PC te maken.
- Voorzie elk bestand dat je aanmaakt of aanpast van een hoofding die jouw naam en studentnummer bevat.
- Je mag gebruik maken van de documentatie die via Inginious beschikbaar is.
- Dien je praktijk-oplossing in via INGinious.
- Er zijn 6 vragen. Deze vragen zijn onafhankelijk van elkaar op te lossen. Het totaal aantal punten telt op tot 22 en met programmeerstijl kan jouw resultaat tot 2 punten beïnvloeden in zowel positieve als negatieve zin. De som van de punten op alle delen die je correct oploste is jouw definitieve punt, tenzij dat hoger is dan 20. In dat geval behaal je 20/20.

Inleiding

Lees zorgvuldig onderstaande beschrijving voordat je aan het examen begint. Een goed begrip van de code waarvan je start is essentieel!

We gaan in deze opdracht het spelletje "2048" implementeren. 2048 is een puzzelspel dat gespeeld wordt op een veld met 4 rijen en 4 kolommen. Elke cel van het veld kan ofwel leeg zijn, ofwel een getal bevatten dat een macht van 2 is (bijvoorbeeld 2, 4, 8, enz.). Het doel van het spel is om door het samenvoegen van gelijke getallen een tegel met de waarde 2048 te verkrijgen. Een mogelijke opstelling van het spelbord kan er bijvoorbeeld als volgt uitzien:

Je speelt 2048 door telkens het bord in één van de vier richtingen te schuiven: omhoog, omlaag, naar links, of naar rechts. Bij elke schuifbeweging worden de volgende regels gevolgd:

1. Alle tegels schuiven in de gekozen richting zo ver mogelijk op, totdat ze een andere tegel of de rand van het bord tegenkomen.
2. Wanneer twee tegels met dezelfde waarde tegen elkaar schuiven, worden ze samengevoegd tot één tegel waarvan de waarde de som van de oorspronkelijke tegels is.
3. Bij elke schuifbeweging verschijnt er een nieuwe tegel met waarde 2 op een willekeurige lege plaats op het bord.
4. Het spel eindigt wanneer er geen geldige schuifbewegingen meer mogelijk zijn.

Door deze regels te gebruiken, kunnen we strategisch proberen het bord te manipuleren. Bijvoorbeeld: als er in een rij twee naast elkaar liggende tegels met waarde 4 staan, kunnen we door in de juiste richting te schuiven een tegel met waarde 8 creëren. Op die manier kunnen we het spel verder spelen.

Bestanden

Bij de opdracht heb je volgende bestanden gekregen:

- `main.py` : dit bestand bevat code die je verder moet uitbreiden.

Controleer voordat je begint dat je alle bestanden hebt en dat je het bestand `main.py` kan uitvoeren.

Opdracht (op 20)

1. Code uitbreiden (2pt)

Breid de code uit om volgende functionaliteit toe te voegen. Zie onder de vraag een voorbeeld van de uitvoer.

(a) Schrijf een functie `getRow` die gegeven een rijnummer een lijst teruggeeft met de elementen in die rij.

(b) Schrijf een functie `getColumn` die gegeven een kolomnummer een lijst teruggeeft met de elementen in die kolom.

(c) Schrijf een functie `setRow` die gegeven een rijnummer en een lijst, de elementen in die rij vervangt door de elementen in die lijst.

(d) Schrijf een functie `setColumn` die gegeven een kolomnummer en een lijst, de elementen in die kolom vervangt door de elementen in die lijst.

Voorbeeld:

Volgend stukje code geeft onderstaande output:

```
game = Game(4)
game.setRow([4, 16, 64, 2], 0)
game.setRow([2, 16, 8, 0], 1)
game.setRow([4, 0, 0, 2], 2)
game.setRow([2, 0, 0, 0], 3)

for i in range(4):
  print(game.getRow(i))

game.setColumn([4, 2, 4, 2], 0)
game.setColumn([16, 16, 0, 0], 1)
game.setColumn([64, 8, 0, 0], 2)
game.setColumn([2, 0, 2, 0], 3)

for i in range(4):
  print(game.getColumn(i))
```

De output is als volgt:

```
[4, 16, 64, 2]
[2, 16, 8, 0]
[4, 0, 0, 2]
[2, 0, 0, 0]
[4, 2, 4, 2]
[16, 16, 0, 0]
[64, 8, 0, 0]
[2, 0, 2, 0]
```

2. Vul het bord op (2pt)

Het doel van deze opdracht is om het bord op te vullen met de waarde 2 door gebruik te maken van de methode `spawn`. Zorg ervoor dat de methode `spawn` een willekeurige lege cel selecteert en daar een nieuwe tegel met waarde 2 plaatst. Maak hiervoor gebruik van de `random.randint` functie van de `random` package. Het bord begint leeg en na 16 keer `spawn` op te roepen wordt deze volledig opgevuld:

```
game = Game(4)
for i in range(16):
  game.spawn()
game.printBoard()
```

De output is als volgt:

```
[2, 2, 2, 2]
[2, 2, 2, 2]
[2, 2, 2, 2]
[2, 2, 2, 2]
--------------------
```

3. Functies voor manipulatie van rijen (5pt)

In dit deel implementeren we drie functies die rijen manipuleren. Deze functies worden gebruikt om de logica van het spel te ondersteunen.

(a) Voeg een functie `reverse` toe die de volgorde van de elementen in een lijst omkeert.

Voorbeeld:

Volgend stukje code geeft onderstaande output:

```
game = Game(4)
print(game.reverse([4, 2, 4, 2]))
print(game.reverse([16, 16, 0, 0]))
print(game.reverse([64, 8, 0, 0]))
print(game.reverse([2, 0, 2, 0]))
```

De output is als volgt:

```
[2, 4, 2, 4]
[0, 0, 16, 16]
[0, 0, 8, 64]
[0, 2, 0, 2]
```

(b) Implementeer de functie `move1` die alle getallen in een lijst verplaatst naar links. Hierdoor staan alle nullen aan de rechterkant.

Voorbeeld:

Volgend stukje code geeft onderstaande output:

```
game = Game(4)
print(game.move1([4, 2, 2, 4]))
print(game.move1([16, 0, 0, 16]))
print(game.move1([0, 0, 64, 8]))
print(game.move1([2, 0, 2, 0]))
```

De output is als volgt:

```
[4, 2, 2, 4]
[16, 16, 0, 0]
[64, 8, 0, 0]
[2, 2, 0, 0]
```

(c) Tot slot, breid de functie `move2` uit zodat naast elkaar liggende gelijke getallen in een lijst correct worden samengevoegd. Wanneer twee gelijke getallen naast elkaar staan, worden ze vervangen door hun som, die gelijk is aan het getal vermenigvuldigd met 2. De positie van het tweede getal, dat is samengevoegd, wordt vervolgens opgevuld met een 0 om de lijst consistent te houden.

Voorbeeld:

Volgend stukje code geeft onderstaande output:

```
game = Game(4)
print(game.move2([4, 2, 4, 2]))
print(game.move2([16, 16, 0, 0]))
print(game.move2([64, 8, 0, 8]))
print(game.move2([2, 2, 2, 2]))
print(game.move2([0, 0, 2, 2]))
```

De output is als volgt:

```
[4, 2, 4, 2]
[32, 0, 0, 0]
[64, 8, 0, 8]
[4, 0, 4, 0]
[0, 0, 4, 0]
```

4. Beweeg een rij in een bepaalde richting (3pt)

Implementeer de functie `move` die een rij of kolom van het spelbord in een opgegeven richting (direction) manipuleert volgens de spelregels. Het 'direction' argument komt overeen met een booleaanse waarde die aangeeft in welke richting de lijst bewogen wordt. False komt overeen met het verplaatsen naar links en True naar rechts.

Voor het implementeren van deze opgave gebruik je best de functies van het vorige deel. Zo maak je bijvoorbeeld sequentieel gebruik van `move1` en `move2` en gebruik je de `reverse` functie om rechtse bewegingen uit te voeren. Het is dan aan jou om met behulp van de vorige functies een juiste logica te definiëren.

Voorbeeld:

Volgend stukje code geeft onderstaande output:

```
game = Game(4)
for d in [False, True]:
    print(game.move([4, 2, 4, 2], d))
    print(game.move([16, 16, 0, 0], d))
    print(game.move([64, 8, 0, 0], d))
    print(game.move([2, 0, 2, 0], d))
    print(game.move([2, 2, 2, 2], d))
    print(game.move([0, 0, 2, 2], d))
```

De output is als volgt:

```
[4, 2, 4, 2]
[32, 0, 0, 0]
[64, 8, 0, 0]
[4, 0, 0, 0]
[4, 4, 0, 0]
[4, 0, 0, 0]
[4, 2, 4, 2]
[0, 0, 0, 32]
[0, 0, 64, 8]
[0, 0, 0, 4]
[0, 0, 4, 4]
[0, 0, 0, 4]
```

5. Beweeg het bord in een bepaalde richting (4pt)

Implementeer methoden om het volledige spelbord in een specifieke richting te bewegen. Deze methodes zijn:

(a) `moveUp`: Beweeg alle kolommen van het bord naar boven.

(b) `moveDown`: Beweeg alle kolommen van het bord naar beneden.

(c) `moveLeft`: Beweeg alle rijen van het bord naar links.

(d) `moveRight`: Beweeg alle rijen van het bord naar rechts.

Voorbeeld:

Volgend stukje code geeft onderstaande output:

```
game = Game(4)
game.setRow([4, 16, 64, 8], 0)
game.setRow([2, 16, 8, 0], 1)
game.setRow([2, 0, 0, 8], 2)
game.setRow([4, 0, 0, 0], 3)
game.moveUp()
game.printBoard()
game.moveRight()
game.printBoard()
game.moveDown()
game.printBoard()
game.moveLeft()
game.printBoard()
game.moveRight()
game.printBoard()
game.moveDown()
game.moveDown()
game.printBoard()
game.moveLeft()
game.moveRight()
game.moveUp()
game.printBoard()
```

De output is als volgt:

```
[4, 32, 64, 16]
[4, 0, 8, 0]
[4, 0, 0, 0]
[0, 0, 0, 0]
--------------------
[4, 32, 64, 16]
[0, 0, 4, 8]
[0, 0, 0, 4]
[0, 0, 0, 0]
--------------------
[0, 0, 0, 0]
[0, 0, 0, 16]
[0, 0, 64, 8]
[4, 32, 4, 4]
--------------------
[0, 0, 0, 0]
[16, 0, 0, 0]
[64, 8, 0, 0]
[4, 32, 8, 0]
--------------------
[0, 0, 0, 0]
[0, 0, 0, 16]
[0, 0, 64, 8]
[0, 4, 32, 8]
```

```
--------------------
[0, 0, 0, 0]
[0, 0, 0, 0]
[0, 0, 64, 0]
[0, 4, 32, 32]
--------------------
[0, 0, 4, 128]
[0, 0, 0, 0]
[0, 0, 0, 0]
[0, 0, 0, 0]
--------------------
```

6. Verloren en gewonnen (4pt)

In het laatste onderdeel zullen we definiëren wanneer het spel eindigt. Vul de functies `isSolved` en `isLost` aan

(a) `isSolved`: Retourneert `True` als het getal 2048 aanwezig is op het bord. Dit betekent dat de speler het spel heeft gewonnen. Anders geef je `False` terug.

(b) `isLost`: De functie controleert of er geen mogelijke zetten meer zijn. Dit is het geval wanneer er geen lege vakjes (0) op het bord zijn en er geen twee aangrenzende cellen in een rij of kolom gelijke waarden bevatten die gecombineerd kunnen worden. Als er geen geldige zetten mogelijk zijn, retourneert de functie `True` (spel verloren), anders `False`.

Voorbeeld:

Volgend stukje code geeft onderstaande output:

```
game = Game(4)
print(game.isSolved())
print(game.isLost())
game.setRow([2048, 0, 0, 0], 0)
print(game.isSolved())
print(game.isLost())
game.setRow([2, 4, 8, 16], 0)
game.setRow([16, 8, 4, 2], 1)
game.setRow([2, 4, 8, 16], 2)
game.setRow([16, 8, 4, 2], 3)
print(game.isSolved())
print(game.isLost())
game.setRow([16, 8, 4, 16], 1)
print(game.isLost())
```

De output is als volgt:

```
False
False
True
False
False
True
False
```

6. Programmeerstijl (+/-2pt)

Naast de opdrachten kan je ook 2 punten met jouw programmeerstijl verdienen of verliezen; dit houdt onder andere in: correct gebruik van globale variabelen, duidelijke, leesbare en goed gedocumenteerde code, gestructureerd werken zoals doordacht gebruik maken van functies waar nodig, de juiste data structuren gebruiken, onnodig diep geneste code vermijden en in mindere mate de efficiëntie van jouw code.