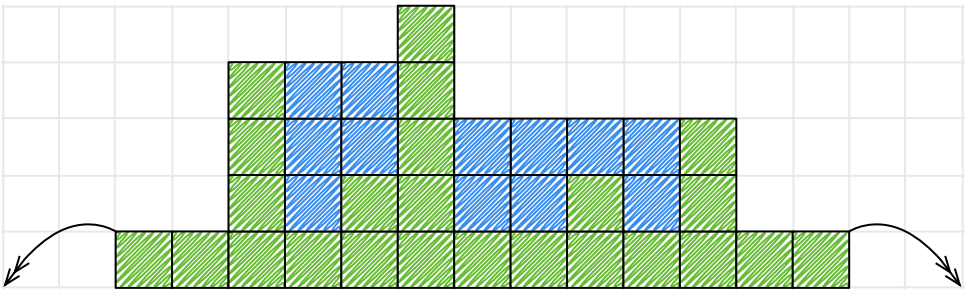
Inleiding tot de Computerwetenschappen
Taak “Wetrix” – Deel 1

Inleiding

Het doel van deze taak is om een eigen versie van het spel “Wetrix” te maken. Wetrix lijkt op het klassieke tetris. Blokken vallen vallen naar beneden en je moet ze positioneren op het speelveld. Het doel van het spel is dijken bouwen die het speelveld omsluiten. Het speelveld stelt een stuk van de aardbodem voor, dat in het begin vlak is. Er zijn verschillende soorten blokken, met elk een eigen functie. De basisblokken zijn “uppers”, “downers” en waterblokken. Als een upper op het speelveld beland wordt de bodem op die plaats opgehoogd. Een downer doet het omgekeerde en verlaagd de bodem. Een waterblok laat water vallen op de bodem. Waar geen dijken voorzien zijn, stroomt het water van het speelveld af. Tussen dijken worden het water vastgehouden. Het spel eindigt wanneer te veel water verloren is gegaan.

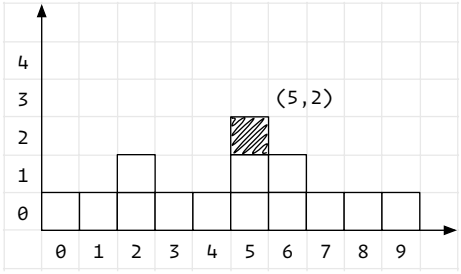
Via volgende links kan je een tutorial en de gameplay van het spel bekijken. Meer video’s kan je uiteraard terugvinden via Google. Het originele spel werkt met een driedimensionaal speelveld. Wij bouwen een vereenvoudigde versie in twee dimensies, waarbij het water links en rechts van het speelveld wegstroomt.



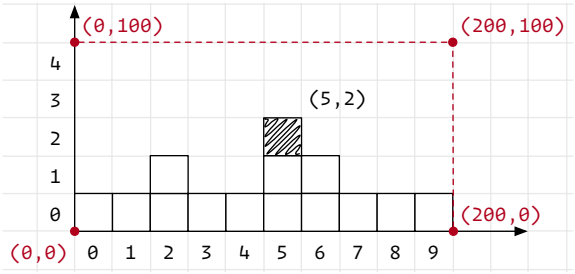
De taak bestaat uit twee delen. In deel 1 zullen we een aantal voorbereidende functies implementeren. In deel 2 zullen we het echte spel bouwen. De functies uit deel 1 kan je hergebruiken in deel 2 of helpen je om deel 2 te bouwen.

Conversie rooster – coördinatenstelsel

Om het spel te bouwen gebruiken we een roostervoorstelling. Het rooster laat ons toe op eenvoudige wijze blokken te plaatsen in het spel en ze te laten bewegen. Hieronder zie je een voorbeeld van een rooster met het speelveld. Het ingekleurde blok staat op positie `(5,2)` in het rooster. Als we een blok willen laten bewegen verplaatsen we het één positie naar onder, boven, links of rechts in het rooster.



In deel 2 van de taak zullen de grafische voorstelling uitbouwen van het spel. Daarbij maken we gebruik van de *VPyhon* module. Deze module maakt gebruik van een carthesisch coördinatenstelsel om visuele elementen te positioneren op het scherm. We hebben dus een manier nodig om posities in het rooster om te rekenen naar posities in het coördinatenstelsel, en vice versa. Hieronder zie je een voorbeeld van een rooster van 10 vakken breed en 5 hoog. Het rooster is aangebracht op een coördinatenstelsel van 200 breed op 100 hoog. Een vak in het rooster is vierkant, elk vak in dit rooster heeft dus breedte en hoogte 20 in het coördinatenstelsel.



Schrijf volgende functies:

- De functie `rooster_naar_coord(vak_grootte, rooster_pos)` neemt als invoer de grootte van een vak in het rooster en een tupel, dat een positie in het rooster voorstelt. Het resultaat van de functie is een tupel dat de positie van het midden van het gegeven vak aangeeft in het coördinatenstelsel.
- De functie `coord_naar_rooster(vak_grootte, coord_pos)` neemt als invoer de grootte van een vak in het rooster en een tupel, dat een coördinaat in het stelsel voorstelt. Het resultaat van de functie is een tupel dat de roosterpositie voorstelt van het vak waarin de coördinaat ligt.

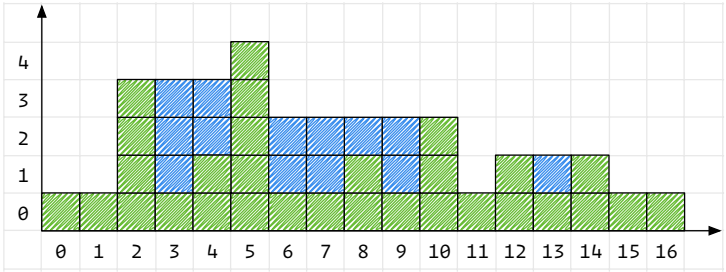
Let op: roosterposities worden voorgesteld door integers, coördinaten worden voorgesteld door floats.

Beschouw volgende voorbeelden:

```
>>> rooster_naar_coord(20, (5,2))
(110.0, 50.0)
>>> coord_naar_rooster(20, (85.7, 34.10))
(4, 1)
```

Meren

Het doel van het spel is dijken bouwen zodat zo weinig mogelijk water afloopt van het speelveld. Dat doen we door dijken te bouwen die het water vasthouden. Een aaneengesloten watermassa tussen dijken noemen we een meer. Hieronder vind je een voorbeeld van een speelveld met 3 meren.



We kunnen het afgebeelde speelveld voorstellen door twee lijsten. De eerste lijst bevat de hoogtes van de bodem, per horizontale positie op het speelveld. De tweede lijst bevat de waterniveau's. Het speelveld hierboven wordt dus gegeven door de volgende lijsten:

```
bodemhoogtes = [1, 1, 4, 1, 2, 5, 1, 1, 2, 1, 3, 1, 2, 1, 2, 1, 1]
waterniveaus = [0, 0, 0, 3, 2, 0, 2, 2, 1, 2, 0, 0, 0, 1, 0, 0, 0]
```

Schrijf volgende functies:

Schrijf volgende functies:

- De functie `aantal_meren(bodemhoogtes, waterniveaus)` neemt twee lijsten als invoer. De eerste lijst bevat de hoogtes van de bodem voor elke horizontale positie in het rooster. De tweede lijst bevat de waterniveaus voor elke horizontale positie in het rooster. Het resultaat van de functie is het aantal meren op het speelveld.
- Het predicaat `is_meer(i, bodemhoogtes, waterniveaus)` neemt een index en de lijsten met bodemhoogtes en waterniveaus als invoer. De functie geeft aan of er zich een meer bevindt op (boven) de horizontale index i van het rooster .
- De functie `start_eind_meer(i, bodemhoogtes, waterniveaus)` neemt een index en de lijsten met bodemhoogtes en waterniveaus als invoer. Het resultaat van de functie is een tupel met de horizontale start en eindindex van het meer rond (boven) index i.
- De functie `verdamp(i, bodemhoogtes, waterniveaus)` neemt een index en de lijsten met bodemhoogtes en waterniveaus als invoer. Het resultaat past de lijst met waterniveaus destructief aan zodat het meer rond (boven) index i verdwenen is.

Beschouw volgende voorbeelden:

```
>>> aantal_meren(bodemhoogtes, waterniveaus)
3
>>> is_meer(8, bodemhoogtes, waterniveaus)
True
>>> is_meer(11, bodemhoogtes, waterniveaus)
False
>>> start_eind_meer(8, bodemhoogtes, waterniveaus)
(6, 9)
>>> verdamp(6, bodemhoogtes, waterniveaus)
>>> waterniveaus
[0, 0, 0, 3, 2, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
```

Nivelleren van het waterniveau

Tijdens het spel komt er regelmatig water op het speelveld terecht. Het water wordt vastgehouden door dijken. Water dat niet wordt vastgehouden stroomt links en rechts van het speelveld af. Op bepaalde momenten tijdens het spel zullen we het waterniveau moeten nivelleren en het overtollige water laten afstromen.

Het proces om dat te doen gaat als volgt.

1) We verplaatsen eerst zoveel mogelijk water naar links:
    - Loop van *links naar rechts* door de horizontale posities in het rooster.
    - Waar er een *negatief drukverschil* is naar de linkerkant verplaats je één eenheid water één positie naar links.
    - Herhaal dit proces tot er geen negatieve drukverschillen naar links meer bestaan.
2) Hierna verplaatsen we zoveel mogelijk overgebleven water naar rechts op analoge wijze:
    - Loop van *rechts naar links* door de horizontale posities in het rooster.
    - Waar er een *negatief drukverschil* is naar de rechterkant verplaats je één eenheid water één positie naar rechts.
    - Herhaal dit proces tot er geen negatieve drukverschillen naar rechts meer bestaan.

Telkens je een eenheid water links of rechts *naast* het speelveld verplaatst geldt dat als "afstromen van het speelveld".

We definiëren de *drukverschillen* links dv_i^l (1) en rechts dv_i^r (2) als volgt:

$$dv_i^l = W_i \times hv_{i-1,i} \tag{1}$$
$$dv_i^r = W_i \times hv_{i+1,i} \tag{2}$$

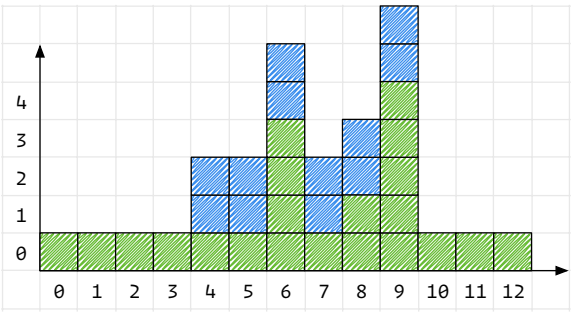
Het *hoogteverschil* $hv_{i,j}$ (3) is gegeven als volgt:

$$hv_{i,j} = (B_i + W_i) - (B_j + W_j) \tag{3}$$

Hierbij zijn:

- i en j horizontale indices in het rooster
- B_i de hoogte van de bodem op positie i
- W_i het waterniveau op positie i.

Beschouw het voorbeeld hieronder voor het afgebeelde rooster. We zien dat er een negatief drukverschil bestaat naar links voor index 8 en een positief drukverschil naar rechts.



$$
\begin{aligned}
hv_{7,8} &= (1+2) - (2+2) \\
&= 3 - 4 \\
&= -1 \\
dv_8^l &= 2 \times hv_{7,8} \\
&= 2 \times -1 \\
&= -2 \\
hv_{9,8} &= (5+2) - (2+2) \\
&= 7 - 4 \\
&= 3 \\
dv_8^r &= 2 \times hv_{9,8} \\
&= 2 \times 3 \\
&= 6
\end{aligned}
$$

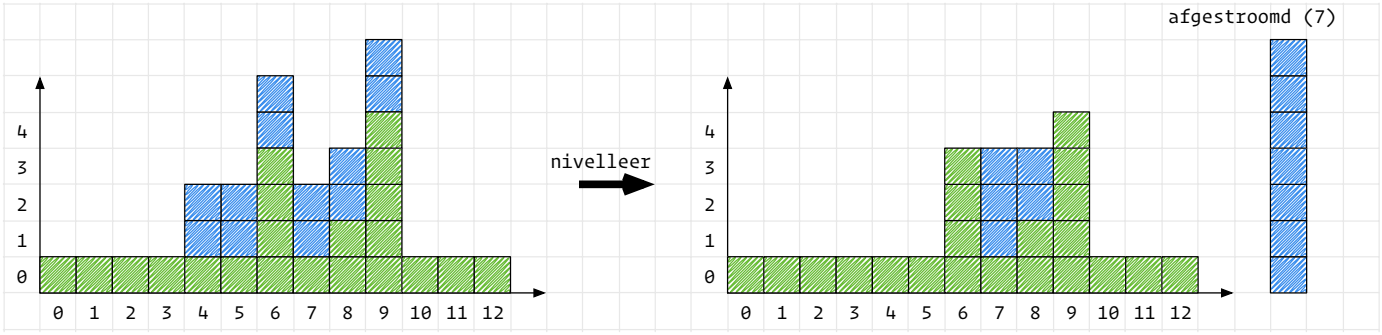
Schrijf volgende functies volgens de definities hierboven:

- De functie `hoogteverschil(i, j, bodemhoogtes, waterniveaus)` neemt twee indices en twee lijsten als invoer. De indices i en j bepalen twee horizontale posities in het rooster. De eerste lijst bevat de hoogtes van de bodem voor elke horizontale positie in het rooster. De tweede lijst bevat de waterniveaus voor elke horizontale positie in het rooster. Het resultaat van de functie is het totale hoogteverschil op deze posities.
- De functies `drukverschil_links(i, bodemhoogtes, waterniveaus)` en `drukverschil_rechts(i, bodemhoogtes,` nemen een index en de lijsten met bodemhoogtes en waterniveaus als invoer. Het resultaat is respectievelijk het drukverschil naar links en rechts op positie i.
- De predicaten `nivellering_links_mogelijk(bodemhoogtes, waterniveaus)` en `nivellering_rechts_mogelijk(bo` neemt de lijsten met bodemhoogtes en waterniveaus als invoer. Het resultaat geeft aan of we nog verder kunnen nivelleren, respectievelijk naar links en rechts.
- De functie `nivelleer(bodemhoogtes, waterniveaus)` neemt de lijsten met bodemhoogtes en waterniveaus als invoer. Deze functie past de lijst met waterniveaus destructief aan zodat de waterniveaus genivelleerd zijn volgens bovenstaand proces. Het resultaat van de functie is het aantal eenheden water die weggevloeid zijn (t.t.z., links en rechts van het speelveld zijn verplaatst).

Beschouw volgende voorbeelden:

```
>>> bodem = [1, 1, 1, 1, 1, 1, 4, 1, 2, 5, 1, 1, 1]
>>> water = [0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 0, 0, 0]
>>> hoogteverschil(7, 8, bodem, water)
-1
>>> drukverschil_links(8, bodem, water)
-2
>>> hoogteverschil(9, 8, bodem, water)
3
>>> drukverschil_rechts(8, bodem, water)
6
>>> nivellering_links_mogelijk(bodem, water)
True
>>> nivelleer(bodem, water)
7
>>> water
[0, 0, 0, 0, 0, 0, 0, 3, 2, 0, 0, 0, 0]
```

Het resultaat na nivellering is dus:



Praktische informatie

Tenslotte volgt hier nog de praktische informatie over de taak.

- Je moet de taak **individueel** oplossen! Samenwerken is **niet** toegelaten. Uitwisselen van code wordt als **plagiaat** beschouwd!
- Er is geen aparte deadline om deel 1 van de taak in te dienen. Dien deze samen in met deel 2.
- De deadline is **zondag 6 januari 2019 om 23u59**.
- Verdere instructies voor het indienen vind je bij deel 2 van de taak.
- Nuttige commentaar[1] in je code is niet verplicht maar wordt aangemoedigd.
- De keuzes die gemaakt werden tijdens de uitvoering van de taak zullen mondeling verdedigd worden tijdens de zittijd in januari.
- De assistenten zijn bereikbaar voor vragen via e-mail: icw@dinf.vub.ac.be
- Jullie kunnen ook steeds terecht bij de studiebegeleider van de computerwetenschappen, Marjon Blondeel. Via e-mail: mblondee@vub.ac.be
- Wacht niet te lang om aan je taak te beginnen!

[1] Als je commentaar toevoegt aan je code, zorg dan dat ze een toegevoegde waarde heeft. Commentaar is dus **geen tekstuele vertaling** van je code.