Integratie Externe Functionaliteiten
Les 2 : NumPy indexing, slicing, copy vs.view

We weten uit vorige les hoe we een multidimensionale array aanmaken. We kunnen deze waarden nu ook opvragen. Dit kon voor 1 specifieke waarde op een specifieke plek of ineens een deel van de array.

Indexeren & Slicing

1D array

Indexering

We kunnen via indexering waarden van een 1D array opvragen zoals we dat gewoon zijn van lists en tuples.

We spreken onze variabele met de array in aan, gevolgd door vierkante haken met het index nummer. Let op deze beginnen vanaf 0.

Maak een nieuw Python bestand.
Maak automatisch een nieuwe 1D array met 12 waarden aan.

Vraag het eerste en de derde waarde op.
Print deze afzonderlijk af.

Slicing

Via slicing vragen we dan weer een heel deel van de array op.
Aan de variabele met de array voegen we tussen de vierkante haken dan 2 getallen toe gescheiden door een dubbele punt. Als we vanaf het begin of tot op het einde willen nemen hoeven we geen indexering in te vullen.

Let op!
Inclusief het eerste, exclusief het tweede getal.

Print uit de vorige 1D array een nieuwe array af van de 3de waarde tot en met de 7de waarde.

Print de eerste 5 waardes af.

Print alle waarden beginnende met 10 af.

Multidimensionale array

Indexering

Als we uit een uit een array met meerde lagen/dimensies een waarde willen opvragen moeten we een pad aanleggen. Deze weg leidt ons naar de gewilde waarde. We onderscheiden dan de dimensies door komma's.
We beginnen vanaf de buitenste laag.

Hervorm de vorige array om een gelijk verdeelde 2D array en sla dit op in een nieuwe variabele.

Print deze af om te controleren of het gelukt is.

Print de 3de waarde uit de 2de array af.

Hervorm de 1D array om naar een 3D array met de vorm 2,2,3.

Print deze af.

Print uit de 2de waarde uit de 1ste array van de 2de array.

Slicing

Bij een multidimensionale array kunnen we een stuk van een bepaalde array opvragen. Dan leggen we eerst een pad naar die array, zoals bij indexering met een komma, en vragen van deze array een 'slice' op.
Hier gebruiken we de dubbele punt tussen de index nummers.

Print uit de 2D array een middelste 2 waarden van de 2de array af.

Print uit de 3D array de eerste 2 waarden van de 1ste array uit de 2de array af.

Een multidimensionale array laat ook toe om van 'dezelfde' array steeds hetzelfde stukje of losse waarde te nemen. We kunnen dus bij elke laag de slice notatie met dubbele punt gebruiken.

Print uit beide array's van de 2D array de laatste waarde af.

Print uit beide array's van de 2D array een slice af van de 2de waarde tot en met de 4de waarde.

Print uit alle binnenste array's van de 3D array de eerste waarde.

Hoe wordt dit weergegeven?
Pas deze weergave aan zodat het wordt weergegeven als de originele array.

Print uit de eerste array van beide arrays middelste waarde af.

Slicing met stappen

Bij he slicen moeten we niet alle waarden nemen, maar kunnen we ook de grootte van de stap instellen. We voegen dan een derde waarde toe. Deze onderscheiden we nogmaals met een dubbele punt.

Print alle even getallen uit de 1D array, inclusief 0.

Negatieve index en slice

Om achteraan aan de array te beginnen tellen gebruiken we negatieve waarden. Zowel bij de indexering en het slicen.

Let op!

Hier kunnen we niet vanaf 0 beginnen. We beginnen dus vanaf -1.

Oefenen en herhaling:

Oefn1.

Maak een automatische 'matrix' van 8x8 met alleen nullen aan.

Maak hier nu een dambord patroon met eentjes.

Oefn2.

Maak een automatische 'matrix' van 4x4 met alleen eentjes aan.

Pas de matrix zo aan dat het centrum nullen worden en alleen aan de rand eentjes overblijven.

View vs. Copy

Soms willen we een array aanpassen, maar de originele waarden behouden.

Dan maken we best een kopie. Rechtstreeks overhevelen in een nieuwe variabelen gaat standaard een nieuw view, geen kopie. Als we dan een waarde zouden veranderen in de nieuwe array, wordt die waarde ook aangepast in de originele.

Probeer zelf even uit:

Maak een nieuwe 1D array met getallen. [1,10,100,1000,10000]
Maak een tweede variabele nieuwe_getallen waarin de eerste variabele zit.

Print ze beiden af.

Verander nu in de nieuwe array 1000 door 999 met behulp van indexering.

Print beide variabelen nog eens af.

Wat merken we op?

Eigenlijk hebben we gewoon een pad naar de array opgegeven.

We gaan dit anders doen.

Als we ***.copy()*** toevoegen maken we een echte kopie.
Deze past de waarden van de originele array niet aan.

Als we ***.view()*** toevoegen maken we een dubbel beeld.
Deze past de waarden van de originele array wel aan.

nieuwe_getallen = getallen.copy()

of

nieuwe_getallen = getallen.view()

Nu is het mogelijk voor ons om te kijken of een array een originele of een kopie is.
NumPy heeft het attribuut ***base***.
Als de array origineel, of een view, is dan zal er **none** worden teruggegeven.
Is de array een kopie dan zal de array worden weergegeven.

```
print(getallen.base)
print(nieuwe_getallen.base)
```

Wanneer dan een view?
Een view gebruiken we al heel de tijd. Een slice is slechts een view van de originele array.

Neem een slice van de 3D array en slaag deze op in een nieuwe variabele, arrDeel.

Print deze af.

Vraag de base van arrDeel op.

Verander het de eerste waarde van de eerste array in de eerste array.

Vraag de base op of print de originele 3D array af.

Is deze array eigenlijk wel de originele?

Hoe zouden we de 3D array wel een originele kunnen maken?

Oefn.

Oefn 3

Maak een automatische 4x4 matrix met 16 waarden.

Print hiervan afzonderlijk af:

- De 2de rij.
- De 3de kolom.
- De laatste 2 waarden van rij 1 en 2.
- De 1ste en 3de waarden van de 1ste en 3de rij.
- Een nieuwe matrix waar de 1ste en 2de kolom van plek veranderen.