



Pythoncursus

week 2

cs.ru.nl/pythoncursus

Algoritmes

- “Een algoritme is een eindige reeks instructies die vanuit een gegeven begintoestand naar een beoogd doel leiden.” - Wikipedia
- Een probleem stapsgewijs oplossen
- While-loops of for-loops nuttig
- Input die variabelen van het probleem definieert

Algoritmes met loops

- Elke stap brengt je iets dichterbij de oplossing
- Variabelen nodig om je 'voortgang' in op te slaan

```
x, exp = 1.1538, 8
```

```
# reken x^exp uit
```

```
tussenresultaat = 1
```

```
for _ in range(exp):  
    tussenresultaat *= x
```

normaal:

`for i in range(exp):`
maar we hebben `i` niet nodig

```
# print het resultaat
```

```
print(tussenresultaat)
```


Algoritmes met loops

- Elke stap brengt je iets dichterbij de oplossing
- Variabelen nodig om je 'voortgang' in op te slaan
- Faculteit uitrekenen: $5! = 1 * 2 * 3 * 4 * 5$

Algoritmes met loops

- Elke stap brengt je iets dichterbij de oplossing
- Variabelen nodig om je 'voortgang' in op te slaan
- Faculteit uitrekenen: $5! = 1 * 2 * 3 * 4 * 5$

stap	1	2	3	4	5
resultaat	1	2	6	24	120



- Structuur: elk **getal op de onderste rij**, is het getal **links ervan** maal het getal **erboven**

Algoritmes met loops

stap	1	2	3	4	5
resultaat	1	2	6	24	120

n = 5

```
# reken n! uit
```

```
resultaat = 1
```

```
for stap in range(1, n+1): # [1, 2, 3, 4, 5]  
    resultaat = resultaat * stap
```

```
# print het resultaat
```

```
print(resultaat)
```

Uitvoer: 120

Algoritmes met loops en parameters

```
n = int(input()) # waarom int?

# reken n! uit
resultaat = 1
for stap in range(1, n+1): # [1, 2, 3, .., n]
    resultaat = resultaat * stap

# print het resultaat
print(resultaat)
```

For-loop vs While-loop

- Wanneer de één, wanneer de ander?
- For-loop: je weet van te voren hoe vaak je iets wilt herhalen.

Voorbeeld: `for i in range(n):`

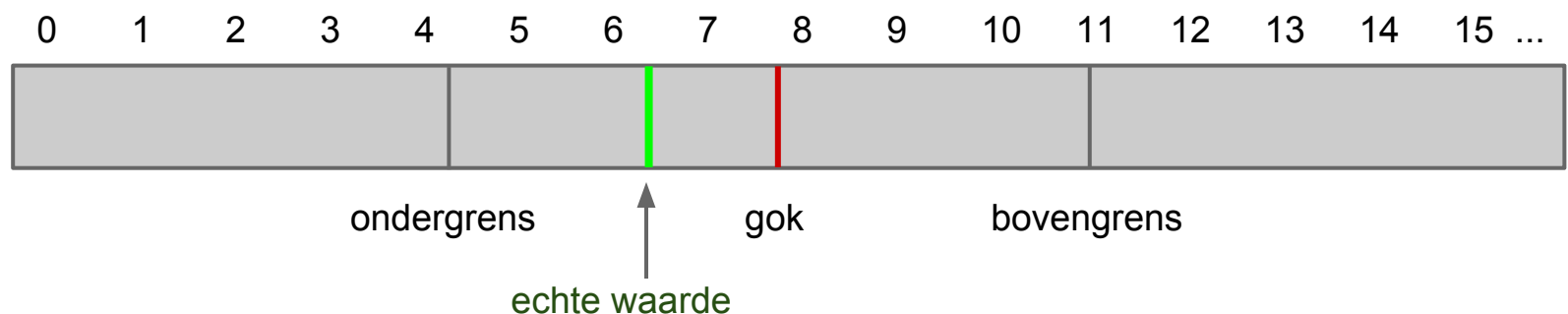
- While-loop: je weet iets dat je wilt bereiken
Voorbeeld: `while x < n:`

Wortel uitrekenen (opdracht 4)

- Voordat we de wortel van x uitgerekend hebben, weten we nog niet wat de uitkomst gaat worden
- Maar als we de wortel gokken, weten we of we er ver vanaf zitten: $|gok^2 - x|$
- En we weten of we te hoog of te laag zitten:
is $gok^2 > x$ of is $gok^2 < x$?

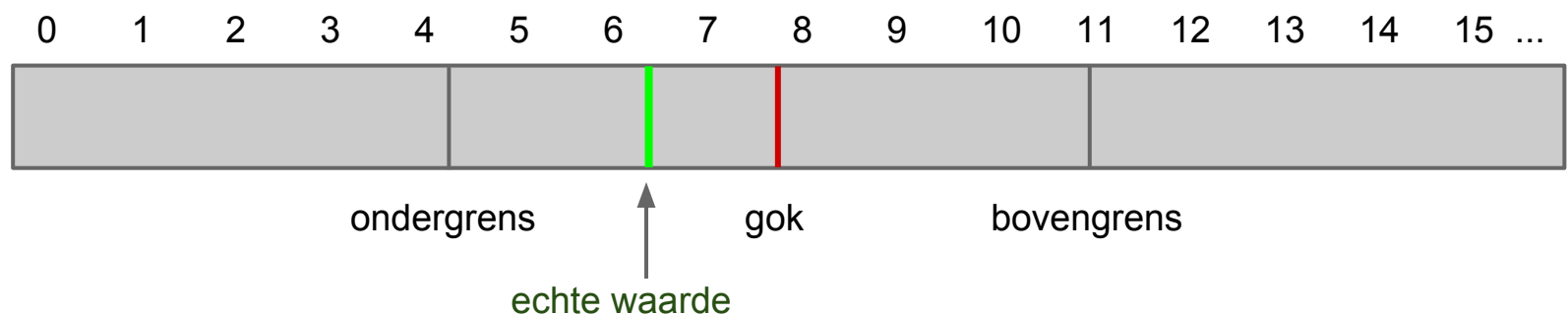
Wortel uitrekenen

- Neem een ondergrens en bovengrens waar de wortel zeker tussen zal zitten
- Eerste gok: halverwege tussen de grenzen



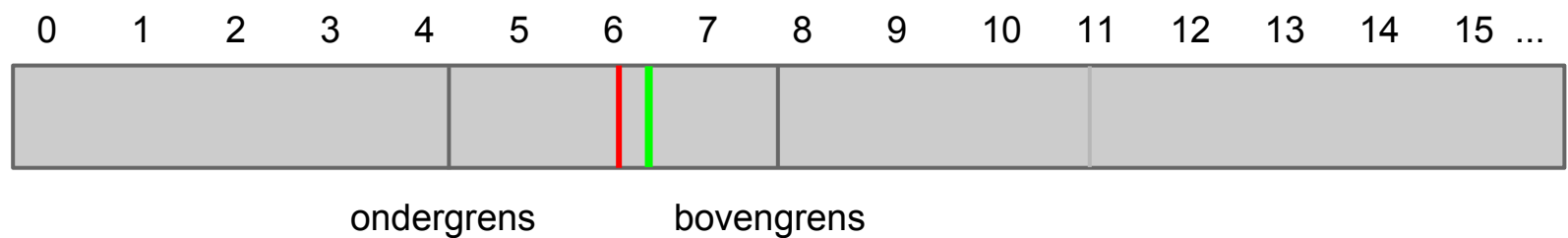
Wortel uitrekenen

- Kijk of de gok te hoog of te laag is
- Als hij te hoog is, is het een goede (nieuwe) bovengrens
- Anders een nieuwe ondergrens



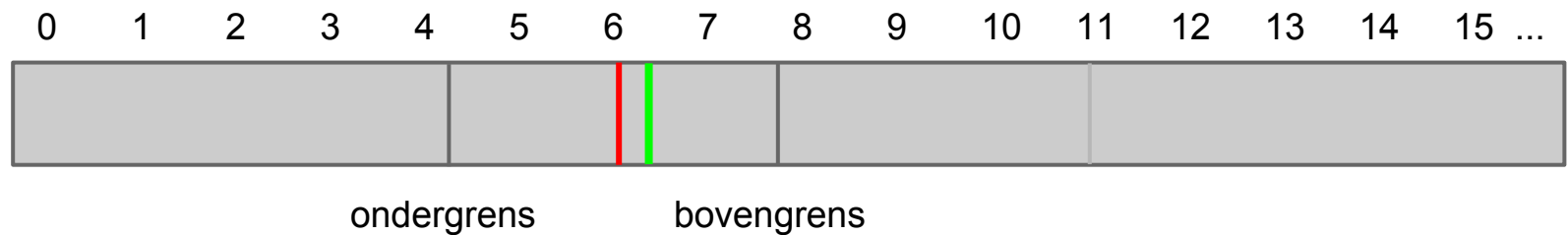
Wortel uitrekenen

- Kijk of de gok te hoog of te laag is
- Als hij te hoog is, is het een goede (nieuwe) bovengrens
- Anders een nieuwe ondergrens



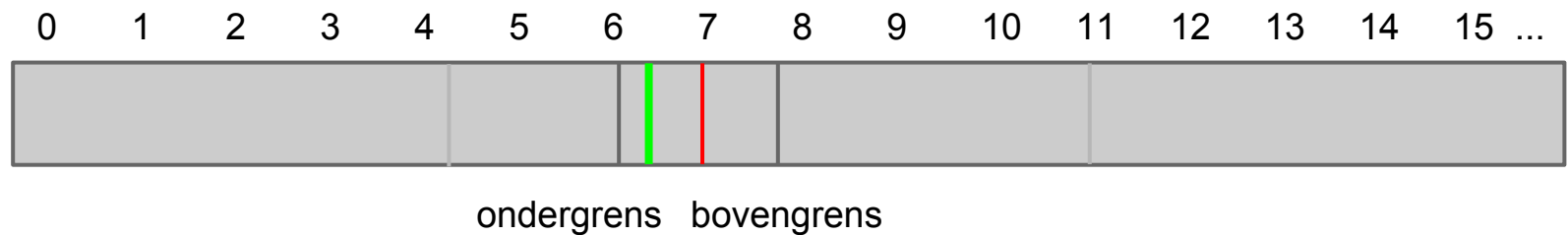
Wortel uitrekenen

- De schatting komt vanzelf steeds dichterbij de echte waarde...



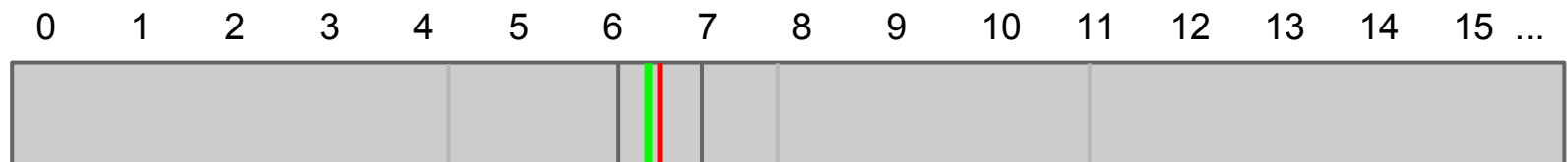
Wortel uitrekenen

- De schatting komt vanzelf steeds dichterbij de echte waarde...



Wortel uitrekenen

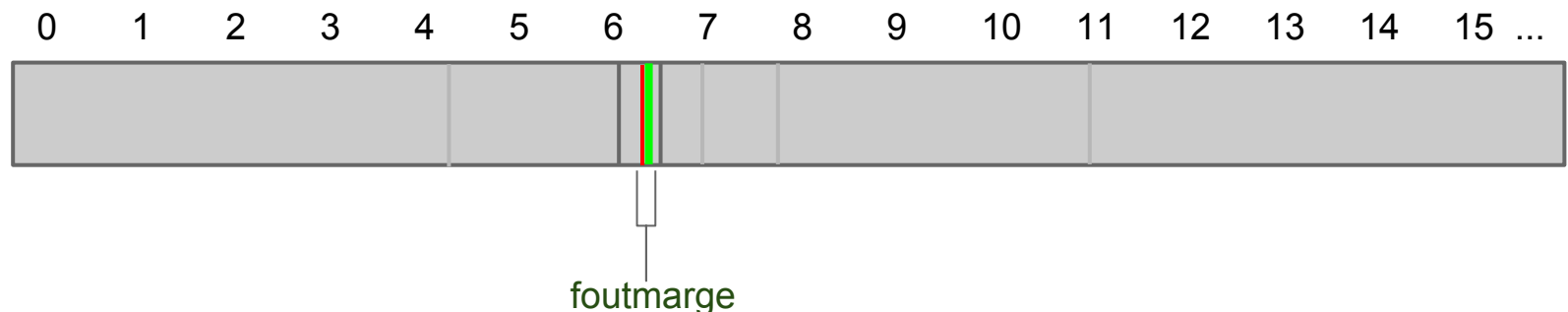
- De schatting komt vanzelf steeds dichterbij de echte waarde...



Wortel uitrekenen

- De schatting komt vanzelf steeds dichterbij de echte waarde...
- Tot de afstand zo klein is, dat je tevreden bent
- Conditie van de loop wordt dan ongeveer:

$\text{afstand} < \text{foutmarge}$



Wortel uitrekenen

- We willen net zolang doorgaan, tot we iets bereikt hebben
- Oftewel: een `while`-loop!

**Veel succes met de
opdrachten!**