



Unit 3 : Beginselen van programmeren in Python

Oefenblad 1 : Voorwaardelijke opdrachten

In deze Unit gaan we wat dieper in op programmeren in Python.

Doelen :

- Meer over voorwaardelijke opdrachten.
- True en False.
- Deelbaarheid.

Een positief geheel getal heet priemgetal als het slechts twee delers heeft (1 en zichzelf).

Zo is 7 een priemgetal want er zijn maar twee delers (1 en 7), maar 10 is geen priemgetal omdat 10 deelbaar is door 1, 2, 5 en 10.

Het kleinste priemgetal is 2.

We gaan een functie maken waarmee je kunt bepalen of een getal een priemgetal is.

De uitkomst van deze functie is True (waar) als het getal een priemgetal is en False (onwaar) als het getal geen priemgetal is.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Met behulp van deze functie berekenen we daarna het honderdste priemgetal.

We maken gebruik van een speciale deling: %

In Python geeft dit de rest bij een deling.

Zo is bijvoorbeeld $20\%3$ gelijk aan 2 want $20/3=6$ met een rest van 2.

(3 past 6 keer in 20 en je houdt 2 over)

Dit betekent: als $a\%b = 0$ dan is b een deler van a. (b past een heel aantal keren in a want de rest is nul).

Begin met de definitie van de functie ispriem(n).

Als $n < 2$ kan het geen priem zijn. Dan is de uitkomst dus False.

Als $n \geq 2$ dan controleren we of er een deler is. Zo ja dan is de uitkomst False, anders is de uitkomst True.

True en **False** kun je vinden met **F2(a A #)**.

```

EDITOR: PRIEM
PROGRAM LINE 0012
def is_priem(n):
    if n < 2:
        return False
    for i in range(2,n):
        if n%i==0:
            return False
    return True
    
```

Om het honderdste priemgetal te vinden maken we een while-lus.

Eerst kiezen we een teller (i) en geven die de waarde 0.

De variabele p doorloopt de getallen 2 en verder.

Telkens als p een priemgetal is wordt i een opgehoogd tot dat i=100.

Ga na dat je op deze manier het honderdste priemgetal krijgt.

```

EDITOR: PRIEM
PROGRAM LINE 0018
        return False
    return True

i=0
p=1
while i<100:
    p=p+1
    if is_priem(p):
        i=i+1

print(i,p)
    
```



De functie `is_priem(n)` in het programma is erg traag.

Als je bijvoorbeeld het 200-ste priemgetal wilt berekenen kost dat veel tijd.

Je kunt de functie aanmerkelijk sneller maken.

Als 3 een deler is van 27, dan is 9 dat ook want $3 \cdot 9 = 27$.

Telkens als je een deler hebt gevonden heb je er automatisch nog een.

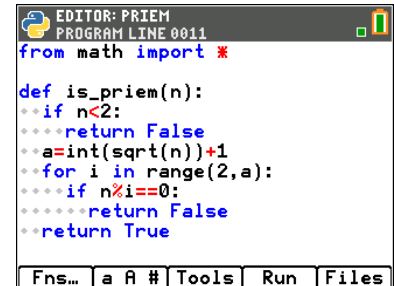
Eigenlijk hoef je alleen maar getallen te controleren die kleiner zijn dan of gelijk zijn aan de wortel van n .

Dit voegen we toe in de functie.

Om de wortel te kunnen gebruiken hebben we de **math**-module nodig.

Het getal a in de functie wordt berekend door eerst de wortel te berekenen, dan het gehele deel ervan nemen en dan nog 1 erbij optellen want we willen controleren t/m de wortel van n .

Ga na dat deze functie sneller is.



```
EDITOR: PRIEM
PROGRAM LINE 0011
from math import *

def is_priem(n):
    if n<2:
        return False
    a=int(sqrt(n))+1
    for i in range(2,a):
        if n%i==0:
            return False
    return True

Fns... a A # Tools Run Files
```