Unit 3: Programmeren in Python | **Oefenblad 1 : Voorwaardelijke opdrachten**

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

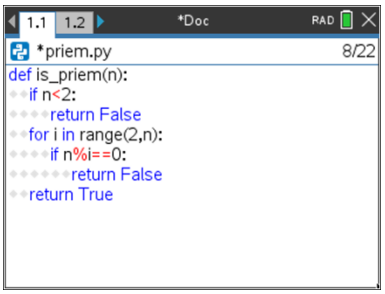
Doelen :

- 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.
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 dat b dan een deler is van a. (b past een heel aantal keren in a, de rest is nul).

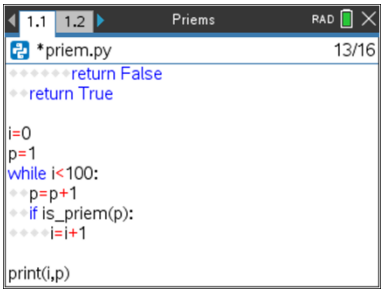
Begin met de definitie van de functie is_priem(n).
Als n<2 kan het geen priem zijn. Dan is de uitkomst dus False.
Als n>=2 dan controleren we of er een deler is.
Zo ja dan is de uitkomst False, anders is de uitkomst True.
(Je kunt **True** en **False** vinden in het menu bij Built-ins en dan bij 3:Ops)



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.

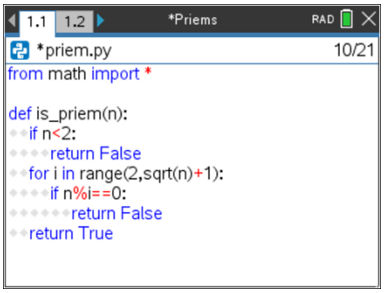


De functie is_priem(n) in het programma is erg traag.
Als je bijvoorbeeld het 1000-ste priemgetal wilt berekenen kost dat veel tijd (zeker op de handheld).
Je kunt de functie aanmerkelijk sneller maken.
Als 3 een deler is van 27, dan is 9 dat ook want 3*9=27.
Telkens als je een deler hebt gevonden heb je er automatisch nog een.
Eigenlijk hoef je alleen maar getallen te controleren die kleiner of gelijk zijn dan de wortel van n.

Dit voegen we toe in de functie.

Om de wortel te kunnen gebruiken hebben we de **math**-module nodig.
Voeg aan het begin van het programma deze module in met **from math import ***.

Ga na dat deze functie sneller is.


1.1 1.2 *Priems RAD
*priem.py 10/21
from math import *
def is_priem(n):
if n<2:
return False
for i in range(2,sqrt(n)+1):
if n%i==0:
return False
return True