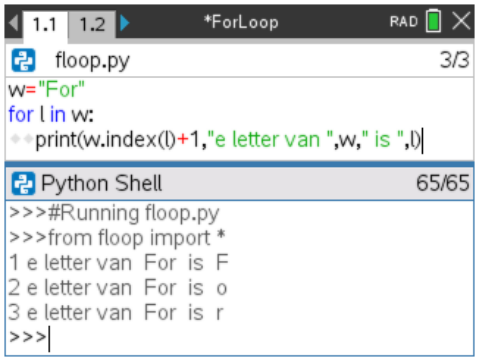
1. For-lus

Lussen worden gebruikt voor het automatisch uit voeren van code of blokken van code. We starten met de For-lus die handelt als een iterator. Het doorloopt items uit een geordende rij, b.v. lijsten, strings en tuples.

De structuur van een for-lus is als volgt:

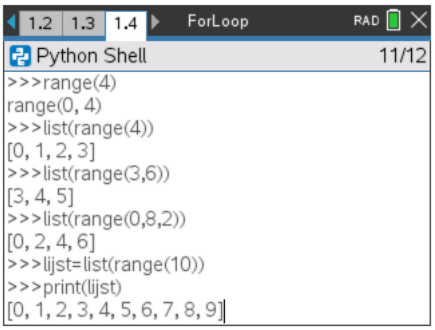
```
for index in object:
    block
```

Twee voorbeelden, baserend op een lijst en een string:



Een operator die vaak gebruikt wordt voor een for-lus is de range()-operator.
De range()-operator heeft de volgende syntax:

- range(4) van 0 tot 4, 4 niet inbegrepen
- range(3,6) van 3 to 6, 6 niet inbegrepen
- range(0,8,2) van 0 tot 8 met stapgrootte 2, 8 niet inbegrepen



range() is een generator!

Een generator genereert data zonder deze data op te slaan in het geheugen.

De combinatie van range() en de list()-functie creëert effectief een lijst.

Voorbeeld 1

Het werpen van een dobbelsteen

De volgende code simuleert het werpen van een dobbelsteen, waarbij voor iedere worp gecheckt wordt of het aantal ogen zes is. In het geval van een zes wordt de teller verhoogd met 1.
Uiteindelijk printen we de benadering van de kans op zes bij het werpen van een dobblesteen: $P(zes) = \frac{1}{6}$.

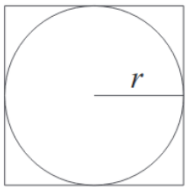
```
from random import *

aantal=int(input("Aantal worpen: "))
teller=0

for i in range(aantal):
    worp=randint(1,6)
    if worp ==6:
        teller+=1

prob=teller/aantal

print("# 6: ",teller,"in", aantal, "worpen")
print("Kans op zes ≈ {0:5.4f}".format(prob))
```

Voorbeeld 2

Monte Carlo benadering voor π

Een Monte Carlo method is een algoritme dat gebruik maakt van random sampling om een probleem op te lossen of te benaderen.

Om π te benaderen tellen we het aantal random gegenereerde punten in een vierkant die binnen de ingeschreven cirkel vallen zoals hiernaast afgebeeld.

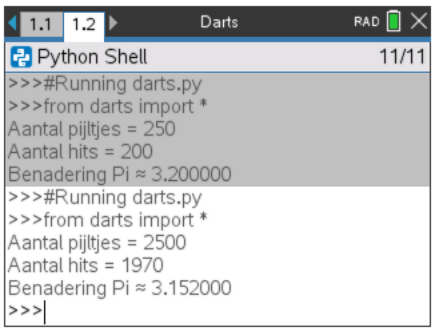


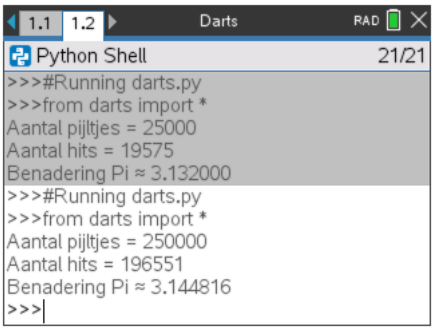
De verhouding van het aantal punten in de cirkel tot het totale aantal gegenereerde punten geeft als volgt een benadering voor π:

$$\frac{N_{\text{cirkel}}}{N_{\text{totaal}}} \approx \frac{\text{Oppervlakte}_{\text{cirkel}}}{\text{Oppervlakte}_{\text{totaal}}} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4} \quad \Rightarrow \quad \pi \approx 4\frac{N_{\text{cirkel}}}{N_{\text{totaal}}}$$

De volgende code simuleert a.h.v. een Monte Carlo methode een benadering van het getal π.

```
from math import *
from random import *

darts=int(input("Aantal pijltjes = "))
hits=0

for worp in range(darts):
    x=random()*2
    y=random()*2
    if sqrt(x**2+y**2)<1:
        hits+=1

prob=4*hits/darts

print("Benadering Pi ≈ {0:5.6f}".format(prob))
```





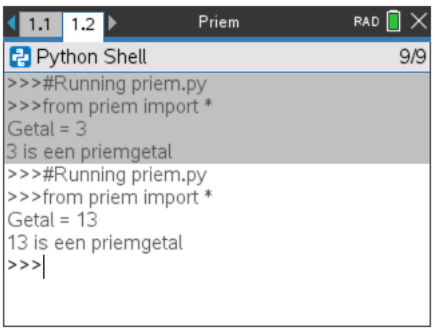
Voorbeeld 3

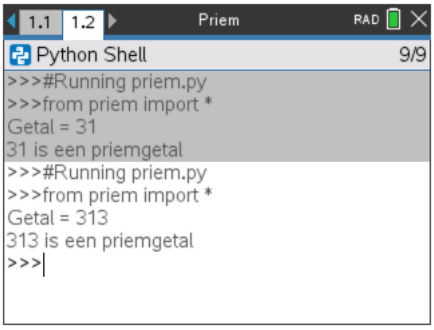
Wanneer is een getal een priemgetal?

Een priemgetal is een geheel getal groter dan 1 dat niet kan geschreven worden als een product van twee kleinere natuurlijke getallen. Priemgetallen worden veel gebruikt in cryptografie, b.v. voor de RSA-code, omdat het ontbinden van grote getallen in een product van priemgetallen niet zo eenvoudig is.

Een methode om te bepalen of een getal n een priemgetal is, is te testen of n een veelvoud is van een van de gehele getallen z met $2 \le z \le \sqrt{n}$.

```
from math import *
n=int(input("Getal = "))
deler=0
if n<=1:
  print("Input ≤ 1 :-(")
else:
  for i in range(2,floor(sqrt(n))+1):
    if n%i == 0:
      deler+=1
  if deler==0:
    print(n,"is een priemgetal")
  else:
    print(n,"is geen priemgetal")
```





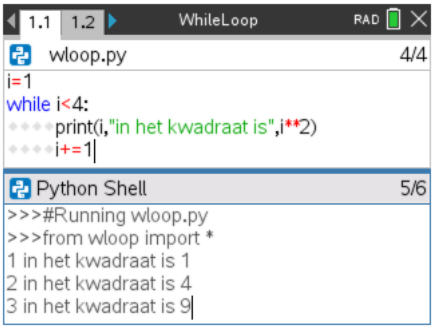
2. While-lus

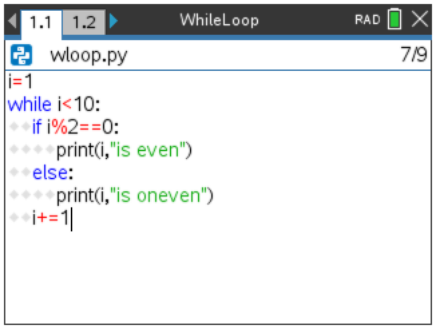
Een while-lus wordt gebruikt voor het uitvoeren van code of blokken van code zolang aan een bepaalde voorwaarde voldaan is.

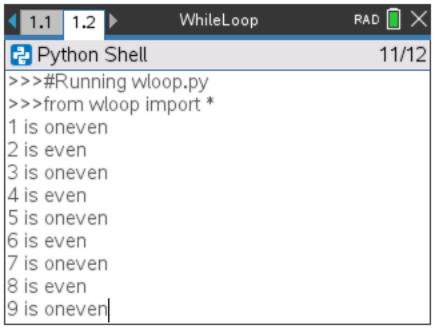
De structuur van een while-lus is als volgt:

```
while BooleanExpr:
    block
```

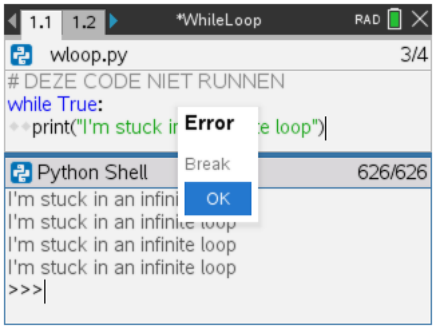
Twee eenvoudige voorbeelden voor het illustreren van de syntax van een while-lus:







Wat te doen in het terecht komen in een oneindige loop, b.v. bij het vergeten toe te voegen van i+=1 in bovenstaande voorbeelden?



- Handheld Druk op esc of on
- Windows Druk F12
- MacOs Druk F5

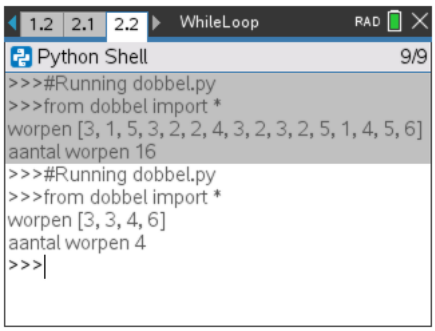
Afhankelijk van het programma, wordt het programma niet altijd onmiddellijk onderbroken; best dan de knop langer ingedrukt te houden

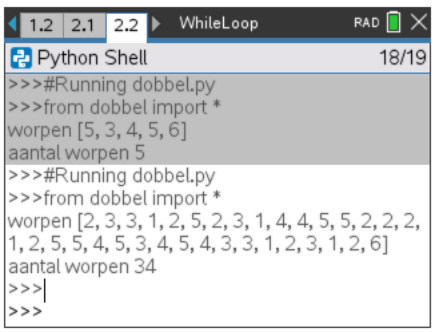
Voorbeeld 1

Het werpen van een dobbelsteen

De volgende code simuleert het werpen van een dobbelsteen tot dat het aantal ogen gelijk is aan zes met als output de ogen van alle worpen en het aantal worpen.

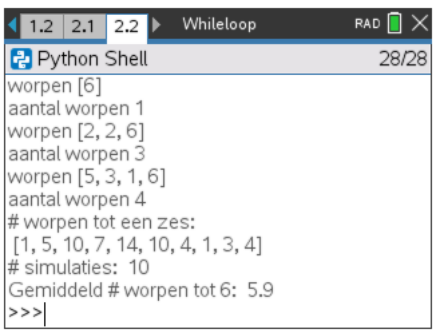
```
from random import *

aantal=0
ogen=0
worpen=[ ]

while ogen != 6:
    ogen=randint(1,6)
    worpen.append(ogen)
    aantal+=1

print("worpen",worpen)
print("aantal worpen",aantal)
```

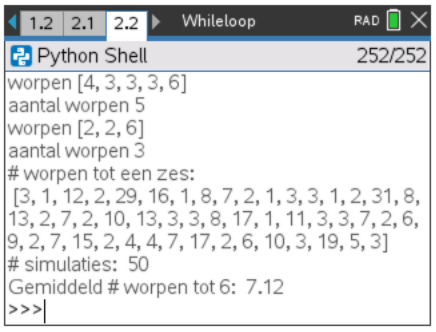


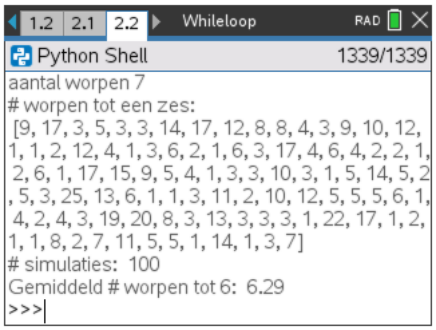


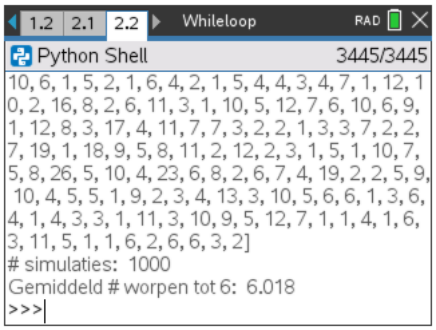
We voegen code toe (for-lus) om deze simulatie een aantal keren na mekaar uit te voeren en telkens het aantal worpen tot zes ogen op te slaan in een lijst. We eindigen het programma met het berekenen van het gemiddelde van deze lijst.

```
from random import *

t=int(input("# experimenten: "))

tot6=[ ]

for n in range(t):
    aantal=0
    ogen=0
    worpen=[ ]

    while ogen != 6:
        aantal=aantal+1
        ogen=randint(1,6)
        worpen.append(ogen)

    print("worpen",worpen)
    print("aantal worpen",aantal)

    tot6.append(aantal)

print("# worpen tot een zes: ",tot6)
print("# simulaties: ",t)

som=0
for i in range(len(tot6)):
    som=som+tot6[i]
print("Gemiddeld # worpen tot zes",som/len(tot6))
```



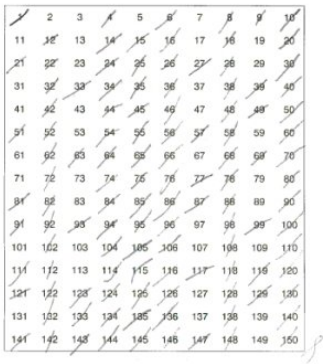






Voorbeeld 2

Zeef van Eratosthenes



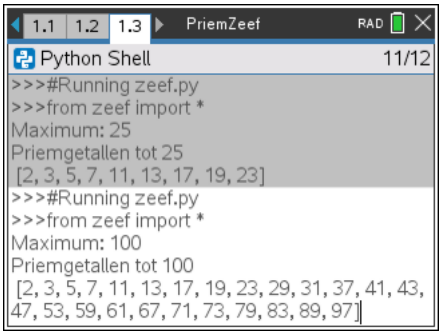
Dit zeer lang gekende algoritme voor het vinden van priemgetallen werkt als volgt:

- Stap 1 Creëer een lijst startend vanaf 2 tot een te kiezen maximum.
- Stap 2 Verwijder alle veelvouden van 2 uit de lijst.
- Stap 3 Kies het kleinste nog overgebleven getal uit de lijst.
- Stap 4 Verwijder alle veelvouden van dit gekozen getal en ga verder met stap 3

Men kan steeds starten met verwijderen van getallen vanaf het kwadraat van het gekozen getal daar alle kleinere veelvouden al verwijderd zijn.

Het algoritme is voltooid als het gekozen getal groter is dan de wortel van het maximum.

```
from math import *
n=int(input("Maximum: "))
priemlijst=[2]
for i in range(3,n+1):
    priemlijst.append(i)

i=2
while i <= sqrt(n):
    if i in priemlijst:
        for j in range(i**2, n+1, i):
            if j in priemlijst:
                priemlijst.remove(j)
    i=i+1

print("Priemgetallen tot",n,"\n",priemlijst)
```

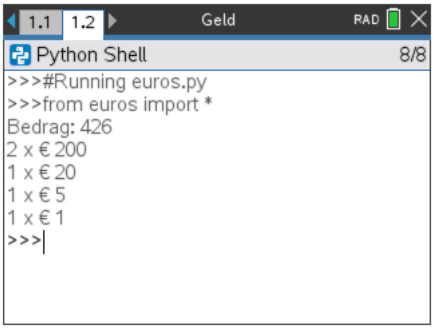


Voorbeeld 3

Verdeling in euros

Het onderstaande programma verdeelt ieder geheel bedrag in € in biljetten van € 200, € 100, € 50, € 20, € 10 of € 5 en munten van € 2 of € 1.

```
bank=[200,100,50,20,10,5,2,1]
bedrag=int(input("Bedrag: "))

for geld in bank:
    aantal=0
    while bedrag>=geld:
        bedrag=bedrag - geld
        aantal=aantal + 1
    if aantal > 0 :
        print (aantal, "x €", geld)
```



We bekijken de code van twee speciale while-lussen.

2.1. break & continue

De statements break en continue doen het volgende:

- break breekt uit de dichtstbijzijnde omsluitende lus
- continue gaat naar het begin van de dichtstbijzijnde omsluitende lus

Het onderstaande programma blijft een dobbelsteen werpen simuleren totdat het aantal ogen 6 is:

```
from random import *

while True:
    w=randint(1,6)
    if w==6:
        break
    else:
        print("Aantal ogen =",w)
        print("... verder werpen")
        continue

print("STOP - Zes ogen geworpen")
```

2.2. get_key

Gebruikmakend van de module TI System kunnen we een while-lus uitvoeren tot het onderbreken met het intikken van een toets. De syntax ziet er b.v. uit zoals hieronder. De while-lus blijft de blok code uitvoeren tot dat de esc-toets wordt ingedrukt.

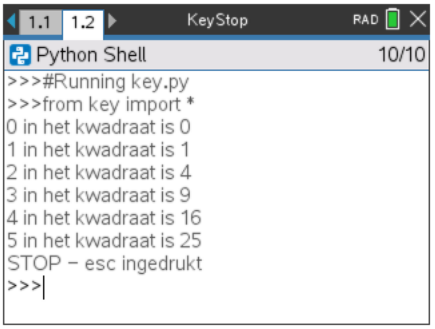

```
key=0
while key != "esc":
    block
    key=get_key()
```

We gebruiken het sleep()-statement van de time-module om het uitvoeren van de code te pauzeren.

De output van kwadraten blijft op het scherm – per 1 seconde – verschijnen tot het indrukken van de esc-toets.

Merk op dat meerdere code op 1 regel kunt plaatsen door met ; de code te scheiden.

```
from ti_system import *
from time import *

i=0 ; key=0

while key != "esc":
    print(i,"in het kwadraat is",i**2)
    sleep(1)
    key=get_key()
    i+=1

print("STOP - esc ingedrukt")
```



De output van kwadraten blijft op het scherm – per 1 seconde – verschijnen tot het indrukken van de esc-toets.

Merk op dat meerdere codes met ; op eenzelfde regel kunnen geplaatst worden.