# Geheim!

# Keuzecollege technasium 2

# Python

Vanmiddag gaan we aan de slag met RSA, een moderne cryptografie methode. De eerste stap van RSA is het kiezen van twee grote priem getallen. We moeten dan wel weten of de getallen priem zijn of niet. We gaan hiervoor een programma schrijven dat voor ons gaat checken of een getal priem is of niet.

Een van de andere stappen binnen RSA is het checken of twee getallen co-priem zijn. Met andere woorden moeten we kunnen bepalen of de grootste gemeenschappelijke deler van twee getallen gelijk is aan 1. Dit doen we liever niet met de hand, zeker niet als het om grote getallen gaat. We gaan ook hiervoor een programma schrijven dat voor ons de grootste gemeenschappelijke deler kan vinden.

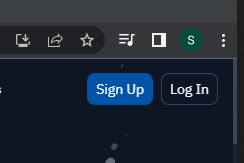
Als bonus opdracht gaan we het programma dat priemgetallen kan berekenen verbeteren.

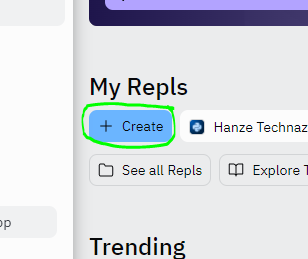
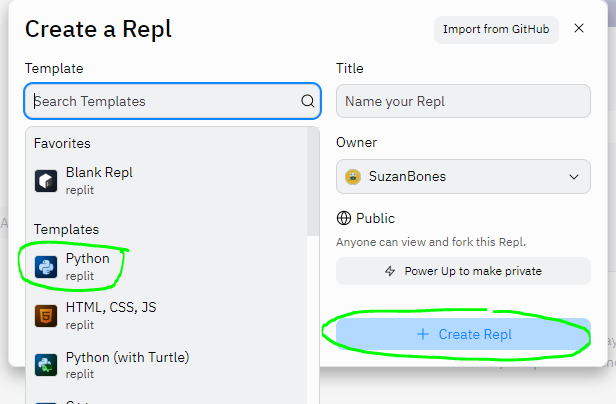
Python is een taal om met een computer opdrachten te geven. Wil de computer je geschreven opdrachten echt begrijpen en uitvoeren, dan heeft hij daar wel een vertaler voor nodig, een zogenaamde interpreter. Normaal gesproken zou je die eerst moeten installeren, maar er bestaan ook websites die Python programma’s kunnen uitvoeren.

# Project opstarten

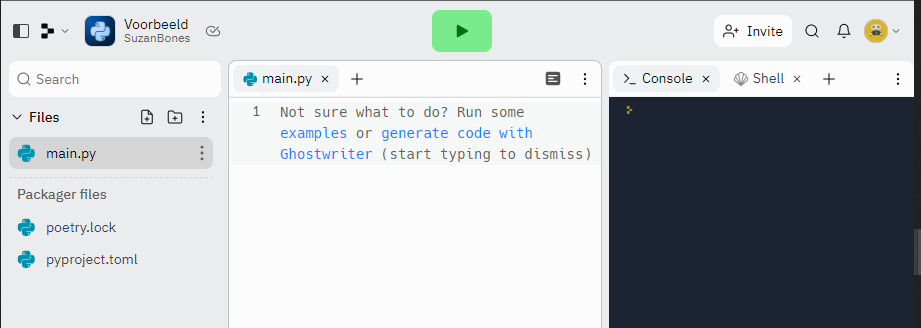
Stap 1: Open <https://replit.com/>

Stap 2: Inloggen met account dat in de eerste keuzecollege is gemaakt

 Heb je nog geen account, maak er 1 aan met je schoolmail.

Stap 3: Zodra je bent ingelogd, klik op +Create om een project te starten en kies voor een Python project. Vervolgens klik je op ‘+Create Repl’

Als alles goed is gegaan heb je nu een project omgeving om mee aan het werk te gaan. Een aantal belangrijke onderdelen zijn hieronder aangegeven.



A

B

C

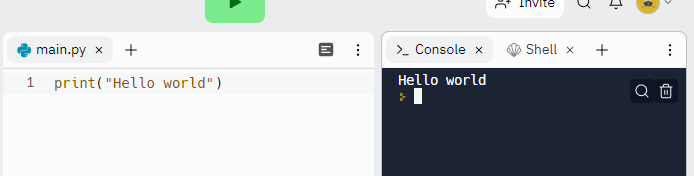
D

E

1. Hier vind je alle files. Wij werken in dit project enkel met de main.py file
2. Schrijf hier al je code. Zorg er voor dat je telkens in de main.py file werkt.
3. Met de RUN knop kunnen we de code uitvoeren
4. Als je met iemand samen wil werken kan je met de C knop iemand uitnodigen. (Generate join link en deel deze link met een medeleerling)
5. Dit is de console. Hier kunnen we de output van ons programma bekijken.

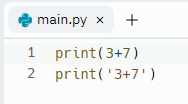
# Warmup

Een heel bekend programma is ‘hello world’. Dit programma zorgt ervoor dat de computer ‘hello world’ print.

Type in print("Hello world") en druk op run. De computer zal nu Hello world printen in de console.

Print is de opdracht aan de pc om output te geven. Tussen de haken geven we aan wat we willen printen. Als we tekst willen printen zetten we aanhalingstekens om de tekst heen. Bij getallen hoeft dat niet.

## Opdracht 1:

Type:

Wat denk je wat de computer gaat printen?

Antwoord:

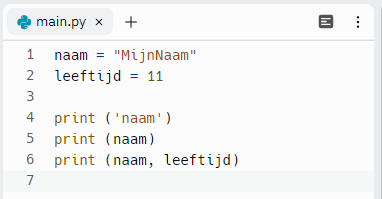
Druk op run

Wat print de computer?

Antwoord:

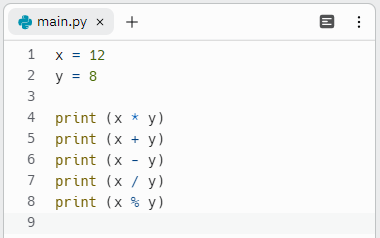
In het vorige keuzecollege zijn we variabelen tegengekomen. Variabelen geeft ons de mogelijkheid om waardes op te slaan en op een ander moment in het programma te gebruiken.

## Opdracht 2:

1. Type in en vul je eigen naam en leeftijd in.

Druk op run. Wat is de output?

1. Type in en druk op run.



Wat is de output en waarom?

# Priem getallen

Priem getallen zijn enkel deelbaar door 1 en zichzelf. Het getal 7 is dus een priemgetal, maar het getal 8 niet. Want 8 is namelijk ook deelbaar door 2 en 4. Dus 8 heeft meer factoren dan enkel 1 en zichzelf.

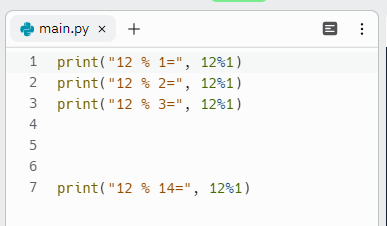
Om te bepalen of een getal deelbaar is door een gegeven getal kunnen we modulo gebruiken.

12 % 3 = 0. In dit voorbeeld is 12 deelbaar door 3 want er is geen rest over.

12 % 7 = 5. 12 is niet deelbaar door 7 want we hebben een rest van 5

## Opdracht 3:

We zoeken alle factoren van 12.

Type in en maak het programma af. (Gaat alles zo wel goed?)

1. Kopieer je output en plak hieronder  
   Antwoord:

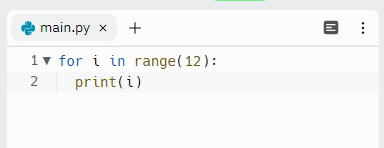
1. Wat zijn de factoren van 12?  
   Antwoord:

Voor een klein getal als 12 het nog te doen om alle regels zelf te schrijven, maar als we alles willen printen voor een getal zoals 63 hebben we 63 regels nodig. Dan wordt het vervelend. Gelukkig is daar een oplossing voor, namelijk een loop. (ook wel lus genoemd)

Met een for-loop kunnen we aangeven aan de computer om een bepaalde opdracht een aantal keer te herhalen, daarbij kunnen we bijhouden in een variabele hoe vaak de lus al is herhaald. In ons voorbeeld gebruiken we de variabele i om bij te houden hoe vaak de lus al is herhaald. Deze i kunnen we ook printen.

## Opdracht 4:

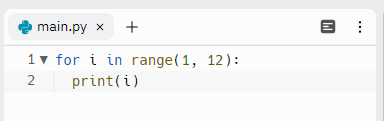
Type in en druk op run



1. Kopieer je output en plak hieronder  
   Antwoord:
2. Pas de code zo aan dat de getallen 0 tot met 63 wordt geprint. Plak je gevonden code hieronder.

We willen vanaf 1 gaan tellen, om dat te kunnen doen moeten we range in de for loop zo aanpassen dat we beginnen bij 1 en niet standaard bij 0.

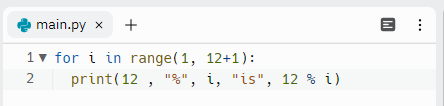
## Opdracht 5:

Type in en druk op run:

1. Kopieer je output en plak hieronder  
   Antwoord:
2. Pas de code zo aan dat de getallen 1 tot en met 63 wordt geprint. Plak je gevonden code hieronder.

We kunnen de computer nu 1 tot en met 63 laten printen. Nu kunnen we dat samenvoegen met onze modulo opdracht.

## Opdracht 6:

Type in en druk op run

1. Kopieer je output en plak hieronder  
   Antwoord:
2. Pas de code zo aan dat we voor het getal 63 alle berekeningen wordt uitgevoerd. Plak je gevonden code hieronder.
3. Wat zijn de factoren van 63? Maak gebruik van de output van je programma.

Nu hebben we een klein programma geschreven dat alle berekeningen voor ons doet zonder dat wij 63 regels code moeten intypen. Het nadeel alleen is dat we nu in de output moeten gaan zoeken om te checken waar de uitkomt 0 is.

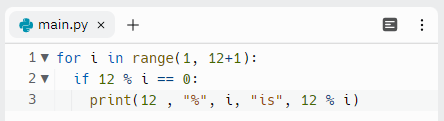
We kunnen python laten zoeken of een uitkomst van een berekeningen gelijk is aan 0. Hiervoor kunnen we een if-statement gebruiken. Als een conditie waar is, voor dan de opdracht uit. (if true then do this). Als de uitkomst van onze modulo berekening 0 is, print dan de berekening.

if 12 % i == 0:

print(…)

## Opdracht 7:

Type in en druk op run



1. Kopieer je output en plak hieronder  
   Antwoord:
2. Pas de code zo aan dat deze voor het getal 132 alle factoren print ( met berekening). Kopieer de output in plak deze hieronder.

We kunnen nu wel elk getal in ons programma gaan testen, maar dan moeten we handmatig op verschillende plekken het programma aanpassen. Om te zorgen dat we maar op 1 plek ons getal moeten veranderen kunnen we een variabele. Hier kunnen we dan onze getal in opslaan.

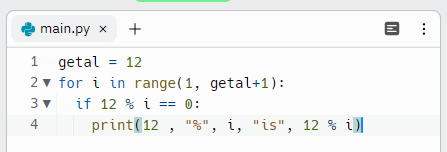
Denk aan wiskunde. Daar heb je ook functies met variabelen, zoals f(x) = 8x + 2. Hier is x de variabele. Als we dan aannemen dat x = 4 kunnen we de x in de functie vervangen met 4 en de uitkomst berekenen. 8x + 2 = 8 \* 4 + 2 = 34

## Opdracht 8:

In deze opdracht gaan we ons getal opslaan in een variabele genaamd getal en deze in de code gebruiken.

Hieronder staat het begin van die aanpassing. In regel 1 kan je zien dat de waarde 12 in de variabele getal wordt opgeslagen. In regel 2 wordt dan vervolgens ook deze variabele gebruikt.

Type over



1. Maak het programma af.
   1. In regel 3 moet de variabele getal 1 keer voorkomen
   2. in regel 4 moet de variabele getal 2 keer voorkomen

Kopieer je gemaakte code hieronder

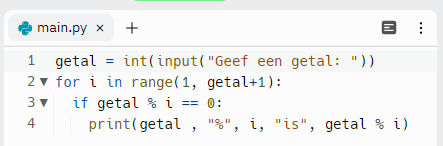
1. Verander de waarde van de variabele getal naar 87 en druk op run. Wat zijn de factoren van 87?

We hebben nu dus een programma waarmee we alle factoren van een getal kunnen vinden. Hiermee kunnen we dus bepalen of een priemgetal is door te controleren dat er maar twee factoren zijn, namelijk zichzelf en 1. Het getal 87 is dus niet priem, gezien ook 3 en 29 factoren zijn van 87.

Het beste zou zijn dat we niet eens onze code hoeven aan te passen om te controleren of een getal priem is of niet. We zouden het programma voor input kunnen laten vragen en dan een getal intypen.

## Opdracht 9:

Verander regel 1 en druk op run.



Je krijgt nu in de terminal de mogelijkheid om een getal in te typen.

1. Gebruik je programma om van de volgende getallen te bepalen of ze priem zijn of niet. Je kan de getallen kopiëren en plakken. Als de berekening lang duurt kan je deze stoppen door op de stopknop te drukken.

182937

2639

6700417

9876544321

1. Probeer uit te zoeken of 170141183460469231731687303715884105727 priem is.

Je zult zien dat met zulke grote getallen de computer er ook heeeel lang over moet rekenen om te controleren of een getal priem is of niet. Er zijn wel aanpassingen die we in het programma kunnen toevoegen om het te enigszins te versnellen. Daar komen we in de verdieping op terug.

Laten we naar de eerste stappen kijken van RSA.:

Stap 1: Kies twee grote priemgetallen, we noemen deze p en q. Mag niet twee dezelfde getallen zijn.

Stap 2: Vermenigvuldig beide priemgetallen met elkaar. We noemen de uitkomst N. Dus p \* q is N.

De rest van de stappen gaan we straks uitleggen.

Het getal N wordt over het internet gestuurd. Deze kan gevonden worden door hackers. Dit is niet erg, zolang p en q maar geheim blijven. Als de hacker namelijk wel p en q kan vinden, dan kan hij de code kraken en geheime berichten lezen.

Het getal N heeft maar 4 factoren. 1, zichzelf, p en q. (Want p en q waren immers priemgetallen en hebben dus ook geen andere delers.) Stel dat wij de hackers zijn en hebben het getal N te pakken gekregen. We kunnen dan proberen p en q te vinden met ons eigen gemaakte programma!

## Opdracht 10:

1. Maak gebruik van je programma om voor de volgende getallen p en q te vinden.   
   559 p = q =   
   8633 p = q =  
   338519 p = q =  
   309715664956111 p = q =
2. In het echte leven zijn p en q groter. Probeer de p en q te vinden van 75613602168854886891707. (Waarschuwing: Dit kan lang duren)

Je merkt al dat als p en q echt grote priemgetallen zijn dat N super groot wordt en het heel lang duurt om p en q te vinden. De veiligheid van RSA ligt in het feit dat het berekenen van p en q heel veel tijd kost. Een Hacker kan uiteindelijk wel p en q berekenen, maar dan is er al zooooveel tijd (jaren) voorbij gegaan dat het bericht dat hij wil ontcijferen inmiddels niet meer belangrijk is.

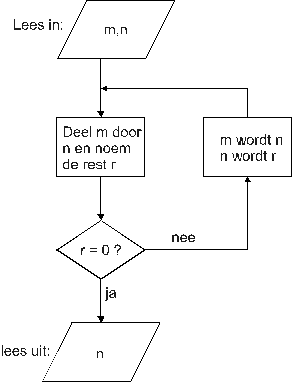
# Grootste gemene deler.

We hebben net naar de eerste twee stappen gekeken van RSA, maar er zijn nog meer stappen.

* Stap 1: Kies twee grote priemgetallen, we noemen deze p en q. Mag niet twee dezelfde getallen zijn.
* Stap 2: Vermenigvuldig beide priemgetallen met elkaar. We noemen de uitkomst N. Dus p \* q is N.
* Stap 3: Bereken (p-1)(q-1) en kies een getal dat relatief priem is met de uitkomst EN tussen de 1 en N ligt. Binnen de RSA noemen we dit getal e.

Tijdens de presentatie over wiskunde hebben we het gehad over co-priem. Het houdt in dat als we twee getallen hebben dan zijn ze co-priem als de grootste gemene deler gelijk is aan 1. Er is dus geen groter getal dan 1 dat beide getallen kan delen.

Om dus te kunnen bepalen of twee getallen co-priem zijn moeten we dus hun grootste gemene deler bereken. We hebben op het bord de ggd van 84 en 35 berekend. Dit deden we volgens een stappenplan. In de ICT wordt dit een algoritme genoemd. We gaan dit algoritme programmeren in Python.

m = 84 en n = 35

m % n = 84 % 35 = 14

m = 35 en n = 14

m % n = 35 % 14 = 7

m = 14 en n = 7

14 % 7 = 0

We hebben een rest van 0 gevonden. De GGD deler tussen 84 en 35 is 7.

## Opdracht 11:

Het zou zonde zijn om onze priem code te verwijderen. Daarom gaan we er een functie van maken. Dit lijkt op een wiskunde functie.

Wiskunde: f(x) = berekening

In dit voorbeeld heeft de functie de naam f en als input kunnen we een waarde meegeven. In programmeren noemen we de input van een functie een parameter.

1. Type in

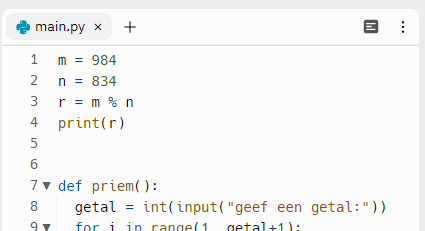
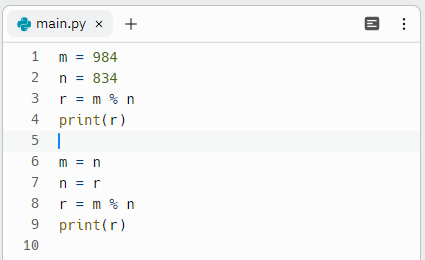


Nu hebben we van onze code een functie gemaakt en via priem() op regel 8 vertellen we het programma om de functie uit te voeren.

1. Druk op run. Als alles goed is gegaan zal het programma om input vragen. Geef als input het getal 123.
2. Verwijder regel 8. Zo zorgen we ervoor dat we de oude code blijft staan, maar niks meer doet tenzij wij het weer aanroepen.

## Opdracht 12:

We gaan de grootste gemene deler tussen 984 en 834 uitrekenen met Python.

1. Type in en druk op run. Je kan zien dat de priem code er ook bij staat, maar deze code doet nu niks want we roepen het niet aan. Ga er bij de volgende opdrachten van uit dat de code voor priem onderaan staat.
2. Druk op run. Wat is de output?
3. We moeten door blijven rekenen totdat de print opdracht ons een 0 oplevert, want dan is de rest (variabele r) ook gelijk aan 0.  
   Type in:   
   
4. Druk op run. Er zal nu 150 en 84 uitkomen. We zijn dus nog niet bij 0 gekomen. Copy en paste regel 6 t/m 9 net zo lang totdat de computer een 0 print. De waarde dat boven de 0 staat, dat is onze GGD.   
   Wat is de grootste gemende deler van 984 en 834 ?

Uiteindelijk moet de computer de berekening 7 keer uitvoeren en dat levert ons 28 regels aan code (lege regels en code voor priem niet meegerekend)

Dit is natuurlijk onhandig. Stel nou dat we de GGD willen vinden tussen 87635 en 83875. Dat kost misschien nog meer regels code, of juist minder.

We hebben in de priem opdrachten een for-loop gebruikt om code een gegeven aantal keer te herhalen. Het nadeel hier is dat we nu niet weten hoe vaak we de code moeten herhalen. Het enige wat we weten is dat we de code net zolang moeten herhalen zolang de modulo berekening niet gelijk is aan 0.

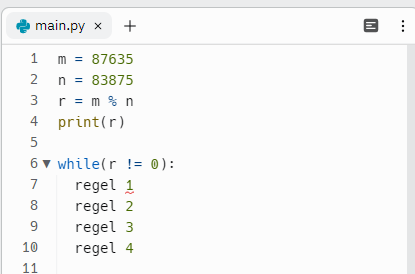
Gelukkig hebben we naast de for-loop ook een while-loop. Met een while-loop kunnen we code laten herhalen zolang een bepaalde voorwaarde waar is.

Waarvoor zou dit de code kunnen zijn?:

while (muziek staat aan)

dans

## Opdracht 13:

1. Type in en vervang regel 7 t/m 10 met de juiste code zodat de juiste code herhaalt wordt.
2. Kopieer en plak je code hieronder:
3. De voorwaarde in de while loop is r != 0. Wat zou != kunnen betekenen?
4. Wat is de GGD van 87635 en 83875?

## Opdracht 14:

Het zal fijn zijn als ons programma netjes om input vraagt en niet dat we de getallen in onze code moeten aanpassen. Daarnaast hoeven we niet alle tussenstappen te zien, maar willen we vooral het antwoord weten.

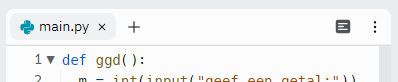
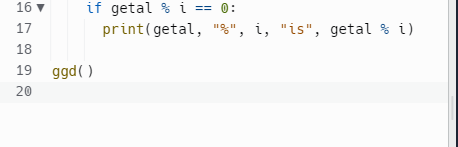
1. Vul regel 1 en 2 aan zodat de computer om input gaat vragen. (check je priem code om te kijken wat de code daarvoor is.
2. Voeg een print statement toe op regel 12 (tussen de while loop en de code voor priem) dat de waarde van de variabele n print.
3. Test je programma met de getallen 786390 en 238740. Als alles is goed gegaan dan print je programma enkel het getal 30 af.
4. Gebruik je programma om erachter te komen of de volgende getallen co-priem zijn of niet ( m.a.w is de GGD gelijk aan 1) **BELANGRIJK**: m moet altijd groter zijn dan n is!  
     
   373 en 619  
   87471 en 123867  
   9283 en 2983

# Een menu

We hebben nu code om factoren te kunnen vinden en we hebben een stuk code om de ggd tussen twee getallen te bepalen. Het zou mooi zijn dat als we op run klikken de computer ons vraagt wat we willen doen, namelijk factoren berekenen of de ggd berekenen.

## Opdracht 15

Onze eerste stap is om de code van de ggd berekening in een functie te plaatsen.

1.  Schuif alle code 1 regel naar beneden en zet boven aan:
2. Zorg dat alle code die bij de ggd berekening hoort een tab in het begin krijgt. hiermee geven we aan dat deze regels bij de functie ggd horen.
3. Zet nu helemaal onderaan ggd() (zelfs onder de code voor priem) :
4. Druk nu op run. Als alles goed is gegaan roept de computer de ggd() functie en voert dus de ggd berekening uit. Test je programma met random getallen.

We hebben nu twee functies in onze code staan. We hebben de functie ggd en de functie priem. Door een functie aan te roepen wordt de code in de functie uitgevoerd. We moeten alleen nog aan de computer duidelijk maken welke functie we aan willen roepen zodra we op de run knop drukken.

## Opdracht 16

Je krijgt bij deze opdracht minder hulp en moet nu zelf meer code schrijven. Gebruik wat je hebt geleerd over code om de opdracht te maken.

1. Verwijder de regel met ggd() die we in opdracht 14.c hadden toegevoegd.
2. Zorg ervoor dat er op regel 20 een print statement komt met de tekst:   
   *kies 1 voor ggd*
3. Zorg ervoor dat er op regel 21 een print statement komt met de tekst:   
   *kies 2 voor priem*
4. Druk op run en check of de computer inderdaad de twee regels print en verder niks doet.
5. Voeg op regel 22 de code toe dat om een getal als input vraagt en dat opslaat in een variabele met de naam keuze.
6. Op regel 23 voegen we een if statement toe dat controleert of de variabele keuze de waarde 1 heeft ( met == kan je checken of iets gelijk aan elkaar is)
7. Op regel 24 roepen we de ggd functie aan (met een tab ervoor)   
   m.a.w., als het if statement op regel 23 waar is roepen we de ggd functie aan
8. Druk op run. Als alles goed is zal de computer om input vragen en als we een 1 als input geven start de ggd functie zoals het hoort. Als we iets anders dan een 1 invoeren gebeurt er niks.
9. Op regel 25 zetten we het woord else neer (zonder tab ervoor). Dit zorgt ervoor dat als het if statement in regel 23 **niet** waar is de computer de code onder regel 25 zal uitvoeren.
10. Op regel 26 roepen we de priem functie aan. ( met tab ervoor).
11. Druk op run en test je programma. Als (if) je keuze een 1 is start hij de ggd functie en anders (else) start hij de priem functie.