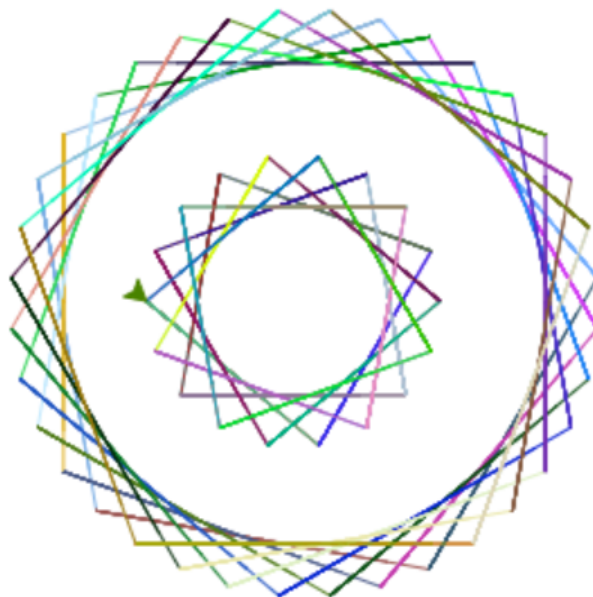


Spirograaf in Python

Een kunstwerk maken

Met programmeren kun je alles maken! Ook een kunstwerk!

In deze les maken we zelf een kunstwerk met Python. Hiervoor zal je werken met herhalingen en variabelen. Weet je nog niet wat dat zijn? Geen zorgen! We zullen je alles uitleggen.



Zo gaat jouw kunstwerk er straks uitzien. Of misschien wel heel anders.

Ga naar deze website: <https://repl.it/@sdewit/Spirograaf>.

Dit materiaal is gemaakt door [Feliene](#) en Shirley de Wit. Het is Creative Commons [by-nc-sa-4.0](#)

Simpel gezegd: Je mag het gebruiken in je lessen, aanpassen, uitprinten, kopiëren, wat je maar wilt.

Maar: Je moet mijn naam erbij zetten, je mag er geen geld mee verdienen en als je het aanpast, moet je dat ook weer Creative Commons maken. Het linkje bevat alle informatie.

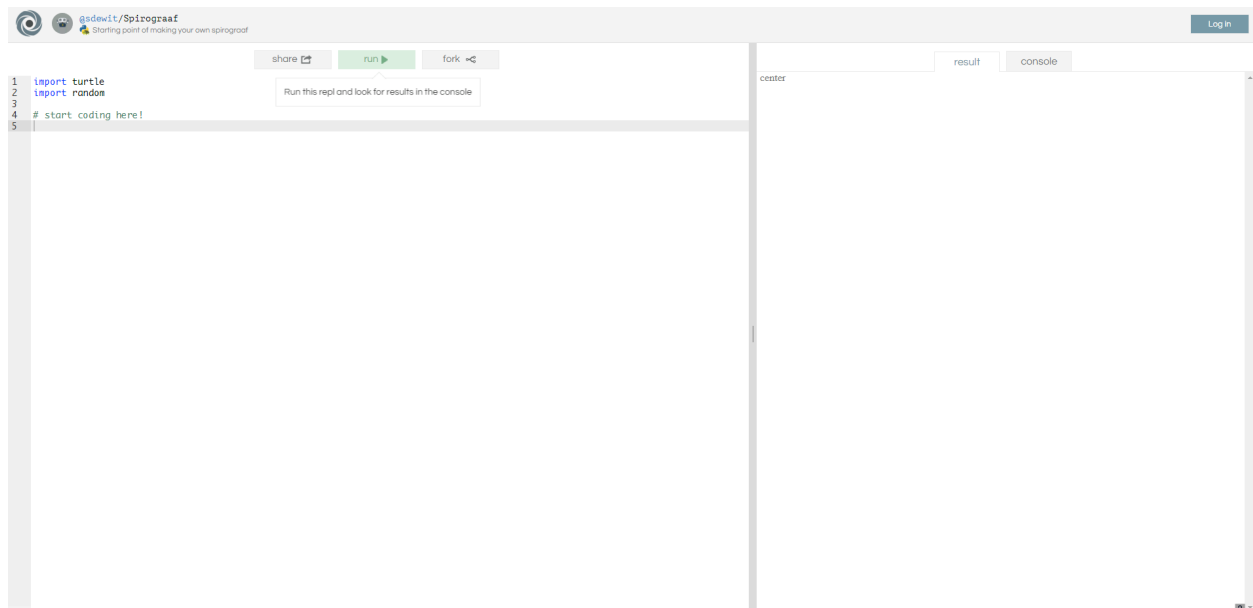
Voordat we gaan beginnen...

Hier zie je het startpunt van jouw programma. Je komt op de onderstaande pagina terecht. Aan de linkerkant van het scherm vindt je de code. Je ziet in de linker kantlijn nummers staan. Deze nummers geven het regelnummer aan.

Aan de rechterkant zie je het resultaat van het programma. In het tab 'result' zie je het resultaat als je iets tekent, zoals wij gaan doen. Wanneer je een tekstuele output verwacht of een foutmelding krijgt zal je die vinden in het tabblad 'console'. Als je nu op 'Run' klikt zie je nog niks gebeuren. Dit klopt, we hebben tenslotte nog niks geprogrammeerd!

Wanneer je probeert om zelf code te typen zal je een melding dat je het project moet 'forken' om het op te kunnen slaan. Door een project te 'forken' maak je een kopie van het project in je eigen account.

Als je het project straks wilt opslaan, klik je op 'fork' en log in als je al een account hebt in Repl en maak anders een account aan.



Het potlood over het scherm bewegen.

Nu gaan we echt beginnen met het programmeren.

Op de eerste regel van de code zie je 'import turtle' staan. Hierdoor kunnen we de library Turtle Graphics gebruiken. Dit betekent dat andere mensen al dingen geprogrammeerd hebben die wij kunnen hergebruiken. In dit geval een pen die je onder andere een locatie, kleur en breedte kan meegeven. Handig want zo kan je snel aan de slag met het maken van jouw spirograaf!

Om gebruik te maken van Turtle beginnen we met een instantie aanmaken van een turtle. Dit kun je doen door een regel toe te voegen zoals hieronder op regel 4 te zien is.

```
1 import turtle
2
3 # make an instance of turtle
4 t = turtle.Turtle()
5
```

Het valt je misschien op dat de tekst op regel 3 er anders uit ziet. Wat je hier ziet is een comment, oftewel een opmerking. Je kunt deze herkennen doordat ze beginnen met '#' en vaak een andere kleur krijgen in het programma (de editor) waarin je programmeert. Door comments toe te voegen maak je je code voor jezelf en voor andere duidelijker te begrijpen. Programmeren doe je vaak in teams, waardoor dit extra belangrijk is. De comments die in deze code staan zijn in het Engels. Dit omdat je vrijwel altijd in het Engels programmeert. Vind jij het makkelijker om de comments in het Nederlands te schrijven? Doe dat dan vooral!

In de variabele 't' zit nu een Turtle. Dit klinkt misschien nog een beetje abstract. Maak je geen zorgen, het wordt waarschijnlijk een stuk duidelijker wanneer je verder gaat met programmeren.

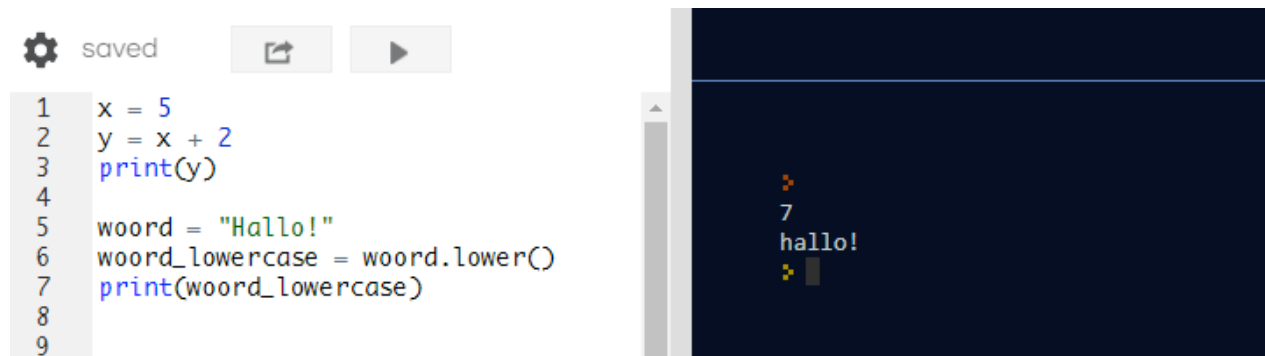
Weet je nog niet wat een variabele is? Lees dan verder in onderstaand blokje. Ben je wel al bekend met het concept van een variabele? Dan kun je verder onder het blokje.

Wat is een variabele?

Waarschijnlijk kom je in je wiskunde les al variabelen tegen zoals x en y. Soms weet je al wat de waarde van x of y is en soms moet je deze uitrekenen. Je kunt de variabele gebruiken in bijvoorbeeld een formule.

Een variabele in een stuk software is vergelijkbaar. Je kunt er een waarde in opslaan waarvan je wilt dat de computer het onthoudt. Deze waarde kan een getal zijn, zoals in de wiskunde, maar kan ook een stukje tekst zijn maar ook een afbeelding. Afhankelijk van het soort waarde die je opslaat kun je de computer operaties laten uitvoeren met je variabelen. Wanneer je een getal opslaat kun je daar, net als bij wiskunde, berekeningen mee doen zoals optellen en vermenigvuldigen. Als je er een andere waarde in opslaat, bijvoorbeeld een stukje tekst kun je daar ook operaties op uitvoeren. Denk bijvoorbeeld aan het splitsen van een zin in aparte woorden of zorgen dat er geen hoofdletters in je woord of tekst zitten.

Hieronder zie je een voorbeeld met zowel een getal als met een stuk tekst.



```
saved [share] [run]
1 x = 5
2 y = x + 2
3 print(y)
4
5 woord = "Hallo!"
6 woord_lowercase = woord.lower()
7 print(woord_lowercase)
8
9
```

```
7
hallo!
```

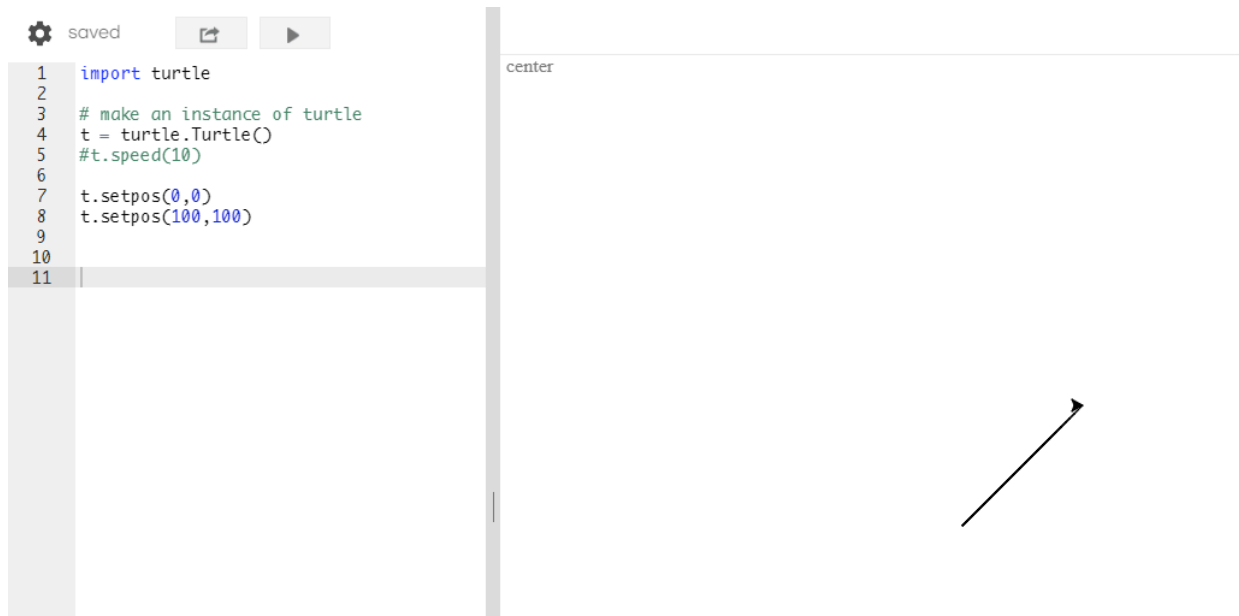
Probeer ook zelf om een variabele te maken en bijvoorbeeld een getal te vermenigvuldigen of delen. Je ziet de prints verschijnen in de 'console'.

Nu is het tijd om de cursor oftewel het potlood in het midden van het resultaat scherm te krijgen. Dit kun je doen door een x en y coördinaten aan het potlood te geven. Je scherm heeft namelijk een x en y as, waarbij het middelpunt in het midden ligt. Je kunt de locatie meegeven aan het potlood (oftewel de variabele 't') door de 'setpos(x,y)' operatie, waarbij je een x en y invult. Probeer dit te programmeren.

Wanneer het gelukt is om het potlood in het midden te krijgen geef je het potlood op een nieuwe regel een nieuwe locatie. Wat gebeurt er nu?

Mocht je er niet helemaal uit komen, vraag dan om hulp of kijk op de volgende pagina.

Hieronder zie je een stuk code die het potlood van het midden van het scherm naar positie 100,100 laat gaan waardoor er een lijn ontstaat!



```
1 import turtle
2
3 # make an instance of turtle
4 t = turtle.Turtle()
5 #t.speed(10)
6
7 t.setpos(0,0)
8 t.setpos(100,100)
9
10
11
```

The drawing window shows a horizontal line from the center to the right. An arrow points from the center towards the right, indicating the direction of the line.

Je kan nu natuurlijk een tekening maken door een aantal keer de positie van de pen te veranderen, maar dit zou wel erg veel werk zijn. Daarom gaan wij dat op een slimme manier doen. Wanneer je de pen hebt neergezet in het midden, kun je ook de operatie 'forward(size)' gebruiken.

Probeer dit uit in je code en zorg ervoor dat er een rechte lijn getekend wordt.

Draaien maar

Voor een mooie spirograaftekening willen we natuurlijk niet alleen een rechte lijn. we willen ook mooie bochtjes. Daarvoor moet de pen gaan draaien. Daar is ook een operatie voor. Om het potlood linksom te laten draaien gebruiken we 'left(angle)'. Laat de pen een stukje tekenen door de 'forward' operatie te gebruiken, te draaien met 'left' en vervolgens weer een stuk te tekenen. Probeer verschillende hoeken uit.

Om rond te gaan moet je het draaien en vooruit gaan een aantal keer herhalen. Dat mag je proberen, maar dat is dus veel werk! Kijk maar eens op de volgende pagina.

```
saved [run] [stop]
1 import turtle
2
3 # make an instance of turtle
4 t = turtle.Turtle()
5
6 # set t in the middle
7 t.setpos(0,0)
8
9 # take several steps
10 t.forward(100)
11 t.left(50)
12 t.forward(100)
13 t.left(50)
14 t.forward(100)
15 t.left(50)
16 t.forward(100)
17 t.left(50)
18 t.forward(100)
19 t.left(50)
20 t.forward(100)
21 t.left(50)
22 t.forward(100)
23
24
25
```

```
1 for i in range(4):
2     print(i)
3
```

Herhaalverhaal

Het herhalen van steeds dezelfde code kunnen we een stuk slimmer doen dan we hierboven gezien hebben. We kunnen namelijk een stukje code meerdere keren herhalen. We doen dit met een 'for-loop'. Typ het volgende stukje code in je editor

Wat verwacht je nu als output? Kijk in je console of de uitkomst overeen komt met je verwachting. Ga na wat er gebeurt en vraag het vooral als je het niet helemaal begrijpt!

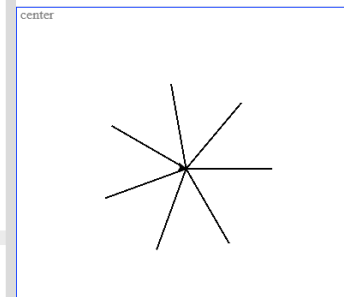
Gebruik nu de 'for-loop' om een cirkel te tekenen.

Let op! In Python gebruik je tabs aan het begin van een regel om aan te geven of een regel nog binnen de loop hoort. Hieronder zie je twee stukken code waarbij alleen de positie van regel 15

```
saved [run] [stop]
1 import turtle
2
3 # make an instance of turtle
4 t = turtle.Turtle()
5
6 # set t in the middle
7 t.setpos(0,0)
8
9 # take 7 steps
10 for i in range(7):
11     t.forward(100)
12     t.left(50)
13
14 # go back to the middle
15 t.setpos(0,0)
16
```

anders is. Je ziet dat de tekening hierdoor heel anders is! Snap jij wat er precies gebeurt in beide situaties?

```
saved [share] [run]
1 import turtle
2
3 # make an instance of turtle
4 t = turtle.Turtle()
5
6 # set t in the middle
7 t.setpos(0,0)
8
9 # take 7 steps
10 for i in range(7):
11     t.forward(100)
12     t.left(50)
13
14 # go back to the middle
15 t.setpos(0,0)
16
```



Tijd voor variatie

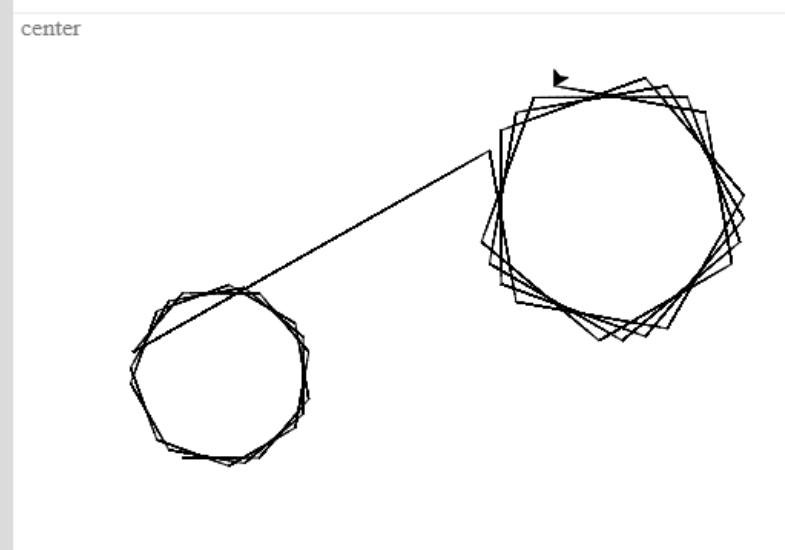
Helaas krijgen we nu steeds dezelfde tekening. Zullen we eens wat variatie toevoegen?

Zorg ervoor dat je vaker over regel 11 en 12 heen gaat lopen. Speel daarna met de lengte van de lijn en de hoek die het potlood maakt. Krijg je nu verschillende figuren?

Probeer ook een tweede rondjes te tekenen op een andere locatie.

Hieronder zie je een voorbeeld van hoe je tekening er nu uit kan zien. Wil je dat je figuur ronder of minder rond wordt? Pas dan het aantal herhalingen aan.

```
saved [share] [run]
1 import turtle
2
3 # make an instance of turtle
4 t = turtle.Turtle()
5
6 # set t in the middle
7 t.setpos(0,0)
8
9 # take 20 steps
10 for i in range(20):
11     t.forward(50)
12     t.left(50)
13
14 # go to another position
15 t.setpos(200, 200)
16
17 # take 20 steps
18 for i in range(20):
19     t.forward(100)
20     t.left(70)
21
```



Misschien heb je ook soms zo'n streepje in je rondje zoals hierboven. Dat is niet mooi, hè? We kunnen dat oplossen door de pen op te tillen voordat we het potlood verplaatsen. Gebruik hiervoor de 'penup()' en 'pendown()' operaties.

Tip! Wil je dat je tekening sneller of langzamer gemaakt wordt? Voeg dan nog een 'speed(z)' operatie toe aan t.

Er zijn nog wat leuke dingen die je kan doen om je tekening mooier te maken.

Kleurtjes

Alle tekeningen zijn nu zwart. Dat is best mooi, maar misschien wil je wel iets anders.

Je kunt de kleur veranderen door de operatie 'color(r,g,b)' toe te passen op het potlood. Je kunt een kleur kiezen door de RGB (Rood, Groen, Blauw) code in te vullen. Deze RGB code kun je op verschillende manieren vinden, bijvoorbeeld door gebruik te maken van <http://www.rgbtool.com/>.

Willekeurige tekeningen

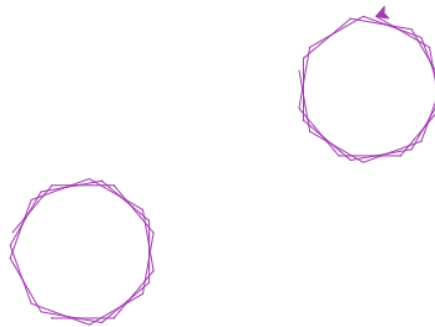
Tot slot gaan we de tekening nog wat gekker maken. Vind je het ook een beetje saai dat je steeds dezelfde tekening krijgt? Zou het niet leuker zijn als Python zelf wat verzint?

We gaan hiervoor gebruik maken van variabelen voor de lengte van de lijnen en de graden. Maak een variabele aan voor de lengte en graden en geef ze een duidelijke naam, bijvoorbeeld 'length' en 'angle'. Geef beide variabelen een waarde. Vervang nu overal in je code de getallen die de lengte en hoek representeren.

Wanneer je dit gedaan hebt ziet je code er ongeveer zo uit

```
1 import turtle
2
3 # make an instance of turtle
4 t = turtle.Turtle()
5
6 # set the speed of t
7 t.speed(20)
8
9 # set the color of t
10 t.color(159, 51, 176)
11
12 length = 50
13 angle = 50
14
15 # set t in the middle
16 t.setpos(0,0)
17
18 # take 20 steps
19 for i in range(20):
20     t.forward(length)
21     t.left(angle)
22
23 # go to another position
24 t.penup()
25 t.setpos(200, 200)
26 t.pendown()
27
28 # take 20 steps
29 for i in range(20):
30     t.forward(length)
31     t.left(angle)
32
```



center



Nu zijn de figuren precies hetzelfde geworden terwijl we juist een andere tekening wilden. Hiervoor gaan we met random getallen werken. We laten Python zelf een getal kiezen!



Het kiezen van een random getallen is iets wat gelukkig al geprogrammeerd is. Met een extra import kunnen we de 'random' operatie gebruiken. Typ op regel 2: 'import random'.

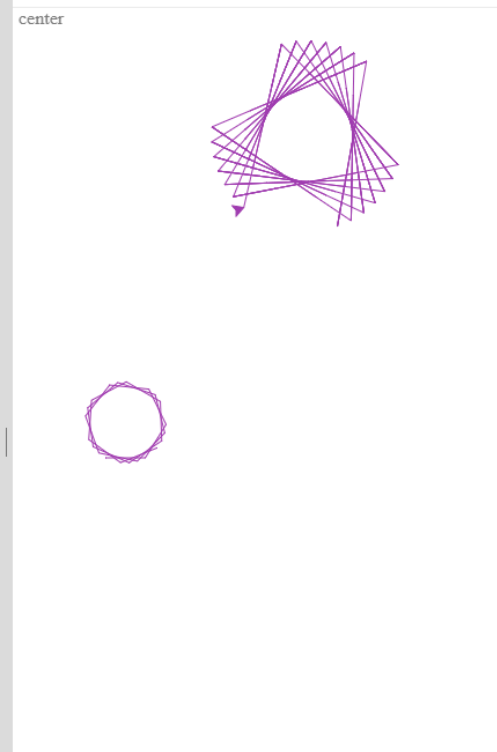
We kunnen nu een random getal (0, 1, 2, ..., enz.), ook wel integer genoemd, krijgen door de volgende code: 'random.randint(min,max)' waarbij je een minimale en maximale waarde meegeeft. Run de volgende code zelf een aantal keer met verschillende minimale en maximale waarde.

```
saved    
1 import random  
2  
3 random_number = random.randint(10,15)  
4 print(random_number)
```

Gebruik de 'random.randint(min,max)' om de lengte en de hoek random te maken.

De tekening ziet er al iets anders uit. Een voorbeeld zie je hieronder.

```
saved    
1 import turtle  
2 import random  
3  
4 # make an instance of turtle  
5 t = turtle.Turtle()  
6  
7 # set the speed of t  
8 t.speed(20)  
9  
10 # set the color of t  
11 t.color(159, 51, 176)  
12  
13 # set random length and angle  
14 length = random.randint(10,150)  
15 angle = random.randint(10,150)  
16  
17 # set t in the middle  
18 t.setpos(0,0)  
19  
20 # take 20 steps  
21 for i in range(20):  
22     t.forward(length)  
23     t.left(angle)  
24  
25 # set random length and angle  
26 length = random.randint(10,150)  
27 angle = size = random.randint(10,150)  
28  
29 # go to another position  
30 t.penup()  
31 t.setpos(200, 200)  
32 t.pendown()  
33  
34 # take 20 steps  
35 for i in range(20):  
36     t.forward(length)  
37     t.left(angle)  
38
```



Aan jou de taak om er voor te zorgen dat er meerdere cirkels op een random locatie getekend worden. Gebruik hiervoor de concepten die je tot nu toe geleerd hebt, zoals variabelen, herhalingen en random getallen.

Ben je klaar? Goed gedaan!

Hier zijn nog wat dingen die je kan proberen om je code te verbeteren en nog leukere resultaten te krijgen.

- Zorg ervoor dat de kleuren van je potlood tijdens het tekenen veranderd

- Kun jij de lengte en hoek tijdens het tekenen aanpassen?
- Denk je dat jouw code begrijpbaar en leesbaar is voor anderen? Zo niet, probeer je code duidelijker te structureren en comments toe te voegen om uit te leggen wat je doet.
- Er kan nog veel meer met de Turtle library! Kijk op <https://docs.python.org/2/library/turtle.html> om te zien wat er nog meer mogelijk is. Lukt het jou nu om de dikte van de pen aan te passen? Of een stempel te gebruiken? Probeer vooral dingen uit als je iets leuks zit in de library of als je benieuwd bent wat het is.