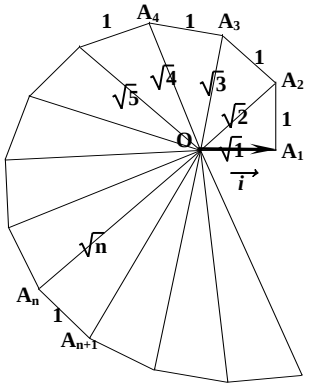
Algorithme de tracé du colimaçon de Pythagore.



Le colimaçon de Pythagore est une figure permettant de construire à partir d'un segment de longueur 1, des segments de longueur les racines carrés des entiers naturels successifs. On a donc : $1 = A_1A_2 = A_2A_3 = A_3A_4 = \ldots$

Les triangles tracés sont des triangles rectangles. Donc le triangle OA_1A_2 est rectangle en A_1, le triangle OA_2A_3 est rectangle en A_2 et ainsi de suite.

I. Elaboration des formules.

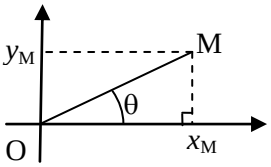
1. Soit $n \in \mathbb{N}\setminus\{0\}$. Vérifier que dans le triangle OA_nA_{n+1} rectangle en A_n, si $OA_n = \sqrt{n}$ et si $A_nA_{n+1} = 1$, alors $OA_{n+1} = \sqrt{n+1}$.

2. **a.** On note $\alpha_n = \widehat{A_nOA_{n+1}}$. Dans le triangle OA_nA_{n+1} rectangle en A_n, exprimer $\tan(\alpha_n)$ en fonction de n.

b. Lorsque l'on sait que $\tan(\alpha) = x$, alors on peut noter que l'angle $\alpha = \text{atan}(x)$. En déduire l'expression de α_n en fonction de n.

c. On note θ_n une mesure de $(\vec{i}, \overrightarrow{OA_n})$. En déduire l'expression de θ_{n+1} en fonction de θ_n et de n.

d. Sachant que M est un point quelconque du plan tel que $(\vec{i}, \overrightarrow{OM}) = \theta$.
Justifier que les coordonnées $(x_M ; y_M)$ de M sont telles que :
$x_M = OM\cos(\theta)$ et $y_M = OM\sin(\theta)$.



e. En déduire les coordonnées $(x_{n+1} ; y_{n+1})$ de A_{n+1} en fonction de n, θ_n et α_n.

3. Récapitulatif. Compléter le tableau suivant sans calculer les valeurs, mais en faisant apparaître les formules.

point	OA	α	θ	x	y
A_1	$OA_1 = \sqrt{\ldots}$	$\alpha_1 = \text{atan}\left(\frac{\ldots}{\sqrt{\ldots}}\right)$	$\theta_1 = \ldots$	$x_1 = \ldots$	$y_1 = \ldots$
A_2	$OA_2 = \sqrt{\ldots}$	$\alpha_2 = \ldots\ldots\ldots$	$\theta_2 = \ldots + \ldots$	$x_2 = \ldots \times \ldots\ldots$	$y_2 = \ldots \times \ldots\ldots$
A_3	$OA_3 = \sqrt{\ldots}$	$\alpha_3 = \ldots\ldots\ldots$	$\theta_3 = \ldots + \ldots$	$x_3 = \ldots \times \ldots\ldots$	$y_3 = \ldots \times \ldots\ldots$
A_4	$OA_4 = \sqrt{\ldots}$	$\alpha_4 = \ldots\ldots\ldots$	$\theta_4 = \ldots + \ldots$	$x_4 = \ldots \times \ldots\ldots$	$y_4 = \ldots \times \ldots\ldots$
A_n	$OA_n = \sqrt{\ldots}$	$\alpha_n = \ldots\ldots\ldots$	$\theta_n = \ldots + \ldots$	$x_n = \ldots \times \ldots\ldots$	$y_n = \ldots \times \ldots\ldots$
A_{n+1}	$OA_{n+1} = \sqrt{\ldots}$	$\alpha_{n+1} = \ldots\ldots\ldots$	$\theta_{n+1} = \ldots + \ldots$	$x_{n+1} = \ldots \times \ldots\ldots$	$y_{n+1} = \ldots \times \ldots\ldots$

II. Tracer le colimaçon.

A l'aide des résultats précédents, en partant du segment $[OA_1]$ de longueur 1, écrire un algorithme permettant de créer le colimaçon de Pythagore.
On demandera au départ jusqu'à quelle valeur de n l'utilisateur souhaite tracer la racine carré.
Programmer cet algorithme sous Python en vous aidant des renseignements suivants.

Pour pouvoir utiliser un graphique sous Python, il faut commencer le programme par : `from turtle import*`
et le terminer par : `mainloop()`

Pour pouvoir tracer un segment il faut dès le départ saisir le "sous programme" suivant :

```
def segment(x1,y1,x2,y2):        on définie l'instruction segment et ses variables
    up()                         on lève le stylo
    goto(x1,y1)                  on se déplace jusqu'au point de coordonnées cartésiennes (x1 ; y1)
```

down() *on baisse le stylo*

goto(x2,y2) *on se déplace jusqu'au point de coordonnées cartésiennes (x2 ; y2)*

Une fois que ce sous programme a été créé, on peut utiliser l'instruction :

segment(*valeur de x1,valeur de y1,valeur de x2,valeur de y2*)

Etant donné la fenêtre d'affichage très large, et donc pour que le colimaçon soit visible, il sera nécessaire de multiplier par 50 les coordonnées cartésiennes calculées.

On dispose des instructions suivantes :

Pour calculer $\sqrt{x}$: sqrt(x) exemple : $\sqrt{2x+3}$ s'écrit sqrt(2*x+3).

Pour calculer x^n : pow(x,n) exemple : x^7 s'écrit pow(x,7) .

Pour calculer un angle, connaissant sa tangente qui vaut x : atan(x) exemple : si tan(x) = 3/5 alors on obtient la valeur de x par atan(3/5) .

II. bis. Tracer le colimaçon.

Avec le module graphique graphsecondev2_3.py écrit par M. Body et disponible sur le site Planète MATHS.

Programmer cet algorithme sous Python en vous aidant des renseignements suivants.

Pour pouvoir utiliser le module graphique sous Python, il faut commencer le programme par :

from graphsecondev2_3 import*

et le terminer par : affiche()

Comme sur une calculatrice, il faut obligatoirement définir la fenêtre d'affichage avec l'instruction :

fenetre(x_{min},x_{max},y_{min},y_{max})

On dispose des instructions suivantes :

Pour tracer un segment : segment(*valeur de x1,valeur de y1,valeur de x2,valeur de y2,'couleur'*)

exemple : segment(x1,y1,0,0,'bleu') trace en bleu le segment de A(x_1 ; y_1) à B(0 ; 0).

Pour calculer $\sqrt{x}$: sqrt(x) exemple : $\sqrt{2x+3}$ s'écrit sqrt(2*x+3).

Pour calculer x^n : pow(x,n) exemple : x^7 s'écrit pow(x,7) .

Pour calculer un angle, connaissant sa tangente qui vaut x : atan(x) exemple : si tan(x) = 3/5 alors on obtient la valeur de x par atan(3/5) .

Mise en œuvre de cette activité.

Il est souhaitable que la partie **I.** soit préparée à la maison. Ce travail peut être évalué.
La deuxième partie ne peut être correctement exécutée que si un corrigé de la partie **I.** a été fourni.
Voici une ébauche d'une grille d'auto-évaluation pour l'algorithme.

capacités		je sais faire	je sais faire avec de l'aide	je comprends avec la solution	je ne comprends pas même avec la solution
Respect des consignes :	j'ai bien demandé la valeur de n				
	j'ai obtenu un colimaçon s'arrêtant à $\sqrt{n}$				
Initialisation des variables :	j'ai affecté 0 à θ au départ				
	j'ai affecté les coordonnées de A_1 à x et y au départ				
Mise en œuvre d'une boucle itérative "pour i variant de … à …"	j'ai correctement choisi la valeur de départ et de fin pour i dans la boucle				
	j'ai conservé les coordonnées du dernier point calculé pour calculer le suivant				
Auto évaluation de l'algorithme	J'ai su utiliser un tableau donnant pour chaque valeur de i les valeurs des variables				

Propositions de programmes sous Python avec le module tortue :

Cette version trace deux fois [OA_1], mais on utilise $\sqrt{i}$ aussi bien pour le calcul de α que celui de x2 et y2. La boucle est exécutée pour i variant de 1 à n.

```
from turtle import*
from math import*

def segment(x1,y1,x2,y2):
    up()
    goto(x1,y1)
    down()
    goto(x2,y2)

teta=0
x1=50
y1=50
n=int(input("indiquer jusqu'à quel entier vous souhaitez tracer la racine carré "))
for i in range(1,n+1):
    x2=sqrt(i)*cos(teta)*50
    y2=sqrt(i)*sin(teta)*50
    alpha=atan(1/sqrt(i))
    teta=teta+alpha
    segment(0,0,x1,y1)
    segment(x1,y1,x2,y2)
    x1=x2
    y1=y2
segment(x2,y2,0,0)

mainloop()
```

Cette version trace une seule fois [OA_1], mais on utilise $\sqrt{i - 1}$ pour le calcul de α et $\sqrt{i}$ pour celui de x2 et y2. La boucle est exécutée pour i variant de 2 à n.

```
from turtle import*
from math import*

def segment(x1,y1,x2,y2):
    up()
    goto(x1,y1)
    down()
    goto(x2,y2)

teta=0
x1=50
y1=50
n=int(input("indiquer jusqu'à quel entier vous souhaitez tracer la racine carré "))
segment(0,0,x1,y1)
for i in range(2,n+1):
    alpha=atan(1/sqrt(i-1))
    teta=teta+alpha
    x2=sqrt(i)*cos(teta)*50
    y2=sqrt(i)*sin(teta)*50
    segment(0,0,x2,y2)
    segment(x1,y1,x2,y2)
    x1=x2
    y1=y2
```

mainloop()

Propositions de programmes sous Python avec le module graphique de M. Body téléchargeable sur Planète Maths:

Cette version trace deux fois le segment [OA_1], mais on utilise $\sqrt{i}$ aussi bien pour le calcul de α que celui de x2 et y2. La boucle est exécutée pour i variant de 1 à n.

```
from graphsecondev2_3 import*

teta=0
x1=1
y1=0
n=int(input("indiquer jusqu'à quel entier vous souhaitez tracer la racine carré "))
fenetre(-1*sqrt(n+2),sqrt(n+2),-1*sqrt(n+2),sqrt(n+2))
for i in range(1,n+1):
   x2=sqrt(i)*cos(teta)
   y2=sqrt(i)*sin(teta)
   alpha=atan(1/sqrt(i))
   teta=teta+alpha
   segment(0,0,x1,y1,'rouge')
   segment(x1,y1,x2,y2,'vert')
   x1=x2
   y1=y2
segment(x2,y2,0,0,'rouge')

affiche()
```

Cette version trace une seule fois [OA_1], mais on utilise $\sqrt{i - 1}$ pour le calcul de α et $\sqrt{i}$ pour celui de x2 et y2. La boucle est exécutée pour i variant de 2 à n.

```
from graphsecondev2_3 import*

teta=0
x1=1
y1=0
n=int(input("indiquer jusqu'à quel entier vous souhaitez tracer la racine carré "))
fenetre(-1*sqrt(n+2),sqrt(n+2),-1*sqrt(n+2),sqrt(n+2))
segment(0,0,1,0,'rouge')
for i in range(2,n+1):
   alpha=atan(1/sqrt(i-1))
   teta=teta+alpha
   x2=sqrt(i)*cos(teta)
   y2=sqrt(i)*sin(teta)
   segment(0,0,x2,y2,'rouge')
   segment(x1,y1,x2,y2,'vert')
   x1=x2
   y1=y2

affiche()
```