Spécialité Mathématiques ⇨ Informatique
TP 1 : Comment programme-t-on des suites en Python ?

Les programmes demandés seront tous tapés dans le même fichier, séparés par une ligne de commentaire du type : """Etape 1""" … Ce fichier sera ensuite déposé dans Pronote pour la rentrée.

Etape n°1 : **Boucle bornée**

1 **Devinez** ce qu'affiche le programme suivant :

Programme	*Affichage*

```
def u(n):
    return 1-3*n
for n in range(10):
    print(u(n))
```

2 Ecrire un programme pour qu'il affiche les vingt premiers termes de la suite définie, pour tout entier naturel non nul n, par $\sqrt{n^3-1}$.

Etape n°2 : **Condition**

1 **Devinez** ce qu'affiche le programme suivant :

Programme	*Affichage*

```
def u(n):
    return n**2
for n in range(20):
    if u(n)>100:
        print(n,u(n))
    else:
        print(n,"Trop petit")
```

2 **Devinez** ce qu'affiche le programme suivant :

Programme	*Affichage*

```
from random import *
def u(n):
    return randint(0,n)
for n in range(50):
    a=u(n)
    if a%2==0:
        print(n,a)
    else:
        print(n,"impair")
```

3 Ecrire un programme pour qu'il calcule les cinquante premiers termes de la suite définie, pour tout entier naturel n, par $u_n = 2^n$, affiche le terme s'il est plus petit qu'un million et affiche « Trop grand » sinon.

Etape n°3 : **Boucle non bornée**

1 **Devinez** ce qu'affiche le programme suivant :

Programme	*Affichage*

```
def u(n):
    return 7*n+10
n=0
while u(n)<10**6:
    n=n+1
print(n,u(n))
```

2 Ecrire un programme pour qu'il affiche le rang du premier terme de la suite définie, pour tout entier naturel n, $u_n = 3 + n^2$ qui passe sur le seuil du milliard.

Etape n°4 : **Suite définie par récurrence**

1 **Devinez** ce qu'affiche le programme suivant :

Programme	***Affichage***

`def u(n):`
`  a=0`
`  for i in range(n):`
`  a=3*a+1`
`  return a`
`print(u(4))`	

2 **Devinez** ce qu'affiche le programme suivant :

Programme	***Affichage***

`def u(n):`
`  if n==0:`
`  a=1`
`  else:`
`  a=2*u(n-1)+1`
`  return a`
`for n in range(10):`
`  print(n,u(n))`	

3 Ecrire un programme pour qu'il affiche les dix premiers termes de la suite définie par $u_0 = 3$ et, pour tout entier naturel n, par $u_{n+1} = \frac{u_n^2}{2} - 3$.

Etape n°5 : **Créer un affichage**

1 **Devinez** ce qu'affiche le programme suivant :

Programme	***Affichage***

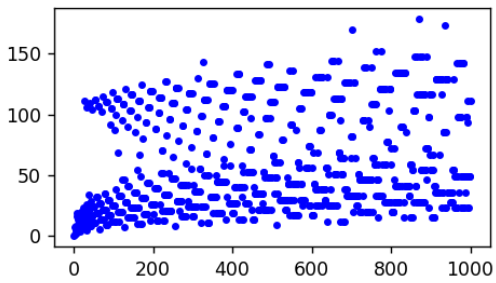
`from math import *`
`import matplotlib.pyplot as plt`
`def u(n):`
`  return sin(n/100)`
`X=[]`
`Y=[]`
`for n in range(628):`
`  X.append(n)`
`  Y.append(u(n))`
`plt.plot(X,Y,'bo')`
`plt.show()`
`plt.close()`	

2 Ecrire un programme pour qu'il place les trente premiers termes de la suite définie, pour tout entier naturel n, par $u_n = (-1{,}1)^n$.

Etape n°6 : **Suite de Syracuse**

On appelle **suite de Syracuse** des nombres qui se suivent en utilisant le principe de calcul suivant. On choisit le premier terme de la suite.

- Si le terme est pair alors le terme suivant est la moitié de ce terme.
- Si le terme est impair alors le terme suivant est le triple de ce terme augmenté de 1.



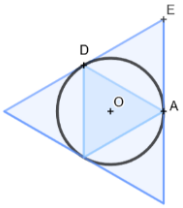
1 Créer une fonction qui, à partir d'un terme de la suite, calcule les termes suivants jusqu'à obtenir 1.

2 On appelle ***Temps de vol de la suite de Syracuse***, le nombre de termes nécessaires pour atteindre 1. Modifier la fonction précédente pour qu'à partir d'un premier terme de la suite, elle affiche le temps de vol.

3 Modifier l'algorithme pour qu'il fasse varier le 1er terme de 1 à 1000 et qu'il crée une liste contenant les temps de vol de chaque suite de Syracuse obtenue alors.
4 Modifier l'algorithme pour qu'il place des points où l'abscisse est le 1er terme et l'ordonnée son temps de vol.

Etape n°7 : **Comment Archimède a-t-il approximé π ?**

Archimède, dans De la mesure du cercle, a créé le premier algorithme pour le calcul de π, basé sur l'idée que le périmètre de n'importe quel polygone inscrit dans un cercle est inférieur à la circonférence du cercle qui, à son tour, est inférieur au périmètre de tout polygone circonscrit à ce cercle. Soit $n \in \mathbb{N}^*, n \geq 3$, on note P_n et G_n les périmètres respectifs des polygones réguliers à n côtés, inscrits et circonscrits autour du même cercle de rayon 1.

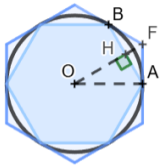


1 Calculer P_3 et G_3.
2 Démontrer que $G_4 = 8$ et $P_4 = 4\sqrt{2}$.
3 Archimède établit une méthode pour calculer les périmètres des polygones réguliers ayant deux fois plus de côtés qui sont inscrits et circonscrits autour du même cercle. Les formules obtenues sont :

$$\forall n \in \mathbb{N}^*, n \geq 3, \quad G_{2n} = \frac{2P_nG_n}{P_n + G_n} \text{ et } P_{2n} = \sqrt{P_nG_{2n}}$$

a) Ecrire un algorithme qui permet de calculer G_{4096} et P_{4096}.
b) En déduire une approximation de π.

4 **APPROFONDIR**



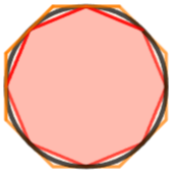
a) Démontrer que $\forall n \in \mathbb{N}^*, n \geq 3$, $P_n = 2n \sin\left(\frac{\pi}{n}\right)$ et $G_n = 2n \tan\left(\frac{\pi}{n}\right)$.
b) Donner les formules d'addition du cosinus et du sinus :

$$\forall a \in \mathbb{R}, \forall b \in \mathbb{R} \quad \cos(a+b) = \cdots \text{ et } \sin(a+b) = \cdots$$

c) En déduire que $\forall a \in \mathbb{R}$,

$$\sin(2a) = 2\cos(a)\sin(a)$$
$$\text{et que } \cos(2a) = 2\cos^2(a) - 1.$$

d) Démontrer alors les formules obtenues par Archimède pour P_{2n} et G_{2n}.



Etape n°8: **Vitesse de convergence d'une suite**

1 Algorithme de Babylone :

Soit a un nombre réel positif, on pose $u_0 = 1$ et, $\forall n \in \mathbb{N}, u_{n+1} = \frac{1}{2}\left(u_n + \frac{a}{u_n}\right)$

a) Démontrer, par récurrence, que $\forall n \in \mathbb{N}, u_n > 0$.
b) Pour tout $n \in \mathbb{N}$, démontrer que $u_{n+1} - \sqrt{a} = \frac{(u_n - \sqrt{a})^2}{2u_n}$
c) En déduire que, pour tout $n \in \mathbb{N}^*$, $u_n \geq \sqrt{a}$.
d) Pour tout $n \in \mathbb{N}$, démontrer que $u_{n+1} - u_n = \frac{(\sqrt{a} + u_n)(\sqrt{a} - u_n)}{2u_n}$
e) En déduire que la suite (u_n) est décroissante à partir du rang 1. La suite (u_n) est-elle convergente ?
f) Résoudre l'équation $\frac{1}{2}\left(x + \frac{a}{x}\right) = x$. Que peut-on en déduire ?
g) Programmer cet algorithme pour obtenir une valeur approchée de $\sqrt{2}$.

2 Algorithme de dichotomie :

a) Encadrer $\sqrt{2}$ par deux entiers naturels notée a et b où $a < b$.
b) Ecrire une fonction dichotomie qui partant de a et b, va prendre la moyenne des deux, tester si cette moyenne est inférieure ou supérieure à $\sqrt{2}$ puis remplacer a par cette moyenne dans le 1er cas et remplacer b par cette moyenne dans le 2e cas.
c) Répéter cette fonction jusqu'à obtenir la précision obtenue dans l'algorithme de Babylone.

NOMBRE ENTIER -NOMBRE REEL	`Int( )  float( )`
BOOLEENS	`True  False`
AFFICHER DU TEXTE OU UNE VALEUR	`print( )`
DEMANDER DU TEXTE OU UNE VALEUR	`input( )`
QUOTIENT DE LA DIVISION EUCLIDIENNE	`//`
RESTE DE LA DIVISION EUCLIDIENNE	`%`
PUISSANCE	`**`
EGALITE - DIFFERENT	`==  !=`
INEGALITE STRICTE	`<  >`
INEGALITE LARGE	`<=  >=`
MODULE ALEATOIRE	`from random import *`
NOMBRE ENTIER ENTRE BORNES	`randint(`*min*`,`*max*`)`
MODULE MATH	`from math import *`
RACINE CARREE	`sqrt( )`
MODULE DE GRAPHIQUE	`import matplotlib.pyplot as plt`
PLACER UN POINT (X,Y)	`plt.scatter( )`
MONTRER – FERMER LA FENETRE	`plt.show()  plt.close()`
LISTE VIDE	*nom* `=[]`
AJOUT D'UN ELEMENT	*nom*`.append(`*élément*`)`
1ER ELEMENT	*nom*`[0]`
2E ELEMENT	*nom*`[1]` …
NOMBRE D'ELEMENT	`len(`*nom*`)`
INVERSE LES ELEMENTS	*nom*`.reverse()`
SUPPRIME LA VALEUR	*nom*`.remove(`*valeur*`)`
ORDONNE LA LISTE	`sorted(`*nom*`)`

Fonction

```
def nom(parametre1,parametre2):
    instructions
    return valeur
```

Boucle bornée (qui commence à 0)

```
for i in range(nombre):
    instructions
```

Boucle bornée (qui s'arrête à *fin*-1)

```
for i in range(début,fin):
    instructions
```

Condition

```
if condition1:
    instructions si condition1 Vraie
elif condition2:
    instructions si condition2 Vraie
else:
    instructions si condition 1 et 2 fausses
```

Boucle non bornée

```
while condition:
    instructions si condition vraie
```