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 : **Afficher les termes d'une suite définie explicitement**

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

Programme	*Affichage*

`def u(n):`
`  return 3*n-2`
`print(u(5))`	

2 Ecrire un programme pour qu'il affiche les termes de rang 0 et 7 de la suite définie, pour tout entier naturel n, par $u_n = \frac{1}{n+1}$.

Etape n°2 : **Saisir, entrer une valeur**

1 En entrant 7 dans la variable n, **devinez** ce qu'affiche le programme suivant :

Programme	*Affichage*

`from math import sqrt`
`def u(n):`
`  return sqrt(1+n**2)`
`n=int(input("Quel rang souhaitez-vous ?"))`
`print(u(n))`	

2 Ecrire un programme pour qu'il affiche les termes souhaités de la suite définie, pour tout entier naturel n, par $u_n = 2 \times 3^n$.

Etape n°3 : **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°4 : **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°5 : **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°6 : **Suite définie par récurrence avec boucle**

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 Ecrire un programme pour qu'il affiche les termes de rang 2 et 7 de la suite définie par $u_0 = 1$ et, pour tout entier naturel n, par $u_{n+1} = \frac{1}{u_n+1}$.

Etape n°7 : **Suite définie par récurrence récursivité de la fonction**

1 **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))
```

2 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 $u_{n+1} = 2u_n - 3$.

Etape n°8 : **Créer un affichage**

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

Programme	***Affichage***

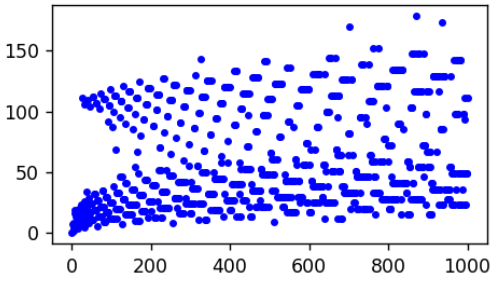
```
from math import *
import matplotlib.pyplot as plt
def u(n):
    return sin(n/100)
X=[]
Y=[]
for n in range(628):
   X.append(n)
   Y.append(u(n))
plt.plot(X,Y,'bo')
plt.show()
plt.close()
```

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°9 : **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 le terme suivant.

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

3 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.

5 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.

6 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°10 : **Le Juste Prix**

L'ordinateur choisit un nombre au hasard compris entre 1 et 100 et vous devez le deviner.
Vous pouvez faire des propositions et l'ordinateur répond "trop grand" ou "trop petit" ou encore "gagné".
Le jeu s'arrête lorsque vous avez trouvé le nombre caché et le programme vous dit combien de propositions vous avez faites.

1 Programmer ce jeu en python.
2 Quelle est la stratégie gagnante à ce jeu ?
3 En combien de propositions en moyenne réussissez-vous à gagner ?

Glossaire des instructions en Python

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
```