Listes Python [I] et programmation

```
1 liste=[1,2,3]
2 print(liste)
3 print("liste[0]= ",liste[0])
```

```
deg PYTHON
>>> from tuto import *
[1, 2, 3]
liste[0]=  1
>>>
```

```
1 liste=[i for i in range(6)]
2 print(liste)
3
```

```
deg PYTHON
>>> from tuto import *
[0, 1, 2, 3, 4, 5]
>>>
```

```
1 liste=[i for i in range(1,6)]
2 print(liste)
3
```

```
deg PYTHON
>>> from tuto import *
[1, 2, 3, 4, 5]
>>>
```

```
1 liste=[i for i in range(1,13,2)]
2 print(liste)
3
```

```
deg PYTHON
>>> from tuto import *
[1, 3, 5, 7, 9, 11]
>>>
```

```
1 liste=[i for i in range(13,1,-2)]
2 print(liste)
3
```

```
deg PYTHON
>>> from tuto import *
[13, 11, 9, 7, 5, 3]
>>>
```

```
1 liste=[10**i for i in range(0,-6,-1)]
2 print(liste)
3
```

```
deg PYTHON
>>> from tuto import *
[1, 0.1, 0.01, 0.001, 0.0001,
>>>
```

Quelques instructions utiles pour programmer :

- La longueur d'une liste nommée Liste est $\mathrm{len}(\mathrm{Liste})$

```
1 liste=[1,1,1,1,1,1,1,1]
2 print(len(liste))
3
```

```
deg PYTHON
>>> from tuto import *
8
>>>
```

Listes Python [I] et programmation

- La commande «input » pour inviter l'utilisateur à entrer une valeur numérique au clavier

```
1  a=float(input("a=? "))
2  b=int(input("b=? "))
```

```
deg                PYTHON
>>> from tuto import *
a=? 2
b=? |
```

- La commande « print » pour afficher

```
1  a=2
2  print(a)
```

```
deg                PYTHON
>>> from tuto import *
2
>>> |
```

```
1  print("Alice et Bob","et Toto")
```

```
deg                PYTHON
>>> from tuto import *
Alice et Bob et Toto
>>> |
```

```
1  a=2
2  b=5
3  print("f(",a,")=",b)
```

```
deg                PYTHON
>>> from tuto import *
f( 2 )= 5
>>> |
```

- Définir une fonction

```
1  def f(x,y,z):
2    return x+y+2*z
3
4  print (f(1,2,3))
```

```
deg                PYTHON
>>> from tuto import *
9
>>> |
```

- Boucle non bornée « Tant que »

```
1  n=0
2  u=4
3  while u<100:
4    u=3*n+4
5    n=n+1
6  print(n)
```

```
deg                PYTHON
>>> from tuto import *
33
>>> |
```

Listes Python [I] et programmation

- Quelques applications

1) Impression d'une liste de coefficients directeurs de sécantes à une courbe passant par les point A d'abcsisse a et M d'abscisse a+h

```
1  liste=[10**i for i in range(0,-10,-1)]
2  def f(x):
3    return x**3+2*x-3
4  def secante(a,liste_h):
5      coeff_dir=[(f(a+h)-f(a))/h
6      for h in liste_h]
7      return coeff_dir
8  liste2=secante(1,liste)
9  for k in range(10):
10   print(liste2[k])
```

```
deg                PYTHON
>>> from tuto import *
9.0
5.3100000000000006
5.030100000000015
5.003000999999507
5.000300009996295
5.000030000124056
5.000002999633324
5.000000302679553
4.999999969612645
5.000000413701855
>>>
```

2) Résolution d'une équation du second degré

Racine d'un trinôme de la forme $ax^2 + bx + c$ en traitant tous les cas possibles (on admettra cependant qu'il s'agit bien d'un trinôme et donc $a \neq 0$).	
Python en utilisant une saisie (*input*) de la part de l'utilisateur	

```
from math import*
a = float(input("Coefficient a ? "))
b = float(input("Coefficient b ? "))
c = float(input("Coefficient c ? "))

Delta = b**2 - 4 * a * c

if Delta < 0:
    print("Le trinôme n'a pas de racine réelle.")
elif Delta == 0:
    x0 = -b / (2 * a)
    print("Le trinôme a une seule racine :", x0, ".")
else:
    x1 = (-b - sqrt(Delta))/(2 * a)
    x2 = (-b + sqrt(Delta))/(2 * a)
    print("Le trinôme a deux racines :", x1, "et", x2, ".")
```

Remarque importante : quelques instructions doivent être importées d'une bibliothèque spécifique « math ». C'est le cas de la racine carré « sqrt », du nombre « pi » par exemple. L'instruction « from math import * » en début de programme permet cette importation.

Listes Python [I] et programmation

Python en utilisant une fonction	

```python
from math import*

# Définition de la fonction
def Racines(a,b,c):
    Delta = b**2 - 4 * a * c
    if Delta < 0:
        print("Le trinôme n'a pas de racine réelle.")
    elif Delta == 0:
        x0 = -b / (2 * a)
        print("Le trinôme a une seule racine :", x0, ".")
    else:
        x1 = (-b - sqrt(Delta))/(2 * a)
        x2 = (-b + sqrt(Delta))/(2 * a)
        print("Le trinôme a deux racines :", x1, "et", x2, ".")

# Utilisation de la fonction sur 3 exemples :
Racines(1, 2, 1)
Racines(1, -1, -6)
Racines(1 , 2 , 2)
```

Remarque : lorsque ligne de code Python commence par le caractère #, celle-ci n'est pas exécutée. Le caractère # peut être utilisé pour écrire des **commentaires** dans un programme