CONTROLE FLASH - PYTHON *(45 minutes)*

DATE : Novembre 2016 **GROUPE 4**

NOM : _____ **PRENOM :** _____

Calcul des racines d'un polynôme dans l'intervalle [a, b]

Dans tout le contrôle on utilisera la représentation dense vu en cours pour les polynômes.
Exemple :
Le polynôme $2x^3 - 4x^2 + 7$ sera représenté par la liste de réels [7,0,-4,2]

Toute fonction définie pour répondre à une question pourra être utilisée dans les questions suivantes. Même si vous passez une question, vous pouvez utiliser la fonction de la question dans les questions suivantes.

Q1. Ecrire une fonction valeur qui prend un polynôme *p* et un réel *x* et qui renvoie le réel *p(x)*

```
def valeur(p,x) :
```

Q2. Ecrire une fonction *derive* qui prend un polynôme *p* et renvoie le polynôme *p'*

```
def derive(p) :
```

Q3. Ecrire une fonction racine_degre1 qui prend un polynôme *p* **de degré exactement 1**, un réel *a* et un réel *b* et qui renvoie une liste vide si la racine de p n'est pas dans l'intervalle *[a, b]* et une liste contenant sa racine si celle ci se trouve l'intervalle *[a, b]*.

Exemple :

```
>>> racine_degre1( [2,1] , -4 , 4 )
[-2]
>>> racine_degre1( [2,1] , -1 , 4 )
[ ]
```

```
def racine_degre1(p,a,b) :
```

Q4. Ecrire une fonction dichotomie qui prend un polynôme *p*, un réel *a*, un réel *b* et un réel *epsilon*. **Sachant que :**

1 *p* admet une unique racine dans l'intervalle [a ,b]

2 *p* est strictement monotone dans l'intervalle [a ,b]

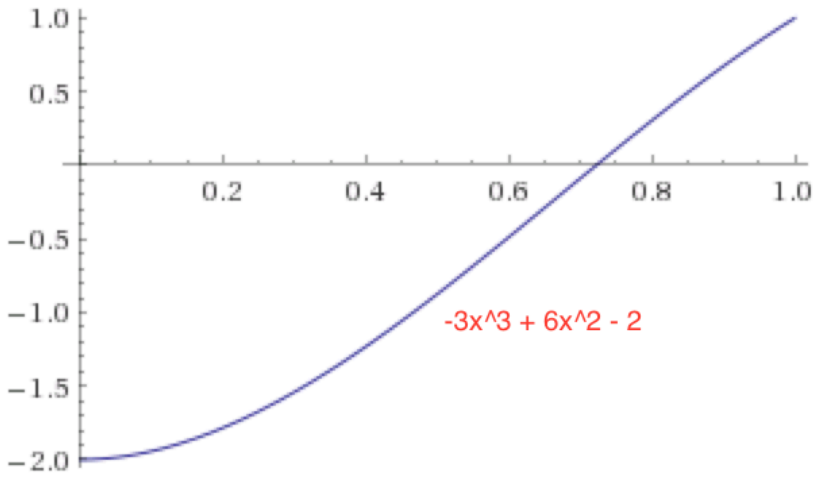
(3) *p(a)* et *p(b)* ne sont donc pas du même signe

En utilisant la recherche par dichotomie, la fonction renverra un réel *x* situé dans l'intervalle *[a ,b]* tel que $|p(x)| < epsilon$, c'est à dire une approximation de la racine de *p* située dans l'intervalle *[a ,b]*.

Exemple :

```
>>> dichotomie( [-2,0,6,-3] , 0 , 1 , 0.3)
0.65
```

Remarque : *0.65* est une réponse correcte car $|p(0,65)| \simeq 0.29 < epsilon = 0.3$, il y a évidemment d'autres réponses valides.



```
def dichotomie(p,a,b,epsilon) :
```

Q4. Ecrire une fonction *dichotomie*_opt qui utilise la fonction dichotomie et prend un polynôme *p*, un réel *a*, un réel *b* et un réel *epsilon*. **Sachant que :**

1' **Si** *p* admet une racine dans l'intervalle [a ,b] **alors** elle est unique

2 *p* est strictement monotone dans l'intervalle [a ,b]

(3') *p(a)* et *p(b)* sont donc du même signe **si et seulement si** *p* admet une racine dans l'intervalle [a, b].

À la manière de la Q3, La fonction renvoie une liste vide si p n'a pas de racine dans l'intervalle *[a, b]* et une liste contenant le résultat de la fonction dichotomie (appelée avec les mêmes paramètres) sinon.

Exemple :

```
>>> dichotomie_opt( [-2,0,6,-3] , 0 , 1 , 0.3)
[0.65]
>>> dichotomie_opt( [-2,0,6,-3] , 0 , 0.5 , 0.3)
[ ]
```

Remarque : Le second résultat est valide car dans l'intervalle [0, 0.5] le polynôme (qui est bien monotone sur cet intervalle) n'admet pas de racine.

```
def dichotomie_opt(p,a,b,epsilon) :
```

Q5. *On suppose qu'on dispose d'une fonction* racines *qui prend un polynôme p de degré supérieur ou égal à 1, un réel a, un réel b et un réel epsilon et qui renvoie une liste contenant toutes les racines de p situées dans l'intervalle [a, b], par ordre croissant.*
On rappelle que les extremums locaux d'un polynôme correspondent aux racines de sa dérivée.
Ecrire une fonction extremums qui prend un polynôme *p* de degré supérieur ou égal à 1, un réel *a*, un réel *b* et un réel *epsilon* et qui renvoie une liste contenant *a*, puis par ordre croissant tous les extremums locaux de *p* situés dans l'intervalle *[a, b]*, puis *b*.

```
def extremums(p,a,b,epsilon) :
```

Q6. En remarquant qu'entre deux extremums locaux consécutifs *x* et *y* d'un polynôme *p* on a :

1' **Si** *p* admet une racine dans l'intervalle *[x, y]* **alors** elle est unique

2 *p* est strictement monotone dans l'intervalle *[x, y]*

Ecrire la fonction racines qui prend un polynôme *p* de degré supérieur ou égal à 1, un réel *a*, un réel *b* et un réel *epsilon* et qui renvoie une liste des « racines approchées », telles que définies en Q4, avec précision *epsilon*.
Remarque 1: La fonction extremums utilise la fonction racines et la fonction racines doit utiliser (entre autres) la fonction extremums, il s'agit d'une récursivité mutuelle.

```
def racines(p,a,b,epsilon) :
```

Remarque 2: On implémente une fonction qui ne renvoie que des « racines approchées » alors qu'à la Q5 on a supposé qu'on disposait des racines exactes. Pour que cette approximation soit correcte, Il faut supposer que ni le polynôme *p* et ses dérivées n'admettent d'extremum local problématique, c'est à dire d'extremum local tel que $|p^{(i)}(m)| < epsilon$. Cette méthode ne marchera donc pas pour les polynômes admettant une racine multiple notamment.