Polynômes : comment les multiplier par un scalaire et les additionner

<sxh python; title : poly07-scal-add.py> #!/usr/bin/python # -*- coding: UTF-8 -*- “”“ écriture d'un programme pour évaluer des polynomes + fonction de multiplication d'un polynome pas un scalaire + fonction d'addition de deux polynomes ”“” from math import *

def polyeval(x,a):

```
  """application de l'agorithme de Horner
  cf. http://fr.wikipedia.org/wiki/M%C3%A9thode_de_Ruffini-Horner
  """
  n=len(a)-1 # n = ordre du polynome
  p=a[n]
  for i in range(n-1,-1,-1):
      p=p*x+a[i]
  return p
```

def polyscal(a,s):

```
  """polynome multiplié par un scalaire s """
  b=[]
  for coef in a:
      b.append(coef*s)
  return b    # on retourne les coefficients multipliés par s
```

def polyadd(a,b):

```
  """ Addition de deux polynomes de coefficients a et b
  """
  r=a[:] # on travaille sur une copie de a pour ne pas le modifier
  t=b[:] # idem pour b
  g=[]   # polynome somme
  n1=len(r) # ordre du premier polynome
  n2=len(t) # ordre du second polynome
  if n1>n2: # premier polynome de plus haut degré que le second
      for i in range (n1-n2):
          t.append(0)
  elif n1<n2: # second polynome de plus haut degré que le premier
       for i in range (n2-n1):
           r.append(0)
  # r et t ont à présent la même longueur
  for i in range (len(r)):
      g.append(r[i]+t[i])
  return g  # on retourne les coefficients additionnés dans la liste g
```

différents tests : x=2. a=[1,1,1,1,1,1,1,1,1,1,1] print polyeval(x,a)

varx=0.5 varcoef=[1.,2.,3.,4.,5.,6.,7.,8.,9.,10.] print polyeval(varx,varcoef)

for j in range(0,11,1):

```
  rep=sin(polyeval(float(j)*0.1,varcoef))
  print rep
```

u=[1,2,4,8,16] v=[1,3,9,27,81,243,729] print u,v,polyadd(u,v) </sxh>

Il est temps de créer des graphes de fonctions polynomiales. Comment faire ? Quelles librairies utiliser ?

Un exemple de graphe a été montré dans l'introduction du cours !

Voici un programme simple pour une fonction particulière (une sinusoïde amortie) : <sxh python; title : simple-plot-sinus-amorti.py> #!/usr/bin/python # -*- coding: iso-8859-15 -*- # sinusoïde amortie

from pylab import *

def sin_amort(t):

```
  s1 = sin(pi*t)
  e1 = exp(-t/8.)
  return s1*e1
```

xvals = arange(0., 40., 0.1) plot(xvals, sin_amort(xvals)) show() # le même genre de graphique, avec des symboles rouges et lignes noires plot(xvals, sin_amort(xvals), 'ro', xvals, sin_amort(xvals), 'k') show() </sxh>

Il faut donc faire un programme similaire en introduisant la fonction nécessaire et en adaptant les appels.

Réponse à la page suivante !