```
def Maximum(L) :
....if len(L) > 0 :
........maxim = L[0]
........for elt in L :
............if elt > maxim :
................maxim = elt
........return(maxim)
....return('Liste vide')
```

```
def Maximum_index(L) :
....if len(L) > 0 :
........maxim = L[0]
........indice = 0
........for k in range(len(L)) :
............if L[k] > maxim :
................maxim = L[k]
................indice = k
........return(maxim, index)
....return('Liste vide')
```

```
def moyenne(L) :
....if len(L) > 0 :
........cumul = 0
........for elt in L :
............cumul += elt
........return(cumul/len(L))
....return('Liste vide')
```

après avoir importé `randrange`

```
def melange(L) :
....mix = []
....LL = L[ :  ]  #copie de sécurité
....while len(LL) > 0 :
........extrait = LL.pop(randrange(len(LL)))
........mix.append(extrait)
....return(mix)
```

```
def convertisseur(n, base) :
....L = []
....while n>0 :
........L = [n % base]+L
........n = n/base #division euclidienne
....return(L)
```

```
def convertisseur(L, base) :
....cumul = 0
....puiss = 1
....for k in range(len(L)) :
........chiffre = L[-k-1]
........cumul = cumul + chiffre*puiss
........puiss = puiss*base
....return(cumul)
```

après avoir importé `matplotlib.pyplot` sous le nom de `pl`

```
def graphique(f, a, b, NbPoints) :
....pas = (b-a)/float(NbPoints) #le float c'est pour éviter la division euclidienne
....abscisses = [a+k*pas for k in range(NbPoints+1)]
....ordonnees = [f(x) for x in abscisses]
....graphe = pl.plot(abscisses, ordonnees)
....pl.show()
```

ajouter avant `pl.grid()` et `pl.title('graphe de ...')` si on y tient

```
def factorielle(n) :
....if n = int(n) and n >= 0 :
........fact = 1
........for k in range(n) :
............fact = fact*(k+1)
........return(fact)
....return('euh pardon ?  vous voulez
dire quoi ?')
```

```
def factorielle(n) :
....if n <= 1 :
........return(1)
....return(n*factorielle(n-1))
```

```
def binomial(n, k) :
....if 2*k > n :
.........return(binomial(n, n-k))
....if k < 0 :
........return(0)
....prod = 1
....for i in range(k) :
........prod = prod*(n-i)/(i+1)
....return(prod)
```

```
def Unite(n) :
....return([[int(ligne==colonne) for colonne in range(n)] for ligne in range(n)])

def produit(A, B) :
....if len(A[0]) == len(B) :
........P = []
........for IndiceLigne in range(len(A)) :
............Ligne = []
............for IndiceCol in range(len(B[0])) :
................s = 0
................for k in range(len(B)) :
....................s = s + A[IndiceLigne][k]*B[k][IndiceCol]
................Ligne.append(s)
............P.append(Ligne)
........return(P)
```

```
def puissance(A, p) :
....puiss =
Identite(len(A))
....for k in range(p) :
........puiss =
produit(puiss, A)
....return(puiss)
```

```
def dichotomie(f, a, b, epsi) :
....fa, fb = f(a), f(b)
....if fa*fb < 0 :
........while abs(b-a)>epsi :
............c = (a+b)/2
............if fa*f(c) < 0 :
................b = c
............else :
................a = c
```

```
def EquaDiff(F, xDebut, xFin, yDebut, NbPas) :
....pas = (b-a)/float(NbPas)
....abscisses = [xDebut]
....ordonnees = [yDebut]
....x, y = xDebut, yDebut
....for k in range(NbPas) :
........dy = F(x, y)
........y = y+dy
........x=x+pas
........abscisses.append(x)
........ordonnees.append(y)
....return(abscisses, ordonnees)
```

```
def Euclide(a, b) :
....aa, bb = abs(a), abs(b) #par sécurité
....while bb > 0 :
.........aa, bb = bb, (aa%bb)
....return(aa)
```

```
def IntegraleRiemannGauche(f, a, b,
NbSegments) :
....cumul = 0
....pas = (b-a)/float(NbSegments)
....x = a
....for k in range(NbSegments) :
........cumul = cumul+f(x)
........x = x+a
....return(pas*cumul)
```

```
def IntegraleTrapezes(f, a, b,
NbSegments) :
....cumul = -f(a)/2
....pas = (b-a)/float(NbSegments)
....x = a
....for k in range(NbSegments) :
........cumul = cumul+f(x)
........x = x+a
....cumul = cumul+f(b)/2
....return(pas*cumul)
```

```
def IntegraleMilieux(f, a, b, NbSegments) :
....cumul = 0
....pas = (b-a)/float(NbSegments)
....x = a+pas/2
....for k in range(NbSegments) :
........cumul = cumul+f(x)
........x = x+a
....return(pas*cumul)

def IntegraleParaboles(f, a, b, N) :
....return((2*IntegraleMilieux(f,a,b,N)+IntegraleTrapezes(f,a,b,N))/3)
```

```
def Combinaison(A, i, j, coeff) :
....for k in range(len(A[0]) :
........A[i][k] = A[i][k]+coeff*A[j][k]

def PermuteLignes(A, i, j) :
....for k in range(len(A[0]) :
........A[i][k], A[j][k] = A[j][k], A[i][k]
```

```
def CherchePivot(A, Col) :
....MeilleureLigne = Col
....for Lig in range(Col+1, len(A)) :
........if abs(A[Lig][Col]) >
abs(A[MeilleureLigne][Col]) :
............MeilleureLigne = Lig
....return(MeilleureLigne)
```

```
def Pivot(Matrice, SecondMembre) :
....n = len(Matrice)
....A = [Ligne[:]  for Ligne in Matrice] #copie compliquée de sécurité
....Y = [Ligne[:]  for Ligne in SecondMembre] #copie de sécurité
....for i in range(n) :
........BestLigne = CherchePivot(A, i) :
........if BestLigne > i :
............PermuteLignes(A, i, BestLigne)
............PermuteLignes(Y, i, BestLigne)
........for k in range(i+1, n) :
............coeff = -A[k][i]/float(A[i][i])
............Combinaison(A, k, i, coeff)
............Combinaison(Y, k, i, coeff)
....#maintenant, on remonte
....X = [float(0) for i in range(n)]
....for k in range(n) :
........i = len(A)-k-1
........X[i] = (Y[i][0]-sum(A[i][j]*X[j] for j in range(i+1, n))) / A[i][i]
....return(X)
```

```
def TestPremier(n) :
....MaxPetitDiviseur =
int(sqrt(n))
....for k in range(1,
MaxPetitDiviseur+1) :
........if n%k == 0 :
............return(False)
....return(True)
```

```
def GenerePremiers(Max) :
....Premiers = [2, 3]
....Nombre = 3
....while Nombre < Max :
........NbDivPremiers = 0
........for p in Premiers :
............if Nombre%p == 0 :
................NbDivPremiers += 1
........if NbDivPremiers == 0 :
............Premiers.append(Nombre)
........Nombre = Nombre+2
....return(Premiers)
```