Divers Python

- input(text)
- print(x, y, sep=’ ’, end=’\n’)
- range(start, stop, step)
- len(s)
- IndexError, KeyError, NameError, TypeError, ValueError, ZeroDivisionError
- <, >, <=, >=, ==, is, is not, and, or, in, not in

Listes

- L = [1,2,3]
- x in L
- L.append(x)
- L.insert(i, x)
- L.pop()
- L.pop(i)
- L.remove(x)
- L1.extend(L2)
- L.count(x)
- L.index(x)
- L.copy()

Dictionnaires

- d = {key1: val1, key2: val2,}
- d.get(key)
- del d[key]
- d.keys()
- d.values()
- d.items()
- x in d/d.items()/d.values()/d.keys

NumPy

- import numpy as np
- np.sqrt(x), np.cos(x), np.sin(x)
- a = np.array(x)
- np.arange(start, stop, step)
- np.linspace(start, stop, N)
- np.zeros(shape)
- np.ones(shape)
- np.full(shape, x)
- np.empty(shape)
- np.eye(N)
- np.ndim(a)
- np.shape(a)
- np.reshape(a,shape)
- np.dot(a,b)
- np.round(a)
- np.concatenate(a, b)
- np.loadtxt(filename, usecols = None, skiprows = 0, unpack = False)
- np.savetxt(filename, data, delimiter = ’ ’, newline = ’\n’, header = ’ ’)

Matplotlib

- import matplotlib.pyplot as plt
- plt.plot(x, y, "–b", label=my_label)
- plt.xlabel(my_x_label)
- plt.ylabel(my_y_label)
- plt.legend()
- plt.title(my_title)
- plt.xlim(xmin, xmax)
- plt.ylim(ymin, ymax)

Algorithmique

- $f \sim \mathcal{O}(g)$ si
  $\exists C, N > 0$ tq. $\forall n > N, f(n) \leq C \cdot g(n)$
- $f \sim \Omega(g)$ si
  $\exists C, N > 0$ tq. $\forall n > N, f(n) \geq C \cdot g(n)$
- $f \sim \Theta(g)$ si
  $f \sim \mathcal{O}(g)$ et $f \sim \Omega(g)$

Algorithme : Recherche binaire

Input : Une liste triée L, une valeur x
Output : L'indice de x dans L ou `None` si non trouvé

$a \leftarrow 0$
$b \leftarrow$ longueur$(L) - 1$
while $b \geq a$ **do**
  $m \leftarrow \left\lfloor \frac{a+b}{2} \right\rfloor$
  if $L[m] = x$ **then**
    return m
  else
    if $L[m] < x$ **then**
      $a \leftarrow m + 1$
    else
      $b \leftarrow m - 1$
return `None`

Algorithme : Tri par sélection

Input : Une liste L de taille n
Output : La liste L triée par ordre croissant

$n \leftarrow$ longueur(L)
for $i \leftarrow 0$ **to** $n - 1$ **do**
  $\mathrm{min\_ind} \leftarrow \mathrm{i}$
  for $j \leftarrow i + 1$ **to** n **do**
    if $L[j] < L[\mathrm{min\_ind}]$ **then**
      $\mathrm{min\_ind} \leftarrow \mathrm{j}$
  échanger $L[i]$ et $L[\mathrm{min\_ind}]$

Algorithme : Tri à bulles

Input : Une liste L de taille n
Output : La liste L triée par ordre croissant

$n \leftarrow$ longueur(L)
for $i \leftarrow 0$ **to** $n - 1$ **do**
  for $j \leftarrow 0$ **to** $n - 1 - i$ **do**
    if $L[j] > L[j + 1]$ **then**
      échanger $L[j]$ et $L[j + 1]$

Algorithme : Tri par insertion

Input : Une liste L de taille n
Output : La liste L triée par ordre croissant

$n \leftarrow$ longueur(L)
for $i \leftarrow 1$ **to** $n - 1$ **do**
  $j \leftarrow i$
  while $j > 0$ ***et*** $L[j] < L[j - 1]$ **do**
    échanger $L[j]$ et $L[j - 1]$
    $j \leftarrow j - 1$

Équations non linéaires

Algorithme : Méthode de la bissection

Input : Une fonction f continue sur $[a, b]$ avec $f(a) \cdot f(b) < 0$, et une précision $\varepsilon > 0$
Output : Une approximation d'une racine de f sur $[a, b]$

while $b - a > \varepsilon$ **do**
  $m \leftarrow \frac{a+b}{2}$
  if $f(m) = 0$ **then**
    return m
  else
    if $f(a) \cdot f(m) < 0$ **then**
      $b \leftarrow m$
    else
      $a \leftarrow m$
return $\frac{a+b}{2}$

- $k_{\min} > \log_2\left(\frac{|b_0 - a_0|}{\varepsilon}\right) - 1$

Algorithme : Méthode de la sécante

Input : Une fonction f, deux approximations x_0 et x_1, tolérance $\varepsilon > 0$
Output : Une approximation d'une racine de f

repeat
  $x \leftarrow x_1 - \frac{(x_1 - x_0)}{f(x_1) - f(x_0)} f(x_1)$
  $x_0 \leftarrow x_1$
  $x_1 \leftarrow x$
until $|f(x_1)| < \varepsilon$
return x_1

Algorithme : Méthode de Newton

Input : Une fonction dérivable f, sa dérivée f', une valeur initiale x_0, une tolérance $\varepsilon > 0$

Output : Une approximation d'une racine de f

repeat

  if $f'(x_0) = 0$ **then**

    return Erreur : dérivée nulle

  $x_0 \leftarrow x_0 - \dfrac{f(x_0)}{f'(x_0)}$

until $|f(x_0)| < \varepsilon$

return x_0

- Ces deux méthodes se basent sur $x_{k+1} = x_k - q_k^{-1} f(x_k)$

Algorithme : Méthode de Picard

Input : Une fonction ϕ, une valeur initiale x_0, une tolérance $\varepsilon > 0$

Output : Une approximation d'un point fixe de ϕ

repeat

  $x_1 \leftarrow \phi(x_0)$

  $x_0 \leftarrow x_1$

until $|x_1 - x_0| < \varepsilon$

return x_1

Intégration numérique

- $\varphi_k(t) = \prod\limits_{\substack{j=1 \\ j \neq k}}^{M} \dfrac{t - t_j}{t_k - t_j}$
- Changement de variable : $x = \dfrac{x_{i+1} - x_i}{2} \cdot t + \dfrac{x_i + x_{i+1}}{2}$
- $\omega_j = \displaystyle\int_{-1}^{+1} \varphi_j(t)\,\mathrm{d}t$
- $J(g) = \displaystyle\sum_{j=1}^{M} \omega_j\, g(t_j)$
- Méthode du point de gauche : $J_i^{\mathrm{g}}(f) = (x_{i+1} - x_i) \cdot f(x_i)$ Exacte pour les polynômes de degré 0
- Méthode du point de droite : $J_i^{\mathrm{d}}(f) = (x_{i+1} - x_i) \cdot f(x_{i+1})$ Exacte pour les polynômes de degré 0
- Méthode de Riemann : $J_i^{\mathrm{R}}(f) = (x_{i+1} - x_i) \cdot f(\xi)$, où ξ est un nombre aléatoire entre x_i et x_{i+1} Exacte pour les polynômes de degré 0
- Méthode du point milieu : $J_i^{\mathrm{PM}}(f) = (x_{i+1} - x_i) \cdot f\left(\dfrac{x_i + x_{i+1}}{2}\right)$ Exacte pour les polynômes de degré ≤ 1
- Méthode des trapèzes : $J_i^{\mathrm{TR}}(f) = (x_{i+1} - x_i) \cdot \dfrac{f(x_i) + f(x_{i+1})}{2}$ Exacte pour les polynômes de degré ≤ 1
- Méthode de Simpson : $J_i^{\mathrm{S}}(f) = \dfrac{x_{i+1} - x_i}{6} \cdot \left(f(x_i) + 4f\left(\dfrac{x_i + x_{i+1}}{2}\right) + f(x_{i+1})\right)$ Exacte pour les polynômes de degré ≤ 3
- $e_{\mathrm{abs}}(h) \leq C \cdot h^{r+1}$