♦ Révisions – Quelques exemples de fonctions

I. — Sur les listes simples

Exemple 1. (Test d'appartenance) Écrire une fonction `appartenance_liste` qui prend en arguments une liste `L` et un objet `x` et qui renvoie `True` si `x` est un élément de `L` et `False` sinon.

Solution.

```
def appartenance_liste(L,x):
  for e in L:
    if e == x:
      return True
  return False
```

Exemple 2. (Première occurrence) Écrire une fonction `premiere_occ_liste` qui prend en arguments une liste `L` et un objet `x` et qui renvoie l'indice de la première occurrence de `x` dans `L` si `x` est un élément de `L` et `None` sinon.

Solution.

```
def premiere_occ_liste(L,x):
  for i in range(len(L)):
    if L[i] == x:
      return i
```

Exemple 3. (Nombre d'occurrences) Écrire une fonction `nb_occ_liste` qui prend en arguments une liste `L` et un objet `x` et qui renvoie le nombre d'occurrences de `x` dans `L` (ce nombre étant égal à 0 si `x` n'apparaît pas dans `L`).

Solution.

```
def nb_occ_liste(L,x):
  nb = 0
  for e in L:
    if e == x:
      nb += 1
  return nb
```

Exemple 4. (Fréquence d'un élément) Écrire une fonction `frequence_liste` qui prend en arguments une liste `L` et un objet `x` et qui renvoie la fréquence d'apparition de `x` parmi les éléments de `L`.

Solution.

```
def frequence_liste(L,x):
  nb = 0
  for e in L:
    if e == x:
      nb += 1
  return nb/len(L)
```

Exemple 5. (Sous-liste) Écrire une fonction `sous_liste` qui prend en arguments deux listes `L` et `M` et qui renvoie `True` si `M` est une sous-liste de `L` (i.e. si les éléments de `M` apparaissent consécutivement et dans le même ordre dans `L`) et `False` sinon.

Solution.

```
def sous_liste(L,M):
  l = len(L)
  m = len(M)
  for k in range(l-m+1):
    j = 0
    while j < m and L[k+j] == M[j]:
      j += 1
    if j == m:
      return True
  return False
```

II. — Sur les listes de nombres

Exemple 6. (Somme des éléments) Écrire une fonction `somme_liste` qui prend en argument une liste de nombres `L` et qui renvoie la somme des éléments de `L`.

Solution.

```
def somme_liste(L):
  S = 0
  for e in L:
    S += e
  return S
```

Exemple 7. (Moyenne des éléments) Écrire une fonction `moyenne_liste` qui prend en argument une liste de nombres `L` et qui renvoie la moyenne des éléments de `L`.

Solution.

```
def moyenne_liste(L):
  S = 0
  for e in L:
    S += e
  return S/len(L)
```

Exemple 8. (Maximum d'une liste) Écrire une fonction `maximum_liste` qui prend en argument une liste de nombres `L` et qui renvoie le plus grand élément de `L`.

Solution.

```
def maximum_liste(L):
  M = L[0]
  for e in L:
    if e > M:
      M = e
  return M
```

Exemple 9. (Première occurrence du maximum d'une liste) Écrire une fonction `premiere_occ_max` prenant en argument une liste de nombres `L` et qui renvoie le rang de la première occurrence du plus grand élément de `L`.

Solution.

```
def premiere_occ_max(L):
  M = L[0]
  k = 0
  for i in range(len(L)):
    if L[i] > M:
      M = L[i]
      k = i
  return k
```

III. — Sur les listes de listes

Exemple 10. (Test d'appartenance) Écrire une fonction `appartenance_liste2` qui prend en arguments une liste de listes `L` et un objet `x` et qui renvoie `True` si `x` est un élément d'une des listes de `L` et `False` sinon.

Solution.

```
def appartenance_liste2(L,x):
  for liste in L:
    for e in liste:
      if e == x:
        return True
  return False
```

Exemple 11. (Génération d'une liste de listes) Écrire une fonction `Repetition` qui prend en arguments un objet `x` et deux entiers naturels non nuls `n` et `m` et qui renvoie une liste de `n` listes composées chacune de `m` fois l'objet `x`.

Solution.

```
def Repetition(x,n,m):
  return [ [ x for i in range(m)] for j in range(n) ]
```

IV. — Sur les chaînes de caractères

Exemple 12. (Test d'appartenance) Écrire une fonction `appartenance_chaine` qui prend en arguments une chaîne de caractères `ch` et un caractère `x` et qui renvoie `True` si `x` apparaît dans `ch` et `False` sinon.

Solution.

```
def appartenance_chaine(ch,x):
  for e in ch:
    if e == x:
      return True
  return False
```

Exemple 13. (Première occurrence) Écrire une fonction `premiere_occ_chaine` qui prend en arguments une chaîne de caractères `ch` et un caractère `x` et qui renvoie l'indice de la première occurrence de `x` dans `ch` si `x` est un caractère de `ch` et `None` sinon.

Solution.

```
def premiere_occ_chaine(ch,x):
  for k in range(len(ch)):
    if ch[k] == x:
      return k
```

Exemple 14. (Nombre d'occurrences) Écrire une fonction `nb_occ_chaine` qui prend en arguments une chaîne de caractères `ch` et un caractère `x` et qui renvoie le nombre d'occurences de `x` dans `ch` (ce nombre étant égal à 0 si `x` n'est pas un caractère de `ch`).

Solution.

```
def nb_occ_chaine(ch,x):
  k = 0
  for e in ch:
    if e == x:
      k += 1
  return k
```

Exemple 15. (Liste des occurrences) Écrire une fonction `occ_chaine` qui prend en arguments une chaîne de caractères `ch` et un caractère `x` et qui renvoie la liste, éventuellement vide, des indices des différentes occurrences de `x` dans `ch`.

Par exemple, `occ_chaine('abcaaca')` doit renvoyer `[0,3,4,6]` car, dans la chaîne `'abcaaca'`, le caractère `'a'` apparaît aux indices 0, 3, 4 et 6.

Solution.

```
def occ_chaine(ch,x):
  L = []
  for k in range(len(ch)):
    if ch[k] == x:
      L.append(k)
  return L
```

Exemple 16. (Liste des caractères) Écrire une fonction `liste_caracteres` qui prend en argument une chaîne de caractères `ch` et qui renvoie la liste des différents caractères de `ch` (chaque caractère ne doit apparaître qu'une seul fois dans la liste). On pourra utiliser la fonction `appartenance_liste`.

Solution.

```
def liste_caracteres(ch):
  L = []
  for e in ch:
    if not appartenance_liste(L,e):
      L.append(e)
  return L
```

Exemple 17. (Fréquence d'un caractère) Écrire une fonction `frequence_chaine` qui prend en arguments une chaîne de caractères `ch` et un caractère `x` et qui renvoie la fréquence d'apparition de `x` dans la chaîne `ch`

Solution.

```
def frequence_chaine(ch,x):
  nb = 0
  for e in ch:
    if e == x:
      nb += 1
  return nb/len(ch)
```

Exemple 18. (Sous-chaîne) Écrire une fonction `sous_chaine` qui prend en arguments deux chaînes de caractères `ch1` et `ch2` et qui renvoie `True` si `ch2` est une sous-chaîne de `ch1` i.e. si les caractères de `ch2` apparaissent consécutivement et dans le même ordre dans `ch1` et `False` sinon.

Solution.

```
def sous_chaine(ch1,ch2):
  l = len(ch1)
  m = len(ch2)
  for k in range(l-m+1):
    j = 0
    while j < m and ch1[k+j] == ch2[j]:
      j += 1
    if j == m:
      return True
  return False
```

Exemple 19. (Miroir d'une chaîne) Écrire une fonction `miroir` qui prend en argument une chaîne de caractères `ch` et qui renvoie la chaîne obtenu à partir de `ch` en écrivant les caractères du dernier au premier.

Par exemple, `miroir('abcd')` doit renvoyer `'dcba'`.

Solution.

```
def miroir(ch):
  mir = ''
  for e in ch:
    mir = e + mir
  return mir
```

V. — Sur les probabilités

Dans cette section, on suppose que le module `random` est importé.

Exemple 20. (Liste d'entiers aléatoires) Écrire une fonction `liste_aleatoire` qui prend en arguments deux entiers naturels non nuls `n` et `N` et qui renvoie une liste de `N` entiers choisis aléatoirement et de façon équiprobable entre `1` et `n`.

Solution.

```
def liste_aleatoire(n,N):
  return [ randint(1,n) for i in range(N) ]
```

Exemple 21. (Simulation d'une loi de Bernoulli) Écrire une fonction `bernoulli` qui prend en argument un réel `p` compris entre 0 et 1 et qui simule une variable aléatoire suivant une loi $\mathscr{B}(p)$ i.e. qui renvoie 1 avec une probabilité `p` et 0 avec une probabilité `1-p`.

Solution.

```
def bernoulli(p):
  if random() < p:
    return 1
  else:
    return 0
```

Exemple 22. (Simulation d'une loi binomiale) En utilisant la fonction `bernoulli`, écrire une fonction `binomiale` qui prend en arguments un réel `p` compris entre 0 et 1 et un entier naturel non nul `n` et qui simule une variable aléatoire suivant une loi $\mathscr{B}(\mathtt{n},\mathtt{p})$.

Solution.

```
def binomiale(n,p):
  c = 0
  for i in range(n):
    c += bernoulli(p)
  return c
```

Exemple 23. (Simulation d'une loi géométrique) En utilisant la fonction `bernoulli`, écrire une fonction `geometrique` qui prend en argument un réel `p` compris entre 0 et 1 et qui simule une variable aléatoire suivant une loi $\mathscr{G}(\mathtt{p})$.

Solution.

```
def geometrique(p):
  n = 1
  while bernoulli(p) == 0:
    n += 1
  return n
```