CHAPITRE

1

ÉLÉMENTS DE CORRECTION

```
''' Dans les premiers exemples, on précise la signature des fonctions mais ce n est pas
    indispensable. '''

## premiers exemples

def f(x: float) -> float:
    """ entrée : flottant x
        sortie : valeur de x**2+x-1
    """
    return x**2 + x - 1


from math import exp

def g(x: float) -> float:
    """ entrée : flottant x
        sortie : valeur de exp(x+1)
    """
    return exp(x+1)


def dernier(ch: str) -> str:
    """ entrée : chaîne de caractères non vide
        sortie : dernier caractère
    """
    return ch[-1] # ou ch[len(ch)-1]


def f(a: float, b: float, x:float) -> float:
    """ entrée : a, b, x flottants
        précondition : a < b
        sortie : valeur de la fonction de l'énoncé en x
    """
    if x >= b:
        return 1
    elif x <= a:
        return 0
    else:
        return (x-a)/(b-a)
```

```
def impairs(n: int) -> int:
    """ entrée : entier naturel n
        sortie : somme des n premiers nombres impairs
    """
    s = 0
    for k in range(n):
        s = s + 2*k+1
    return s


''' Dans l exemple suivant <<plus grande puissance de 2>> est interprété
    comme étant l exposant de ladite puissance (i.e. 4 pour 2**4=16). '''

def puissance(n: int) -> int:
    """ entrée : entier naturel n
        sortie : plus grande puissance de 2 inférieure ou égale à n
    """
    p = 0
    while 2**p <= n:
        p = p + 1
    return p-1

def puissance_bis(n: int) -> int:
    """ entrée : entier naturel n
        sortie : plus grande puissance de 2 inférieure ou égale à n
    """
    p = 0
    puiss = 1
    while puiss <= n:
        p = p + 1
        puiss = puiss * 2
    return p-1



## Exercice 1

def long(n: int) -> int:
    """ entrée : entier naturel n
        sortie : nombre de chiffres dans l'écriture en base 2 de n
    """
    return len(bin(n)[2:])

def f(n):
    if n == 1:
        return 0
    if n > 1:
        if n%2 == 0:
            return 2*f(n//2) + 1
        if n%2 == 1:
            return 2*f(n//2)

def iter(n):
    k = 0
    while n != 0:
        n = f(n)
        k += 1
    return k
```

```python
def maxi():
    m = 0
    n_max = 1
    for n in range(1, 101):
        new = iter(n)
        if new > m:
            m = new
            n_max = n
    return m, n_max



## sur les listes

def appartient(a, L):
    ind = 0
    while ind < len(L):
        if L[ind] == a:
            return True
        ind += 1
    return False


def appartientbis(L, a):
    ind = 0
    while ind < len(L) and L[ind] <= a:
        if L[ind] == a:
            return True
        ind += 1
    return False


def tous(L):
    for x in L:
        if x < 0:
            return False
    return True


def aumoins(L):
    for x in L:
        if x >= 0:
            return True
    return False


def seuil(L, x):
    lis = []
    for y in L:
        if y >= x:
            lis.append(y)
    return lis


def extremes(L):
    m, M, ind = L[0], L[0], 1
    while ind < len(L):
        if L[ind] > M:
            M = L[ind]
        if L[ind] < m:
            m = L[ind]
        ind += 1
    return (m, M)
```

```python
def indmax(L):
    i, M = 0, L[0]
    ind = 1
    while ind < len(L):
        if L[ind] >= M:
            M = L[ind]
            i = ind
        ind += 1
    return i


def nombre(lis, x, n):
    c = 0
    for element in lis:
        if element == x:
            c += 1
    return c == n


def deuxieme(lis):
    a = lis[0]
    b = lis[1]
    if a > b:
        a, b = b, a
    for k in range(2,len(lis)):
        c = lis[k]
        if c > b:
            a, b = b, c
        elif c > a:
            a = c
    return(a)


def distmin(lis):
    n = len(lis)
    d = abs(lis[1]-lis[0])
    for i in range(n):
        for j in range(n):
            e = abs(lis[i] - lis[j])
            if 0 < e < d:
                d = e
    return d

# un jeu de tests pour cette dernière fonction:
assert distmin([2, 5, 6, 0, 11]) == 1
assert distmin([2, 5, 6, 0, 6, 11]) == 1
assert distmin([2, 6, 5, 0, 11]) == 1
```