université PARIS-SACLAY | FACULTÉ DES SCIENCES D'ORSAY

vendredi 17 octobre 2025

Introduction à l'informatique
L1

TP n° 6

Consignes les exercices ou questions marqués d'un ⋆ devront être rédigés sur papier (afin de se préparer aux épreuves écrites du partiel et de l'examen). En particulier, il est recommandé d'être dans les même conditions qu'en examen : pas de document ni de calculatrice. Les questions marquées d'un ⋄ sont des questions supplémentaires permettant d'aller plus loin (et ne seront pas forcément corrigées en TP). Tous les TPs se font sous Linux.

But du TP

Le but de cette feuille d'exercice est :
— de se familiariser avec la notion de fonction en Python
— de consolider les acquis sur les autres concepts, il y a donc une utilisation variée des tests, boucles, tableaux.

Exercices

Exercice 0

Ouvrir un terminal :
— créer un répertoire TP6 à l'intérieur du répertoir `IntroInfo`
— se placer à l'intérieur du répertoire `TP6`
On suppose pour les autres exercices que le répertoire `TP6` est le répertoire courant.

Exercice 1

⋆

1. Pour chacun des programmes Python ci-dessous, dire ce qu'ils affichent.

(a)
```
1 def f (x):
2     print(x)
3
4 f(42)
```

(b)
```
1 def f (x):
2     x = 43
3     print(x)
4
5 f(42)
```

(c)
```
1 def f(x):
2     return x + 10
3
4 def g (y):
5     z = y + 1
6     print(f(z))
7
8 g(42)
```

(d)
```
1 y = 19
2 def f (x):
3     print(y)
4
5 f(42)
```

(e)
```
1 y = 19
2 def f (x):
3     y = 17
4     print(y)
5
6 f(42)
7 print(y)
```

(f)
```
1 y = 19
2 def f(x):
3     global y
4     y = 17
5     print(y)
6
7 f(42)
8 print(y)
```

Exercice 2

Pour chacune des fonctions ci-dessous, vous devez, en plus de code de la fonction, proposer des jeux de tests permettant de vérifier le bon comportement du code écrit. Une façon d'écrire un jeu de test est d'utiliser l'instruction **`assert`** e où e est une expression booléene. Cette instruction ne fait rien si e est vrai et provoque une erreur si e est faux. Pour les questions 1 et 2, on pourra par exemple tester les fonctions écrites avec :

```
1  def bissextile(a):
2      ... #votre code
3
4  assert bissextile(2024)
5  assert bissextile(2025) == False
6  assert bissextile(2000)
7  assert bissextile(1900) == False
8
9  def nbjoursannee(a):
10     ... #votre code
11
12 assert nbjoursannee(2024) == 366
13 assert nbjoursannee(1900) == 365
```

Le fichier ci-dessus ne doit pas lever d'erreur. On procèdera de façon similaire pour toutes les fonctions.

1. Écrire une fonction `bissextile(a)` qui renvoie `True` si l'année `a` est bissextile et `Flase` sinon. Une année est bissextile si elle est divisible par 4 et qu'elle n'est pas divisible par 100 ou alors si elle est divisible par 400.
2. Écrire une fonction `nbjoursannee(a)` qui renvoie le nombre de jours d'une année (on devra réutiliser la fonction précédente).
3. Écrire une fonction `nbjoursmois(a, m)` qui renvoie le nombre de jours dans le mois `m` (compris entre 1 et 12) de l'année `a`. Il est suggéré d'utiliser un tableau dans cette fonction, plutôt qu'une suite de **`if`**/**`elif`**.
4. Écrire une fonction `nbjoursdate(a,m,j)` qui calcule combien de jours complets se sont écoulés entre le `01/01/a` et le `j/m/a`. Attention, le jour considéré doit être exclu. Par exemple, l'appel `nbjoursdate(2020, 1, 1)` doit renvoyer `0` et l'appel `nbjoursdate(2020, 2, 1)` doit renvoyer `31`.
5. Écrire une fonction `nbjoursentre(a1, m1, j1, a2, m2, j2)` qui calcule le nombre de jours écoulés entre la date `j1/m1/a1` et `j2/m2/a2`. Vous pouvez supposer que les dates sont valides et que `j1/m1/a1` est avant `j2/m2/a2`. La deuxième date est exclue. Par exemple, `nbjoursentre(2020, 1, 1, 2020, 1, 1)` renvoie `0` et `nbjoursentre(2019, 1, 1, 2020, 1, 1)` renvoie `365`.

Exercice 3

On souhaite simuler des entiers en base 2 de taille 8 bits. De tels entiers peuvent être représentés en Python par un tableau de taille 8. La case i contient le bit en position 2^i.

Par exemple, l'entier $1001\,0111_2$ peut être représenté par le tableau `[1, 1, 1, 0, 1, 0, 0, 1 ]` (attention à l'ordre on écrit le nombre de gauche à droite, mais dans le tableau les bits sont stockés des indices 0 à 7, donc « dans le sens inverse »).

Pour toutes les fonctions ci-dessous, si l'argument de la fonction n'est pas valide, exécuter l'instruction : **`raise`** `ValueError("argument invalide")`. Cette instruction permet de signaler une erreur à l'utilisateur de la fonction.

1. Écrire une fonction `conv_base10(tab)` qui prend en argument un tableau et renvoie l'entier Python représentant ce nombre (en base 10 donc). Par exemple : `conv_base10([1, 1, 1, 0, 1, 0, 0, 1 ])` renvoie `151`.

   Votre fonction doit s'assurer que le tableau est de taille 8 et ne contient que des 0 et des 1.

   Vous pouvez dans un premier temps ignorer cette vérification, puis la rajouter à votre fonction lorsque votre code fonctionne pour un tableau bien formé. Dans toute la suite, on ne travaille qu'avec des tablaux de taille 8.

2. Écrire une fonction `conv_base2(n)` qui prend en argument un entier `n` compris entre 0 et 255 et qui renvoie le tableau correspondant. Par exemple `conv_base2(151)` renvoie `[1, 1, 1, 0, 1, 0, 0, 1 ]`. Votre fonction doit s'assurer que la valeur est dans l'intervalle 0-255 et lever un erreur sinon.
3. Écrire une fonction `print_base2(tab)` qui affiche l'entier en base 2 représentée par le tableau `tab)`. Ainsi, `print_base2([1, 1, 1, 0, 1, 0, 0, 1])` doit afficher `10010111`.

   Remarque on peut demander à la fonction `print` de ne pas afficher de retour à la ligne en passant l'argument optionnel `end=""`. Ainsi

```
print("A", end="")
print("B", end="")
print()              #pour le retour à la ligne final
```

   affiche

```
AB
```

   au lieu de

```
A
B
```

4. Écrire une fonction `add_base2(tab1, tab2)` qui utilise les fonctions ci-dessus pour renvoyer un tableau contenant la somme de `tab1` et `tab2`. Si cette somme est supérieure à 255, alors renvoyer le resultat de la somme modulo 256. Ainsi, si le tableau `tab1` représente l'entier 128 et l'entier `tab2` représente l'entier 205, la fonction doit renvoyer le tableau correspondant à l'entier `(128 + 205) % 256` soit `77`.
5. A-t-on besoin de tester dans `add_base2` que les tableaux sont au bon formats ?
6. On propose la fonction d'affichage suivante :

```
def print_base2_bad(tab):

    #on inverse l'ordre des éléments du tableau tab
    for i in range(len(tab)//2):
        tmp = tab[i]
        tab[i] = tab[len(tab)-1-i]
        tab[len(tab)-1-i] = tmp

    for i in range(len(tab)):
        print(tab[i], end="")
    print()
```

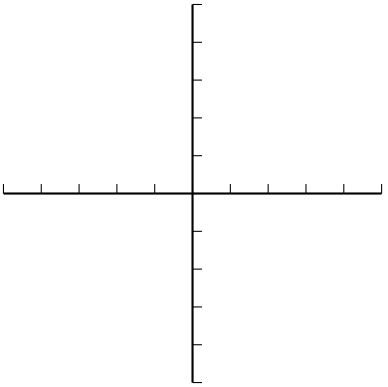
   Indiquer pourquoi le code ci-dessous calcule des résultats incorrects :

```
a = conv_base2(29)
b = conv_base2(101)
print_base2_bad(a)
print_base2_bad(b)
c = add_base2(a, b)
print("29 + 101 = ",conv_base10(c))
#Affiche 29 + 101 = 94
```

◇ Exercice 4

On se place dans un fichier où le module `turtle` a été importé. Le but de l'exercice est de tracer la courbe de fonctions mathématiques.

1. Définir une variable globale `F` représentant l'échelle, c'est à dire le nombre de pixels correspondant à une unité. On pourra définir `F` à 100 par exemple.
2. Définir une fonction `dessine_axes()` qui dessine les deux axes du repère orthonormé, avec une graduation de `F/10` pixel de long toutes les unités. On souhaite donc avoir un dessin comme celui-ci :



3. Définir une fonction `f(x)`, qui est la fonction mathématique dont on souhaite tracer le graphe. Cela peut être par exemple `x**2 -5x + 3` ou toute autre fonction de votre choix.
4. Définir une fonction `dessine_f(a, b, s)` qui dessine le graphe de `f` pour des valeurs de `x` allant entre `a` inclus et `b` exclu, par pas de `s`. Attention, on souhaite que `s` puisse être un nombre flottant, on ne peut donc pas utiliser la fonction **`range`**.
5. Appeler les deux fonctions pour tracer le repère et le graphe de `f` (ne pas oublier d'appeler `done()` à la suite afin de laisser la fenêtre ouverte.
6. Que peut-il se produire si on définit

```
1     def f (x):
2         return 1/x
```

et qu'on trace cette fonction ? Modifier `dessine_f` pour gérer ce cas et ne pas dessiner le graphe pour les valeurs de `x` ou `f` n'est pas définie.