Errata à destination des lecteurs qui possèdent le premier tirage de la première édition (*Python 3*, éditions Dunod)

— Page 2 : au lieu de « qui ne compte aucun », lire « qui ne comptent aucun »
— Page 5 : dans l'extrait de script qui trace les villes, au lieu de « `map` », lire « `carte` »
— Page 6 : au lieu de « un gestionnaire de paquets », lire « un dépôt en ligne de paquets »
— Page 10 : au lieu de « d'une séquence », lire « d'une séquence d'instructions »
— Page 12, Figure 1.1 : au lieu de « `sources → compilateur et lieur → objet → exécution → résultat` », lire « `sources → compilateur et lieur → objet (langage machine) → exécution → résultat` »
— Page 15 : au lieu de « Dans un fichier », lire « Dans un fichier source »
— Page 15 : au lieu de « tape la commande `python` dans », lire « tape la commande `python3` dans »
— Page 16 : au lieu de « sensibles à la casse », lire « sensibles à la casse (distinction minuscule/majuscule) »
— Page 16 : au lieu de « s'éloignent du style préconisé », lire « s'éloignent du style préconisé dans l'ouvrage »
— Page 17 : au lieu de « attribut de classe» », lire « attribut lié à la classe» »
— Page 21 : au lieu de « n'offre pas la notion de variable comme adresse de mémoire identifiée (comme avec le langage C) mais plutôt celle de références d'objets par des noms », lire « offre la notion de variables comme étant des références d'objets accessibles par des noms (là où d'autres langages comme le langage C utilisent la notion d'adresse mémoire fixe identifiée) »
— Page 23, en 2.7.4 : au lieu de « `un tuple` », lire « `demande des deux valeurs (sous la forme d'un tuple)` »
— Page 26 : au lieu de « sur 2 chiffres », lire « sur 2 chiffres hexadécimaux »
— Page 27 : au lieu de « `# ... finit par` », lire « `# ... finit par (casse prise en compte)` »
— Page 27 : au lieu de « les méthodes qui les modifient» », lire « les méthodes qui effectuent des modifications »
— Page 27 : au lieu de « `# suppression de caractères en début de chaîne` », lire « `# suppression de caractères e, espace ou L en début de chaîne` »
— Page 27 : au lieu de « un indice », lire « un index »
— Page 27 : au lieu de « dans la sous-chaîne », lire « dans la sous-chaîne entre les index »
— Page 29, en 2.8.10 : nouvel encart de la syntaxe : « Où `format` est une chaîne contenant des spécificateurs de format `%` et `valeurs` est un tuple contenant une ou plusieurs valeurs. Chaque spécificateur `%` est suivi d'un caractère dont la signification est donnée dans le tableau ci-dessous. À l'affichage, cette spécification est remplacée, dans l'ordre, par son correspondant dans valeurs. Valeurs peut aussi être un dictionnaire, on utilise alors le spécificateur %(clé) pour établir la correspondance. »
— Page 30 : Simplification de l'exemple :

```
# adaptation du message de saisie :
msg = "Êtes-vous {} {} ? (o/n) : "
qualite = "Madame"
nom = "Plume"
rep = input(msg.format(qualite, nom))
```

- Page 30 : au lieu de « l'argument de `format()` à utiliser », lire « l'argument de `format()` à utiliser, donc ici un dictionnaire, »
- Page 30 : au lieu de :

```
print("math.pi = {0.pi}, epsilon = {1.float_info.epsilon}".format(math, sys))
# math.pi = 3.14159265359, epsilon = 2.22044604925e-16
```

lire :

```
print("math.pi = {0.pi}, sys.float_info.epsilon = {1.float_info.epsilon},
       math.e = {0.e}".format(math, sys))
# math.pi = 3.141592653589793, sys.float_info.epsilon = 2.220446049250313e-16,
# math.e = 2.718281828459045
```

- Page 30 : au lieu de :

```
print("{:.2%}".format(n/(47*pi)))  # 67.73% - Ce format évite de multiplier le ratio par 100
```

lire :

```
print("{:.2%}".format(2 * pi))  # 628.32% - Ce format évite de multiplier l'argument par 100
```

- Page 33 : au lieu de :

```
>>> print(a, end='@') # affiche un autre caractère qu'un espace en fin de ligne
2@
```

lire :

```
>>> print(a, b, sep="+++", end='@') # utilise un séparateur et une fin de ligne spécifiques
2+++5@
```

- Page 37 : au lieu de « notion d'ordre entre ces données », lire « une notion de classement ordonné entre ces données. »
- Page 37 : au lieu de « une série de valeurs entières », lire « une série de valeurs entières servant d'index »
- Page 38 : au lieu de « (on doit préciser le type », lire « (on doit adapter le type »
- Page 39 : au lieu de « reprend à la ligne de l'en-tête de la boucle », lire « reprend à la ligne d'introduction de la boucle »
- Page 39 : au lieu de « Une exception sépare », lire « Le système d'exceptions sépare »
- Page 40 : au lieu de :

```
except <exception_1> as e1:
```

lire :

```
except <classe_exception_1> as e1:
```

- Page 43 : au lieu de « vues précédemment », lire « vues au chapitre 2.8.1, p. 25 »
- Page 44 : ajout d'une note de bas de page à « itérateur d'entiers `range()` » : « En réalité `range()` est un *itérateur*. La syntaxe `list(range())` signifie que l'on transtype un itérateur en liste. »

- Page 45 : au lieu de « ou bien », lire « autre syntaxe »
- Page 45 : au lieu de « on peut indiquer une tranche », lire « on doit indiquer une tranche »
- Page 45 : au lieu de « sont séparés par des virgules, et entourés », lire « sont séparés par des virgules, et optionnellement entourés »
- Page 47 : au lieu de « Or celle-ci peut », lire « Or en Python celle-ci peut »
- Page 49 : au lieu de « n'appartiennent pas aux séquences », lire « n'appartiennent pas aux séquences car ils n'en partagent pas les propriétés »
- Page 50 : au lieu de « Un `set` est la transposition », lire « Un `set` est une transposition »
- Page 53 : au lieu de « on évitera en général les caractères `/ * ? < > " | :` », lire « on évitera en général les caractères `\ / * ? < > " | :` »
- Page 54 : au lieu de « signifie toujours Unicode », lire « signifie toujours chaînes de caractères Unicode »
- Page 56 : dans l'exemple de lecture d'une partie d'un fichier, au lieu de :

```
s = f.read(3)        # lit au plus n octets --> chaîne
```

lire :

```
s = f.read(3)        # lit au plus n caractères --> chaîne
```

- Page 56 : dans l'exemple sur la construction des noms de chemin, au lieu de :

```
>>> import os.path
>>> os.path.join('home','bob/Esperanto', 'Baza_kurso')  # concaténer les noms d'un chemin
'home/bob/Esperanto/Baza_kurso'
>>> os.path.join('/home', 'bob', 'Esperanto', 'Baza_kurso')
'/home/bob/Esperanto/Baza_kurso'
>>> os.path.expanduser('~/Esperanto')  # remplace ~ par le 'home' de l'utilisateur
>>> os.path.join(os.path.expanduser('~'), 'Esperanto')
'/home/bob/Esperanto'
```

lire :

```
>>> import os.path
>>> os.path.join('Esperanto', 'Baza_kurso')  # concaténer les noms d'un chemin relatif
'Esperanto/Baza_kurso'
>>> os.path.join('/home', 'bob', 'Esperanto', 'Baza_kurso')  # idem (chemin absolu)
'/home/bob/Esperanto/Baza_kurso'
>>> os.path.join(os.path.expanduser('~'), 'Esperanto')  # remplace ~ par le 'home' de l'utilisateu
'/home/bob/Esperanto'
```

- Page 56 : dans l'exemple sur les opérations sur les noms de chemin, au lieu de :

```
>>> os.path.isfile('/home/bob/Esperanto/brassens')  # l'OS Linux est sensible à la casse !
False
```

lire :

```
>>> os.path.exists('/home/bob/Esperanto/brassens')  # l'OS Linux est sensible à la casse !
False
```

- Page 66 : dans la figure 6.2, au lieu de « une fonction ou un module », lire « une fonction »
- Page 67 : au lieu de « en utilisant la formule », lire « en utilisant la formule d'Euler »
- Page 70 : au lieu de « risques de masquages), et on perd l'origine », lire « risques d'homonymie et donc de masquages), on perd alors l'origine »

— Page 85 : au lieu de :

```
# 200 valeurs flottantes réparties entre -10 et 10
```

lire :

```
# 200 valeurs flottantes réparties entre -10 et 10 compris
```

— Page 86 : au lieu de :

```
def f(a, b, c, d):
    x = np.linspace(-10, 10, 20)
    y = a*(x**3) + b*(x**2) + c*x + d
    title = '$f(x) = (%s)x^3 + (%s)x^2 + (%s)x + (%s)$' % (a, b, c, d)
    plt_arrays(x, y, title=title)
```

lire :

```
def f(a, b, c, d):
    x = np.linspace(-10, 10, 20)
    y = a*(x**3) + b*(x**2) + c*x + d
    # Encadrer le titre entre $ est une syntaxe LaTeX qui permet
    # beaucoup d'enrichissements typographiques
    title = '$f(x) = (%s)x^3 + (%s)x^2 + (%s)x + (%s)$' % (a, b, c, d)
    plt_arrays(x, y, title=title)
```

— Page 92 : au lieu de « `lst = [3, 5, 1]` », lire « `lst = [3, 5, 1, 9]` »
— Page 93 : au lieu de « Les attributs de classe sont partagés entre tous les objets de cette classe », lire « Les attributs de classe existent de façon unique et sont partagés entre tous les objets de cette classe »
— Page 100 : au lieu de « par un nom d'instance », lire « par une référence d'instance »
— Page 121 : au lieu de « en assurant l'appel à une méthode », lire « en assurant l'appel à une méthode spéciale »
— Page 123 : au lieu de « tant que l'élément examiné est un répertoire », lire « si l'élément examiné est un répertoire »
— Page 126 : au lieu de :

```
# Initialisation d'une liste 2D
multi_liste = [[0]*2 for ligne in range(3)]
print(multi_liste)      # [[0, 0], [0, 0], [0, 0]]
```

lire :

```
# Initialisation d'une liste 2D
multi_liste = [[0, 0] for ligne in range(3)]
print(multi_liste)      # [[0, 0], [0, 0], [0, 0]]
```

— Page 126 : au lieu de :

```
>>> {n : x**2 for n, x in enumerate(range(5))}
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

lire :

```
>>> {n : n**2 for n in range(5)}
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

— Page 126 : au lieu de « qu'elles ne calculent », lire « qu'ils ne calculent »
— Page 127 : au lieu de :

```
>>> def countDown(n):
...     """génère un décompteur à partir de <n>.
...
...     Un générateur ne peut retourner que None.
...     """
```

lire :

```
>>> def countDown(n):
...     """génère un décompteur à partir de <n>.
...
...     Un générateur ne peut retourner que None (implicite en l'absence d'instruction return).
...     """
```

— Page 137 : nouveaux exemples de directive `lambda` :

```
>>> # On peut utiliser la fonction print() en tant qu'expression
>>> majorite = lambda x : print('mineur') if x < 18 else print('majeur')
>>> majorite(15)
mineur
>>> majorite(25)
majeur
>>> # Retourne le tuple somme et différence de ses deux arguments
>>> t = lambda x, y: (x+y, x-y)
>>> t(5, 2)
(7, 3)
```

— Page 172 : au lieu de :

```
>>> conda install numpy scipy pyqt matplotlib pandas sympy ipywidgets notebook
```

lire :

```
>>> conda install numpy scipy pyqt matplotlib pandas sympy ipywidgets notebook jupyter
```

— Page 172 : au lieu de « d'ajouter un nouveau shell », lire « d'ajouter un nouveau shell avec son propre paramétrage »
— Page 174 : au lieu de « avec la commande `notebook password` », lire « avec la commande `notebook password` saisie dans pyzo ou `jupyter-notebook password` si on est en ligne de commande »
— Page 178 : au lieu de « le type `str()` », lire « le type `str` »
— Page 194 : au lieu de « Lorsqu'on lance l'exécution », lire « Lorsqu'on lance l'exécution (utiliser F5 si les menus de débogage ne sont pas actifs) »
— Page 207 : au lieu de « Décalage binaire de X vers la gauche ou vers la droite de Y bits », lire « Décalage des bits de X, vers la gauche ou vers la droite, de Y positions »
— Page 215 : ajout dans le tableau des opérations ensemblistes :

```
x not in Z    x ∉ E : la non-appartenance
```