Modularité

Mise au point des programmes

Exercice 1 *QCM d'activation*

Pour chaque question, une seule réponse est correcte.

1. On dispose d'un fichier `outils.py` contenant une fonction `carre(n)`. Pour utiliser cette fonction depuis un autre fichier Python situé dans le même répertoire, on écrit :

   A. `include outils`

   B. `import outils` puis `outils.carre(5)`

   C. `import carre from outils`

   D. `outils.carre(5)` directement, sans import

2. Une docstring est :

   A. un commentaire commençant par `#`

   B. une chaîne de caractères placée en première ligne d'une fonction ou d'un module, servant de documentation

   C. un fichier texte accompagnant le module

   D. une variable spéciale contenant le nom du module

3. Le module `doctest` permet de :

   A. générer automatiquement la documentation d'un module

   B. vérifier que les exemples écrits dans les docstrings produisent le résultat attendu

   C. créer des tests unitaires dans un fichier séparé

   D. corriger automatiquement les erreurs dans les docstrings

4. Quelle est la différence entre `import math` et `from math import sqrt` ?

   A. Il n'y a aucune différence

   B. Avec `import math`, on écrit `math.sqrt(4)` ; avec `from math import sqrt`, on écrit `sqrt(4)`

   C. `from math import sqrt` est plus lent car il charge tout le module

   D. `import math` n'importe que la fonction `math`, pas le module entier

5. Un paquet (*package*) Python est :

   A. un fichier `.py` contenant plusieurs fonctions

   B. un répertoire contenant un fichier `__init__.py` et un ou plusieurs modules

   C. un fichier compressé `.zip` contenant du code Python

   D. une bibliothèque externe installée avec `pip`

Exercice 2 *Utiliser un module (guidé)*

On dispose du module `geometrie.py` vu en cours, contenant les fonctions `aire_cercle(rayon)`, `perimetre_cercle(rayon)` et `aire_rectangle(longueur, largeur)`.

1. Écrire un programme qui importe le module `geometrie` et affiche l'aire et le périmètre d'un cercle de rayon 7.
2. Réécrire le même programme en n'important que les fonctions `aire_cercle` et `perimetre_cercle`. Quelle syntaxe faut-il utiliser ?
3. Que se passe-t-il si on écrit `from geometrie import *` ? Pourquoi cette pratique est-elle déconseillée ?
4. Écrire l'instruction permettant d'afficher la documentation de la fonction `aire_cercle` depuis l'interpréteur Python.

Exercice 3 *Écrire et tester des docstrings (guidé)*

1. Compléter les docstrings de la fonction suivante en ajoutant une description et deux exemples testables avec `doctest` :

```
def est_palindrome(mot):
    """
    -----------------------------------

    >>> est_palindrome("kayak")
    ----
    >>> est_palindrome("python")
    ----
    """
    return mot == mot[::-1]
```

2. Écrire une fonction `pgcd(a, b)` qui calcule le PGCD de deux entiers par l'algorithme d'Euclide. Ajouter une docstring complète avec au moins trois exemples, dont un cas limite (par exemple `pgcd(0, 5)`).
3. Écrire le code permettant de lancer les tests `doctest` sur ces deux fonctions.

Exercice 4 *Créer un module de statistiques*

On souhaite créer un module `stats.py` contenant les fonctions suivantes, chacune documentée avec une docstring et des exemples `doctest` :

1. `moyenne(liste)` : retourne la moyenne arithmétique d'une liste de nombres.
2. `variance(liste)` : retourne la variance de la liste, définie par $\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$ où $\bar{x}$ est la moyenne.
3. `ecart_type(liste)` : retourne l'écart type (racine carrée de la variance).
4. `mediane(liste)` : retourne la médiane d'une liste triée. Si la liste a un nombre pair d'éléments, la médiane est la moyenne des deux éléments centraux.
5. Ajouter en fin de fichier le bloc suivant et expliquer son rôle :

```
if __name__ == "__main__":
    import doctest
    doctest.testmod()
```

Exercice 5 *Structurer un projet (synthèse)*

On souhaite organiser un petit projet de gestion de notes en trois modules, regroupés dans un paquet `gestion_notes/`.

1. Proposer une structure de fichiers pour ce paquet, avec les modules suivants :
   - `eleve.py` : contient une classe `Eleve` avec un nom et une liste de notes ;
   - `calculs.py` : contient les fonctions `moyenne(notes)` et `meilleure_note(notes)` ;
   - `affichage.py` : contient une fonction `bulletin(eleve)` qui affiche le nom, la moyenne et la meilleure note.
2. Écrire le contenu du fichier `eleve.py` avec une classe `Eleve` dotée d'un constructeur, d'une méthode `ajouter_note(note)` et d'une méthode `__repr__`. Documenter chaque méthode.
3. Écrire le contenu du fichier `calculs.py` avec les deux fonctions documentées.
4. Écrire le contenu du fichier `affichage.py`. Ce module devra importer les fonctions de `calculs.py`.
5. Écrire un fichier `main.py` (en dehors du paquet) qui crée deux élèves, leur ajoute des notes, puis affiche leur bulletin.