

Modularité

Fiche de synthèse

① MODULES

Définition 1 (Module) Un **module** est un fichier `.py` qui regroupe des définitions de fonctions, de classes ou de variables, réutilisables dans d'autres programmes.

La **modularité** consiste à découper un programme en modules, chacun chargé d'une tâche bien définie. Le programme devient plus lisible, chaque module peut être réutilisé et testé séparément, et plusieurs personnes peuvent travailler en même temps sur des modules différents.

Le fichier `geometrie.py` étant placé dans le même répertoire que le programme :

```
import geometrie
print(geometrie.aire_cercle(10))

from geometrie import aire_rectangle
print(aire_rectangle(3, 7))
```

Avec `import`, on préfixe chaque nom par celui du module ; avec `from ... import`, on n'importe que les noms cités et on les emploie sans préfixe.

② DOCUMENTATION ET TESTS

Définition 2 (Docstring) Une **docstring** est la chaîne placée en première ligne d'un module ou d'une fonction. Elle décrit son rôle, ses paramètres et sa valeur renvoyée.

```
def aire_rectangle(longueur, largeur):
    """Renvoie l'aire d'un rectangle.

    >>> aire_rectangle(3, 4)
    12
    """
    return longueur * largeur
```

- `help(aire_rectangle)` affiche cette documentation sans ouvrir le code source.
- Les lignes `>>>` sont des exemples que le module `doctest` exécute : `doctest.testmod()` n'affiche rien si chaque résultat obtenu est celui écrit sous l'exemple, et signale chaque écart sinon.

③ EXÉCUTER OU IMPORTER

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

Python donne à chaque module une variable `__name__`. Elle vaut `"__main__"` quand on lance le fichier lui-même : le bloc s'exécute et vérifie les exemples. Elle vaut le nom du module, ici `"geometrie"`, quand un autre programme l'importe : le bloc est ignoré.

④ API ET IMPLÉMENTATION

Définition 3 (API) L'**API** (interface de programmation) d'un module est l'ensemble des fonctions, classes et constantes qu'il met à disposition, avec leur spécification.

L'**implémentation** est le code qui réalise ces fonctions. On utilise un module à travers son API, sans lire son implémentation : celle-ci peut changer sans gêner les utilisateurs, tant que l'interface reste la même. Un nom qui commence par un tiret bas, comme `_carre`, désigne par convention une fonction interne, qui ne fait pas partie de l'API.

📖 **Utiliser une bibliothèque inconnue.** Lire la documentation avec `help(random.choice)`, repérer les paramètres attendus et ce qui est renvoyé (une séquence non vide, l'un de ses éléments au hasard), puis écrire un appel conforme : `random.choice(['rouge', 'vert'])`.

⑤ PAQUETS

Définition 4 (Paquet) Un **paquet** (*package*) est un répertoire qui contient plusieurs modules et un fichier `__init__.py`, qui peut être vide. On importe alors `from mon_jeu.personnage import Guerrier`.

📌 À retenir.

- Un module est un fichier `.py` ; on l'importe avec `import` ou `from ... import`.
- Chaque fonction publique a une docstring, avec des exemples vérifiés par `doctest`.
- Le bloc `if __name__ == "__main__":` ne s'exécute jamais lors d'un import.
- On utilise un module par son API, documentée par `help`, sans dépendre de son implémentation.