



ENSGTI
ÉCOLE D'INGÉNIEURS



Programmation Orientée Objet (Python)

TP2 : Fonctionnalités avancées de Python

[A.Martin](#)

2026-2027



Visual Studio Code



Objectif

Cette séance va vous initier à écrire du code plus concis et efficace en Python.

Vous apprendrez à :

- Manipuler des collections avec des fonctions d'ordre supérieur (`map()`, `filter()`, `reduce()`) et les fonctions anonymes `lambda`.
- Utiliser les compréhensions de listes et les générateurs pour simplifier la création de structures de données.
- Organiser votre code en créant des modules et des paquetages réutilisables.
- Appliquer ces outils à un cas pratique.

Durée estimée : 4 heures

1. Travail préparatoire

1. Ouvrez un terminal.
2. Dans votre dossier « `tp_python` », créez le dossier « `tp2` » et, dans ce dernier, le sous-dossier « `src` » qui contiendra vos scripts.
3. Créez l'environnement virtuel « `env_tp2` ». Vous devriez obtenir l'arborescence suivante :

```
tp2/  
|-- env_tp2_/          # les fichiers de l'environnement virtuel  
|-- src/               # vos scripts Python
```

4. Activez l'environnement virtuel et installez les bibliothèques Numpy et Matplotlib. Vous pouvez réutiliser le fichier « `requirements.txt` » du TP précédent :

```
pip install -r requirements.txt
```

5. Vérifiez l'installation :

```
pip list
```

2. Exercices

Chaque script de ce TP doit suivre le squelette suivant :

```
#!/usr/bin/env python3  
# Imports nécessaires (numpy, matplotlib, etc.)  
  
# Fonctions utiles (si besoin)  
  
def main():  
    # Instructions principales  
    pass  
  
if __name__ == "__main__":  
    main()
```

Exercice n°1 : Fonctions `lambda`, `map()`, `filter()`, `reduce()`

Créez le script « `src/lambda.py` ». Ce script doit :

1. Créer une liste `longueurs = [5, 10, 15, 20]`.
2. Utiliser `map()` et une fonction `lambda` pour calculer les périmètres de carrés dont la longueur d'un côté est stockée dans la liste.
3. Utiliser `filter()` pour garder les longueurs supérieures ou égales à 10.

4. Utiliser `reduce()` pour calculer le produit de l'ensemble des longueurs (module `functools`).

Rappel :

- `map(f, iterable)` : applique `f` à chaque élément.
- `filter(f, iterable)` : conserve les éléments pour lesquels `f(e)` est vrai.
- `reduce(f, iterable)` : combine les éléments 2 à 2.

Convertir en liste pour affichage avec `list(...)`.

Exercice n°2 : Compréhensions de listes et générateurs

1. Créez le script « `src/ages.py` » qui :

- Définit `ages = [18, 21, 25, 30, 35]`
- Crée une liste « `ages_en_mois` » en utilisant une compréhension.
- Affiche « `ages_en_mois` ».

2. Créez le script « `src/generateur.py` » qui :

- Contient une fonction génératrice « `gen_nombres_pairs(n)` » produisant les entiers pairs strictement inférieurs à `n`.
- Affiche les nombres pairs `< 50`.

Rappel : les générateurs utilisent `yield` au lieu de `return`, et permettent d'éviter de stocker toute la séquence en mémoire.

Exercice n°3 : Modules et paquetages

1. Créez le fichier « `mon_paquetage/operations.py` » avec deux fonctions :

- `addition(a, b)`
- `multiplication(a, b)`

2. Dans le dossier « `mon_paquetage` », ajoutez-y un fichier « `__init__.py` » (fichier vide)

3. Créez un script « `test_operations.py` » à la racine du dossier « `tp3` » qui importe ce module et utilise les fonctions avec :

- `addition(15, 30)`
- `multiplication(6, 7)`

4. Testez que tout fonctionne en gardant « `test_operations.py` » à la racine.

Exercice n°4 : Simulation de la réponse d'un capteur thermique (thermistance NTC)

Objectif

Utiliser la formule de Steinhart–Hart pour modéliser la température (en K) en fonction de la résistance d'une thermistance NTC (en Ω).

Formule directe ($R \rightarrow T$) :

$$\frac{1}{T} = A + B \ln(R) + C \ln(R)^3$$

Formule inversée ($T \rightarrow R$) :

$$R = \exp(\sqrt[3]{(y - x/2)} - \sqrt[3]{(y + x/2)})$$

avec :

$$x = \frac{1}{C} \left(A - \frac{1}{T} \right), \quad y = \sqrt{\left(\frac{B}{3C} \right)^3 + \left(\frac{x}{2} \right)^2}$$

Coefficients recommandés :

```
A = 0.001129148
B = 0.000234125
C = 8.76741e-8
```

Travail demandé

Écrivez le script « src/thermistance.py » qui réalise les opérations suivantes :

- Définit deux fonctions :
 - « calculer_temperature(R, A, B, C) » : retourne T en °C à partir de R en ohms.
 - « calculer_resistance(T_K, A, B, C) » : retourne R à partir de T en Kelvin.
- Génère les températures de 0°C à 100°C (pas de 5°C) et les convertit en Kelvin.
- Calcule la résistance correspondante pour chaque valeur de T.
- Vérifie l'inversion en recalculant T à partir de R.
- Trace la courbe $R = f(T)$ avec Matplotlib (T en °C, R en ohms).
- Détermine les plages où la variation de R est presque linéaire en :
 - Proposant un critère simple (pente constante ou erreur < seuil).
 - Utilisant filter() pour les identifier.

Squelette de départ

```
#!/usr/bin/env python3
import numpy as np
import matplotlib.pyplot as plt
import math

A = 0.001129148
B = 0.000234125
C = 8.76741e-8

def calculer_temperature(R, A, B, C):
    pass # À compléter

def calculer_resistance(T_K, A, B, C):
    pass # À compléter

def main():
    pass # À compléter

if __name__ == "__main__":
    main()
```

Références

[Équation de Steinhart–Hart](#)

Résumé

Cette séance vous a permis de découvrir des fonctionnalités pour écrire du code Python plus efficace et maintenable. Les fonctions `lambda`, les fonctions d'ordre supérieur comme `map()`, `filter()` et `reduce()`, ainsi que les compréhensions de listes et les générateurs sont très utiles lorsqu'il s'agit de travailler avec des collections de données.

La création de modules et de paquetages vous aidera à organiser votre code et à favoriser sa réutilisation, ce qui est essentiel dans le développement de projets complexes.