# TP 1 : Découverte de Copilot

## Objectif

- Formuler un commentaire clair pour guider Copilot.
- Générer automatiquement des fonctions simples, des structures de contrôle (conditions, boucles), et des déclarations de variables/objets.
- Comprendre comment Copilot interprète le contexte du fichier et des prompts.
- Apprendre à améliorer un prompt pour obtenir un meilleur code.

## Énoncé

1. **Initialiser le projet:**
    - Créer un dossier: tp1_copilot_intro.
    - Fichier principal: main.py (ou index.js en JS).

***Attention Windows** : pour exécuter Python, utilisez la commande py main.py (et non python main.py).*

2. **Demander une fonction de somme d'une liste**:
    - Dans main.py, écrire :

```
# Écrire une fonction sum_list(nums: list[int]) -> int qui retourne la somme des éléments
```

→ Copilot propose une fonction. Si elle utilise sum(), ajouter :

```
# Implémenter sans utiliser la fonction built-in sum()
```

3. **Générer des commentaires explicatifs** :
    - **Ajouter**:

```
# Ajouter des commentaires ligne par ligne pour expliquer la logique aux débutants
```

    - **Action :** laisser Copilot détailler; ajuster si besoin pour clarté pédagogique.
4. **Boucles et fonctions**
    - **Ecrire**:

```
# Parcourir une liste [1, 2, 3, 4, 5] et afficher chaque carré
```

    - **Puis,** transformer en fonction :

```
# Transformer en fonction squares (nums: list[int]) -> list[int]
```

5. **Validation rapide :**
    - **Ajouter un bloc d'essai:**

```
if __name__ == "__main__":
    print(sum_list([1, 2, 3]))  # attendu: 6
    print(squares([1, 2, 3, 4]))  # attendu: [1, 4, 9, 16]
```

    - Exécuter avec py main.py
    - vérifier les sorties, corriger si nécessaire.
6. **Amélioration du prompt:**
    - Expérimenter : reformuler les commentaires pour guider Copilot vers un code plus lisible et conforme aux conventions de l'entreprise
    - Exemple :

```
# Écrire une fonction qui respecte les conventions PEP8 et ajoute des annotations de type
```