Introduction à Python

- Créé en 1991 par Guido van Rossum.
- Syntaxe claire et simple, favorisant la lisibilité du code.
- Utilisé pour un large éventail de tâches, de l'analyse des données au développement web.

Les avantages de Python

- **Facilité d'apprentissage :**

  Syntaxe intuitive et abondance de ressources pédagogiques.

- **Polyvalence :**

  Adapté à diverses applications, incluant l'automatisation, l'analyse de données, et le développement web.

- **Grande communauté :**

  Une communauté active offrant support et nombreuses bibliothèques.

Environnements d'exécution

- Python peut être exécuté sur différentes plateformes (Windows, MacOS, Linux).
- Utilisé aussi bien sur le côté serveur que dans des environnements locaux pour des scripts et automations.
- Python est également intégré dans des environnements web grâce à des frameworks comme Django et Flask.

L’évolution de Python

- Python a connu plusieurs versions majeures, avec Python 2 et Python 3 comme jalons importants.
- Python 3, introduit en 2008, est maintenant le standard, avec des améliorations significatives en termes de performance et de fonctionnalités.

Pourquoi apprendre Python

- **Popularité croissante :**

  Python est devenu l'un des langages les plus utilisés dans le monde de la technologie.

- **Opportunités de carrière :**

  Compétences en Python recherchées dans de nombreux secteurs (science des données, développement web, etc.).

- **Facilité d'intégration :**

  Python peut être intégré dans divers systèmes et interagit bien avec d'autres langages et technologies.

Les éditeurs de code pour Python

- **Pourquoi un éditeur de code ?** :

  Un bon éditeur aide à écrire, organiser et déboguer le code efficacement.

- **Caractéristiques communes :**

  - Coloration syntaxique pour améliorer la lisibilité.
  - Auto-complétion pour accélérer l'écriture du code.
  - Outils de débogage intégrés.

Visual Studio Code (VSC)

- **Gratuit et open-source** :
  Développé par Microsoft, largement adopté par la communauté des développeurs.

- **Extensions puissantes** :
  Support de Python via des extensions qui ajoutent des fonctionnalités comme l'intelliSense, l'analyse du code, et le débogage.

- **Personnalisable** :
  Une grande flexibilité avec des thèmes, des raccourcis clavier, et des extensions spécifiques.

- **Multi-langages** :
  Supporte divers langages de programmation au-delà de Python.

PyCharm

- **Développé par JetBrains** :

  Un environnement de développement intégré (EDI) conçu spécifiquement pour Python.

- **Outils intégrés** :

  Outils de débogage, analyse de code, et gestion des environnements virtuels.

- **Versions disponibles** :

  Community (gratuite) : Pour les développeurs de base et les projets open-source.

  Professional (payante) : Inclut des outils supplémentaires pour le web et les bdd

Comment choisir son éditeur ?

- **Type de projet :**
  Pour de grands projets, un EDI comme PyCharm pourrait être plus adapté.
  Pour des scripts simples, VSC ou Sublime Text suffisent.

- **Performance** :
  Certains éditeurs sont plus légers et rapides, comme Sublime Text, ce qui est idéal pour les machines moins puissantes.

- **Extensions et plugins** :
  Choisir un éditeur qui offre des extensions adaptées à vos besoins spécifiques (tests, débogage, gestion de versions).

- **Support communautaire :**
  Un éditeur avec une large communauté peut offrir plus de support et de ressources (tutoriels, plugins).

Exemple de code Python

```python
import ipaddress

# Fonction pour vérifier la validité d'une adresse IP
def verifier_ip(ip):
    try:
        # Convertir l'adresse IP en objet IP pour validation
        ip_obj = ipaddress.ip_address(ip)
        return True, ip_obj
    except ValueError:
        return False, None

# Fonction pour vérifier si une adresse IP est privée ou publique
def type_ip(ip_obj):
    if ip_obj.is_private:
        return "privée"
    else:
        return "publique"

# Exemple d'adresse IP
ip_entrante = "192.168.1.1"

# Vérification de l'adresse IP
est_valide, ip_obj = verifier_ip(ip_entrante)

if est_valide:
    print(f"L'adresse IP {ip_entrante} est valide.")
    type_adresse = type_ip(ip_obj)
    print(f"L'adresse IP {ip_entrante} est une adresse {type_adresse}.")
else:
    print(f"L'adresse IP {ip_entrante} n'est pas valide.")
```