PYTHON

Présenté par :

Xavier TABUTEAU

Présentation des participants

- Votre nom et prénom
- Votre entreprise et le poste occupé
- Votre expérience en programmation
- Vos attentes de cette formation
- Votre système d’exploitation
- Les accès administrateurs de votre machine

Déroulement de la formation

Horaire
- Début tous les jours à 9h
- Mini-pause de 15 min le matin
- Une pause de 1h à 12h30
- Seconde mini-pause l’après midi de 15 min
- Fin à 17h30

Formation
- Exposé des théories + lien documentation
- Travaux pratiques
- Partage des liens et ressources grâce à Slack
- Evaluation finale

Présence
- Il faut émarger à chaque demi journée sur le site sign.m2iformation.fr

Prérequis

- Avoir les bases de la programmation (variables, fonctions, expressions, etc…).
- Savoir utilisé un IDE (Integrated Development Environment).
- Avoir des notions de classes, héritages, encapsulations et polymorphismes (concept de Programmation Orienté Objet).

Pourquoi Python ?

- Python est un langage interprété avec un typage dynamique fort, portable, extensible, gratuit, qui permet (sans l'imposer) une approche modulaire et orientée objet de la programmation.
- Particularité importante : Python n'utilise pas d'accolades ou d'autres délimiteurs (begin/end…) pour repérer les blocs d'un programme, mais l'indentation.

- Un exemple : en fonction d'une liste de valeurs, nous souhaitons savoir celles qui sont des nombres pairs et celles qui ne le sont pas.

En PHP

```php
<?php
function valeurs_paires($liste_valeurs) {
    $classement = array();

    for($i=0; $i<count($liste_valeurs); $i++) {
        if($liste_valeurs[$i] % 2 == 0) {
            array_push($classement, True);
        }
        else {
            array_push($classement, False);
        }
    }

    return $classement;
}

echo var_dump(valeurs_paires([51, 8, 85, 9]));
?>
```

En Python

```python
def valeurs_paires(liste_valeurs):
    return [(False if v % 2 else True) for v in liste_valeurs]

print(valeurs_paires([51, 8, 85, 9]))
```

Champs d’application de Python

Nous trouvons Python dans le Web, les multimédias, la bureautique, les utilitaires, l’intelligence artificielle, …

Nous le retrouvons dans tous les domaines professionnels tel que :
Le domaine scientifique, les finances, la programmation système, les base de données, …

Python à cette force de pouvoir réunir des profils d’informaticiens assez différents (administrateur système, développeur généraliste, développeur web, etc…).

Positionnement de Python

- **Indice TIOBE des langages de programmations**

https://www.tiobe.com/tiobe-index/

Au jour où sont écrits ces lignes, Python est le langage le plus utilisé avec 15% d'avance sur C++, C et Java.

- **Historique de Python**

Décembre 1989 : Guido van Rossum commence à développer Python.
Février 1991 : Première version publique (v0.9.0).
26 Janvier 1994 : Python 1.0.
Octobre 2000 : Python 2.0.
3 décembre 2008: Python 3.0 (refonte majeure du code et rupture de compatibilité).
2010 : Python devient un projet communautaire sous la Python Software Fundation.
2020 : Fin du support de Python 2 (après presque 20 ans).
Octobre 2021 : Python 3.10.

Depuis, tous les ans en octobre, Python évolue d'une version (3.11, etc…)

Avantages et inconvénients

- **Avantages**

  - Simplicité et lisibilité
    Syntaxe claire, facile à apprendre même pour les débutants.
    Idéal pour l'enseignement et le prototypage rapide.

  - Polyvalence
    Utilisable pour le web, la data science, l'IA, les scripts, la robotique, etc.

  - Grande communauté
    Nombreux forums, documentations et ressources d'apprentissage.

  - Enorme bibliothèque standard + packages externes
    Modules intégrés et frameworks (Django, Flask, NumPy, Pandas, TensorFlow…).

  - Multi-plateforme
    Fonctionne sur Windows, macOS, Linux, et même sur certaines cartes embarquées (Raspberry Pi).

Avantages et inconvénients

- **Inconvénients**

  - Vitesse d'exécution plus lente
    Interprété → moins rapide que C, C++ ou Java pour des calculs très lourds.

  - Consommation mémoire
    Peut être gourmand en ressources pour de gros projets.

  - Pas toujours adapté au mobile
    Moins utilisé pour les applications mobiles natives par rapport à Swift ou Kotlin.

  - Gestion du multithreading limitée
    A cause du Global Interpreter Lock (GIL), certaines tâches parallèles CPU sont moins efficaces.

- **Conclusion**

Python est idéal pour apprendre à programmer, démarrer rapidement un projet et travailler sur des domaines modernes comme l'IA ou l'analyse de données.
Cependant, pour des applications très performantes ou embarquées, d'autres langages peuvent être plus adaptés.

Les interpréteurs Python

Bien que Python possède son propre interpréteur et soit le standard, il en existe d'autres tel que IPython et bpython. IPython est un interpréteur interactif avancé, très populaire dans la communauté scientifique. bpython est une alternative plus légère et plus colorée à Ipython.

- **Python (standard)**

  Avantages :

  Léger et toujours disponible.
  Suffisant pour tester des commandes simples.

  Inconvénients :

  Pas très convivial.
  Pas de complétion automatique ni d'historique riche.
  Pas de coloration syntaxique. (sauf pour les sorties d'erreurs depuis les nouvelles version 3.10+).

- **IPython (Interactive Python)**

  Avantages :

  Complétion automatique avec tabulation.
  Historique des commandes amélioré.
  Coloration syntaxique.
  Support du debugging, du profiling, et des magics (commandes spéciales comme %time, %run, etc.).
  Intégration avec Jupyter Notebook.

  Usage typique :

  Très utilisé en science des données, recherche...

Les interpréteurs Python

Le "b" dans bpython ne correspond pas à un mot précis officiellement documenté par les créateurs, mais il est généralement compris comme signifiant "better" ou "beautiful", pour insister sur le fait que bpython est une version améliorée, plus "jolie" et ergonomique que l'interpréteur Python standard.

- **bpython**

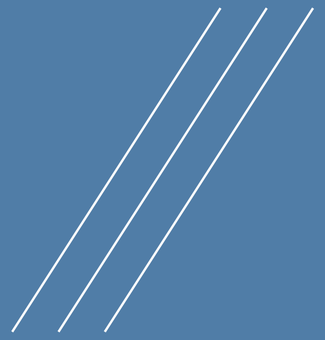
  Avantages :

  Interface colorée et agréable.
  Suggestions automatiques de code.
  Possibilité de voir le code source d'une fonction.
  Rewind : possibilité d'effacer la dernière commande.
  Fonctionne bien dans des environnements limités.

  Usage typique :

  Pour les développeurs qui aiment travailler en terminal, avec un outil léger et interactif.

PYTHON



Présenté par

Xavier TABUTEAU