Soirée 10/03/2025 Info Python Prepa

plan soirée (idées)

1. Presentation rapide du langage python : Pourquoi programmer en python? ; Avantages/ Inconvenients
2. Les differentes versions de Python: Version 2, version 3 , Micropython, Cpython
3. Comment programmer , les outils: IDE , Plugins,
4. Un exemple de programme simple : print(); Choix If ... else; Programme calculatrice
5. Programmer en Ligne avec micropython : Wokwi ....
6. Python dans node-red : Librairies , Exemples
7. Mise en pratique sur votre PC: Avec l'aide de ChatGPT...

programmes de test

Programme python proposé par Xavier

menu_prenom_option_devinette_calculette_0000.py.zip

Modification du programme python proposé par Xavier modifié par GL

menu_prenom_option_devinette_calculette_0003.py.zip

rectangleentourcercle.py.zip

python:neopixel

bonjour en python dans rectangle

Cercle en mode blocs en python

exemples

Presenatation PYthon3

matplotlib exemples

http://tableauxmaths.fr/spip/spip.php?rubrique26

https://ww2.ac-poitiers.fr/math/spip.php?article943

https://lamerci-maths-2nde.jimdofree.com/programmation-en-python/1%C3%A8re-s%C3%A9ance-ordinateur-langages-et-python/

https://fr.slideserve.com/xander/formation-python

https://cours.brosseau.ovh/cours/python.html

https://www.slideserve.com/aldon/introduction-python-powerpoint-ppt-presentation

https://docs.python.org/fr/3/index.html

https://python.guillod.org/python.pdf

http://exo7.emath.fr/cours/livre-python1.pdf

https://www.zonensi.fr/Maths/Seconde/AlgoBases/ProgrammationPythonThonny/

https://inforef.be/swi/download/apprendre_python3_5.pdf

https://zestedesavoir.com/tutoriels/pdf/2514/un-zeste-de-python.pdf

http://www.larsen-b.com/static/intro_python/

https://brython.info/cours_python/cours_python.html

https://inforef.be/swi/download/apprendre_python3_5.pdf

https://koor.fr/Python/Tutorial/python_ide_spyder.wp

http://scol.edubois.info/blockly/documents/python/Synth%C3%A8se_Python.pdf

http://python.lycee.free.fr/presentation-python/#slide1

https://eric.univ-lyon2.fr/ricco/cours/slides/PE%20-%20fichiers%20sous%20python.pdf

https://www.mycompiler.io/fr/new/python

https://www.ludomag.com/wp-content/uploads/2021/02/Scratch-to-Python.pdf

https://fr.slideshare.net/slideshow/introduction-python-238780636/238780636

Proposition pour la presentation de python

Pour faire une présentation de Python3 à des débutants et les motiver à l'utiliser, il est important de rendre la présentation accessible, engageante et dynamique. Voici quelques conseils et étapes pour organiser cette présentation :

1.

- **Commencer avec une question ou un problème simple :**
  - Exemple : "Avez-vous déjà rêvé de créer vos propres applications ou automatiser des tâches répétitives sur votre ordinateur ?"
- **Mettre en avant la simplicité et l'accessibilité de Python :**
  - Expliquez que Python est l'un des langages de programmation les plus simples à apprendre grâce à sa syntaxe claire et sa communauté active.

2.

- **Facilité de lecture et d’écriture :**
  - Montrez quelques exemples de code Python simples et comparé à d'autres langages plus complexes comme C++ ou Java.
  - Exemple : Montrez comment "Hello World" peut être écrit en Python en une seule ligne :

    ```
    print("Hello, World!")
    ```

  - Comparez avec d’autres langages où c'est beaucoup plus complexe.
- **Polyvalence :**
  - Python est utilisé dans de nombreux domaines : développement web (Django, Flask), science des données (pandas, NumPy), intelligence artificielle (TensorFlow, PyTorch), automatisation, etc.
- **Communauté et ressources :**
  - Mettez en avant la richesse de la communauté Python, qui propose des tutoriels, des forums, des bibliothèques et des frameworks pour quasiment toutes les problématiques.

3.

- **Automatisation des tâches :**
  - Montrez comment Python peut être utilisé pour automatiser des tâches simples comme renommer plusieurs fichiers ou récupérer des informations sur internet.
  - Exemple de script pour renommer des fichiers automatiquement.
- **Création de jeux et applications :**
  - Expliquez que Python permet de créer des jeux simples ou des applications utiles avec des bibliothèques comme **Pygame**.
- **Analyse de données :**
  - Montrez une analyse simple avec des bibliothèques comme **pandas** pour manipuler des données et **matplotlib** pour les visualiser.

4.

- **Exécuter un petit script Python** en direct :
  - Créez un programme simple et interagissez avec le public. Par exemple, un générateur de mot de passe ou un programme qui effectue un calcul mathématique de base.
- **Faites-les participer** : Demandez aux participants de proposer des idées simples de programmes qu'ils aimeraient créer (calculatrices, générateur de citations, etc.)

5.

- **Démontrez que la programmation peut être amusante et accessible.**
  - Expliquez que les erreurs font partie du processus d'apprentissage et qu'il existe de nombreuses ressources pour obtenir de l'aide (StackOverflow, documentation officielle).
- **Faire un parallèle avec des choses qu'ils connaissent** : par exemple, un programme Python pourrait être comparé à une recette de cuisine (étapes bien définies).

6.

- **Donner des ressources pour commencer** :
  - Encouragez-les à rejoindre des forums, des communautés en ligne et à consulter des cours gratuits sur des sites comme **Codecademy**, **Coursera** ou **Khan Academy**.
- **Petits projets à réaliser** :

    - Proposez des mini-projets faciles à réaliser : un chatbot, un générateur de mots de passe, un jeu simple.
- **Rappeler qu'ils peuvent apprendre à leur rythme** :
    - Python permet de progresser à son propre rythme, en commençant par des petites bases pour ensuite aborder des concepts plus complexes.

7.

- **Encourager à persévérer et à s'amuser :**
    - "La programmation est un jeu, plus vous jouez, plus vous devenez compétent !"
- **Insister sur les possibilités infinies** : avec Python, il n'y a pas de limite à ce qu'ils peuvent créer. Tout est possible avec ce langage.

Exemple d'agenda pour la présentation :

1. **Introduction à Python (5 minutes)**
2. **Pourquoi choisir Python ? (5-10 minutes)**
3. **Applications concrètes et exemples de projets (10 minutes)**
4. **Démonstration en direct (10 minutes)**
5. **Encouragement à se lancer avec des ressources (5 minutes)**
6. **Conclusion et questions (5 minutes)**

Conseils supplémentaires :

- **Utilisez des analogies simples** : Les concepts de programmation peuvent être difficiles au début, donc faites des analogies avec des choses qu'ils connaissent.
- **Restez interactif et engageant** : Posez des questions au public et encouragez-les à participer.
- **Utilisez des visuels attractifs** : Montrez des images ou des vidéos qui illustrent les applications de Python (ex. : des graphiques créés avec Python, des jeux, des analyses de données).

Cela devrait rendre la présentation plus dynamique, moins intimidante, et plus attrayante pour les débutants.

Pour expliquer Python3 de manière visuellement attractive et dynamique, l'utilisation de visuels clairs et pertinents peut aider à rendre les concepts plus accessibles et engageants. Voici quelques idées de visuels que vous pouvez utiliser :

1.

- **Exemple avec matplotlib** : Montrez un graphique simple généré avec **matplotlib** ou **seaborn**, deux bibliothèques Python populaires pour la visualisation de données.
    - **Exemple de graphique** : Un graphique à barres, un nuage de points, ou un histogramme. Montrez comment Python peut transformer des données brutes en visualisations facilement compréhensibles.
- **Comparaison de visualisations** : Avant et après l'utilisation de Python pour analyser des données. Par exemple, un tableau Excel brut transformé en un graphique avec Python.

2.

- **Automatisation** : Montrez une capture d'écran d'un script Python automatisant une tâche (par exemple, renommer des fichiers ou télécharger des fichiers depuis internet).
    - Un exemple visuel d'un script simple qui parcourt un dossier et renomme les fichiers en fonction d'un certain critère.
- **Création d'un jeu simple avec Pygame** : Vous pouvez montrer une capture d'écran d'un petit jeu que vous avez créé avec **Pygame**, comme un jeu de type "Snake" ou "Pong".
    - Un gif animé montrant une partie du jeu ou le mouvement des objets à l'écran peut rendre la présentation plus interactive.

3.

- **Code Python propre et lisible** : Affichez un extrait de code Python simple, bien structuré, en utilisant une syntaxe colorée (idéalement dans un éditeur comme **VSCode** ou **PyCharm**). Utilisez des commentaires dans le code pour expliquer brièvement ce qu'il fait.
    - Exemple : Un petit programme qui demande à l'utilisateur son nom et affiche un message de bienvenue :

      ```
      name = input("Quel est ton nom ? ")
      print("Bonjour, " + name + " !")
      ```

    - Vous pouvez mettre en évidence les éléments du code comme les variables ou les instructions de manière à rendre les concepts plus clairs visuellement.

4.

- **Avant/Après** : Montrez l'évolution d'un petit projet Python, en commençant par un programme basique qui ne fait qu'une chose simple, puis en l'enrichissant pour inclure des fonctionnalités plus complexes.
    - Exemple : Commencez par un script qui imprime un message, puis passez à un projet plus complexe (comme une application de gestion de contacts ou une calculatrice).

5.

- **Concepts de base de Python** : Utilisez des cartes mentales pour expliquer les concepts de base de Python comme les variables, les types de données (entiers, chaînes de caractères, etc.), les boucles, les conditions, et les fonctions. Cela aide à rendre la structure du langage plus claire.
- **Diagramme de flux** : Pour expliquer la logique des programmes avec des conditions et des boucles, un diagramme de flux visuel peut aider à comprendre les étapes du programme.
    - Exemple : Montrez un diagramme pour expliquer comment une boucle `for` fonctionne, ou un diagramme pour un programme de conditionnelle (`if` / `else`).

6.

- **Visualisation des différences de syntaxe** : Affichez des exemples de code pour résoudre un même problème dans différents langages de programmation. Montrez comment un simple programme "Hello World" ou une boucle fonctionne en Python comparé à un autre langage (par exemple, C, Java ou JavaScript).
    - Exemple : Comment un programme en Python est beaucoup plus concis et facile à comprendre par rapport à d'autres langages.

7.

- **Jupyter Notebooks ou Google Colab** : Montrez des captures d'écran d'un environnement interactif comme **Jupyter Notebook** ou **Google Colab** où le code Python peut être exécuté en temps réel et voir les résultats instantanément.
- Vous pouvez également intégrer des exemples interactifs directement dans un notebook avec des graphiques ou des calculs.

8.

- **Exemple d'IA avec Python** : Affichez un visuel montrant un modèle d'intelligence artificielle en train de prédire quelque chose à partir de données. Par exemple, la création d'un modèle prédictif pour des données sur les ventes, l'analyse des tendances, ou la reconnaissance d'image avec **TensorFlow** ou **Keras**.
    - **Exemple visuel** : Une image montrant l'évolution d'un modèle qui prédit des résultats basés sur des données.

9.

- **Icônes des ressources populaires** : Affichez des icônes et logos des bibliothèques Python populaires comme **Django**, **Flask**, **Pandas**, **NumPy**, **TensorFlow**, etc., avec des descriptions courtes de ce que chaque bibliothèque permet de faire.
- **Capture d'écran d'une communauté** : Montrez un forum ou une ressource en ligne où les débutants peuvent trouver de l'aide, comme StackOverflow ou des blogs Python.

10.

- **Animations de concepts** : Créez ou utilisez des animations GIF pour illustrer des concepts comme les boucles (par exemple, une animation montrant une boucle `for` qui parcourt une liste).
- **Démonstration de l'exécution du code** : Utilisez un GIF montrant l'exécution d'un petit programme Python et l'affichage des résultats à l'écran.

En résumé :

Les visuels doivent rendre l'apprentissage de Python intuitif et démontrer sa simplicité. Utilisez des graphiques de données, des exemples de code clairs et interactifs, des diagrammes de flux, des comparaisons avec d'autres langages, des animations, et des démonstrations concrètes d'applications. Cela permettra de garder les débutants engagés et de mieux visualiser comment Python peut être utilisé pour créer des projets passionnants et utiles.

Qu'est-ce que Python ? - Python est un langage de programmation interprété, de haut niveau, et largement utilisé. - Il a été conçu par Guido van Rossum et a été lancé en 1991. - Python est connu pour sa syntaxe simple et claire, ce qui en fait un langage populaire pour les débutants.

Pourquoi choisir Python ? - Facilité d'apprentissage et de lecture. - Polyvalence : utilisé dans divers domaines tels que le web, la science des données, l'intelligence artificielle, l'automatisation, etc. - Grande communauté et de nombreuses ressources.

Historique et évolution de Python - Python 2 a été largement utilisé, mais la version Python 3 a été lancée en 2008 pour corriger certaines limitations de Python 2. - La version 3 est la version recommandée aujourd'hui, car Python 2 a été officiellement abandonné en 2020.

Installer Python 3 - Télécharge Python 3 depuis python.org. - L'installation sur Windows et macOS est simple via des exécutables. - Sur Linux, Python est généralement déjà préinstallé.

Configurer l'environnement de développement - Choisir un IDE (environnement de développement intégré) comme PyCharm, Visual Studio Code ou même un simple éditeur de texte comme Sublime Text. - Vous pouvez aussi utiliser des notebooks comme Jupyter pour les projets de science des données.

Syntaxe et structure de base - Python est indenté, ce qui signifie que la structure du code est définie par l'indentation et non par des accolades.

Exemple de code simple :

```
# Ceci est un commentaire
print("Hello, Python!")
```

Variables et types de données - Les variables sont dynamiquement typées en Python. On peut utiliser des types comme `int`, `float`, `str`, `bool`, etc.

Exemple :

```
x = 10          # Un entier
y = 3.14        # Un nombre à virgule flottante
nom = "Alice"   # Une chaîne de caractères
```

Opérateurs - Python supporte des opérateurs arithmétiques (addition, soustraction, multiplication, etc.), logiques (AND, OR, NOT), et de comparaison (égalité, inégalité, supérieur à, etc.).

Exemple :

```
a = 5
b = 3
print(a + b)   # Affiche 8
```

Contrôle de flux - Utilisation des structures de contrôle comme `if`, `else`, `elif`, et les boucles `for`, `while`.

Exemple :

```
x = 5
if x > 3:
    print("x est supérieur à 3")
```

Déclaration de fonctions - Les fonctions en Python sont définies avec le mot-clé `def`.

Exemple :

```
def saluer(nom):
    print("Bonjour, " + nom + "!")
saluer("Alice")
```

Paramètres et valeurs de retour - Une fonction peut accepter des paramètres et renvoyer des valeurs avec `return`.

Exemple :

```
def additionner(a, b):
    return a + b

print(additionner(5, 3))  # Affiche 8
```

Fonctions lambda - Les fonctions lambda sont des fonctions anonymes, souvent utilisées pour des opérations rapides.

Exemple :

```
carre = lambda x: x**2
print(carre(4))  # Affiche 16
```

Listes, tuples, dictionnaires, ensembles - Python fournit des structures de données natives pour stocker des collections d'éléments. - **Listes** : Mutables, ordonnées. - **Tuples** : Immutables, ordonnés. - **Dictionnaires** : Clé-valeur, non ordonnés. - **Ensembles** : Collections non ordonnées d'éléments uniques.

Exemple :

```
# Liste
fruits = ["pomme", "banane", "cerise"]
# Tuple
couleurs = ("rouge", "vert", "bleu")
# Dictionnaire
personne = {"nom": "Alice", "âge": 25}
# Ensemble
animaux = {"chat", "chien", "oiseau"}
```

Compréhensions de listes - Une manière concise de créer des listes à partir d'autres itérables.

Exemple :

```
carres = [x**2 for x in range(5)]
print(carres)  # Affiche [0, 1, 4, 9, 16]
```

Classes et objets - Python supporte la programmation orientée objet (POO) avec des classes, des objets, de l'héritage, et du polymorphisme.

Exemple de classe :

```
class Chien:
    def __init__(self, nom):
        self.nom = nom
    def aboyer(self):
        print(self.nom + " dit Woof!")

chien = Chien("Rex")
chien.aboyer()  # Affiche "Rex dit Woof!"
```

Importer des modules - Python permet d'importer des bibliothèques externes via le mot-clé `import`.

Exemple :

```
import math
print(math.sqrt(16))  # Affiche 4.0
```

Bibliothèques populaires - **NumPy** : pour les calculs scientifiques. - **Pandas** : pour la manipulation de données. - **Matplotlib** : pour la visualisation de données.

Try-except - Python utilise le bloc `try-except` pour gérer les erreurs sans interrompre l'exécution du programme.

Exemple :

```
try:
    x = 10 / 0
except ZeroDivisionError:
    print("Erreur : division par zéro!")
```

Quelques projets que vous pouvez réaliser pour approfondir vos connaissances de Python : - **Calculatrice simple** : Créez une application de calculatrice avec des opérations de base. - **Jeu de devinettes** : Créez un jeu où l'utilisateur doit deviner un nombre choisi au hasard par l'ordinateur. - **Gestion de contacts** : Créez une petite application pour ajouter, supprimer et afficher des contacts.

Utilisation de l IDE Thonny

https://www.zonensi.fr/Maths/Seconde/AlgoBases/ProgrammationPythonThonny/