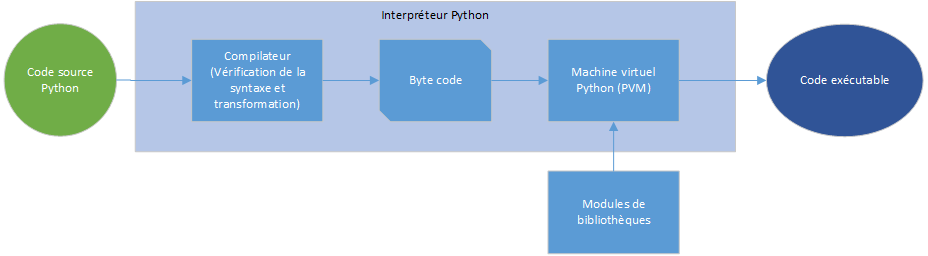
**1. Donner et expliquer le schéma de l’architecture interne de Python.**

Notre code source, tel que nous l’avons écrit n’est pas directement compréhensible et exécutable par la machine. Python étant un langage de programmation interprété, c’est-à-dire notre programme est exécuté ligne par ligne par la machine

Pour analyser notre code et savoir si celui-ci est bien écrit, python produit une sorte de code intermédiaire qui lui permet de faire cette vérification syntaxique, puis produit un byte code (code en langage machine), qui sera ensuite transmis à la machine virtuelle python pour l’intégration des modules et produire le code final en langage machine directement exécutable par l’ordinateur.

Voici un schéma qui présente l’architecture interne du langage python



**2. Quelle différence faites-vous entre une variable globale et une variable locale? Dites**

**Comment est-ce qu’on parvient à obtenir des variables globales en Python (les 2 cas**

**Possibles).**

Une variable est dite local si elle est définie dans une fonction ou une procédure et ne pourra être accessible que dans le cadre de l'exécution de cette procédure ou fonction tant dis qu'une variable globale est définie dans le programme est accessible durant toute l'exécution de celui-ci sans aucune contrainte.

On parvient à obtenir des variables globales en python au travers de ces deux méthodes:

* En les déclarant à l'extérieur des fonctions
* En les déclarant à l'intérieur des fonctions et procédures tout en utilisant le mot clé **"global"** avant celle-ci.

3. A quoi peut servir un décorateur ? (1 exemple d’utilisation suffit)

Un décorateur est une fonction qui permet de modifier le comportement d'une autre fonction.

Exemple :

def appel(func):

def inner():

print("Maison... Téléphone... Maison...")

return func()

return inner

@appel

def jeff():

print("Je lui ai dit de se calmer !")

@appel

def karl():

print("Il semble en colère contre le monde")

Notre décorateur @appel s’exécute sa fonction à chaque fois avant la fonction qui le suit

4. Quelle différence faites-vous entre Git et GitHub

**Git** est un logiciel VCS local qui permet aux développeurs de sauvegarder des instantanés de leurs projets au fil du temps. Il est généralement mieux adapté à un usage individuel tant dis que **GitHub** est une plateforme web qui intègre les fonctionnalités de contrôle de version de **Git** afin de pouvoir les utiliser en collaboration.

**Git** est le système de contrôle de version distribué. Git est chargé de suivre les modifications apportées au contenu – généralement des fichiers de code source.

**GitHub** est une entreprise qui fournit un hébergement de référentiel Git. Cela signifie qu’ils fournissent une solution clé en main pour héberger des référentiels Git sur leurs serveurs. Cela peut être utile pour conserver une sauvegarde de votre référentiel (Git ne suit que les modifications apportées à vos fichiers au fil du temps, le référentiel doit encore être sauvegardé) et pour disposer d’un emplacement centralisé pour conserver et partager votre code avec d’autres.