UNIVERSITÉ COTE D'AZUR
L1
2023-2024

Julien Deantoni

Introduction à l'Informatique par le Web

TP numéro 6

Fonctions utiles pour définir un serveur web en Python

Objectif

L'objectif de ce TP est de vous faire concevoir différentes fonctions utiles ainsi que la mise en place d'un serveur web en python.

Démarrage et prise en main de vscode

Nous allons travailler incrémentalement. Il va falloir s'organiser correctement dès le démarrage.
Dans un répertoire dédié à cette UE (que vous avez du créer la semaine précédente), créer un répertoire $serveurPython$ qui servira pendant plusieurs semaines. À l'intérieur de ce répertoire, mettez le fichier $iiwServerHelper.py$ qui se trouve sur moodle et sur ma page web. Créez maintenant un fichier python nommé $gestionRequeteFichierGet.py$.
Nous allons toujours travailler dans *Visual Studio Code* mais cette fois nous n'utiliserons pas de "notebook", uniquement des fichiers textes contenant du Python. Lancez vscode comme lors des autres TPs.
Une fois lancé, ouvrez le **dossier** $serveurPython$ (et non pas un fichier) en allant dans $File->OpenFolder$.
Une fois ceci fait, vous verrez votre fichier dans la partie gauche du logiciel cliquez dessus pour l'ouvrir.
Vous pouvez maintenant écrire du code python dans ce fichier. Essayer par exemple de taper le code suivant:

chaine de caractères multi lignes

```python
i: int = int(0)
s: str = str("Hello")
print(s,i)
```

Vous avez plusieurs manières d'exécuter ce code.

1. vous pouvez ouvrir la vue $Terminal$ (Menu $Terminal->newTerminal$) puis dans cette *invite de commande* vous tapez `python3 gestionRequeteFichierGet.py`. Vous verrez `Hello 0` s'afficher en dessous de la ligne précédente. à tout moment, si votre code ne s'arrête pas, vous pouvez taper la combinaison de touches `Ctrl+C` pour forcer l'arrêt de votre programme.

2. Vous pouvez utiliser l'interface graphique. Pour cela, identifiez en haut à droite la flèche "run" (▷). Si vous cliquez directement dessus, cela exécute votre programme dans l'*invite de commande* (le terminal) et vous pouvez y lire `Hello 0`. Vous pouvez taper la combinaison de touches `Ctrl+C` pour forcer l'arrêt de votre programme. Notez que vous pouvez aussi cliquer sur la petite flèche à côté de la flèche "run" pour lancer votre programme en mode *debug*. Cela vous permet de mettre des "points d'arrêt" dans votre code. Par exemple vous pouvez cliquer juste à gauche

du numéro de ligne (par exemple de la ligne $print(s,i)$). Un point rouge apparait, c'est un point d'arrêt. Lancer maintenant votre programme en mode debug ( ). Cela ouvre une nouvelle barre de commande:  . De plus, vous devriez voir les valeurs des différentes variables dans une fenêtre sur la gauche. La barre de commande qui s'est ouverte vous permet, depuis un point d'arrêt, d'avancer dans l'exécution du programme. Vous pouvez avancer jusqu'au prochain point d'arrêt (continue), ligne par ligne (step over), rentrer dans la définition de la fonction de la ligne concernée (step into) ou sortir de la fonction où se trouve le point d'arrêt (step out). Enfin, vous pouvez aussi arrêter à tout moment l'exécution en cliquant sur le carré rouge ou redémarrer l'exécution en cliquant sur la flèche verte.

Vous êtes maintenant prêts à définir les fonctions ci dessous. **Pensez bien à tester chacune de vos fonctions !**

1 Introduction

On désire dans un premier temps définir un serveur de fichiers html existants. Le serveur va donc attendre une requête et lorsque la requête arrive il devra faire les étapes suivantes:

1. Vérifier si le fichier demandé existe.
2. Si le fichier n'existe pas:
   (a) répondre un code 404
   (b) Dans une entête, prévenir qu'il va envoyer un contenu au format text/html
   (c) envoyer une page html stipulant l'erreur 404 et le chemin du fichier initialement demandé.
3. Si le fichier existe
   (a) répondre un code 200
   (b) Regarder quel est le type de fichier qui est demandé
   (c) Dans une entête, prévenir qu'il va envoyer un contenu au format trouvé précédemment.
   (d) Ouvrir le fichier demandé en binaire ou non
   (e) Envoyer le contenu du fichier demandé.

Nous allons donc définir plusieurs fonctions qui nous serons utiles pour coder ce comportement. Pour pouvoir définir le corps de ces fonctions, une description du type de paramètre et donnée.

2 Fonction `create404Page`

Définissez une fonction `create404Page` qui prend en paramètre le chemin vers le fichier initialement demandé et renvoie une chaine de caractères avec le contenu HTML de la page.
Note: En Python, vous pouvez définir une chaine de caractères sur plusieurs lignes en utilisant des triples guillemets, simples ou doubles. Voir ci dessous:

chaine de caractères multi lignes

```
s1: str = str("""Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.""")

s2: str = str('''Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
```

```
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.
''')
```

3 Fonction `getFileExtension`

Cette fonction va permettre d'extraire du `path` de la requête l'extension du fichier. Elle prend en paramètre le chemin vers le fichier demandé et renvoie une chaine de caractères contenant l'extension du fichier. Si le fichier demandé ne contient pas d'extension, la fonction renverra une chaine de caractères vide. Note: Une chaine de caractères peut être parcourue de la même manière qu'une liste, ou chaque élément de la liste est un caractère. Voir ci dessous

chaine de caractères utilisée comme une liste

```python
a: str = str('zaza')
print(a[0]) # affiche 'z'
print(len(a)) # affiche 4
print(a[3]) # affiche 'a'
```

⚠Vous n'utiliserez pour faire cette fonction que les choses que l'on a vu en cours et TD. En d'autres mots vous n'utiliserez pas la fonction `split`

4 Fonction `getContentType`

Afin de permettre au navigateur web de traiter correctement un fichier qu'on lui envoie il est nécessaire de lui dire le type de fichier qu'on lui envoie. C'est le `content-type`. Celui-ci peut s'obtenir en fonction de l'extension des fichiers. Ainsi pour un fichier html (ayant l'extension html), le content type sera `text/html`. Pour un fichier ayant l'extension .jpeg, ce sera `image/jpeg`. Vous allez écrire une fonction `getContentType` qui prend une fonction en paramètre et qui renvoie une string contenant la valeur associée du content type. Nous ne traiterons pas tous les cas mais uniquement ceux dont la règle d'association est donnée dans le tableau ci dessous:

extension	content-type
html	text/html
css	text/css
csv	text/csv
txt	text/plain
png	image/png
jpeg	image/jpeg
jpg	image/jpeg
gif	image/gif
webp	image/webp

5 Fonction `isBinaryFile`

Afin que le serveur ouvre et envoie correctement le fichier demandé dans une requête, il est nécessaire de savoir si il s'agit d'un fichier binaire ou non. Écrivez une fonction `isBinaryFile` qui renvoie *vrai* si un fichier est un fichier binaire; *faux* autrement. Pour cela nous ne considérerons que les types de fichier traités à la question précédente. Dans cette liste, seulr les fichiers ayant un content type commençant par `image` sont des fichiers binaires.

6 Fonction `handleGetRequest`

Écrire une fonction `handleGetRequest` permettant d'honorer une requête envoyée par un navigateur web. La fonction prend en paramètre une chaine de caractère spécifiant le chemin vers le fichier que le client veut recevoir. Cette fonction utilisera les fonctions déjà définies plus haut mais aussi les fonction suivantes dont le nom devrait vous permettre de deviner leur comportement:

- `sendResponse(code: int) -> None`
- `sendHeader(headerKeyword: str, headerValue: str) -> None`
- `sendFileContent(content: str, isBinary: bool = False) -> None`
- `doesFileExist(path: str) -> bool`