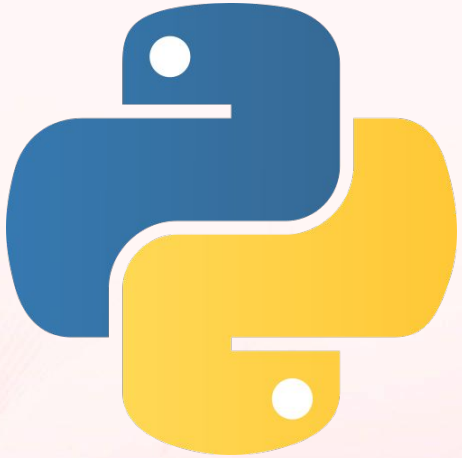
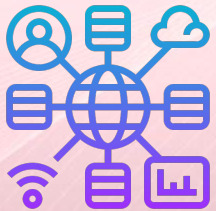


PROJET PYTHON DU JOUR…

TOULOUSE
ynov
CAMPUS

PROJET SCRIPTING FINAL

Automatisation et configuration d’une application



TOULOUSE
ynov
CAMPUS

Pitch

Vous êtes un jeune cadre

dans une start-up
qui vous envoie
en consultant d'un
nouveau client
qui travaille
dans la mode

celio
feels badz*

TOULOUSE
INOV
CAMPUS

Pitch

celio feels badz* vous fournissent un csv avec les produits de leur nouveau catalogue.

Leur idée est de créer un site vitrine où ils pourront montrer leur collection de vêtements.

TOULOUSE Ynov CAMPUS

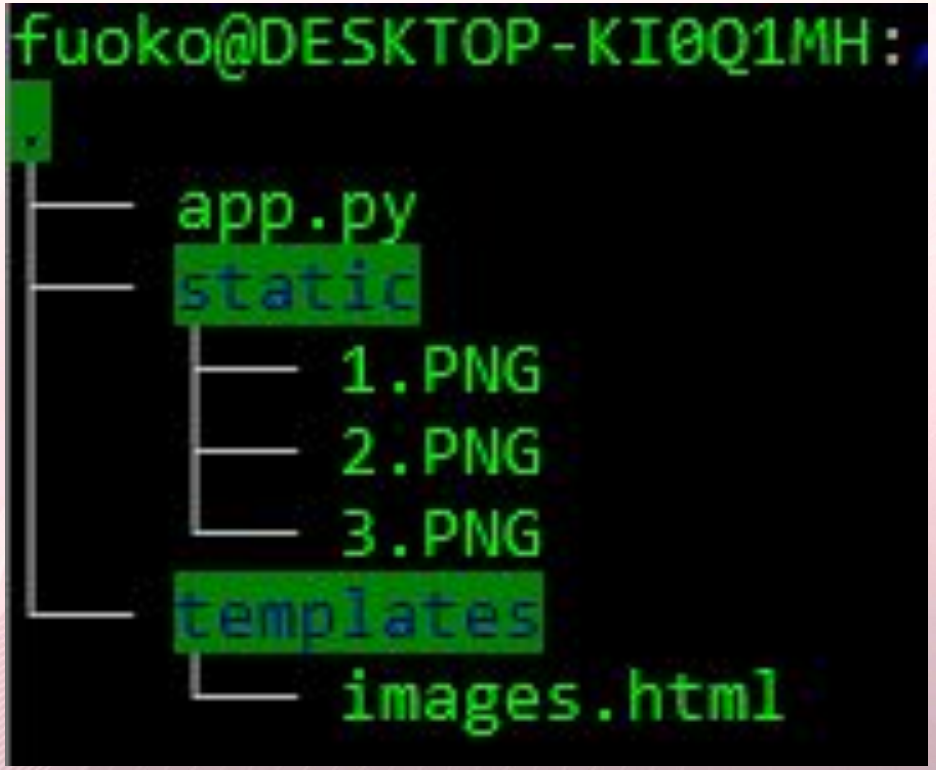
Etape 1

Ils vous donnent à disposition une machine qui tourne sous linux avec seulement python d’installé.

Créer un script `configure_and_run_serv.py` qui permet d’automatiser la création d’un serveur web et de le configurer (utiliser la lib os system)

TOULOUSE YNOV CAMPUS

Architecture de l’application



Etape 1

```
import os
import subprocess

# Install dependencies
subprocess.run(['sudo', 'apt-get', 'update'])
subprocess.run(['sudo', 'apt-get', 'install', '-y', 'python3-pip', 'python3-venv'])

# Create virtual environment and activate it
subprocess.run(['python3', '-m', 'venv', 'env'])
os.chdir('env')
subprocess.run(['source', 'bin/activate'])

# Install Flask and other dependencies
subprocess.run(['pip3', 'install', 'Flask'])

# Create Flask app
with open('app.py', 'w') as f:
    f.write('from flask import Flask\\\\n\\\\napp = Flask(__name__)\\\\n\\\\n@app.route
("/")\\\\ndef index():\\\\n    return "Hello, World!"\\\\n')

# Run Flask app
subprocess.run(['export', 'FLASK_APP=app.py'])
subprocess.run(['flask', 'run'])
```

TOULOUSE YNOV CAMPUS

Etape 2

Lire le fichier client csv présent sur le moodle grâce à la librairie pandas.

Ecrire un script dl_image.py qui permet de télécharger toutes les images à partir de ce csv sur la machine et les déplacer dans un dossier static

Etape 3

Créer une expression cron qui permet de lancer l’application web au lancement du serveur web (à son démarrage)

Ajouter ce cron dans le script d’automatisation avec la syntaxe suivante :

```
echo "*/5 * * * * /path/to/command" | crontab -
```

Etape 4

Créer un script qui permet de vérifier à intervalles réguliers (tester toutes les 10 secondes) si le serveur web tourne toujours.

(Utiliser le module requests et le module sleep)

Etape 4

Créer un fichier de logs pour piloter le déroulement de ce script.

- S’il ne tourne plus, relancer le script. Logger en Warning : “server down, trying to restart…

- S’il tourne encore , logger en info : “server is up !”

Etape 5

Pour tester l’architecture de votre serveur web, le client vous demande d’afficher une vingtaine de produits aléatoires de leur catalogue sur le site.

A chaque rechargement de la page, de nouvelles images vont être affichées.
Modifier les fichiers app.py et template.html

Etape 5 - aide

Utiliser la fonction os.listdir pour afficher la liste de tous les fichiers présents dans le dossier static

Importer la fonction random.shuffle pour mélanger cette liste, et extraire les 20 premiers éléments de cette dernière.

Etape 6
Permettre seulement certaines adresses

Le client ne souhaite pas mettre en ligne directement ce produit à tout internet. Créer un fichier qui contient une liste d'adresses IP autorisées.
Si l'IP d'un utilisateur qui tente de se connecter n'est pas la liste, rediriger l'utilisateur vers une page "FORBIDDEN"

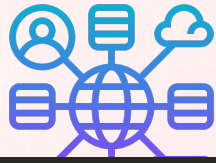
(utiliser request.remote_addr from flask.request)

```python
import os
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/images')
def show_images():
    image_dir = './static'
    images = [f for f in os.listdir(image_dir)]
    print(images)
    return render_template('images.html', images=images)

if __name__ == '__main__':
    app.run(debug=True)
```



```
{% block content %}
    <h1>Images</h1>
    <div class="row">
        {% for image in images %}
            <div class="col-md-3">
                <img src="{{ url_for('static', filename=image) }}">
            </div>
        {% endfor %}
    </div>
{% endblock %}
```