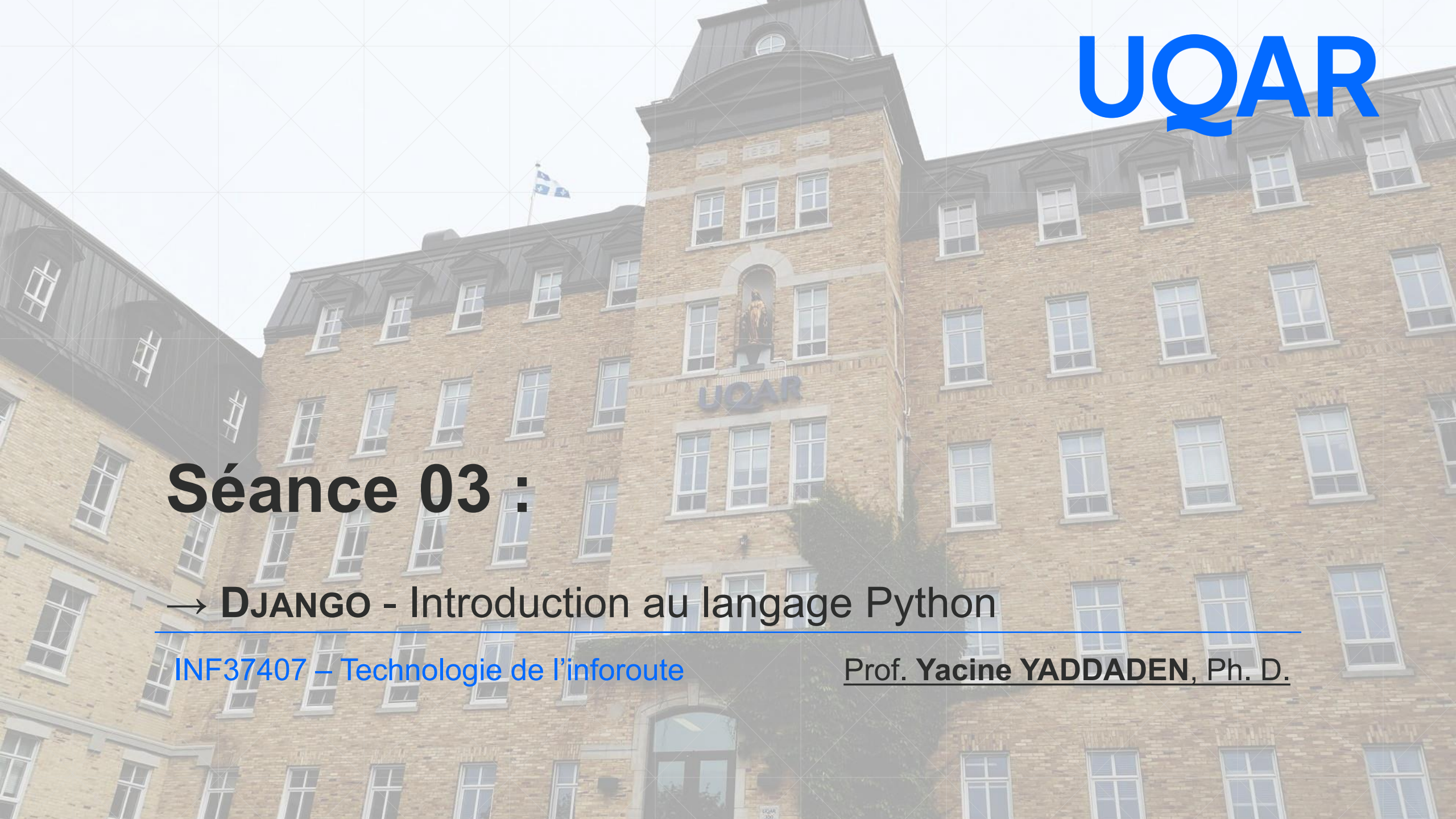


Séance 03 :

→ DJANGO - Introduction au langage Python

INF37407 – Technologie de l'inforoute

Prof. **Yacine YADDADEN**, Ph. D.

UQAR

Plan

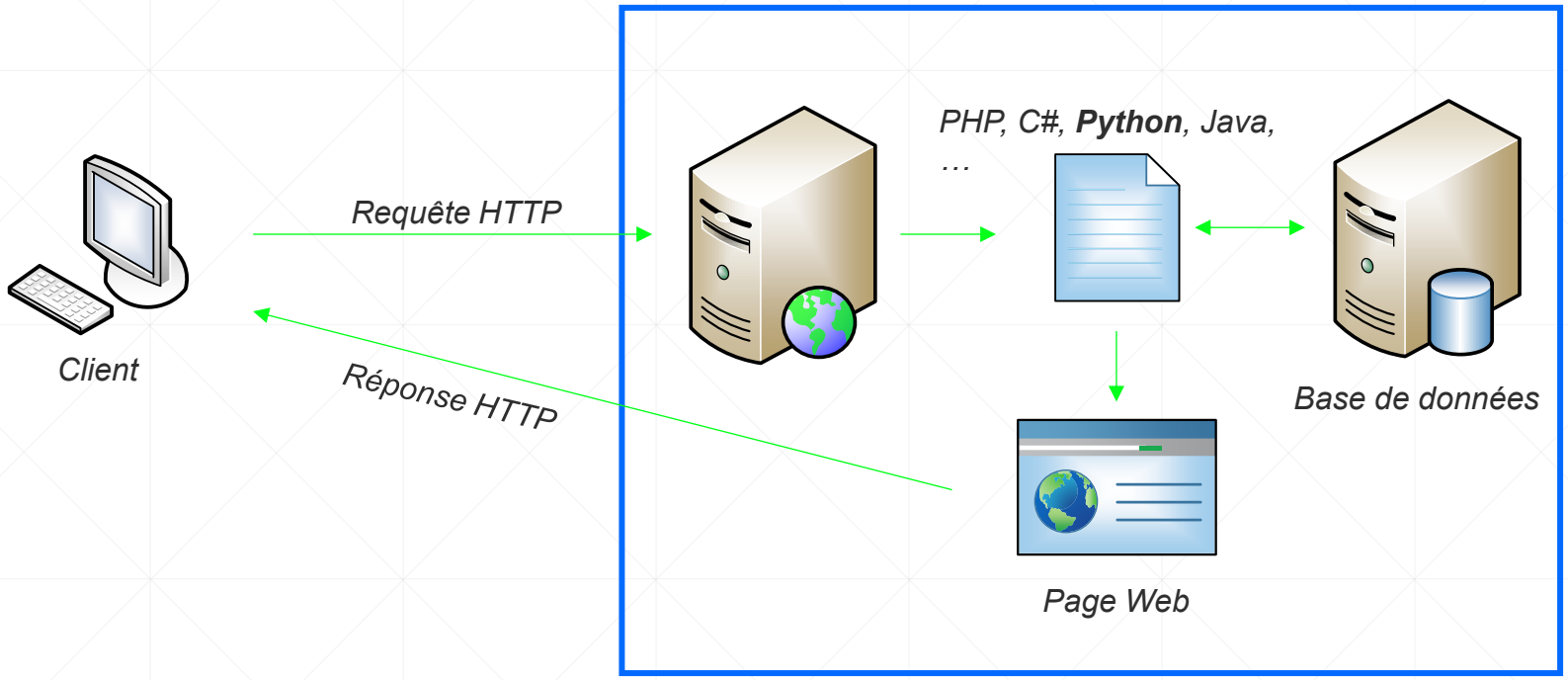
1. Introduction & Contexte
2. Le motif d’architecture logiciel MVC
3. Présentation de Django
4. Initiation au langage Python
5. Pratique → Préparation de l’environnement
6. Questions et discussion

UQAR

Introduction & contexte

- **Contexte :**
  - La création d'application Web complexe nécessite certains **services centralisés** : *authentification*,
  - Elle nécessite également la **persistance des données** utilisateurs.
- **Problématique :**
  - ☒ Le code de l'application côté ***front-end*** est insuffisant pour satisfaire les exigences précédentes,
  - ☒ Qu'est-ce qui permettrait d'implémenter les services centralisés et la persistance des données ?
- **Solution :**
  - ☑ Création d'un ***back-end*** adéquat,
  - ☑ Utiliser le *langage* et les *Framework* adéquats.

Rappel : *front-end & back-end*



UQAR

Design Pattern MVC

- **Définition :** « *C'est un motif d'architecture logiciel crée en 1978 par Trygve Reenskaug et destiné à être utiliser pour le développement d'application avec interface graphique.* »,
- Il est constitué de trois élément fondamentaux :
  - **Modèle :**

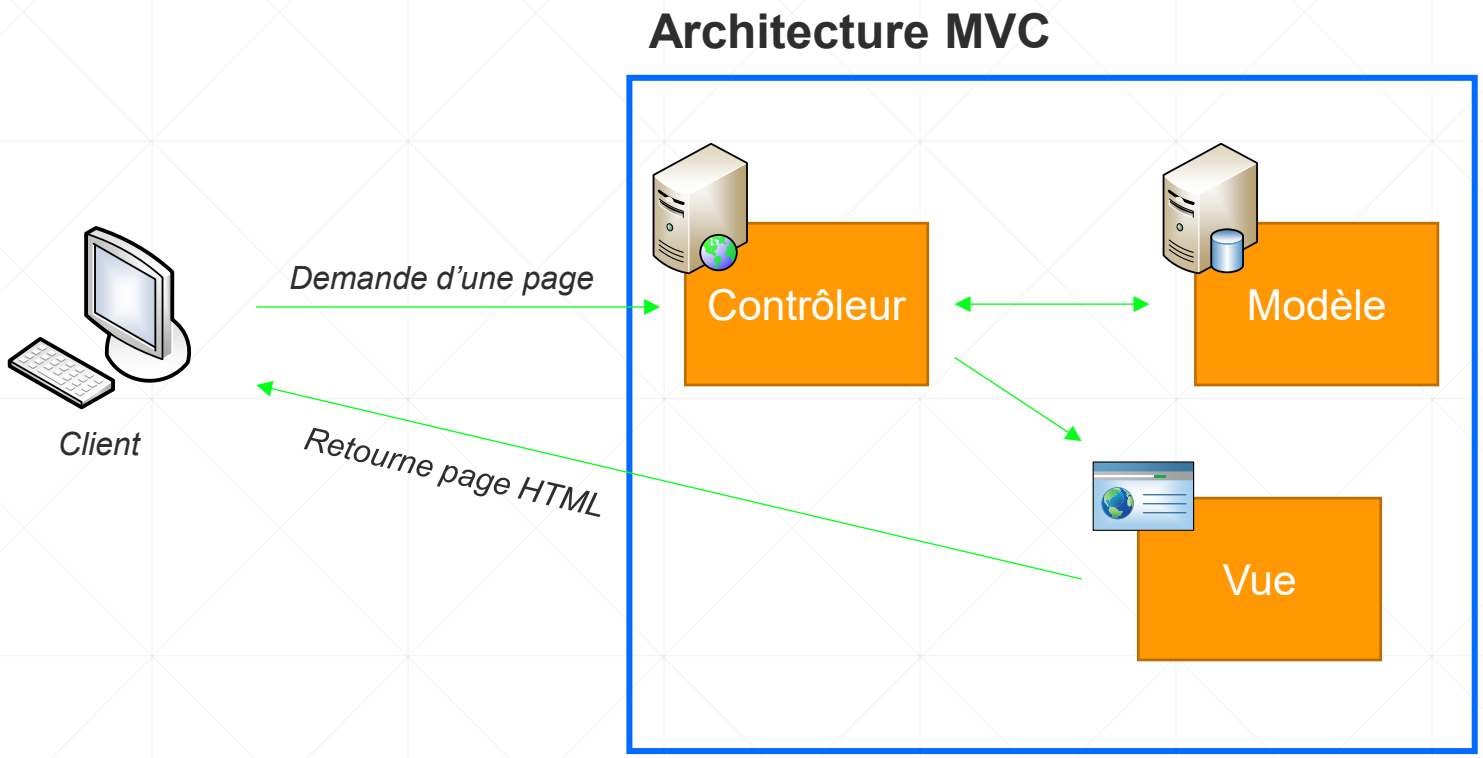
    → Contient la *représentation des données* ainsi que les *traitements* et la *logique* en rapport,
  - **Vue :**

    → Représente l'interface et le *rendu graphique*. Dans le cas du Web, c'est sous forme d'**HTML**,
  - **Contrôleur :**

    → Englobe le *traitement des actions utilisateur* en modifiant les données du modèle ou de la vue,

    → Joue également le rôle d'*intermédiaire* entre le modèle et la vue.

Motif d'architecture logicielle MVC



Langage & Framework pour *back-end*

- **PHP :** Laravel, Symphony, CakePHP, …
- **Ruby :** Ruby on Rails, Sinatra, …
- **Java :** Spring, Struts, …
- **Python : Django**, Pyramid, **Flask**, Bottle, …
- **C# :** ASP.net
- **JavaScript :** Express, …
- …

Flask

Micro Framework

django

Full-Stack Framework

Le Framework Django

- **Définition :** « *C'est un cadre de développement Web Open Source pour Python. Il a été créé par Adrian Holovaty et Simon Willison en 2005 pour le compte du Lawrence Journal-World.* »,
- Il inclut un ensemble d'outils prédéfinis rendant le développement Web aisé,
- Il se base sur le **MVC** ou **MVT** (**M**odel-**V**iew-**T**emplate),
- Il dispose de plusieurs versions : 1.x, 2.x, 3.x, 4.x et **5.x** (courante),
- Slogan : *Le Framework pour les perfectionnistes avec des deadlines*.
- Dépôt GitHub : https://github.com/django/django
- Plateformes : Instagram, Spotify, Pinterest, NASA, …

UQAR

Pourquoi Django ?

- Il inclut une panoplie d'outils et fonctionnalités :
  - ☑ Serveur Web et interface d'administration,
  - ☑ Un **ORM** (**O**bject **R**elational **M**apper),
  - ☑ Outil pour les tests unitaires, …
- Excellente documentation et vaste communauté,
- Très sécurisé,
- Rapide et facile d'utilisation,
- Adéquat pour n'importe quel type de projet Web, …

Initiation à Python

Les fondamentaux

UQAR

Plan

1. Présentation du langage Python
2. Exécution de code Python
3. Les variables et leurs types
4. Structures de données
5. Instructions de contrôle
6. Les fonctions
7. Programmation Orientée Objet
8. Utilisation des modules et bibliothèques

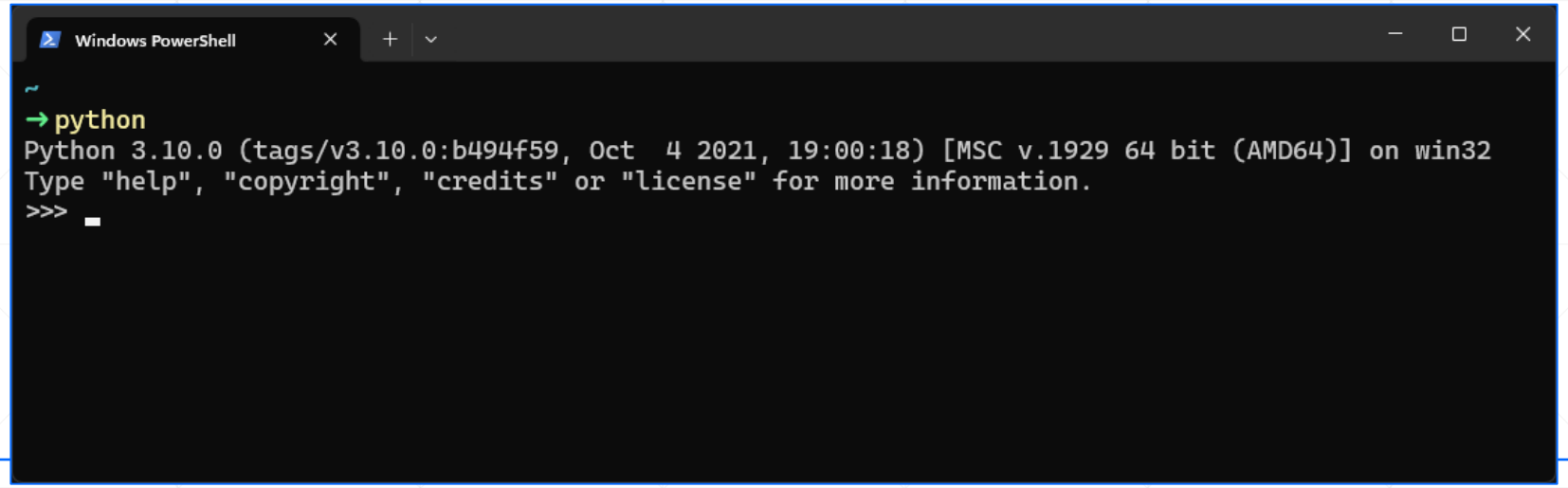
Présentation du langage Python

- **Définition :** « *C'est un langage de programmation interprété, impératif et orienté objet. Il a été conçu par Guido van Rossum en 1991.* »,
- Il est *multiplateforme* (Windows, MacOS, Linux, …),
- Il est utilisé dans beaucoup de cas :
  - ☑ Développement logiciel,
  - ☑ **Programmation Web**,
  - ☑ Machine Learning & Data Science,
  - ☑ Programmation Système, …
- Site Web : https://www.python.org/

UQAR

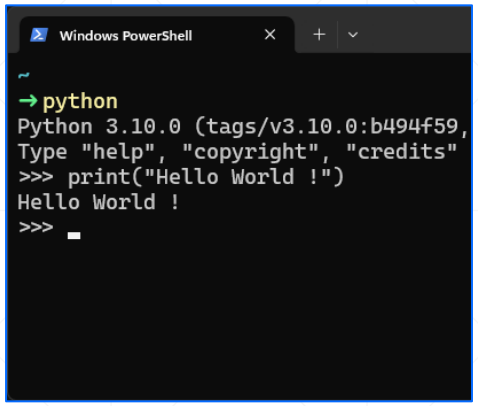
Installation de *Python*

1. Il faut se rendre sur le site : https://www.python.org/
2. Dans la section Downloads, télécharger la version : **3.x.x**
3. Choisir la version adéquate selon votre système et l'architecture (x86 ou x64),
4. Pour procéder à l'installation, il faut lancer l'exécutable et suivre les étapes.



Exécution de code Python

- Il est nécessaire d'installer au préalable l'*interpréteur Python*,
- Il est possible d'exécuter du code Python de deux façons :
  - Ligne par ligne :



  - À partir d'un fichier (exemple : `main.py`) à l'aide de la commande : **`python`** *`main.py`*

UQAR

Structure d’un fichier Python

```python
# -*- coding: utf-8 -*-

def main_function():
    print("This is the main function !")

def secondary_function():
    print("This is the secondary function !")

if __name__ == "__main__":
    main_function()
```

main.py

UQAR

Les variables et leurs types

- Une variable sert à *contenir une valeur* ou une donnée.
- L'opération permettant de *modifier la valeur* d'une variable est l'**affectation**,

```
variable = "value"
```

- Une variable est définie par un **type** : **`float`**, **`bool`**, **`int`**, **`string`**, …
- Contrairement au Java, C++, C#, … qui ont un typage *statique*, Python possède un typage *dynamique*,
- Dans ce cas (dynamique), le *type de la variable peut changer* à tout moment dépendamment de la valeur affectée.

```
# Ceci est un commentaire !
```

UQAR

Structures de données

- Les **listes** : structure de données où chaque élément est indexé à l'aide d'une valeur entière numéroté à partir de 0.

```
this_is_a_list = ["Québec", 'Montréal', 24, True]
```

Valeur à l'indexe 0

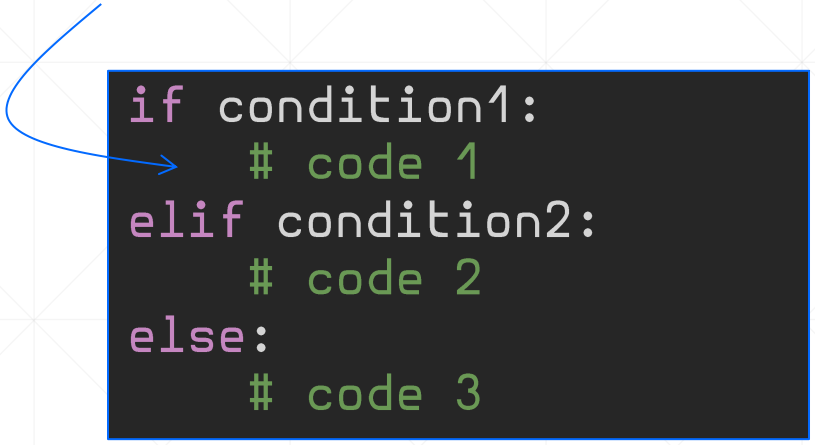
- Les **dictionnaires** : structure de données où chaque élément est indexé à l'aide d'une clé unique.

```
this_is_a_dictionnary = {"firstname": "Yacine", "lastname": "Yaddaden"}
```

Clé *Valeur*

Structures de contrôle – *Les conditions*

- Utilisation de système d'**indentation** au lieu des *accolades* **{}** :



```
if condition1:
    # code 1
elif condition2:
    # code 2
else:
    # code 3
```

- Il est également possible d'utiliser des opérateurs logiques : **and**, **or** et **not**

UQAR

Structures de contrôle – *Les boucles*

- La structure **`while`** :

```
while condition1:
    # code
```

- La structure **`for`** :

```
for i in range(0, 10):
    # code
```

UQAR

Les fonctions

- C'est un *bout de code* qui possède :
  - des arguments ou paramètres,
  - des retours ou sorties.
- Il **réutilisable** (*déclaration* et *appel*) :
  - Déclaration :

```
def addition_function(nbr_1, nbr_2):
    result = nbr_1 + nbr_2
    return result
```

  - Appel :

```
addition_function(5, 4)
```

Programmation Orientée Objet – Partie I

- Les **classes** :

```
class Main_class:
    # code
```

- Le **constructeur** :

```
def __init__(self):
    # code
```

- Les **attributs** :

```
self.__first_attribute = 5
self.second_attibute = 4
```

privé

publique

- Les **méthodes** :

```
def print_first_attribute(self):
    print(self.first_attribute)
```

UQAR

Programmation Orientée Objet – Partie II

Héritage

Polymorphisme

```
class Person:
    def __init__(self, name):
        self.name = name

    def info(self):
        print(self.name)

class Student(Person):
    def __init__(self, name, univ):
        super().__init__(name)
        self.univ = univ

    def info(self):
        print("{} in {}".format(self.name, self.univ))
```

UQAR

Utilisation des modules et bibliothèques

- Il faut commencer par importer la bibliothèque :

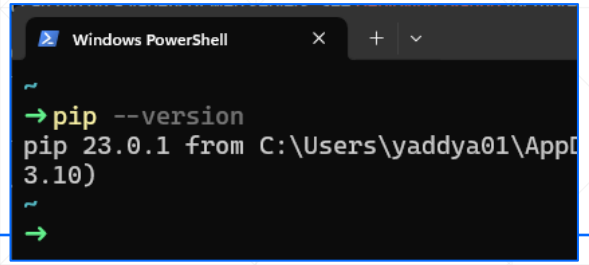
```
import random
```

- Ensuite faire appel à la fonction ciblée :

```
print(random.randint(1, 10))
```

Installation du *gestionnaire de paquets*

- **Objectif :**
  - Permettre une gestion optimisée des différents modules externe à installer.
- **Installation :**
  - **`pip`** est le gestionnaire de paquets Python : https://pip.pypa.io/en/stable/
  - Il est inclus lors de l'installation de Python **3 >=3.10**
  - Dans le cas où il est nécessaire de l'installer :
    - Télécharger **`get-pip.py`** à partir de : https://bootstrap.pypa.io/get-pip.py
    - Lancer la commande : **`python get-pip.py`**
- **Mise à jour :**
  - Lancer la commande → **`pip install --user -U pip`**



Guide d'utilisation de `pip`

- **Installation d'un nouveau paquet :**
  - ☑ Lancer la commande → **`pip install`** *`package_name`*
- **Désinstaller un paquet existant :**
  - ☑ Lancer la commande → **`pip uninstall`** *`package_name`*
- **Liste des paquets :**
  - ☑ Lancer la commande → **`pip list`**
- **Gérer les prérequis :**
  - ☑ Générer le fichier → **`pip freeze >`** *`requirements.txt`*
  - ☑ Installer à partir du fichier → **`pip install -r`** *`requirements.txt`*
- **Pour plus d'information :**
  - ☑ Se rendre sur le page de documentation → https://pip.pypa.io/en/stable/user_guide/

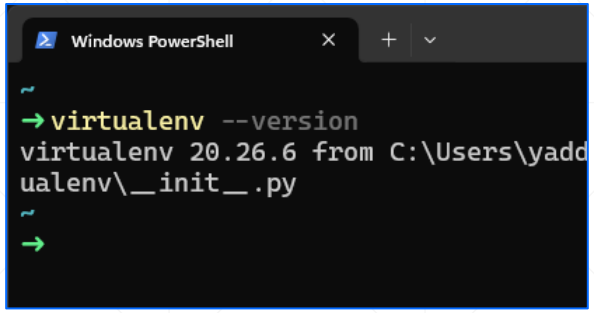
Installation de l'*environnement virtuel*

- **Objectif :**

  ☑ Permettre de créer un environnement Python **isolé**.

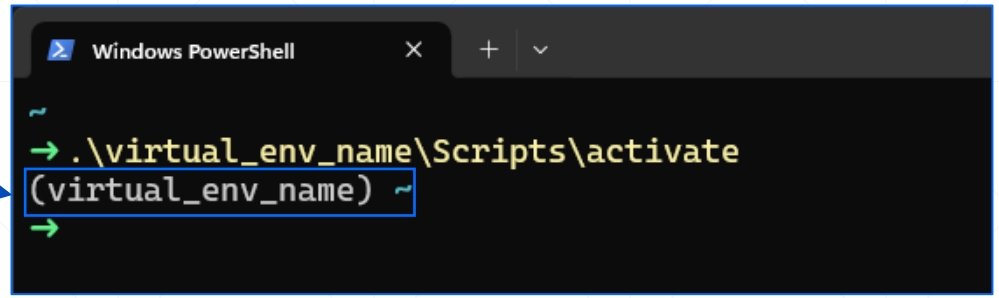
- **Installation :**

  1. Il y a **`virtualenv`** est le paquet Python → : https://virtualenv.pypa.io/en/stable/
  2. Alternatives → **venv** et **pyenv**,
  3. Il faut lancer la commande → `python -m pip install` **`virtualenv`**



Guide d'utilisation de `virtualenv`

- Créer un nouvel environnement : **`virtualenv`** *`virtual_env_name`*
- Un dossier avec ***`virtual_env_name`*** sera créé,
- Activer l'environnement virtuel → **`.\`***`virtual_env_name`***`\Scripts\activate`**



- Afin de désactiver l'environnement virtuel : **`deactivate`**

UQAR

Installation de Django

1. Il faut commencer par créer un environnement virtuel :
   ☑ Commande → **`virtualenv`** *`venv_django`*
2. Activer l'environnement virtuel :
   ☑ Commande → **`.\`***`venv_django`***`\Scripts\activate`**
3. Installer Django :
   ☑ Commande → `python -m pip install` **`django`**
4. Django est installé avec l'ensemble de ses dépendances.

Premier Travail pratique

Aperçu général

UQAR

Premier travail pratique

- **Contexte**

  → L'**OGSL** surveille l'état écologique du fleuve Saint-Laurent. Le TP consiste à **récupérer des données externes, les parser, les structurer et les rendre accessibles via une API REST ou GraphQL**.

- **Objectif**

  ☑ Développer un **parseur Python**.

  ☑ Créer une **base de données relationnelle**.

  ☑ Exposer les données via une **API sécurisée**.

  ☑ Interface conviviale simple.

- **Technologies**

  ☑ Python, Django, Requests, Django REST Framework, Graphene (GraphQL), drf-yasg (Swagger).

  ☑ MySQL Server.

  ☑ Authentification par token.

Premier travail pratique

- **Étapes**
  1. Analyser les sources de données (CSV, JSON, API).
  2. Développer le parseur et nettoyer les données.
  3. Créer la base MySQL et les modèles Django.
  4. Développer l’API REST/GraphQL.
  5. Sécuriser l’accès aux données (token).
  6. Tester et documenter.

Questions & Discussion

UQAR

Bibliographie | EN

1. Duckett, J. (2011). *HTML & CSS: design and build websites* (Vol. 15). Indianapolis, IN: Wiley.
2. Duckett, J. (2014). *JavaScript and JQuery: interactive front-end web development*. Wiley & Son.
3. Banks, A., & Porcello, E. (2017). *Learning React: functional web development with React and Redux*. " O'Reilly Media, Inc.".
4. Haverbeke, M. (2014). *Eloquent javascript: A modern introduction to programming*. No Starch Press.

UQAR

Bibliographie | FR

1. Templier, Thierry & Gougeon, Arnaud (2007). JavaScript pour le Web 2.0 Programmation objet, DOM, Ajax, Prototype, Dojo, Script.aculo.us, Rialto. Éditions Eyrolles.
2. Porteneuve, Christophe (2008). Bien développer pour le Web 2.0 : Bonnes pratiques Ajax. Éditions Eyrolles.
3. Engels, Jean (2012). HTML5 et CSS3 : Cours et exercices corrigés. Éditions Eyrolles.
4. Martin, Michel (2014). HTML5, CSS3 & jQuery : Créez votre premier site web. Éditions Pearson.