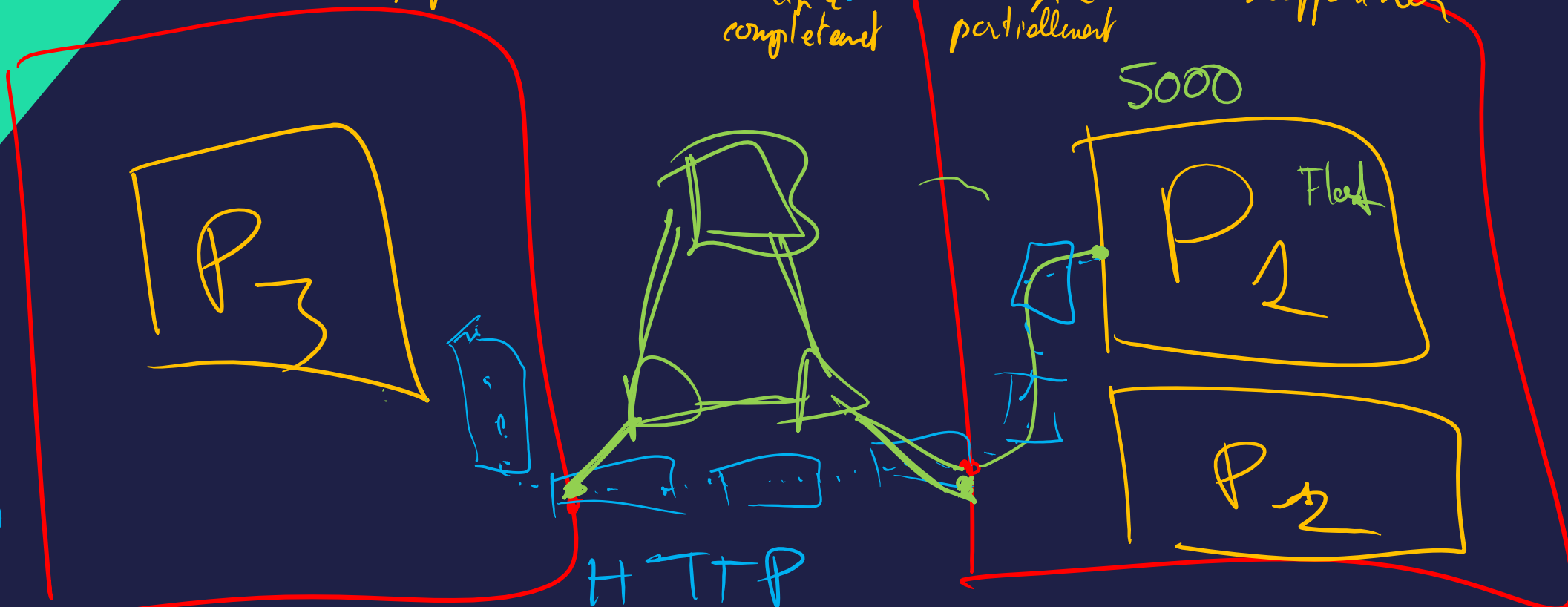


Les clients

Python dans le cadre des
développements des clients

GET, POST, PUT, PATCH, DELETE
 récupérer créer modifier compléter modifier partiellement supprimer

method
 header
 body



M2

TCP/IP

M1

127.0.0.1
 localhost



Compétence demandée :
Intégrer Python dans une interface pour
un client léger et un client lourd

1. **Flask**
2. **TKinter**

1. **Flask**
2. **TKinter**

Flask

Flask est un framework web, c'est un module Python qui vous permet de développer facilement des applications web. Il a un noyau petit et facile à étendre : c'est un microframework qui n'inclut pas d'ORM (Object Relational Manager) ou de telles fonctionnalités.

Il a de nombreuses fonctionnalités intéressantes comme le routage d'URL, le moteur de modèle.

Le routing

Une route est un chemin permet d'exposer une fonction Python afin qu'elle soit accessible grâce au protocole HTTP.

Toutes méthodes HTTP peuvent être utilisées pour les routes, par exemple GET, POST, PUT, PATCH et DELETE.

Un exemple avec GET

```
@app.route('/list')  
def list():  
    result = controller.search('')  
    return jsonify(result)
```

```
@app.route('/list', methods=['GET'])  
def list():  
    result = controller.search('')  
    return jsonify(result)
```

h https:// www .google.com /search ?q=tob
protocol sous domaine chemin query string

```
@app.route('/query', methods=['GET'])  
def query():  
    q = request.args.get('q')  
    result = controller.search(q)  
    return jsonify(result)
```

'go'
↖

http://localhost:5000/query?q=go

Flask utilise le moteur de template Jinja afin de générer dynamiquement une réponse textuelle.

Dans le domaine du WEB, cette réponse textuelle est majoritairement du code HTML.

```
@app.route('/', methods=['GET'])  
def index():  
    return render_template('index.html.jinja')
```


Jinja2 est un langage de template moderne pour les développeurs Python. Il a été créé d'après le modèle de Django.

Il est utilisé pour créer des formats HTML, XML ou d'autres formats de balisage qui sont renvoyés à l'utilisateur via une requête HTTP.

```
@app.route('/search', methods=['POST'])  
def search():  
    data = request.json  
    result = controller.search(data['searchText'])  
    return jsonify(result)
```

Les hooks

```
@app.after_request
def after_request(response):
    response.headers.add("Access-Control-Allow-Origin", "*")
    response.headers.add("Access-Control-Allow-Headers", "*")
    response.headers.add("Access-Control-Allow-Methods", "*")
    return response
```

Exécution d'une application Flask

```
set FLASK_APP=app  
set FLASK_ENV=development  
set FLASK_DEBUG=true  
flask run
```

Retrouvez toutes les variables d'environnement ici :
<https://flask.palletsprojects.com>

Par exemple :

```
FLASK_RUN_HOST=localhost  
FLASK_RUN_PORT=8000
```

1. **Flask**
2. **TKinter**

Flask

Le package tkinter ("interface Tk") est l'interface Python standard de la boîte à outils Tcl/Tk GUI.

Tk et tkinter sont disponibles sur la plupart des plates-formes Unix, y compris macOS, ainsi que sur les systèmes Windows.

Retrouvez la documentation officielle ici :

<https://docs.python.org/fr/3/library/tkinter.html>

Un bonne vision d'ensemble est disponible ici :

<https://python.doctor/page-tkinter-interface-graphique-python-tutoriel>

Tkinter peut être considéré comme une librairie (blocs de code) pour la création de Widget pour client lourd cross-plateforme.

Une fenêtre s'affiche dans un thread séparé avec
une boucle d'événement afin de détecter les
actions de l'utilisateur

```
window = Tk()
```

```
# Ajout de widget
```

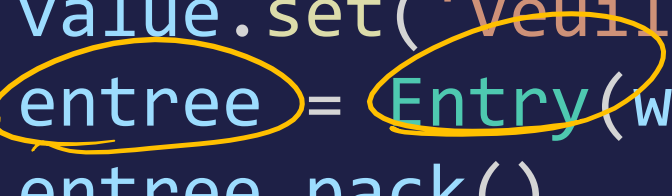
```
window.mainloop()
```

Afficher du texte

```
# Adding label  
label = Label(window, text='Du texte trop bien !')  
label.pack()
```


Afficher une boîte de saisie

```
# Adding input
value = StringVar()
value.set('Veuillez écrire ici ...')
entree = Entry(window, textvariable=value)
entree.pack()
```

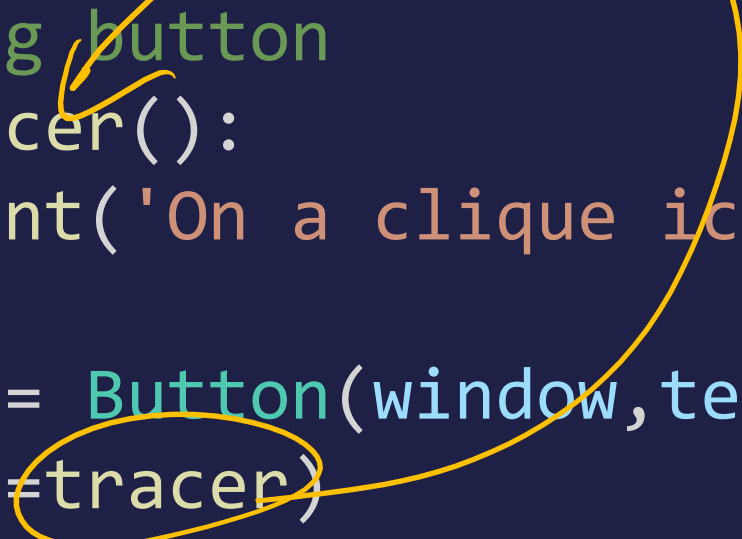


```
value: str = entree.get()
```

Afficher un bouton

```
# Adding button
def tracer():
    print('On a clique ici !')

button = Button(window, text="Vazy click voir",
                 command=tracer)
button.pack()
```



Afficher une liste

```
# Adding list
result_list = Listbox(window)
result_list.insert(1, 'Bien')
result_list.insert(2, 'Encore mieux !')
result_list.insert(3, 'Ce cours est ouf !')
result_list.pack()
```

```
result_list.delete(0, END)
```

↑
pour vider

**Votre mission si vous
l'acceptez**

Développez une interface simple client lourd pour
rechercher dans votre annuaire 😊

En utilisant les widgets suivant : Label, Entry,
Button et Listbox

