Lezione 19/11/2025

Contenuti

Introduzione a ‘PyWebView’ {

[Creare interfacce grafiche per app Python]

}

Realizzato da: Gabriele Cocchetti, Michele Quaini e Stefano Losio

Punti fondamentali:

01 → Cos’è PyWebView
02 → OOP in Python
03 → Creare una finestra
04 → Fondamenti di JS
05 → Compilazione di un eseguibile
06 → Tips

Conclusione e link alla repository

01 {

[Cos’è PyWebView]

<Una libreria che integra una finestra web nativa nel tuo programma Python>

}

```
PyWebView; {

    ‘PyWebView è un wrapper leggero e multipiattaforma che
    semplifica l’uso del componente WebView del sistema
    operativo, permettendo di mostrare pagine HTML in una
    finestra nativa dell’app. “Nativa” significa che la
    finestra è integrata nel sistema, come qualsiasi altra
    applicazione desktop. PyWebView include il supporto al DOM,
    che permette a Python di leggere e modificare elementi
    della pagina, e un piccolo server interno per gestire
    meglio la comunicazione tra frontend e backend.’

}
```

02 {

[OOP in Python]

<Sintassi OOP in Python>

}

```
1  class Persona:
2
3
4
5
```

Creazione di una classe

<per creare una classe in Python si utilizza il termine riservato class>

Creazione di una classe

Creazione di un metodo

```python
class Persona:

    def __init__(self,nome):
        self.nome = nome

    def metodo(self):
        return self.nome
```

Creazione di un metodo

<In Python, un metodo di una classe si definisce con def e richiede come primo parametro self, che rappresenta l'oggetto stesso; se il metodo deve restituire un valore, si utilizza return.
Il costruttore della classe si crea con __init__ e gli attributi dell'oggetto si definiscono con self.nome_attributo = valore.>

03 {

[Creare una finestra]

<Come istanziare una
finestra PyWebView>

}

```
1  import webview
2  window=webview.create_window(‘finestra’, url=’index.html’,js_api=api)
3  webview.start(Debug=False)
4  window.destroy()
5
```

Utilizzare PyWebView

<Per utilizzare PyWebView in Python, è necessario importare la classe webview. Per creare una finestra si utilizza il metodo create_window(). Per avviare l’interfaccia grafica si chiama start(), mentre per chiudere forzatamente la finestra si può usare destroy().>

Parametri create_window()

- Primo parametro: nome della finestra che verrà creata.
- URL: percorso dell'HTML da mostrare nella finestra.
- width: larghezza iniziale della finestra.
- height: altezza iniziale della finestra.
- resizable: valore booleano che determina se la finestra può essere ridimensionata.

Parametro start()

- Debug: valore booleano che abilita la funzione "ispeziona" della finestra, permettendo di aprire la console, visualizzare il codice HTML/JS/CSS, controllare errori e ispezionare gli elementi della pagina, simile agli strumenti di sviluppo di un browser.

04 {

[Fondamenti di JS]

<elenco dei principali comandi JS e delle funzioni per manipolare il DOM>

}

```
1 const Pi = 3.14
2 let flag = false
3
4
5
```

Dichiarazione di variabili e constanti

<In JavaScript, let serve a dichiarare variabili il cui valore può cambiare, mentre const serve a dichiarare costanti il cui valore rimane fisso dopo l’inizializzazione.>

```
1 nomeFunzione (parametri);
2
3 function nomeFunzione (parametri) {
4     operazioni
5     return //opzionale
6 }
```

Definizione di funzioni JS

<Una funzione dichiarata in JavaScript si crea con function, può essere usata anche prima di essere scritta e opzionalmente può restituire un valore con return.>

Manipolazione del DOM

<La manipolazione del DOM in JavaScript si fa usando metodi e funzioni del linguaggio, che permettono di selezionare elementi, modificarli, aggiungerne di nuovi, cambiare stili e gestire eventi.>

{In JavaScript, un element rappresenta un singolo nodo HTML su cui possiamo agire, document rappresenta l'intera pagina HTML e permette di selezionare o creare elementi, mentre window rappresenta la finestra del browser e fornisce metodi e proprietà globali come alert o l'URL corrente.}

Manipolazione del DOM

- `document.getElementById("id")` → permette di selezionare un elemento HTML con l'id specificato.
- `document.getElementsByClassName("classe")` → permette di selezionare un insieme di elementi HTML appartenenti alla classe specificata.
- `element.innerHTML` → permette di leggere o definire il contenuto HTML dell'elemento specificato.
- `element.textContent` → permette di leggere o definire il testo contenuto nell'elemento specificato.
- `document.createElement("tag")` → crea un nuovo elemento HTML della tipologia specificata da "tag".
- `element.addEventListener("evento", funzione)` → aggiunge un listener per un evento; quando l'evento si verifica, esegue la funzione specificata.
- `console.log()` → stampa messaggi nella console del browser, utile per controllare valori e fare debug.
- `confirm()` → mostra una finestra di dialogo con OK e Annulla; restituisce true se l'utente clicca OK, false se clicca Annulla.
- `element.remove()` → metodo del DOM che rimuove direttamente un elemento dalla pagina.

Gestione funzioni asincrone

<In JavaScript, una Promise è un oggetto che rappresenta il risultato futuro di un'operazione asincrona.
Le parole chiave async e await servono a gestire le Promise in modo più semplice e leggibile:

- async si mette davanti a una funzione per renderla asincrona, facendola restituire automaticamente una Promise.
- await si usa dentro funzioni async per fermare temporaneamente l'esecuzione finché una Promise non viene risolta, ottenendo così il valore restituito senza usare .then().>

```
async function eseguiOperazione() {
    let risultato = await window.pywebview.api.getLista();  // attende che la Promise sia risolta
    console.log(risultato);                                 // esegue la stampa
}
```

Richiamare metodi Python da JS

<In PyWebView, js_api è un oggetto Python contenente i metodi che vogliamo rendere accessibili da JavaScript, e da JS li richiamamo tramite window.pywebview.api.nomeMetodo(parametri).>

```
--Python--
class API:
    def saluta(self, nome):
        return f"Ciao, {nome}!"

    def somma(self, a, b):
        return a + b

    def moltiplica(self, a, b):
        return a * b

api_instance = API()
webview.create_window("Finestra", "index.html", js_api=api_instance)

--JS--
window.pywebview.api.saluta("Anna").then(r => console.log(r));   // Ciao, Anna!
window.pywebview.api.somma(5, 3).then(r => console.log(r));      // 8
window.pywebview.api.moltiplica(4, 7).then(r => console.log(r)); // 28
```

05 {

[Compilazione di un EXE]

<Come trasformare il
programma in un eseguibile>

}

La libreria PyInstaller

<La libreria pyinstaller consente, una volta che le vengono indicati i file necessari, di creare dal progetto un unico eseguibile in formato EXE.
Questà è utile perchè permette di distribuire programmi Python senza richiedere l'installazione di Python stesso sul PC dell'utente.>

Opzioni principali:

- --onefile: crea un singolo file eseguibile.
- --windowed: nasconde il terminale per applicazioni GUI.
- --add-data "file;dest": include file aggiuntivi (HTML, immagini, ecc.).

Per utilizzarlo eseguo questo comando nel CMD, aperto nella cartella del progetto:

```
1  pip install pyinstaller
```

La compilazione di un eseguibile

Nel caso del nostro progetto, il comando completo da eseguire nella shell sarà:

```
{

`pyinstaller --onefile --windowed
 --add-data "index.html;."
 --add-data "form.html;."
 --add-data "style.css;.”
 --add-data "addNota.js;."
 --add-data "bottoni.js;."
 --add-data "graficaBacheca.js;."
 --add-data "agenda.csv;."
 main.py

}
```

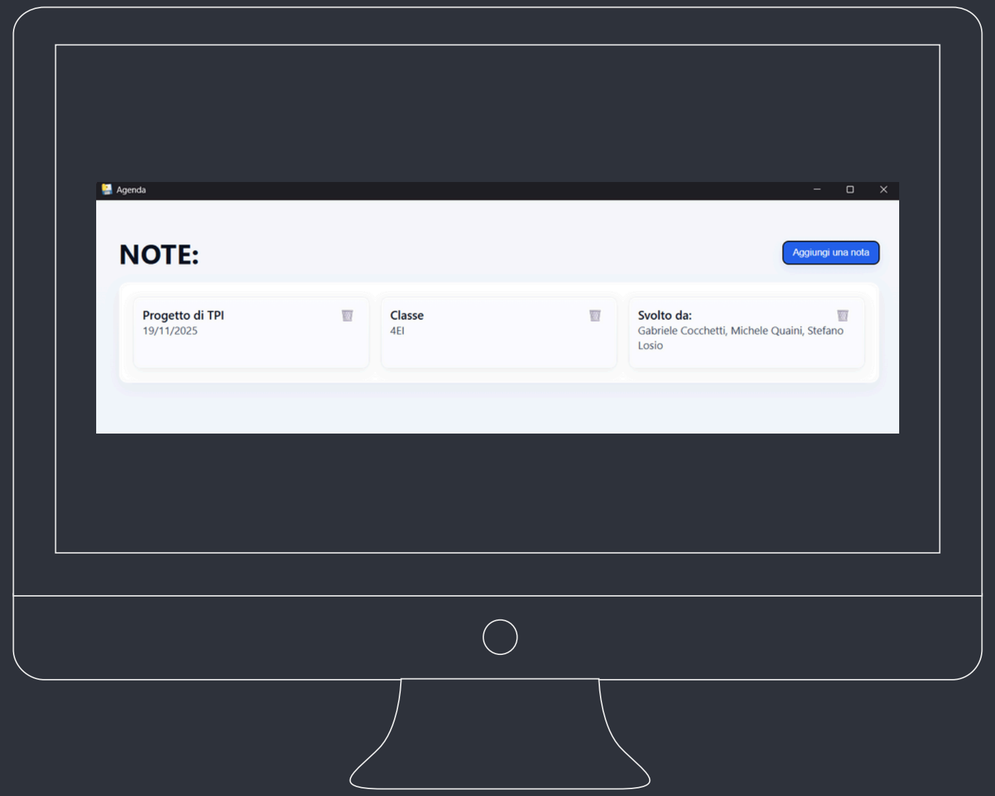
Conclusione

Dopo la compilazione, abbiamo finalmente il nostro file EXE utilizzabile da chiunque.

Grazie a PyWebView e PyInstaller possiamo trasformare script Python in applicazioni complete e portabili.

PyWebView gestisce l’interfaccia grafica, PyInstaller si occupa del packaging.

Distribuire software Python non richiede più che l’utente abbia Python installato.



06 {

[Tips]

<Consigli pratici che ci hanno risolto problemi o che pensiamo siano utili a livello di gestione dell’app>

}

Consigli:

- Gestire la creazione delle finestre tramite API.
- Creare riferimenti alle finestre all’interno delle API.
- In alcuni casi può essere utile popolare l’html tramite JavaScript.
- Collegare gli script JavaScript in fondo al body, oppure inserirli nell’head usando la parola riservata defer.

Grazie!; {

Hai bisogno di consultare ancora il codice?

La repository è disponibile
pubblicamente su GITHUB al link:

https://github.com/StefanoLosio/lezione-python-compilazione

CREDITI: Il template della presentazione è stato preso da **Slidesgo**, incluse le icone di **Flaticon**, e le infografiche di **Freepik**

}