

Titolo del progetto

Aggiornamento automatico della lista degli indirizzi IP autorizzati

Descrizione del progetto

In questo progetto, ho creato un piccolo programma in Python per aiutare a gestire gli accessi a dati riservati in un'azienda sanitaria.

Esiste una lista di indirizzi IP autorizzati ad accedere a contenuti sensibili.

Esiste una lista di IP da rimuovere, corrispondente ai dipendenti che non devono più avere accesso.

L'algoritmo confronta le due liste, rimuove gli IP non autorizzati e stampa la lista aggiornata.

Lo scopo principale è automatizzare questo controllo, rendendolo veloce, sicuro e privo di errori manuali.

Spiegazione del codice

Immagina di avere due liste: una contiene tutti gli indirizzi IP dei dipendenti che possono accedere a dati riservati, e l'altra contiene gli IP dei dipendenti che non devono più avere accesso. Il codice serve a **aggiornare automaticamente** la lista degli IP autorizzati, rimuovendo quelli indicati nella lista di rimozione.

1. Definizione delle liste

```
ip_autorizzati = [  
    "192.168.1.10",  
    "192.168.1.15",  
    "192.168.1.20",  
    "192.168.1.25"  
]  
  
ip_da_rimuovere = [  
    "192.168.1.15",  
    "192.168.1.25"  
]
```

Qui abbiamo due liste Python.

ip_autorizzati contiene tutti gli IP attualmente autorizzati.

ip_da_rimuovere contiene solo gli IP che vogliamo eliminare.

2. Creazione della funzione

```
def aggiorna_ip_autorizzati(ip_aut, ip_rimozione):
```

```
ip_aggiornati = [ip for ip in ip_aut if ip not in ip_rimozione]
```

```
return ip_aggiornati
```

Abbiamo definito una funzione chiamata `aggiorna_ip_authorized`.

La funzione prende due liste come input: `ip_aut` e `ip_rimozione`.

All'interno, la **list comprehension** `[ip for ip in ip_aut if ip not in ip_rimozione]` crea una **nuova lista** contenente solo gli IP che **non sono presenti** nella lista di rimozione.

In altre parole, filtra tutti gli IP da eliminare.

Infine, la funzione restituisce (return) la lista aggiornata.

3. Aggiornamento della lista

```
ip_authorized = aggiorna_ip_authorized(ip_authorized, ip_da_rimuovere)
```

Qui chiamiamo la funzione e salviamo il risultato nella variabile originale `ip_authorized`.

Dopo questa operazione, la lista contiene solo gli IP ancora autorizzati.

4. Stampa della lista aggiornata

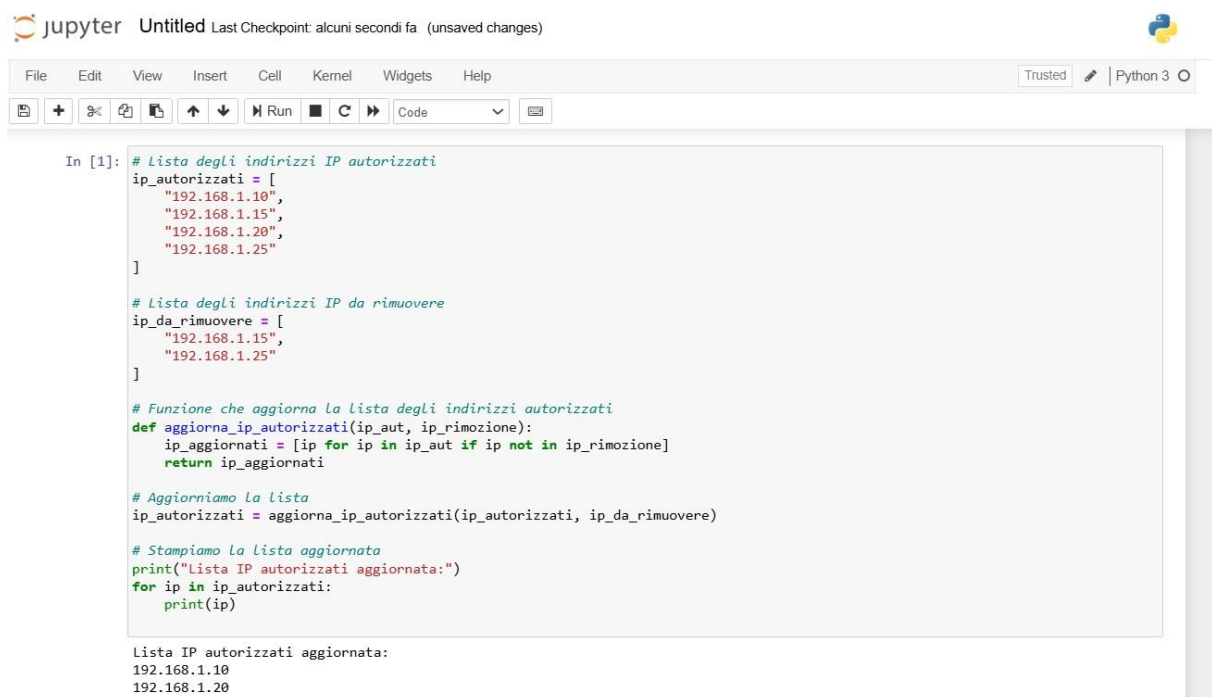
```
print("Lista IP autorizzati aggiornata:")
```

```
for ip in ip_authorized:
```

```
    print(ip)
```

Stampiamo un messaggio per indicare che la lista è stata aggiornata.

Poi utilizziamo un ciclo `for` per scorrere tutti gli IP nella lista aggiornata e li stampiamo uno alla volta.



The screenshot shows a Jupyter Notebook window titled "Untitled" with a status bar indicating "Last Checkpoint: alcuni secondi fa (unsaved changes)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The code cell contains the following Python code:

```
In [1]: # Lista degli indirizzi IP autorizzati
ip_authorized = [
    "192.168.1.10",
    "192.168.1.15",
    "192.168.1.20",
    "192.168.1.25"
]

# Lista degli indirizzi IP da rimuovere
ip_da_rimuovere = [
    "192.168.1.15",
    "192.168.1.25"
]

# Funzione che aggiorna la lista degli indirizzi autorizzati
def aggiorna_ip_authorized(ip_aut, ip_rimozione):
    ip_aggiornati = [ip for ip in ip_aut if ip not in ip_rimozione]
    return ip_aggiornati

# Aggiorniamo la lista
ip_authorized = aggiorna_ip_authorized(ip_authorized, ip_da_rimuovere)

# Stampiamo la lista aggiornata
print("Lista IP autorizzati aggiornata:")
for ip in ip_authorized:
    print(ip)
```

The output of the code cell is:

```
Lista IP autorizzati aggiornata:
192.168.1.10
192.168.1.20
```