

MANUALE SOFTWARE

Sommario

- 1) Introduzione e un pò di storia CGI
- 2) Dove sono stati trovati i codice e come sono stati sviluppati
- 3) Pagina index HTML
- 4) Script CPU.py
- 5) Script Tem.py
- 6) Script temp.py
- 7) Librerie usate datetime ,random,time,matplotlib.pyplot as plt,base64,psutil
- 8) Run script python web server

La storia del CGI (Common Gateway Interface) in Python è un'affascinante testimonianza dell'evoluzione del web e della programmazione.

Origini del CGI

1. **Anni '90:** Con l'emergere del World Wide Web, si sentiva la necessità di creare contenuti dinamici. Il CGI è diventato uno standard che ha permesso ai server web di eseguire script e interagire con gli utenti. Ha rappresentato una prima soluzione per la generazione di pagine web personalizzate.
2. **Python e CGI:** Python, con la sua sintassi chiara e concisa, si è affermato come linguaggio ideale per la scrittura di script CGI. Le versioni iniziali di Python includevano supporto nativo per il CGI, rendendo facile la creazione di applicazioni web dinamiche.

Evoluzione

3. **Sviluppo di Framework:** Con il tempo, l'uso del CGI ha cominciato a diminuire. Tecnologie più moderne come CGI-Fast, mod_php e framework come Flask e Django hanno preso piede, offrendo metodi più efficienti e sicuri per lo sviluppo web. Questi strumenti hanno introdotto paradigmi più avanzati, come il modello MVC, che semplificano la gestione del codice.
4. **Python Web Frameworks:** Oggi, l'uso di CGI in Python è meno diffuso, ma rimane una scelta valida per progetti semplici o didattici. La comunità Python ha creato numerosi framework che semplificano notevolmente la creazione di applicazioni web, rendendo il processo più intuitivo e strutturato.

Conclusione

Il CGI ha avuto un ruolo cruciale nello sviluppo del web dinamico, e Python ha giocato un ruolo importante nell'adozione di questa tecnologia. Anche se oggi si utilizzano strumenti più avanzati, il concetto di CGI continua a rappresentare una parte fondamentale della storia della programmazione web, testimoniando la continua evoluzione delle tecnologie e delle pratiche di sviluppo.

CPU (Central Processing Unit)

La CPU è il "cervello" del computer, responsabile dell'esecuzione delle istruzioni e dell'elaborazione dei dati. Lavora in collaborazione con la RAM per gestire le operazioni di sistema.

RAM (Random Access Memory)

La RAM è una memoria volatile utilizzata per memorizzare temporaneamente i dati e le istruzioni necessarie per l'esecuzione dei programmi. Più RAM significa una migliore capacità di multitasking e performance.

ROM (Read-Only Memory)

La ROM è una memoria non volatile che contiene istruzioni permanenti, come il firmware del computer. A differenza della RAM, i dati nella ROM non vengono persi quando il computer viene spento.

Disco Rigido (HDD/SSD)

Il disco rigido è il dispositivo di archiviazione principale di un computer. Gli HDD usano dischi magnetici, mentre gli SSD utilizzano memoria flash, offrendo prestazioni più elevate e tempi di accesso più rapidi.

BIOS (Basic Input/Output System)

Il BIOS è un firmware di basso livello che inizializza l'hardware del computer durante l'avvio e gestisce il processo di boot. Fornisce anche un'interfaccia per configurare le impostazioni di sistema.

UEFI (Unified Extensible Firmware Interface)

L'UEFI è un'interfaccia più moderna rispetto al BIOS, che offre funzionalità avanzate, come il supporto per dischi di dimensioni maggiori e avvisi più rapidi. Può funzionare in modalità legacy per compatibilità con sistemi più vecchi.

Sistemi Operativi

Windows

Windows è un sistema operativo sviluppato da Microsoft, noto per la sua interfaccia utente grafica e la compatibilità con un'ampia gamma di software e hardware. È molto diffuso in ambito personale e aziendale.

Linux

Linux è un sistema operativo open source, noto per la sua stabilità e sicurezza. Esistono molte distribuzioni (come Ubuntu, Fedora, e CentOS) adatte a diversi utilizzi, da desktop a server.

Sommario

Codici

Relazione sul Supporto Didattico nel Corso di Python

Durante il percorso di formazione in Python, il supporto del docente è stato fondamentale, specialmente durante le esercitazioni relative al compendio di Python base. Il docente ha fornito una

guida costante e ha svolto un ruolo cruciale nell'assicurare che tutti gli studenti avessero le competenze necessarie per affrontare gli argomenti trattati.

Supporto del Docente

Il docente ha utilizzato una metodologia pratica, presentando codici Python sviluppati da lui stesso. Questi esempi sono stati un prezioso strumento per la comprensione dei concetti chiave. Attraverso l'analisi di questi script, gli studenti hanno potuto osservare le migliori pratiche nella scrittura del codice e apprendere come risolvere problemi comuni.

Esercitazioni Pratiche

Le esercitazioni pratiche erano organizzate in modo da permettere a ciascuno di mettere in pratica quanto appreso. Gli studenti avevano l'opportunità di modificare e migliorare i codici forniti, esplorando diverse soluzioni ai problemi. Questo approccio ha favorito un apprendimento attivo e ha incoraggiato la sperimentazione.

Feedback e Interazione

Il docente ha anche incoraggiato la discussione e il confronto tra gli studenti, creando un ambiente collaborativo. Il feedback immediato durante le esercitazioni ha permesso di chiarire dubbi e correggere errori in tempo reale, facilitando così l'acquisizione delle competenze.

Conclusione

Il supporto del docente si è rivelato essenziale per il successo del corso. Grazie ai codici forniti e all'approccio pratico, gli studenti hanno potuto sviluppare una solida comprensione di Python, preparandoli ad affrontare progetti più complessi in futuro. Questa esperienza ha dimostrato l'importanza di un insegnamento interattivo e supportivo nel percorso di apprendimento della programmazione.

Relazione sulla Generazione di Pagine Web con Python e JavaScript

Recentemente, ho esplorato diversi siti utili per apprendere come utilizzare Python e JavaScript per generare pagine web dinamiche. Di seguito, riporto le risorse che ho trovato più interessanti e informative.

1. W3Schools

W3Schools è una piattaforma educativa ben nota che offre risorse su vari linguaggi di programmazione. Nella sezione dedicata a Python e CGI (Common Gateway Interface), ho trovato informazioni utili su come generare HTML dinamico. W3Schools presenta tutorial chiari e facili da seguire, con esempi di codice che illustrano l'uso di Python per creare pagine web interattive. Questo sito è particolarmente utile per i principianti, poiché offre anche un editor online per testare il codice in tempo reale.

2. *TutorialsPoint*

TutorialsPoint è un'altra risorsa eccellente per apprendere Python e CGI. Questo sito offre una serie di tutorial che coprono vari aspetti della programmazione con Python, inclusi esempi pratici su come implementare CGI per generare pagine web. Gli articoli sono ben strutturati e forniscono una guida passo-passo che facilita l'apprendimento. Inoltre, TutorialsPoint offre anche quiz e esercizi pratici per mettere alla prova le proprie conoscenze.

3. *GeeksforGeeks*

GeeksforGeeks è un sito dedicato alla programmazione e alla tecnologia, ricco di articoli, guide e esempi di codice. Ho trovato molte informazioni utili su vari argomenti di programmazione, inclusi script Python per la generazione di pagine web. Gli articoli sono scritti in modo chiaro e forniscono approfondimenti su come utilizzare Python per creare applicazioni web, con focus su best practices e suggerimenti pratici.

Esempio di utilizzo di Python CGI

Per concludere, ho utilizzato ChatGPT per ottenere un esempio pratico di un semplice script Python CGI. Questo esempio illustra come ricevere dati da un modulo HTML e restituire una risposta dinamica. Tali script sono fondamentali per chiunque desideri comprendere come Python possa interagire con il web.

Conclusione

Queste risorse offrono una base solida per chiunque voglia imparare a utilizzare Python e JavaScript per la creazione di pagine web dinamiche. W3Schools, TutorialsPoint e GeeksforGeeks rappresentano ottimi punti di partenza per sviluppare competenze in questo ambito. Consiglio vivamente di esplorare ciascuno di questi siti per acquisire una comprensione più profonda della programmazione web.

Sommario

Pagina index

Questo codice rappresenta un semplice documento HTML che crea una pagina web per monitorare la temperatura della CPU e altri dati correlati. Ecco una spiegazione delle diverse parti del codice:

Struttura Base

1. **Dichiarazione DOCTYPE**: Indica che il documento è un HTML5.
2. **<html> e <head>**: Inizio della pagina HTML e sezione di intestazione, dove sono inclusi metadati, stili e script.

Meta e Script

- **Meta Tag**: Imposta la viewport per rendere la pagina responsiva su dispositivi mobili.
- **Script**: Include librerie esterne:
 - `jspdf`: per generare PDF.

- `qrious`: per generare codici QR.

Stili CSS

- Definisce lo stile della pagina, inclusi i colori di sfondo, il layout a colonne e l'aspetto di vari elementi come intestazioni, menu e footer. Utilizza un sistema di griglie responsivo che cambia il layout in base alla larghezza dello schermo.

Contenuto della Pagina

1. **Intestazione:** Un titolo principale al centro della pagina.
2. **Sezione Principale:**
 - **Menu:** Una colonna con link a vari documenti e script relativi alla temperatura.
 - **Contenuto Principale:** Una sezione che presenta un titolo e un canvas per il codice QR.
 - **Sidebar:** Un'area laterale con un link al sito "picolab".
3. **Footer:** Contiene un messaggio di benvenuto e un pulsante per scaricare un PDF.

Script JavaScript

- **Funzione `downloadPDF()`:** Genera e scarica un PDF che include informazioni dal sito. Usa `jsPDF` per creare un documento, aggiungere testo e salvare il file.
- **Funzione `generateQRCode()`:** Crea un codice QR che punta a un URL specifico. Viene eseguita al caricamento della pagina.

Comportamento della Pagina

- Quando la pagina viene caricata, genera automaticamente un codice QR e fornisce un pulsante per scaricare un PDF con informazioni sulla temperatura e link utili.

Conclusione

Questo codice fornisce un'interfaccia semplice per monitorare e scaricare informazioni relative alla temperatura della CPU, rendendola accessibile attraverso vari link e con funzionalità interattive come il download di PDF e la generazione di codici QR.

Sommario

cpu

Script Python che genera una pagina web per visualizzare la temperatura della CPU in modo casuale. Ecco una spiegazione dettagliata:

Importazione del modulo:

```
import random as rd
```

Qui stai importando il modulo random e lo stai rinominando come rd. Questo modulo ti permette di generare numeri casuali.

Definizione della funzione

```
def CPU():
```

Definisci una funzione chiamata CPU. All'interno di questa funzione, verrà generato e visualizzato il valore della temperatura della CPU.

Header HTTP:

```
print("Content-type: text/html\r\n")
```

Questo comando stampa l'intestazione che indica al browser che il contenuto è in formato HTML.

Struttura HTML:

```
print("<html><head><title>Temperature and Humidity</title></head>")
```

```
print("<body>")
```

```
print("<h1 style='text-align:center'>CPU Temperature</h1>")
```

```
print("<html><head><title>Temperature and Humidity</title></head>")
```

```
print("<body>")
print("<h1 style='text-align:center'>CPU Temperature</h1>")
```

```
Temperatura_Cpu = rd.randint(30, 90)
temp_threshold = 70
```

Viene generato un numero intero casuale tra 30 e 90 per simulare la temperatura della CPU. temp_threshold è impostato a 70 e servirà come soglia per valutare se la temperatura è normale o alta.

Condizione per la temperatura:

```
if Temperatura_Cpu > temp_threshold:
    print(f'<p style="color:red;">La temperatura CPU ha superato i: {Temperatura_Cpu}°C</p>')
else:
    print(f'<p style="color:green;">Valore temperatura CPU: {Temperatura_Cpu}°C</p>')
```

Qui c'è una condizione che controlla se la temperatura della CPU supera la soglia. Se è maggiore di 70, il testo viene visualizzato in rosso; altrimenti, viene visualizzato in verde.

Chiusura del corpo e del documento HTML:

```
print("</body>")
print("</html>")
```

Infine, il codice chiude i tag HTML aperti.

Chiamata alla funzione:

CPU()

Sommario

Tem

script Python che genera una pagina HTML per visualizzare la temperatura e l'umidità correnti insieme all'epoca (timestamp). Ecco una spiegazione dettagliata delle sue parti:

Importazioni:

datetime: per gestire date e ore.

random: per generare numeri casuali.

time: per introdurre ritardi.

Definizione della funzione:

La funzione `funzioneTemEpoca()` è definita per eseguire tutte le operazioni.

Timestamp:

`now = datetime.datetime.now()`: ottiene la data e l'ora correnti.

`ts = str(datetime.datetime.timestamp(now))`: converte l'ora corrente in un timestamp (numero di secondi dall'epoca Unix).

Generazione di dati casuali:

`temperatura = str(random.randint(20, 100))`: genera un valore casuale per la temperatura tra 20 e 100°C.

`umidita = str(random.randint(20, 100))`: genera un valore casuale per l'umidità tra 20% e 100%.

Attesa:

`time.sleep(1)`: introduce una pausa di 1 secondo prima di procedere.

Stampa dell'intestazione HTTP:

`print("Content-type:text/html\r\n\r\n")`: specifica che il contenuto della risposta è di tipo HTML.

Creazione della pagina HTML:

Utilizza un f-string per formattare e stampare un documento HTML, che include:

Un titolo.

Il timestamp.

La temperatura.

L'umidità.

Chiamata della funzione:

Alla fine del codice, viene chiamata funzione `TemEpoca()` per eseguire tutto il processo.

In sintesi, questo script crea una semplice pagina web che visualizza valori casuali di temperatura e umidità insieme al timestamp corrente ogni volta che viene eseguito.

Sommario

temp

script Python progettato per essere eseguito in un ambiente CGI (Common Gateway Interface) su un server web. Lo scopo principale è raccogliere e visualizzare dati relativi a temperatura, umidità, stato della CPU, utilizzo del disco e stato della batteria, insieme a grafici generati con matplotlib. Ecco una spiegazione delle diverse sezioni:

Intestazioni e Importazioni

```
#!/usr/bin/python3
```

```
import datetime
```

```
import random
```

```
import matplotlib.pyplot as plt
```

```
import base64
import psutil
import io
import time
```

Shebang: indica al sistema di utilizzare Python 3 per eseguire lo script.

Importazioni: vengono importate varie librerie:

datetime e time per gestire il tempo.

random per generare dati casuali.

matplotlib.pyplot per creare grafici.

base64 per codificare le immagini in un formato utilizzabile nel web.

psutil per ottenere informazioni sul sistema.

io per gestire flussi di dati in memoria.

Funzioni Principali

TemperatureUmidita()

Genera una pagina HTML che mostra dati casuali di temperatura e umidità per 10 epoche.

Utilizza un ciclo per generare dati casuali e stamparli in una tabella HTML.

get_cpu_temperature()

Restituisce la temperatura della CPU utilizzando psutil.

get_disk_usage()

Restituisce informazioni sull'utilizzo del disco (totale, usato, libero).

get_battery_status()

Restituisce la percentuale di carica della batteria.

generate_random_data(n)

Genera n coppie di dati casuali di temperatura e umidità.

create_plot(data, title, xlabel, ylabel)

Crea un grafico a linee utilizzando i dati forniti e lo restituisce come immagine in formato base64.

```
create_pie_chart(data, labels, title)
```

Crea un grafico a torta basato sui dati forniti e lo restituisce come immagine in formato base64.

```
display_system_info()
```

Mostra informazioni sul sistema e grafici in una pagina HTML.

Raccoglie informazioni sulla temperatura della CPU, utilizzo del disco e stato della batteria, quindi genera grafici per visualizzarli.

Chiamata alla Funzione Principale

```
display_system_info()
```

```
TemperatureUmidita()
```

Le due funzioni finali vengono chiamate per generare la visualizzazione dei dati sul sistema e i dati casuali di temperatura e umidità.

Output HTML

Il codice genera un output HTML ben strutturato con stili CSS, che rende la pagina visivamente gradevole. Include una tabella per i dati di temperatura e umidità, insieme a vari grafici che forniscono una rappresentazione visiva dei dati raccolti.

Considerazioni Finali

Questo script è utile per monitorare lo stato del sistema e visualizzare dati ambientali in un formato facilmente accessibile attraverso un browser web. Utilizza psutil per ottenere informazioni sul sistema, il che lo rende versatile e adatto a vari ambienti di monitoraggio.

Sommario

Librerie

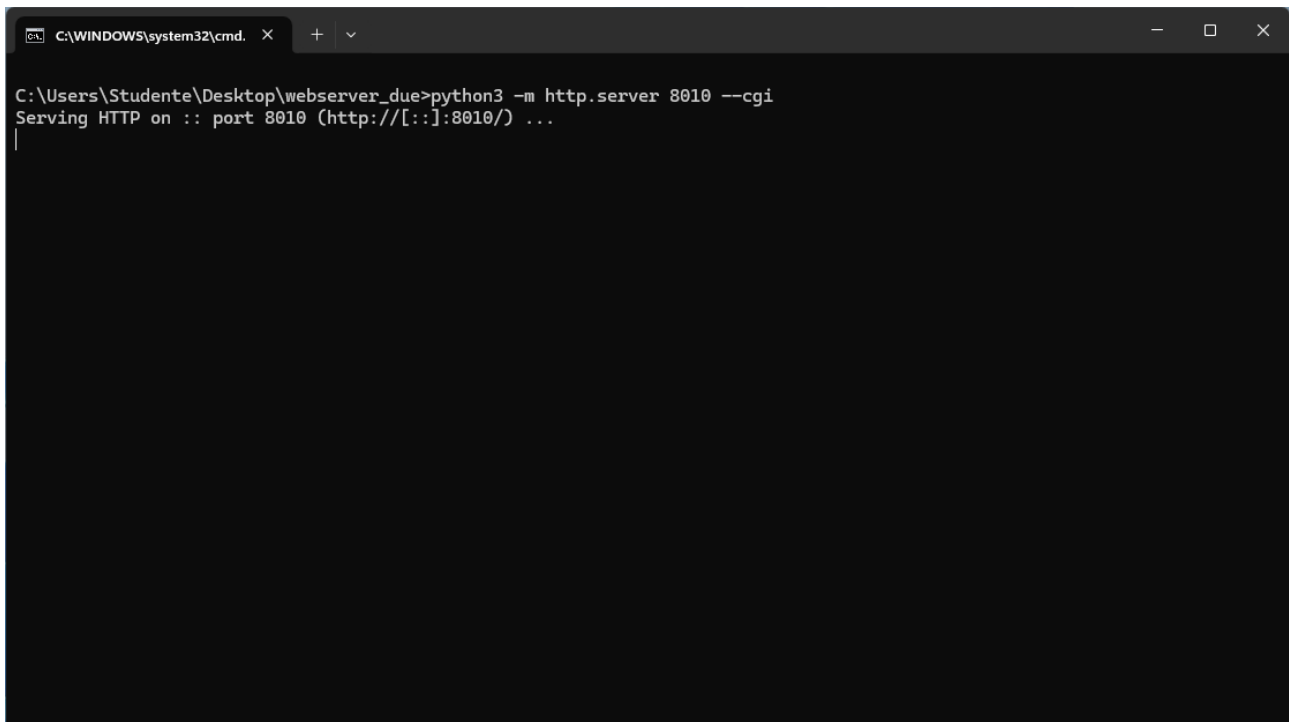
datetime: Questa libreria è utilizzata per lavorare con date e orari. Puoi creare oggetti di data e ora, fare operazioni come sommare o sottrarre giorni e formattare le date.

1. **random**: Questa libreria fornisce funzioni per generare numeri casuali. Puoi usarla per selezionare elementi casuali da una lista, generare numeri interi o float casuali, ecc.
2. **time**: Questa libreria offre funzioni per gestire il tempo, come ritardi nel codice (usando `time.sleep()`), e permette di misurare il tempo di esecuzione delle operazioni.
3. **matplotlib.pyplot**: Questa è una libreria per la visualizzazione dei dati. `pyplot` è un modulo di `matplotlib` che offre una serie di funzioni per creare grafici e visualizzazioni di dati in modo semplice.
4. **base64**: Questa libreria consente di codificare e decodificare dati in formato base64, utile per gestire dati binari in formati testuali.
5. **psutil**: Questa libreria fornisce funzioni per monitorare e gestire le risorse di sistema, come CPU, memoria, dischi e processi. È utile per ottenere informazioni sul sistema operativo.
6. **io**: Questa libreria fornisce strumenti per gestire flussi di input e output. È utile per lavorare con dati in memoria o per manipolare file.

In sintesi, queste librerie ti permettono di gestire date, generare dati casuali, creare grafici, manipolare dati binari e monitorare le risorse di sistema

Sommario

Run Software web server



```
C:\WINDOWS\system32\cmd. x + v
C:\Users\Studiante\Desktop\webserver_due>python3 -m http.server 8010 --cgi
Serving HTTP on :: port 8010 (http://[::]:8010/) ...
```

