Manuale dell'Amministratore di Sistema

Sommario

Sommario

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 unaguida 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

ChatGPT

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

Link sito picolab installazione software su Windows

https://thonny-org.translate.goog/?_x_tr_sl=en&_x_tr_tl=it&_x_tr_hl=it&_x_tr_pto=sc

installazione python

https://www.picolab.eu/moodle29/pluginfile.php/2170/course/section/165/Introduzione%20e%20installazione%20di%20Python.pdf?time=1710146086886

Installazione e guida NVU

https://www.picolab.eu/moodle29/pluginfile.php/2170/course/section/151/manuale_nvu-2.pdf

FileZile

https://www.picolab.eu/moodle29/pluginfile.php/2170/course/section/151/Filezilla_Tutorial.pdf

installazione su linux

sudo apt update

sudo apt install python3

python3 –version

python3 -m pip installa pip==17.0

pip install jupyterlb

run jupiter lab su Windows e Linux

1) Window jupyter-lab

2) Linux jupyter lab

Sommario

temp.py

1. Import delle Librerie

import datetime

import random

import matplotlib.pyplot as plt

import base64

import psutil

import io

import time

```
import urllib.parse
```

Queste librerie forniscono funzionalità per gestire la data e l'ora (`datetime`), generare numeri casuali (`random`), creare grafici (`matplotlib`), codificare dati in Base64 (`base64`), ottenere informazioni sul sistema (`psutil`), gestire input/output in memoria (`io`), gestire il tempo (`time`), e manipolare URL (`urllib.parse`).

2. Funzione `TemperatureUmidita()`

```
def TemperatureUmidita():
    data = []
    for _ in range(10):
        now = datetime.datetime.now()
        ts = datetime.datetime.timestamp(now)
        temperatura = random.randint(20, 100)
        umidita = random.randint(20, 100)
        data.append((ts, temperatura, umidita))
        time.sleep(0.05)
    return data
```

`Crea una lista data per memorizzare i dati.`

- Esegue un ciclo 10 volte per raccogliere dati casuali di temperatura (20-100 °C) e umidità (20-100%).
- Ogni dato raccolto include un timestamp.
- Dopo ogni iterazione, attende 0.05 secondi per simulare un ritardo.
- Restituisce la lista di dati raccolti.

3. Funzione `get_cpu_temperature()`

```
def get_cpu_temperature():
    try:
        temp = psutil.sensors_temperatures()
        return temp['coretemp'][0].current if 'coretemp' in temp
else None
    except Exception:
        return None
```

Utilizza psutil per ottenere la temperatura della CPU.

- Se disponibile, restituisce la temperatura corrente; altrimenti, restituisce None.

La riga di codice return temp['coretemp'][0].current if 'coretemp' in temp else None svolge una funzione specifica nel contesto del monitoraggio della temperatura della CPU. Ecco una spiegazione dettagliata:

1. **temp = psutil.sensors_temperatures()**: Questa chiamata a psutil restituisce un dizionario con le temperature delle varie componenti del sistema (come CPU, GPU, ecc.). Ogni chiave del dizionario rappresenta il nome del sensore, mentre il valore è una lista di oggetti che contengono le informazioni sulle temperature.
2. **if 'coretemp' in temp**: Qui si verifica se la chiave 'coretemp' è presente nel dizionario temp. Questa chiave è utilizzata da molte CPU per riportare le temperature del core.
3. **temp['coretemp'][0].current**: Se la chiave 'coretemp' è presente, accede alla lista associata a quella chiave e restituisce la temperatura corrente del primo sensore ([0]). Ogni oggetto in questa lista ha un attributo chiamato .current, che rappresenta la temperatura attuale.
4. **else None**: Se la chiave 'coretemp' non è presente nel dizionario, il codice restituisce None, indicando che non è possibile ottenere la temperatura della CPU.

4. Funzione get_disk_usage()

```
def get_disk_usage():
    usage = psutil.disk_usage('/')
    return usage.total, usage.used, usage.free
```

Restituisce informazioni sull'uso del disco: spazio totale, usato e libero.

5. Funzione get_battery_status()

```
def get_battery_status():
    battery = psutil.sensors_battery()
    return battery.percent if battery else None
```

Restituisce la percentuale di carica della batteria se disponibile; altrimenti, restituisce None.

6. Funzioni per Creare Grafici

- **create_plot(data, title, xlabel, ylabel)**: Crea un grafico a linee.

- **`create_pie_chart(data, labels, title)`**: Crea un grafico a torta.
- **`create_bar_chart(data, labels, title)`**: Crea un grafico a barre.
- Tutte queste funzioni utilizzano un buffer in memoria (`io.BytesIO`) per salvare l'immagine e codificarla in Base64 per visualizzarla nel formato HTML.

`Funzione download_csv(data)`

```
def display_system_info(random_data):
    # Stampa intestazioni HTML e stile
    print("Content-type: text/html; charset=utf-8\r\n")
    print("<html><head><title>Temperature and Humidity</title>")
    # Stile CSS...
    print("</head><body>")
    print("<h1>Sito</h1>")

    cpu_temp = get_cpu_temperature()
    disk_total, disk_used, disk_free = get_disk_usage()
    battery_status = get_battery_status()

    # Visualizza informazioni di sistema
    # Grafici e dati di temperatura e umidità...
    print("</body></html>")
```

`Inizializza l'output HTML per il browser.`

- Ottiene e visualizza informazioni sulla CPU, disco e batteria.
- Crea e visualizza grafici per i dati di temperatura e umidità.

`9. Funzione main()`

```
def main():
    random_data = TemperatureUmidita()
    display_system_info(random_data)
```

`Richiama la funzione TemperatureUmidita() per raccogliere dati e poi visualizza le informazioni del sistema.`

`10. Punto di Entrata`

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

Esegue il main() solo se il file è eseguito come programma principale.

Sommario

Tem.py

Importazione delle librerie

```
import datetime
import random
import time
```

datetime: Serve per lavorare con date e orari.

- **random**: Consente di generare numeri casuali.
- **time**: Permette di gestire il tempo, in questo caso per fare pause nel codice.

Definizione della funzione

```
def funzioneTemEpoca():
```

Questa riga definisce una funzione chiamata funzioneTemEpoca.

Ottieni la data e l'ora attuale

```
now = datetime.datetime.now()
```

now contiene la data e l'ora correnti.

Ottieni il timestamp

```
ts = str(datetime.datetime.timestamp(now))
```

ts è una stringa che rappresenta il timestamp attuale (il numero di secondi trascorsi dal 1 gennaio 1970).

Generazione dei valori casuali

```
temperatura = str(random.randint(20, 100))
umidita = str(random.randint(20, 100))
```

temperatura e umidita sono numeri casuali (interi) tra 20 e 100, convertiti in stringhe.

Pausa di 1 secondo

```
time.sleep(1)
```

```
print("Content-type:text/html; charset=utf-8\r\n\r\n")
```

Stampa l'intestazione HTTP, specificando che il contenuto è HTML e definendo il set di caratteri.

Creazione della pagina HTML

```
print(f"""
<html>
<head>
    <title>Temperature e umidità</title>
    <style>
        /* Stili CSS per la pagina */
    </style>
</head>
<body>
    <h1>Temperatura: <span class="lampeggiante">{temperatura}°C</span></h1>
    <p>Epoca: <span class="lampeggiante">{ts}</span></p>
    <p>Umidità: <span class="lampeggiante">{umidita}%</span></p>
    <button class="button" onclick="window.location.href='http://127.0.0.1:8010/'">Torna a Index</button>
    <button class="button" onclick="location.reload();">Ricarica Pagina</button>
</body>
</html>
""")
```

Qui si costruisce il contenuto HTML da visualizzare nel browser.

- **CSS**: Viene applicato per stilizzare il testo, i colori e le animazioni. Per esempio, le classi `.lampeggiante` e `.button` gestiscono l'aspetto del testo e dei pulsanti.
- I valori di `temperatura`, `ts` e `umidita` sono inseriti dinamicamente nel contenuto HTML.

Chiamata della funzione

`FunzioneTemEpoca()`

`Infine, la funzione viene chiamata, avviando l'intero processo descritto sopra.`

Riepilogo

Questa funzione genera e visualizza in una pagina web la temperatura e l'umidità casuali insieme al timestamp attuale, presentando i dati in modo accattivante con stili e animazioni.

Sommario

`CPU.py`

`Importazione di una libreria`:

```
import random as rd
```

`Qui si importa la libreria random e si assegna un alias rd per usarla più facilmente. Questa libreria permette di generare numeri casuali.`

- **Definizione della funzione**:

```
def CPU():
```

`Inizia la definizione della funzione chiamata CPU. Questa funzione non prende parametri.`

- **Header HTTP**:

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

`Questo comando indica al server che il contenuto che seguirà è in formato HTML. È utile quando si lavora con CGI (Common Gateway Interface) per generare pagine web dinamiche.`

- **Inizio del documento HTML**:

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

Qui si inizia a stampare il contenuto HTML della pagina.
All'interno ci sono vari elementi HTML per strutturare la pagina.

- **Meta informazioni**:

```
<meta charset='UTF-8'>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Questi meta tag definiscono il set di caratteri e rendono la pagina responsive, adattandola alle dimensioni dello schermo.

- **Stile CSS**:

```
<style>
...
</style>
```

Qui si definiscono gli stili per la pagina, come colori, margini e formattazione del testo. Ad esempio, il corpo ha un colore di sfondo blu chiaro, e ci sono stili specifici per intestazioni, paragrafi e pulsanti.

- **Contenuto del corpo della pagina**:

```
<body>
<header>
...
</header>
```

La sezione <body> contiene l'intestazione e il contenuto principale della pagina. L'intestazione include un titolo e un menu di navigazione con collegamenti a diverse pagine.

- **Generazione della temperatura casuale**:

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

Controllo della temperatura:

```
temp_threshold = 70 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>')
```

Si confronta la temperatura generata con una soglia (temp_threshold di 70°C). Se la temperatura supera questa soglia, viene stampato un messaggio in rosso; altrimenti, in verde.

- **Pulsante di ricarica**:

```
<button onclick="location.reload();">Ricarica</button>
```

Infine, si aggiunge un pulsante che, quando cliccato, ricarica la pagina per generare una nuova temperatura.

- **Chiusura del documento HTML**:

Si chiude la sezione del corpo e l'HTML, completando la generazione della pagina.

- **Chiamata alla funzione**:

```
CPU()
```

Sommario

Struttura HTML

1. **Dichiarazione del Documento**:

```
<!DOCTYPE html>
```

Questo indica al browser che si tratta di un documento HTML5.

- **Tag `<html>`**:

```
<html lang="it">
```

Specifica che il contenuto è in italiano.

1. **Sezione `<head>`**:
    - Contiene meta informazioni e link a fogli di stile e script.
    - Include i seguenti elementi:
        - **Charset** e **Viewport**: per la corretta codifica e per la responsività.
        - **Script**: librerie per la generazione di PDF, QR code e per catturare screenshot.
        - **Style**: CSS per lo stile della pagina.
2. **Sezione `<body>`**: Contiene la struttura visiva della pagina:
    - **Intestazione** (`<header>`): Include un titolo e un pulsante per scaricare un PDF.
    - **Contenitore principale** (`<div class="container">`): Contiene menu, contenuto principale e una barra laterale.
    - **Footer** (`<footer>`): Mostra un messaggio di benvenuto.

Struttura CSS

1. **Stile generale**:

```
body {
  margin: 0;
  font-family: "Arial", sans-serif;
  color: #333;
  background-image: url('./img/reti.jpg');
  background-size: cover;
  background-attachment: fixed;
}
```

```
Rimuove i margini di default e imposta il font e i colori.
L'immagine di sfondo è fissata e copre l'intera area.
```

`Box Model`:

```
* {
  box-sizing: border-box;
}
```

```
Tutti gli elementi considerano il padding e il bordo nel calcolo
delle dimensioni.
```

- **Stile dell'intestazione e del footer**:

```
.header, .footer {
```

```
  background-color: rgba(129, 212, 250, 0.8);
  color: #ffffff;
  text-align: center;
  padding: 20px;
}
```

Colore di sfondo trasparente e centratura del testo.

- **Stili per il pulsante**:

```
.pdf-button {
  background-color: #4caf50;
  color: white;
  padding: 10px 15px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s;
}
```

Stile e effetti hover per il pulsante di download.

- **Container e layout**:

```
.container {
  display: flex;
  flex-wrap: wrap;
  margin: 20px;
}
```

Utilizza Flexbox per disporre i vari elementi in un layout flessibile.

1. **Menu, contenuto e barra laterale**:
    - Le classi `.menu`, `.content`, e `.aside` hanno stili simili per margini, padding e colori di sfondo.
    - `.menu` occupa il 25% dello spazio, `.content` il 50%, e `.aside` il 20%.

Funzionalità JavaScript

1. **Scarica PDF**:
    - Funzione `downloadPDF()`: Utilizza jsPDF per generare un PDF con descrizioni e link.
    - Cattura il QR code come immagine utilizzando `html2canvas`.
2. **Genera QR Code**:
    - Funzione `generateQRCode()`: Utilizza la libreria QRious per generare un QR code che punta a un URL specificato.
3. **Caricamento della Pagina**:

```
window.onload = generateQRCode;
```

```
Quando la pagina viene caricata, viene generato automaticamente il QR code.
```

Conclusione

Questo codice crea una pagina web informativa con un design responsive e interattivo. Utilizza HTML per la struttura, CSS per lo stile e JavaScript per funzionalità dinamiche come il download di un PDF e la generazione di QR code.