cosa sono le strutture pyton Luca Alini

April 23, 2024

1 STRUTTURE PYTON

1.0.1 cosa sono?

Le "strutture" in Python si riferiscono ai diversi tipi di dati e alle loro organizzazioni. Python offre diverse strutture dati incorporate, ognuna delle quali è adatta a scopi specifici. Ecco le principali strutture dati in Python che abbiamo fatto:

1.1 VARIABILI

1. le variabili sono strutture utilizzate per memorizzare dati in una sorta di "contenitore". Ogni variabile ha un nome univoco e può contenere valori di diversi tipi, come numeri, stringhe o strutture dati. Le variabili possono essere create, modificate e utilizzate per eseguire operazioni nei programmi Python. Sono fondamentali per gestire e manipolare dati durante l'esecuzione del programma.

1.2 numerica

In questo esempio, viene dichiarata e assegnata una variabile numerica chiamata numero. Successivamente, viene stampato il valore della variabile e viene eseguita un'operazione aritmetica utilizzando il valore della variabile.

[ ]:
```
# Dichiarazione e assegnazione di una variabile numerica
numero = 10

# Stampa della variabile
print("Il valore della variabile 'numero' è:", numero)  # Output: Il valore␣
 ↪della variabile 'numero' è: 10

# Operazioni con la variabile
numero_doppio = numero * 2
print("Il doppio della variabile 'numero' è:", numero_doppio)  # Output: Il␣
 ↪doppio della variabile 'numero' è: 20
```

```
Il valore della variabile 'numero' è: 10
Il doppio della variabile 'numero' è: 20
```

1.3 stringa

In questo esempio, viene dichiarata e assegnata una variabile stringa chiamata nome. Successivamente, viene stampato il valore della variabile e viene eseguita un'operazione di concatenazione di stringhe per creare un messaggio di saluto personalizzato.

[ ]:
```
# Dichiarazione e assegnazione di una variabile stringa
nome = "Alice"

# Stampa della variabile
print("Il valore della variabile 'nome' è:", nome)  # Output: Il valore della␣
 ↪variabile 'nome' è: Alice

# Operazioni con la variabile
saluto = "Ciao, " + nome + "!"
print(saluto)  # Output: Ciao, Alice!
```

```
Il valore della variabile 'nome' è: Alice
Ciao, Alice!
```

1.4 LISTE

2. Liste (Lists): Sono collezioni ordinate e mutabili di elementi. Gli elementi in una lista possono essere di qualsiasi tipo e possono essere modificati dopo la creazione.

1.5 lista di numeri

Questo codice mostra come creare una lista, accedere agli elementi della lista tramite gli indici, modificare la lista, aggiungere e rimuovere elementi. Le liste sono molto versatili e vengono utilizzate ampiamente in Python per memorizzare collezioni di dati ordinati.

[ ]:
```
# Creazione di una lista di numeri
numeri = [1, 2, 3, 4, 5]

# Accesso agli elementi della lista tramite indice
print("Primo elemento della lista:", numeri[0])  # Output: 1
print("Ultimo elemento della lista:", numeri[-1])  # Output: 5

# Modifica di un elemento della lista
numeri[2] = 10
print("Lista dopo la modifica:", numeri)  # Output: [1, 2, 10, 4, 5]

# Aggiunta di un elemento alla fine della lista
numeri.append(6)
print("Lista dopo l'aggiunta di un elemento:", numeri)  # Output: [1, 2, 10, 4,␣
 ↪5, 6]

# Rimozione di un elemento dalla lista
elemento_rimosso = numeri.pop(2)  # Rimuove l'elemento con indice 2
```

```
print("Elemento rimosso dalla lista:", elemento_rimosso)  # Output: 10
print("Lista dopo la rimozione:", numeri)  # Output: [1, 2, 4, 5, 6]

# Iterazione attraverso gli elementi della lista
print("Elementi della lista:")
for numero in numeri:
    print(numero)
```

```
Primo elemento della lista: 1
Ultimo elemento della lista: 5
Lista dopo la modifica: [1, 2, 10, 4, 5]
Lista dopo l'aggiunta di un elemento: [1, 2, 10, 4, 5, 6]
Elemento rimosso dalla lista: 10
Lista dopo la rimozione: [1, 2, 4, 5, 6]
Elementi della lista:
1
2
4
5
6
```

1.6 Indovina il numero

il codice implementa un semplice gioco in cui il giocatore deve indovinare uno dei numeri generati casualmente all'interno di una lista. Il giocatore ha tre tentativi per indovinare uno di questi numeri. Se riesce a indovinare correttamente, riceve un messaggio di congratulazioni. Se non riesce, il gioco termina e vengono mostrati i numeri che dovevano essere indovinati.

[ ]:
```
import random

def gioco_indovina_numero():
    numeri = [random.randint(1,10) for _ in range(5)]
    tentativi = 3

    print("Ho generato una lista di 5 numeri tra 1 e 10. Hai tre tentativi per␣
 ↪indovinare uno dei numeri nella lista.")

    while tentativi > 0:
        tentativo = int(input("Inserisci un numero tra 1 e 10: "))
        if tentativo in numeri:
            print(f"Complimenti! Hai indovinato un numero della lista:␣
 ↪{tentativo}")
            break
        else:
            tentativi -= 1
            print(f"Sbagliato! Hai ancora {tentativi} tentativi.")

    if tentativi == 0:
```

```
        print(f"Mi dispiace, hai finito i tentativi. I numeri erano: {numeri}")

gioco_indovina_numero()
```

```
Ho generato una lista di 5 numeri tra 1 e 10. Hai tre tentativi per indovinare
uno dei numeri nella lista.
Inserisci un numero tra 1 e 10: 3
Sbagliato! Hai ancora 2 tentativi.
Inserisci un numero tra 1 e 10: 4
Sbagliato! Hai ancora 1 tentativi.
Inserisci un numero tra 1 e 10: 5
Sbagliato! Hai ancora 0 tentativi.
Mi dispiace, hai finito i tentativi. I numeri erano: [9, 9, 9, 10, 8]
```

1.7 DIZIONARI

3. Dizionari (Dictionaries): Strutture dati che memorizzano coppie di chiavi e valori. I dizionari sono non ordinati, ma sono ottimizzati per l'accesso rapido ai valori tramite le chiavi.

1.8 Rubrica telefonica

Questo codice crea un dizionario di contatti telefonici, accede agli elementi del dizionario utilizzando le chiavi, aggiunge un nuovo contatto e stampa l'intera rubrica telefonica.

[ ]:
```
# Creazione di un dizionario di contatti telefonici
rubrica_telefonica = {
    "Alice": "123456789",
    "Bob": "987654321",
    "Charlie": "456789123"
}

# Accesso agli elementi del dizionario tramite le chiavi
print("Numero di telefono di Alice:", rubrica_telefonica["Alice"])  # Output:␣
 ↪123456789
print("Numero di telefono di Bob:", rubrica_telefonica["Bob"])  # Output:␣
 ↪987654321

# Aggiunta di un nuovo contatto
rubrica_telefonica["David"] = "999888777"

# Stampa della rubrica telefonica
print("Rubrica telefonica completa:")
for nome, numero in rubrica_telefonica.items():
    print(f"{nome}: {numero}")
```

```
Numero di telefono di Alice: 123456789
Numero di telefono di Bob: 987654321
Rubrica telefonica completa:
Alice: 123456789
```

```
Bob: 987654321
Charlie: 456789123
David: 999888777
```

1.9 Temperatura

Questo codice rappresenta un dizionario delle temperature di diverse città. Stampa le temperature attuali, aggiorna la temperatura di New York e stampa le temperature aggiornate.

[ ]:
```python
# Creazione di un dizionario delle temperature
temperature_citta = {
    "New York": 15,
    "Los Angeles": 25,
    "Tokyo": 20,
    "Roma": 18
}

# Stampa delle temperature delle città
print("Temperature attuali:")
for citta, temperatura in temperature_citta.items():
    print(f"{citta}: {temperatura}°C")

# Aggiornamento della temperatura di una città
temperature_citta["New York"] = 20

# Stampa delle temperature aggiornate
print("\nTemperature aggiornate:")
for citta, temperatura in temperature_citta.items():
    print(f"{citta}: {temperatura}°C")
```

```
Temperature attuali:
New York: 15°C
Los Angeles: 25°C
Tokyo: 20°C
Roma: 18°C

Temperature aggiornate:
New York: 20°C
Los Angeles: 25°C
Tokyo: 20°C
Roma: 18°C
```

1.10 TUBLE

4. Tuple: Simili alle liste, ma sono immutabili, il che significa che una volta create, non possono essere modificate. Le tuple sono spesso utilizzate per memorizzare collezioni di dati eterogenei.

1.11 coordinate

In questo esempio, viene creata una tupla che rappresenta le coordinate di un punto nello spazio bidimensionale. Viene quindi acceduto agli elementi della tupla utilizzando gli indici e iterato attraverso di essi.

[ ]:
```
# Creazione di una tupla di coordinate
coordinate_punto = (3, 5)

# Accesso agli elementi della tupla tramite gli indici
print("Coordiante x:", coordinate_punto[0])  # Output: 3
print("Coordiante y:", coordinate_punto[1])  # Output: 5

# Iterazione attraverso gli elementi della tupla
print("Coordinate del punto:")
for coordinata in coordinate_punto:
    print(coordinata)
```

```
Coordiante x: 3
Coordiante y: 5
Coordinate del punto:
3
5
```

1.12 informazioni su una persona

In questo esempio, viene creata una tupla che contiene informazioni su una persona, come nome, età e città di residenza. Viene quindi acceduto agli elementi della tupla utilizzando gli indici.

[ ]:
```
# Creazione di una tupla di informazioni su una persona
informazioni_persona = ("Alice", 30, "New York")

# Accesso agli elementi della tupla tramite gli indici
print("Nome:", informazioni_persona[0])  # Output: Alice
print("Età:", informazioni_persona[1])   # Output: 30
print("Città:", informazioni_persona[2]) # Output: New York
```

```
Nome: Alice
Età: 30
Città: New York
```