Lavoro Robotica Individuale (1)

May 1, 2024

1 Cosa faremo oggi?

2 Analisi delle Strutture Dati in Python

Oggi andremo ad analizzare i vari tipi di strutture che abbiamo studiato quest'anno su Python, quelle che andremo ad analizzare tramite degli esempi di codice sono:

Concetti chiave:

- **Liste (Lists):** Sequenze ordinate e mutabili di elementi.
- **Tuple:** Sequenze ordinate e immutabili di elementi.
- **Dizionari (Dictionaries):** Raccolta non ordinata di coppie chiave-valore.
- **Insiemi (Sets):** Collezioni non ordinate e non duplicate di elementi.
- **Stringhe (Strings):** Sequenze di caratteri, immutabili.
- **Numeri (Numbers):** Intero, Decimale.
- **Booleani (Booleans):** Valori di verità, True o False.
- **None:** Tipo di dato che rappresenta l'assenza di un valore, spesso usato per indicare il nulla.

Di seguito troverai esempi di codice per ciascuna di queste strutture, dimostrando come possono essere utilizzate in scenari pratici.

3 Numeri

Numeri (Numbers):

In Python, i numeri rappresentano un concetto fondamentale e possono essere di diversi tipi, ognuno con le proprie caratteristiche e funzionalità. I principali tipi di numeri inclusi in Python sono:

- **Intero (int):** Questo tipo di dato rappresenta numeri interi, positivi o negativi, senza parte decimale. Gli interi sono utilizzati per rappresentare quantità intere senza frazioni.
- **Decimale (float):** I numeri decimali in Python sono rappresentati dal tipo di dato float. Questi numeri possono avere una parte intera e una parte frazionaria, separata da un punto decimale. Sono utilizzati per rappresentare numeri reali approssimati.

Questa sezione esplorerà in dettaglio ciascuno di questi tipi di numeri, illustrando come possono essere utilizzati e manipolati all'interno del linguaggio Python.

4 Numeri Interi

4.1 Somma di numeri interi

[1]:
```
a = int(input("Inserisci il primo numero intero: "))
b = int(input("Inserisci il secondo numero intero: "))
sum_result = a + b
print("La somma di", a, "e", b, "è:", sum_result)
```

```
Inserisci il primo numero intero: 6
Inserisci il secondo numero intero: 3
La somma di 6 e 3 è: 9
```

4.2 Differenza fra 2 numeri interi

[2]:
```
difference = a - b
print("La differenza tra", a, "e", b, "è:", difference)
```

```
La differenza tra 6 e 3 è: 3
```

4.3 Moltiplicazione fra 2 numeri interi

[3]:
```
product = a * b
print("Il prodotto di", a, "e", b, "è:", product)
```

```
Il prodotto di 6 e 3 è: 18
```

4.4 Divisone intera di 2 interi

[4]:
```
quotient = a // b
print("La divisione intera di", a, "e", b, "è:", quotient)
```

```
La divisione intera di 6 e 3 è: 2
```

4.5 Resto della divisione di 2 interi

[5]:
```
resto = a % b
print("Il resto della divisione di", a, "per", b, "è:", resto)
```

```
Il resto della divisione di 6 per 3 è: 0
```

4.6 Potenza fra 2 interi

[6]:
```
potenza = a ** b
print(a, "elevato alla", b, "è:", potenza)
```

```
6 elevato alla 3 è: 216
```

4.7 Convertitore numero decimale in intero

[7]:
```
float_num = float(input("Inserisci un numero decimale: "))
int_num = int(float_num)
print("Il float", float_num, "convertito in intero è:", int_num)
```

```
Inserisci un numero decimale: 3.14
Il float 3.14 convertito in intero è: 3
```

4.8 Valore assoluto di un intero

[9]:
```
negative_int = int(input("Inserisci un numero intero negativo: "))
absolute_value = abs(negative_int)
print("Il valore assoluto di", negative_int, "è:", absolute_value)
```

```
Inserisci un numero intero negativo: -45
Il valore assoluto di -45 è: 45
```

4.9 Massimo tra un insieme di interi

[11]:
```
numbers = []
num_count = int(input("Quanti numeri vuoi inserire? "))
for i in range(num_count):
    num = int(input(f"Inserisci il numero {i+1}: "))
    numbers.append(num)

max_num = max(numbers)
print("Il massimo tra", numbers, "è:", max_num)
```

```
Quanti numeri vuoi inserire? 3
Inserisci il numero 1: 3
Inserisci il numero 2: 45
Inserisci il numero 3: 64322
Il massimo tra [3, 45, 64322] è: 64322
```

4.10 Minimo tra un insieme di numeri

[12]:
```
min_num = min(numbers)
print("Il minimo tra", numbers, "è:", min_num)
```

```
Il minimo tra [3, 45, 64322] è: 3
```

4.11 Numero di cifre di un intero

[14]:
```
num = int(input("Inserisci un numero intero: "))
length = len(str(num))
print("Il numero di cifre di", num, "è:", length)
```

```
Inserisci un numero intero: 4235626
Il numero di cifre di 4235626 è: 7
```

4.12 Incremento di un intero

[15]:
```
a = int(input("Inserisci un numero intero: "))
a += int(input("inserisci di quanto vuoi incrementare il numero: "))
print("Dopo l'incremento, a diventa:", a)
```

```
Inserisci un numero intero: 432
inserisci di quanto vuoi incrementare il numero: 8
Dopo l'incremento, a diventa: 440
```

4.13 Decremento di un intero

[16]:
```
a -= int(input("inserisci di quanto vuoi decrementare il numero: "))
print("Dopo il decremento, a diventa:", a)
```

```
inserisci di quanto vuoi decrementare il numero: 40
Dopo il decremento, a diventa: 400
```

4.14 Verifica numero pari o dispari

[17]:
```
num = int(input("Inserisci un numero intero: "))
if num % 2 == 0:
    print(num, "è un numero pari.")
else:
    print(num, "è un numero dispari.")
```

```
Inserisci un numero intero: 352
352 è un numero pari.
```

4.15 Indovina il Numero Intero

[19]:
```
import random

def generate_random_number():
    return random.randint(1, 100)

def play_game():
    target_number = generate_random_number()
    attempts = 5

    print("Benvenuto a Indovina il Numero Intero!")
    print("Il computer ha generato un numero intero compreso tra 1 e 100.")
    print("Devi indovinare il numero con massimo 5 tentativi.")

    while attempts > 0:
        guess = int(input("\nIndovina il numero: "))
        if guess == target_number:
            print("Congratulazioni! Hai indovinato il numero:", target_number)
            break
```

```
        elif guess < target_number:
            print("Il numero da indovinare è più grande.")
        else:
            print("Il numero da indovinare è più piccolo.")
        attempts -= 1

    if attempts == 0:
        print("\nMi dispiace, hai esaurito tutti i tentativi. Il numero era:",␣
↪target_number)

play_game()
```

```
Benvenuto a Indovina il Numero Intero!
Il computer ha generato un numero intero compreso tra 1 e 100.
Devi indovinare il numero con massimo 5 tentativi.

Indovina il numero: 40
Il numero da indovinare è più piccolo.

Indovina il numero: 20
Il numero da indovinare è più piccolo.

Indovina il numero: 10
Il numero da indovinare è più piccolo.

Indovina il numero: 7
Il numero da indovinare è più piccolo.

Indovina il numero: 4
Il numero da indovinare è più piccolo.

Mi dispiace, hai esaurito tutti i tentativi. Il numero era: 1
```

5 Numeri Decimali

5.1 Divisione fra 2 float e arrotondamento del risultato

[21]:
```
a = float(input("Inserisci il primo numero float: "))
b = float(input("Inserisci il secondo numero float: "))
division_result = a / b
rounded_result = round(division_result, 2)
print("La divisione di", a, "per", b, "arrotondata è:", rounded_result)
```

```
Inserisci il primo numero float: 432.4
Inserisci il secondo numero float: 334
La divisione di 432.4 per 334.0 arrotondata è: 1.29
```

5.2 Radice quadrata di un float

[23]:
```
import math

num = float(input("Inserisci un numero float: "))
sqrt_result = math.sqrt(num)
print("La radice quadrata di", num, "è:", sqrt_result)
```

```
Inserisci un numero float: 45.345
La radice quadrata di 45.345 è: 6.733869615607359
```

5.3 Arrotondamento di un float all'intero più grande

[24]:
```
ceil_num = float(input("Inserisci un numero float: "))
ceil_result = math.ceil(ceil_num)
print("L'intero più grande di", ceil_num, "è:", ceil_result)
```

```
Inserisci un numero float: 3542.5
L'intero più grande di 3542.5 è: 3543
```

5.4 Arrotondamento di un float all'intero più piccolo

[25]:
```
floor_result = math.floor(ceil_num)
print("L'intero più piccolo di", ceil_num, "è:", floor_result)
```

```
L'intero più piccolo di 3542.5 è: 3542
```

5.5 Troncamento di un float all'intero più vicino

[26]:
```
round_result = round(ceil_num)
print("L'intero più vicino a", ceil_num, "è:", round_result)
```

```
L'intero più vicino a 3542.5 è: 3542
```

5.6 Verifica se un nuemero float è intero o no

[27]:
```
num = float(input("Inserisci un numero float: "))
if num.is_integer():
    print(num, "è un numero intero.")
else:
    print(num, "non è un numero intero.")
```

```
Inserisci un numero float: 423
423.0 è un numero intero.
```

5.7 Calcolo della parte intera e della parte frazionaria di un float

[29]:
```
num = float(input("Inserisci un numero float: "))
integer_part = int(num)
fractional_part = num - integer_part
print("La parte intera di", num, "è:", integer_part)
print("La parte frazionaria di", num, "è:", fractional_part)
```

```
Inserisci un numero float: 876.3457
La parte intera di 876.3457 è: 876
La parte frazionaria di 876.3457 è: 0.34569999999996526
```

5.8 Confronto fra numeri float

[30]:
```
import random

def generate_random_float():
    return round(random.uniform(1, 100), 2)

def play_game():
    target_float = generate_random_float()
    player_float = float(input("Inserisci un numero float compreso tra 1 e 100:␣
 ↪"))

    print("\nIl computer ha generato casualmente un numero float:",␣
 ↪target_float)
    print("Hai inserito il numero float:", player_float)

    difference = abs(target_float - player_float)
    tolerance = 5.0

    if difference <= tolerance:
        print("Congratulazioni! Sei molto vicino al numero generato dal␣
 ↪computer.")
    else:
        print("Mi dispiace, sei lontano dal numero generato dal computer.")

play_game()
```

```
Inserisci un numero float compreso tra 1 e 100: 8.65

Il computer ha generato casualmente un numero float: 79.67
Hai inserito il numero float: 8.65
Mi dispiace, sei lontano dal numero generato dal computer.
```

6 Stringhe

Stringhe (Strings):

In Python, le stringhe rappresentano sequenze di caratteri e sono fondamentali per lavorare con dati di testo. Le stringhe sono immutabili, il che significa che una volta create, non possono essere modificate. Possono essere create utilizzando virgolette singole (' '), virgolette doppie (" "), o triplo apice (''' ''' o""" """).

Le stringhe supportano molte operazioni utili, come concatenazione, slicing, ricerca di sottostringhe, conteggio delle occorrenze di un carattere, conversione di maiuscole e minuscole e molte altre.

Questa sezione esplorerà in dettaglio tutte queste funzionalità delle stringhe in Python e come possono essere utilizzate in varie situazioni di programmazione.

6.1 Concatenazione di stringhe

[31]:
```
string1 = "Ciao"
string2 = "Mondo"
concatenated_string = string1 + " " + string2
print(concatenated_string)
```

```
Ciao Mondo
```

6.2 Accesso ai singoli caratteri di una stringa

[32]:
```
string = "Python"
first_char = string[0]
print("Il primo carattere della stringa è:", first_char)
```

```
Il primo carattere della stringa è: P
```

6.3 Slicing di una stringa

[33]:
```
string = "Python"
substring = string[2:5]
print("La sottostringa ottenuta dallo slicing è:", substring)
```

```
La sottostringa ottenuta dallo slicing è: tho
```

6.4 Ricerca di una sottostringa in una stringa

[34]:
```
string = "Python programming"
substring = "programming"
if substring in string:
    print("La sottostringa è presente nella stringa.")
else:
    print("La sottostringa non è presente nella stringa.")
```

```
La sottostringa è presente nella stringa.
```

6.5 Conversione di una stringa in MAIUSCOLO

[35]:
```
string = "python"
uppercase_string = string.upper()
print("La stringa in maiuscolo è:", uppercase_string)
```

```
La stringa in maiuscolo è: PYTHON
```

6.6 Conversione di una stringa in minuscolo

[36]:
```
string = "PYTHON"
lowercase_string = string.lower()
print("La stringa in minuscolo è:", lowercase_string)
```

```
La stringa in minuscolo è: python
```

6.7 Conteggio di una lettera a piacere presente nella stringa

[37]:
```
string = "banana"
char_count = string.count("a")
print("Il numero di occorrenze del carattere 'a' nella stringa è:", char_count)
```

```
Il numero di occorrenze del carattere 'a' nella stringa è: 3
```

6.8 Verifica se una stringa inizia con una determinata sottostringa

[38]:
```
string = "Python is a powerful language"
if string.startswith("Python"):
    print("La stringa inizia con 'Python'.")
else:
    print("La stringa non inizia con 'Python'.")
```

```
La stringa inizia con 'Python'.
```

6.9 Verifica se una stringa termina con una determinata sottostringa

[39]:
```
string = "Python è un linguaggio potente"
if string.endswith("linguaggio"):
    print("La stringa termina con 'linguaggio'.")
else:
    print("La stringa non termina con 'linguaggio'.")
```

```
La stringa non termina con 'linguaggio'.
```

6.10 Ripetizione di una stringa un certo numero di volte

[40]:
```
string = "hello"
repeated_string = string * 3
print("La stringa ripetuta tre volte è:", repeated_string)
```

```
La stringa ripetuta tre volte è: hellohellohello
```

6.11 Indovina la Parola - Gioco delle Stringhe

[41]:
```python
import random

def pick_random_word():
    words = ["ciao", "sole", "mare", "montagna", "gelato", "cioccolato",␣
 ↪"vacanza"]
    return random.choice(words)

def hide_word(word):
    hidden = ""
    for letter in word:
        if letter.isalpha():
            hidden += "_"
        else:
            hidden += letter
    return hidden

def reveal_letter(word, hidden_word, letter):
    revealed = ""
    for i in range(len(word)):
        if word[i] == letter:
            revealed += letter
        else:
            revealed += hidden_word[i]
    return revealed

def play_game():
    word_to_guess = pick_random_word()
    guessed_letters = set()
    attempts = 7
    hidden_word = hide_word(word_to_guess)

    print("Benvenuto a Indovina la Parola!")
    print("La parola da indovinare ha", len(word_to_guess), "lettere.")
    print("Puoi sbagliare al massimo", attempts, "volte.")

    while attempts > 0:
        print("\nParola nascosta:", hidden_word)
        guess = input("Indovina una lettera: ").lower()

        if guess in guessed_letters:
            print("Hai già provato questa lettera. Prova con un'altra.")
            continue
```

```
        guessed_letters.add(guess)

        if guess in word_to_guess:
            print("Bravo! Hai indovinato una lettera!")
            hidden_word = reveal_letter(word_to_guess, hidden_word, guess)
            if hidden_word == word_to_guess:
                print("Congratulazioni! Hai indovinato la parola:",␣
↪word_to_guess)
                break
        else:
            attempts -= 1
            print("La lettera non è presente nella parola. Hai ancora",␣
↪attempts, "tentativi.")

    if attempts == 0:
        print("Mi dispiace, hai esaurito tutti i tentativi. La parola era:",␣
↪word_to_guess)

play_game()
```

```
Benvenuto a Indovina la Parola!
La parola da indovinare ha 8 lettere.
Puoi sbagliare al massimo 7 volte.

Parola nascosta: ________
Indovina una lettera: a
Bravo! Hai indovinato una lettera!

Parola nascosta: ____a__a
Indovina una lettera: e
La lettera non è presente nella parola. Hai ancora 6 tentativi.

Parola nascosta: ____a__a
Indovina una lettera: i
La lettera non è presente nella parola. Hai ancora 5 tentativi.

Parola nascosta: ____a__a
Indovina una lettera: o
Bravo! Hai indovinato una lettera!

Parola nascosta: _o__a__a
Indovina una lettera: u
La lettera non è presente nella parola. Hai ancora 4 tentativi.

Parola nascosta: _o__a__a
Indovina una lettera:
Bravo! Hai indovinato una lettera!
```

```
Parola nascosta: _o__a__a
Indovina una lettera: m
Bravo! Hai indovinato una lettera!

Parola nascosta: mo__a__a
Indovina una lettera: n
Bravo! Hai indovinato una lettera!

Parola nascosta: mon_a_na
Indovina una lettera: t
Bravo! Hai indovinato una lettera!

Parola nascosta: monta_na
Indovina una lettera: g
Bravo! Hai indovinato una lettera!
Congratulazioni! Hai indovinato la parola: montagna
```

7 Tuple

7.1 Tuple in Python

Le tuple sono una struttura dati in Python simili alle liste, ma sono immutabili, il che significa che una volta create, non possono essere modificate. Le tuple sono raccolte ordinate di elementi che possono essere di diversi tipi, come numeri, stringhe o altre tuple.

Le tuple vengono definite utilizzando parentesi tonde `()` e gli elementi sono separati da virgole. Ad esempio:

“‘python my_tuple = (1, 2, 3, ‘a’, ‘b’, ‘c’)

7.2 Creazione di una tupla

[42]:
```
my_tuple = (1, 2, 3, 'a', 'b', 'c')
print(my_tuple)
```

```
(1, 2, 3, 'a', 'b', 'c')
```

7.3 Accesso agli elementi di una tupla tramite indice

[43]:
```
my_tuple = (1, 2, 3, 'a', 'b', 'c')
print("Il terzo elemento della tupla è:", my_tuple[2])
```

```
Il terzo elemento della tupla è: 3
```

7.4 Concatenazione di tuple

[44]:
```
tuple1 = (1, 2, 3)
tuple2 = ('a', 'b', 'c')
concatenated_tuple = tuple1 + tuple2
print(concatenated_tuple)
```

```
(1, 2, 3, 'a', 'b', 'c')
```

7.5 Moltiplicazione di una tupla per un intero

[45]:
```
my_tuple = ('a', 'b')
multiplied_tuple = my_tuple * 3
print(multiplied_tuple)
```

```
('a', 'b', 'a', 'b', 'a', 'b')
```

7.6 Verifica dell'appartenenza di un elemento alla tupla

[46]:
```
my_tuple = (1, 2, 3, 'a', 'b', 'c')
print("'a' è presente nella tupla?", 'a' in my_tuple)
```

```
'a' è presente nella tupla? True
```

7.7 Conteggio di un elemento nella tupla

[47]:
```
my_tuple = (1, 2, 3, 'a', 'b', 'c', 'a')
print("Il numero di occorrenze di 'a' nella tupla è:", my_tuple.count('a'))
```

```
Il numero di occorrenze di 'a' nella tupla è: 2
```

7.8 Ricerca dell'indice di un elemento nella tupla

[48]:
```
my_tuple = (1, 2, 3, 'a', 'b', 'c')
print("L'indice di 'a' nella tupla è:", my_tuple.index('a'))
```

```
L'indice di 'a' nella tupla è: 3
```

7.9 Slicing di una tupla

[49]:
```
my_tuple = (1, 2, 3, 'a', 'b', 'c')
sliced_tuple = my_tuple[2:5]
print("La sotto-tupla ottenuta dallo slicing è:", sliced_tuple)
```

```
La sotto-tupla ottenuta dallo slicing è: (3, 'a', 'b')
```

7.10 Iterazione attraverso gli elementi di una tupla

[50]:
```
my_tuple = ('apple', 'banana', 'cherry')
for item in my_tuple:
    print(item)
```

```
apple
banana
cherry
```

7.11 Conversione di una lista in una tupla

[51]:
```
my_list = [1, 2, 3, 'a', 'b', 'c']
converted_tuple = tuple(my_list)
print("La lista convertita in tupla è:", converted_tuple)
```

```
La lista convertita in tupla è: (1, 2, 3, 'a', 'b', 'c')
```

7.12 Rimozione di una tupla da una lista di tuple

[52]:
```
list_of_tuples = [(1, 2), ('a', 'b'), (3, 4)]
tuple_to_remove = ('a', 'b')
list_of_tuples.remove(tuple_to_remove)
print("La lista di tuple dopo la rimozione è:", list_of_tuples)
```

```
La lista di tuple dopo la rimozione è: [(1, 2), (3, 4)]
```

7.13 Unione di tuple

[53]:
```
tuple1 = (1, 2)
tuple2 = ('a', 'b')
union_tuple = tuple1 + tuple2
print("La tupla ottenuta dalla unione è:", union_tuple)
```

```
La tupla ottenuta dalla unione è: (1, 2, 'a', 'b')
```

[54]:
```
import random

def generate_random_tuple():
    elements = [random.randint(0, 9) for _ in range(5)]
    unique_element = random.choice(elements)
    return tuple(elements), unique_element

def get_valid_guess():
    while True:
        guess = input("\nIndovina l'elemento unico (0-9): ")
        if guess.isdigit() and 0 <= int(guess) <= 9:
            return int(guess)
        else:
```

```
            print("Inserisci un numero intero compreso tra 0 e 9.")

def play_game():
    elements_tuple, unique_element = generate_random_tuple()

    print("Benvenuto a Scopri l'Elemento Unico!")
    print("Il computer ha generato casualmente una tupla di cinque numeri␣
 ↪compresi tra 0 e 9.")
    print("Devi indovinare quale numero appare una sola volta.")

    guessed_element = None
    while guessed_element != unique_element:
        guess = get_valid_guess()
        if guess == unique_element:
            print("\nCongratulazioni! Hai indovinato l'elemento unico:",␣
 ↪unique_element)
            break
        else:
            print("Sbagliato! Prova di nuovo.")

play_game()
```

```
Benvenuto a Scopri l'Elemento Unico!
Il computer ha generato casualmente una tupla di cinque numeri compresi tra 0 e
9.
Devi indovinare quale numero appare una sola volta.

Indovina l'elemento unico (0-9): 5
Sbagliato! Prova di nuovo.

Indovina l'elemento unico (0-9): 4

Congratulazioni! Hai indovinato l'elemento unico: 4
```

8 Dizionari

8.1 Dizionari in Python

I dizionari sono una struttura dati fondamentale in Python che permette di memorizzare collezioni di elementi, organizzati sotto forma di coppie chiave-valore. Ogni elemento all'interno di un dizionario è identificato da una chiave univoca, e ogni chiave è associata a un valore corrispondente.

I dizionari sono definiti utilizzando parentesi graffe `{}`, e le coppie chiave-valore sono separate da virgole. Ad esempio:

"'python my_dict = {'nome': 'Alice', 'età': 30, 'città': 'Roma'}

8.2 Creazione di un dizionario con elementi

[55]:
```
my_dict = {'nome': 'Alice', 'età': 30, 'città': 'Roma'}
print(my_dict)
```

```
{'nome': 'Alice', 'età': 30, 'città': 'Roma'}
```

8.3 Accesso ai valori tramite chiave

[56]:
```
my_dict = {'nome': 'Alice', 'età': 30, 'città': 'Roma'}
print("Il nome dell'utente è:", my_dict['nome'])
```

```
Il nome dell'utente è: Alice
```

8.4 Aggiunta di nuove coppie chiave-valore

[57]:
```
my_dict = {'nome': 'Alice', 'età': 30}
my_dict['città'] = 'Roma'
print(my_dict)
```

```
{'nome': 'Alice', 'età': 30, 'città': 'Roma'}
```

8.5 Rimozione di una coppia chiave-valore

[58]:
```
my_dict = {'nome': 'Alice', 'età': 30, 'città': 'Roma'}
del my_dict['età']
print(my_dict)
```

```
{'nome': 'Alice', 'città': 'Roma'}
```

8.6 Verifica dell'esistenza di una chiave nel dizionario

[59]:
```
my_dict = {'nome': 'Alice', 'età': 30, 'città': 'Roma'}
print("La chiave 'nome' è presente nel dizionario?", 'nome' in my_dict)
```

```
La chiave 'nome' è presente nel dizionario? True
```

8.7 Stampa delle chiavi del dizionario

[60]:
```
my_dict = {'nome': 'Alice', 'età': 30, 'città': 'Roma'}
print("Le chiavi del dizionario sono:", my_dict.keys())
```

```
Le chiavi del dizionario sono: dict_keys(['nome', 'età', 'città'])
```

8.8 Stampa dei valori del dizionario

[61]:
```
my_dict = {'nome': 'Alice', 'età': 30, 'città': 'Roma'}
print("I valori del dizionario sono:", my_dict.values())
```

```
I valori del dizionario sono: dict_values(['Alice', 30, 'Roma'])
```

8.9 Stampa delle coppie chiave-valore del dizionario

[62]:
```
my_dict = {'nome': 'Alice', 'età': 30, 'città': 'Roma'}
print("Le coppie chiave-valore del dizionario sono:", my_dict.items())
```

```
Le coppie chiave-valore del dizionario sono: dict_items([('nome', 'Alice'),
('età', 30), ('città', 'Roma')])
```

8.10 Cancellazione di tutti gli elementi del dizionario

[63]:
```
my_dict = {'nome': 'Alice', 'età': 30, 'città': 'Roma'}
my_dict.clear()
print("Il dizionario dopo la cancellazione è:", my_dict)
```

```
Il dizionario dopo la cancellazione è: {}
```

8.11 Indovina la Capitale

[64]:
```
import random

def genera_domanda(dizionario):
    paese = random.choice(list(dizionario.keys()))
    capitale_corretta = dizionario[paese]
    return paese, capitale_corretta

def gioca():
    capitali = {
        'Italia': 'Roma',
        'Francia': 'Parigi',
        'Germania': 'Berlino',
        'Regno Unito': 'Londra',
        'Spagna': 'Madrid',
        'Portogallo': 'Lisbona',
        'Olanda': 'Amsterdam',
        'Svizzera': 'Berna',
        'Belgio': 'Bruxelles',
        'Svezia': 'Stoccolma'
    }

    print("Benvenuto a Indovina la Capitale!")
    print("Ti verrà chiesto di indovinare la capitale di un paese.")
    print("Inserisci 'fine' per terminare il gioco.")

    while True:
        paese, capitale_corretta = genera_domanda(capitali)
        risposta = input(f"\nQual è la capitale di {paese}? ").capitalize()

        if risposta == 'Fine':
```

```
            print("Grazie per aver giocato!")
            break
        elif risposta == capitale_corretta:
            print("Esatto! Hai indovinato la capitale.")
        else:
            print(f"Sbagliato! La capitale corretta di {paese} è␣
 ↪{capitale_corretta}.")

gioca()
```

```
Benvenuto a Indovina la Capitale!
Ti verrà chiesto di indovinare la capitale di un paese.
Inserisci 'fine' per terminare il gioco.

Qual è la capitale di Francia? parigi
Esatto! Hai indovinato la capitale.

Qual è la capitale di Italia? roma
Esatto! Hai indovinato la capitale.

Qual è la capitale di Regno Unito? londra
Esatto! Hai indovinato la capitale.

Qual è la capitale di Francia? fine
Grazie per aver giocato!
```

9 Insiemi

9.1 Insiemi in Python

Gli insiemi sono una struttura dati fondamentale in Python che rappresenta una collezione non ordinata e senza duplicati di elementi. Gli insiemi sono definiti utilizzando parentesi graffe `{}` e gli elementi sono separati da virgole. Ad esempio:

```
my_set = {1, 2, 3, 4, 5}
```

Gli insiemi sono utilizzati quando è necessario memorizzare una collezione di elementi unici e non ci interessa l'ordine in cui sono memorizzati. Possono essere utilizzati per effettuare operazioni insiemistiche come unione, intersezione, differenza e differenza simmetrica.

In questa sezione, esploreremo in dettaglio le caratteristiche degli insiemi, come crearli, aggiungere o rimuovere elementi, eseguire operazioni insiemistiche e utilizzarli efficacemente nei programmi Python.

9.2 Creazione di un insieme con elementi

[65]:
```
my_set = {1, 2, 3, 4, 5}
print(my_set)
```

```
{1, 2, 3, 4, 5}
```

9.3 Aggiunta di un elemento a un insieme

[66]:
```
my_set = {1, 2, 3}
my_set.add(4)
print(my_set)
```

```
{1, 2, 3, 4}
```

9.4 Rimozione di un elemento da un insieme

[67]:
```
my_set = {1, 2, 3, 4}
my_set.remove(3)
print(my_set)
```

```
{1, 2, 4}
```

9.5 Verifica se un elemento è presente in un insieme

[68]:
```
my_set = {1, 2, 3, 4, 5}
print(3 in my_set)
```

```
True
```

9.6 Unione di due insiemi

[69]:
```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
union_set = set1.union(set2)
print(union_set)
```

```
{1, 2, 3, 4, 5}
```

9.7 Intersezione di due insiemi

[70]:
```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
intersection_set = set1.intersection(set2)
print(intersection_set)
```

```
{3}
```

9.8 Differenza di due insiemi

[71]:
```
set1 = {1, 2, 3, 4}
set2 = {3, 4, 5}
difference_set = set1.difference(set2)
print(difference_set)
```

```
{1, 2}
```

9.9 Verifica se un insieme è un sottoinsieme di un altro

[72]:
```
set1 = {1, 2, 3}
set2 = {1, 2, 3, 4, 5}
print(set1.issubset(set2))
```

```
True
```

9.10 Controllo se due insiemi sono disgiunti (non hanno elementi in comune)

[73]:
```
set1 = {1, 2, 3}
set2 = {4, 5, 6}
disjoint_check = set1.isdisjoint(set2)
print(disjoint_check)
```

```
True
```

[74]:
```
import random

def genera_insiemi():
    set1 = set(random.sample(range(10), 5))
    set2 = set(random.sample(range(10), 5))
    return set1, set2

def gioca():
    insieme1, insieme2 = genera_insiemi()

    print("Benvenuto a Trova l'Elemento Diverso!")
    print("Il computer ha generato casualmente due insiemi di numeri compresi␣
 ↪tra 0 e 9.")
    print("Uno di questi insiemi contiene un elemento diverso dall'altro.")
    print("Devi indovinare quale.")

    elemento_diverso = random.choice(list(insieme1.
 ↪symmetric_difference(insieme2)))
    while True:
        try:
            guess = input("\nIndovina l'elemento diverso (compreso tra 0 e 9):␣
 ↪")
            if guess.lower() == 'fine':
                print("Grazie per aver giocato!")
                break
            guess = int(guess)
            if guess == elemento_diverso:
                print("\nCongratulazioni! Hai indovinato l'elemento diverso.")
                break
            else:
                print("Sbagliato! Prova di nuovo.")
```

```
        except ValueError:
            print("Inserisci un numero intero compreso tra 0 e 9 o 'fine' per␣
 ↪terminare.")

gioca()
```

```
Benvenuto a Trova l'Elemento Diverso!
Il computer ha generato casualmente due insiemi di numeri compresi tra 0 e 9.
Uno di questi insiemi contiene un elemento diverso dall'altro.
Devi indovinare quale.

Indovina l'elemento diverso (compreso tra 0 e 9): 2
Sbagliato! Prova di nuovo.

Indovina l'elemento diverso (compreso tra 0 e 9): 0
Sbagliato! Prova di nuovo.

Indovina l'elemento diverso (compreso tra 0 e 9): 3
Sbagliato! Prova di nuovo.

Indovina l'elemento diverso (compreso tra 0 e 9): 7
Sbagliato! Prova di nuovo.

Indovina l'elemento diverso (compreso tra 0 e 9): 5

Congratulazioni! Hai indovinato l'elemento diverso.
```

10 None

11 None in Python

In Python, None è un tipo di dato speciale che rappresenta l'assenza di un valore o un valore nullo. Non è la stessa cosa di False, 0 o una stringa vuota. None è unico nel suo genere e ha un significato specifico di "nullità".

11.1 Cosa sono e a cosa servono?

- **Rappresentazione dell'assenza di valore:** None viene utilizzato per indicare che una variabile o un oggetto non ha un valore assegnato o che una funzione non restituisce esplicitamente un valore.
- **Valore di default:** Spesso, None viene utilizzato come valore di default per i parametri di funzione o come valore iniziale per variabili che potrebbero non avere un valore immediatamente.
- **Controllo di flusso:** None può essere utilizzato per controllare il flusso del programma, ad esempio in condizioni if per verificare se una variabile è None prima di procedere con ulteriori operazioni.

- **Terminazione di loop:** None può essere utilizzato come segnale di uscita per terminare cicli o iterazioni in determinate condizioni.

11.2 Perché li utilizziamo?

L'utilizzo di None aiuta a rendere il codice più esplicito e chiaro riguardo alla mancanza di valore o all'assenza di una situazione definita. È utile quando è necessario distinguere tra la presenza di un valore effettivo e l'assenza di un valore, evitando ambiguità nel codice.

Ad esempio, quando una funzione non può restituire un valore specifico in determinate circostanze, restituirà None per indicare l'assenza di un valore anziché lasciare la variabile non definita o restituire un valore "falso".

In sintesi, None è un'importante convenzione nel linguaggio Python per gestire situazioni in cui un valore non è presente o non è definito, contribuendo alla chiarezza e alla coerenza del codice.

11.3 Assegnazione di None a una variabile

[75]:
```
x = None
print(x)
```

```
None
```

11.4 Restituzione di None da una funzione

[76]:
```
def funzione_senza_valore():
    print("Questa funzione non restituisce alcun valore.")

result = funzione_senza_valore()
print(result)
```

```
Questa funzione non restituisce alcun valore.
None
```

11.5 Controllo se una variabile è None

[77]:
```
x = None
if x is None:
    print("La variabile x è None.")
else:
    print("La variabile x non è None.")
```

```
La variabile x è None.
```

11.6 Utilizzo di None come valore di default per un parametro di funzione

[78]:
```
def saluta(nome=None):
    if nome is None:
        print("Ciao!")
    else:
```

```
        print("Ciao,", nome)

saluta()
saluta("Alice")
```

```
Ciao!
Ciao, Alice
```

11.7 Assegnazione di None a un elemento di una lista per rimuoverlo

[79]:
```
lista = [1, 2, 3]
lista[1] = None
print("Lista con elemento rimosso:", lista)
```

```
Lista con elemento rimosso: [1, None, 3]
```

11.8 Utilizzo di None per indicare un valore mancante in un dizionario

[80]:
```
info_persona = {
    'nome': 'Alice',
    'età': None,
    'sesso': 'Femmina'
}
print(info_persona)
```

```
{'nome': 'Alice', 'età': None, 'sesso': 'Femmina'}
```

11.9 Restituzione di None se la lista è vuota

[81]:
```
def primo_elemento(lista):
    if lista:
        return lista[0]
    else:
        return None

lista_vuota = []
print(primo_elemento(lista_vuota))  # Output: None
```

```
None
```

11.10 Utilizzo di None per indicare l'assenza di valore in un'operazione

[82]:
```
x = 5
y = None

if y is not None:
    z = x + y
    print(z)
else:
```

```
    print("Impossibile eseguire l'operazione perché y è None.")
```

```
Impossibile eseguire l'operazione perché y è None.
```

11.11 Ricerca di un elemento in una lista e restituzione della sua posizione o None se non trovato

[83]:
```
def trova_posizione(oggetto, lista):
    for i, elem in enumerate(lista):
        if elem == oggetto:
            return i
    return None

numeri = [10, 20, 30, 40, 50]
posizione = trova_posizione(30, numeri)
print("Posizione dell'elemento:", posizione)
```

```
Posizione dell'elemento: 2
```

11.12 La ricerca del tesoro perduto

[84]:
```
import random

def genera_posizione_tesoro():
    x = random.randint(0, 9)
    y = random.randint(0, 9)
    return [x, y]

def calcola_distanza(posizione, tesoro):
    distanza_x = abs(tesoro[0] - posizione[0])
    distanza_y = abs(tesoro[1] - posizione[1])
    return distanza_x + distanza_y

def trova_direzione(movimenti, tesoro):
    direzioni = {
        'nord': (0, 1),
        'est': (1, 0),
        'sud': (0, -1),
        'ovest': (-1, 0),
        'fine': (0, 0)
    }

    posizione = [0, 0]

    for movimento in movimenti:
        movimento = movimento.lower()  # Converti l'input in minuscolo
        if movimento in direzioni:
            delta_x, delta_y = direzioni[movimento]
```

```
            posizione[0] += delta_x
            posizione[1] += delta_y

            distanza = calcola_distanza(posizione, tesoro)
            if distanza == 0:
                print("Congratulazioni! Hai trovato il tesoro!")
                return
            elif distanza == 1:
                print("Fuoco fuochino fuocobene! Sei vicino al tesoro.")
            elif distanza == 2:
                print("Fuoco! Sei abbastanza vicino al tesoro.")
            elif distanza <= 5:
                print("Sei caldo! Il tesoro non è lontano.")
            elif distanza <= 8:
                print("Acqua! Sei un po' lontano dal tesoro.")
            else:
                print("Sei freddo. Il tesoro è molto lontano.")

        else:
            print("Movimento non valido. Inserisci una direzione valida (Nord,␣
 ↪Est, Sud, Ovest) oppure 'Fine' per terminare.")

    print("Il tesoro è rimasto nascosto. Meglio fortuna la prossima volta!")

def main():
    print("Benvenuto a La Ricerca del Tesoro Perduto!")
    print("Trova il tesoro navigando attraverso l'isola misteriosa.")
    print("Per muoverti, inserisci una direzione (Nord, Est, Sud, Ovest) oppure␣
 ↪'Fine' per terminare.")

    tesoro = genera_posizione_tesoro()
    print("Il tesoro è nascosto in una posizione casuale sull'isola. Inizia a␣
 ↪cercare!")

    movimenti = []
    while True:
        movimento = input("Inserisci la tua prossima mossa: ")
        if movimento.lower() == 'fine':
            print("Grazie per aver giocato!")
            break
        movimenti.append(movimento)
        trova_direzione(movimenti, tesoro)

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

```
Benvenuto a La Ricerca del Tesoro Perduto!
```

```
Trova il tesoro navigando attraverso l'isola misteriosa.
Per muoverti, inserisci una direzione (Nord, Est, Sud, Ovest) oppure 'Fine' per
terminare.
Il tesoro è nascosto in una posizione casuale sull'isola. Inizia a cercare!
Inserisci la tua prossima mossa: sud
Sei freddo. Il tesoro è molto lontano.
Il tesoro è rimasto nascosto. Meglio fortuna la prossima volta!
Inserisci la tua prossima mossa: sud
Sei freddo. Il tesoro è molto lontano.
Sei freddo. Il tesoro è molto lontano.
Il tesoro è rimasto nascosto. Meglio fortuna la prossima volta!
Inserisci la tua prossima mossa: fine
Grazie per aver giocato!
```

12 Booleani

12.1 Introduzione ai Booleani

I booleani sono un tipo di dato fondamentale in programmazione che rappresenta una condizione di verità o falsità. In Python, i booleani sono rappresentati dai valori predefiniti `True` e `False`.

12.1.1 Utilità dei Booleani

I booleani sono estremamente utili perché consentono ai programmatori di gestire il flusso del programma attraverso l'uso di istruzioni condizionali. Le istruzioni condizionali permettono al programma di eseguire determinate azioni solo se una determinata condizione è vera (True), altrimenti può eseguire un'altra azione.

Ad esempio, in un'istruzione `if`, il programma eseguirà un blocco di codice solo se la condizione specificata è vera:

"'python x = 5 y = 10

if x < y: print("x è minore di y") else: print("x non è minore di y")

12.2 Confronto tra stringhe

[14]:
```python
stringa1 = "ciao"
stringa2 = "CIAO"
sono_uguali = stringa1.lower() == stringa2.lower()
print("Le stringhe sono uguali?", sono_uguali)
```

```
Le stringhe sono uguali? True
```

12.3 Controllo della lunghezza di una lista

[13]:
```python
lista = [1, 2, 3, 4, 5]
è_vuota = len(lista) == 0
print("La lista è vuota?", è_vuota)
```

```
La lista è vuota? False
```

12.4 Verifica se un valore è incluso in una lista

[11]:
```
valore = 3
lista = [1, 2, 3, 4, 5]
è_incluso = valore in lista
print("Il valore è incluso nella lista?", è_incluso)
```

```
Il valore è incluso nella lista? True
```

12.5 Controllo di più condizioni

[12]:
```
x = 5
y = 10
condizione = x < y and y > 0
print("La condizione è vera?", condizione)
```

```
La condizione è vera? True
```

12.6 Confronto tra numeri

[10]:
```
a = 10
b = 15
è_maggiore = a > b
print("a è maggiore di b?", è_maggiore)
```

```
a è maggiore di b? False
```

12.7 Confronto tra liste

[9]:
```
lista1 = [1, 2, 3]
lista2 = [1, 2, 3]
sono_uguali = lista1 == lista2
print("Le liste sono uguali?", sono_uguali)
```

```
Le liste sono uguali? True
```

12.8 Controllo della presenza di almeno un elemento vero in una lista

[8]:
```
valori = [False, False, True, False]
almeno_un_true = any(valori)
print("C'è almeno un True nella lista?", almeno_un_true)
```

```
C'è almeno un True nella lista? True
```

12.9 Verifica se una variabile è assegnata

[15]:
```
variabile = None
è_assegnata = variabile is not None
print("La variabile è assegnata?", è_assegnata)
```

```
La variabile è assegnata? False
```

12.10 Verifica se una lista contiene solo valori interi

[16]:
```
lista = [1, 2, 3, "quattro", 5]
solo_interi = all(isinstance(x, int) for x in lista)
print("La lista contiene solo valori interi?", solo_interi)
```

```
La lista contiene solo valori interi? False
```

12.11 Controllo della presenza di almeno un valore negativo in una lista

[17]:
```
valori = [2, 4, -3, 6, 8]
ha_negativi = any(x < 0 for x in valori)
print("La lista contiene almeno un valore negativo?", ha_negativi)
```

```
La lista contiene almeno un valore negativo? True
```

12.12 Confronto tra stringhe

[18]:
```
stringa1 = "python"
stringa2 = "java"
è_maggiore = stringa1 > stringa2
print("La stringa1 è maggiore della stringa2?", è_maggiore)
```

```
La stringa1 è maggiore della stringa2? True
```

12.13 Verifica se una variabile è di tipo stringa

[19]:
```
variabile = "hello"
è_stringa = isinstance(variabile, str)
print("La variabile è di tipo stringa?", è_stringa)
```

```
La variabile è di tipo stringa? True
```

12.14 Indovina il Colore

[21]:
```
import random

def scegli_colore():
    colori = ["rosso", "verde", "blu", "giallo", "arancione", "viola"]
    return random.choice(colori)

def gioca():
```

```
    print("Benvenuto a Indovina il Colore!")
    print("Il computer ha scelto un colore tra rosso, verde, blu, giallo,␣
↪arancione e viola.")

    colore_scelto = scegli_colore()
    while True:
        guess = input("\nIndovina il colore: ").lower()
        if guess not in ["rosso", "verde", "blu", "giallo", "arancione",␣
↪"viola"]:
            print("Inserisci un colore valido tra rosso, verde, blu, giallo,␣
↪arancione e viola.")
            continue

        if guess == colore_scelto:
            print("Complimenti! Hai indovinato il colore.")
            break
        else:
            print("Sbagliato! Prova di nuovo.")

        continua = input("Vuoi continuare a giocare? (sì/no): ").lower()
        if continua != "sì":
            print("Grazie per aver giocato!")
            break

gioca()
```

```
Benvenuto a Indovina il Colore!
Il computer ha scelto un colore tra rosso, verde, blu, giallo, arancione e
viola.

Indovina il colore: verde
Sbagliato! Prova di nuovo.
Vuoi continuare a giocare? (sì/no): sì

Indovina il colore: viola
Sbagliato! Prova di nuovo.
Vuoi continuare a giocare? (sì/no): sì

Indovina il colore:
Inserisci un colore valido tra rosso, verde, blu, giallo, arancione e viola.

Indovina il colore: blu
Sbagliato! Prova di nuovo.
Vuoi continuare a giocare? (sì/no): sì

Indovina il colore: giallo
Sbagliato! Prova di nuovo.
```

```
Vuoi continuare a giocare? (sì/no): sì

Indovina il colore: rosso
Sbagliato! Prova di nuovo.
Vuoi continuare a giocare? (sì/no): no
Grazie per aver giocato!
```

12.15 Calcolo distanza fra 2 punti nel piano cartesiano

[27]:
```python
import math

def calcola_distanza(p1, p2):
    x1, y1 = p1
    x2, y2 = p2
    distanza = math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
    return distanza

def inserisci_coordinate():
    x1 = float(input("Inserisci la coordinata x del primo punto: "))
    y1 = float(input("Inserisci la coordinata y del primo punto: "))
    x2 = float(input("Inserisci la coordinata x del secondo punto: "))
    y2 = float(input("Inserisci la coordinata y del secondo punto: "))
    return (x1, y1), (x2, y2)

punto1, punto2 = inserisci_coordinate()
distanza = calcola_distanza(punto1, punto2)
print("La distanza tra", punto1, "e", punto2, "è:", distanza)
```

```
Inserisci la coordinata x del primo punto: 4.0
Inserisci la coordinata y del primo punto: -7.3
Inserisci la coordinata x del secondo punto: 6.0
Inserisci la coordinata y del secondo punto: 1.0
La distanza tra (4.0, -7.3) e (6.0, 1.0) è: 8.537564055396599
```