



LEZIONE 2

Capitolo 2 – Le stringhe e le strutture di dati complesse

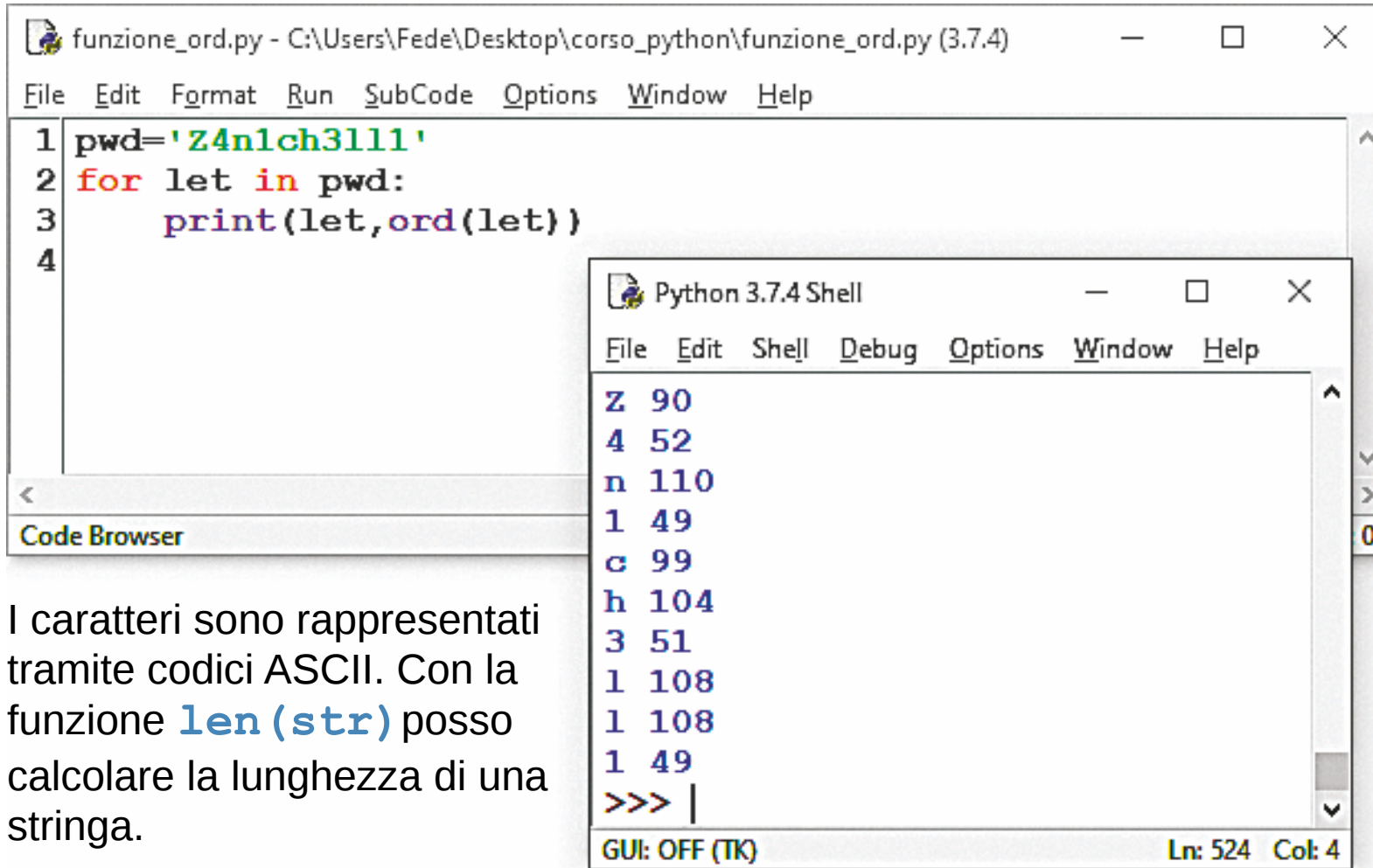
Indice

1. Lavorare con le stringhe
2. L'oggetto stringa e i suoi metodi
3. Le liste e le matrici
4. Tuple, dizionari e set

2.1 Lavorare con le stringhe

Indice

In Python, una stringa è una sequenza di caratteri racchiusa tra apici e ordinata in base a un indice che parte da 0.



The screenshot shows a Python IDE window titled 'funzione_ord.py - C:\Users\Fede\Desktop\corso_python\funzione_ord.py (3.7.4)'. The code in the editor is:

```
1 pwd='Z4n1ch3111'
2 for let in pwd:
3     print(let,ord(let))
4
```

Below the editor is a 'Code Browser' tab. Overlaid on the bottom right is a 'Python 3.7.4 Shell' window showing the output of the script:

```
Z 90
4 52
n 110
1 49
c 99
h 104
3 51
1 108
1 108
1 49
>>> |
```

The shell window also shows 'GUI: OFF (TK)' and 'Ln: 524 Col: 4' at the bottom.

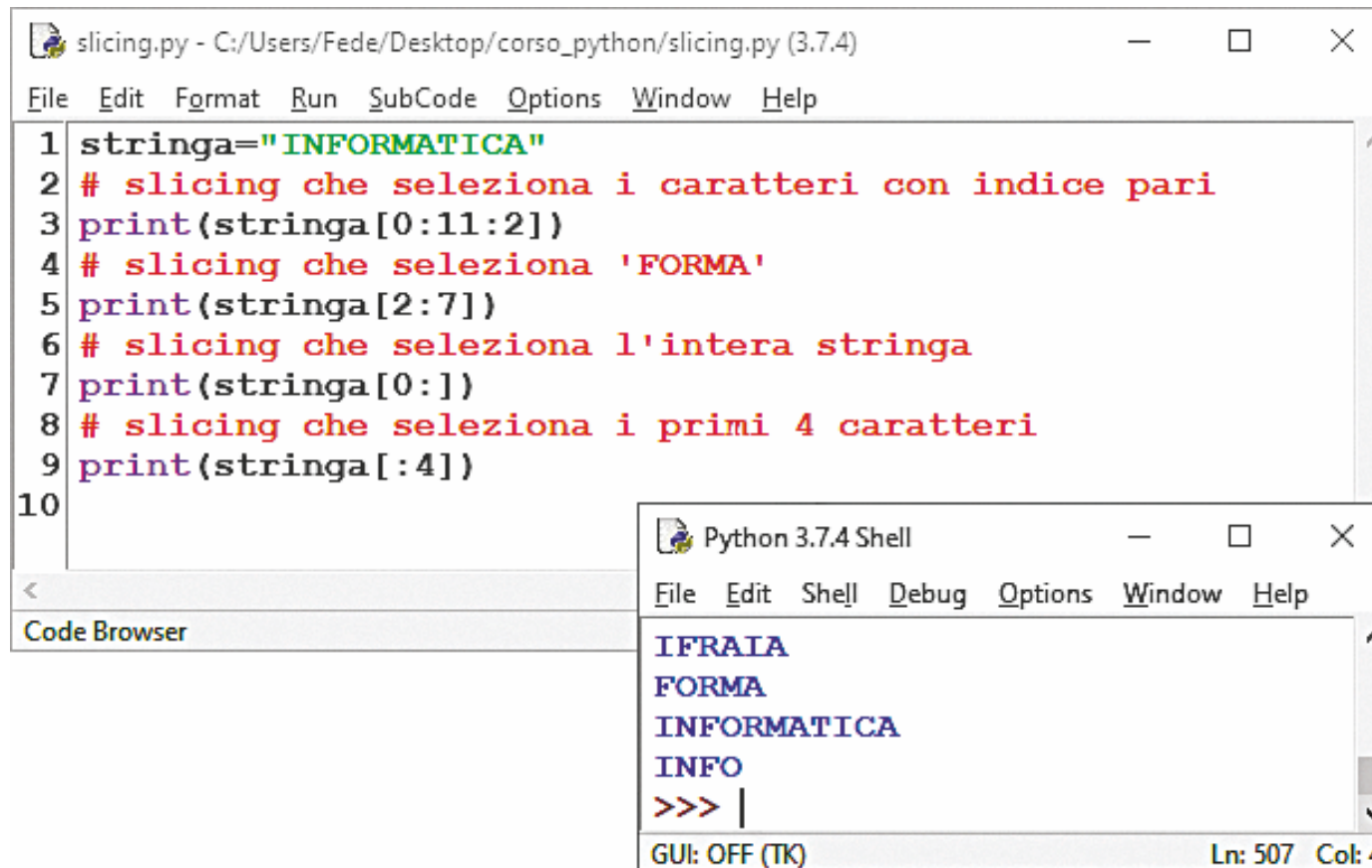
I caratteri sono rappresentati tramite codici ASCII. Con la funzione `len(str)` posso calcolare la lunghezza di una stringa.

2.1 Lavorare con le stringhe

Indice

Lo **slicing** permette di selezionare parti di una stringa.

`nome_stringa[inizio:fine:passo]`



The screenshot shows a Python IDE window titled 'slicing.py - C:/Users/Fede/Desktop/corso_python/slicing.py (3.7.4)'. The code in the editor is as follows:

```
1 stringa="INFORMATICA"
2 # slicing che seleziona i caratteri con indice pari
3 print(stringa[0:11:2])
4 # slicing che seleziona 'FORMA'
5 print(stringa[2:7])
6 # slicing che seleziona l'intera stringa
7 print(stringa[0:])
8 # slicing che seleziona i primi 4 caratteri
9 print(stringa[:4])
10
```

Below the code editor is a 'Code Browser' tab. To the right, a 'Python 3.7.4 Shell' window displays the output of the code:

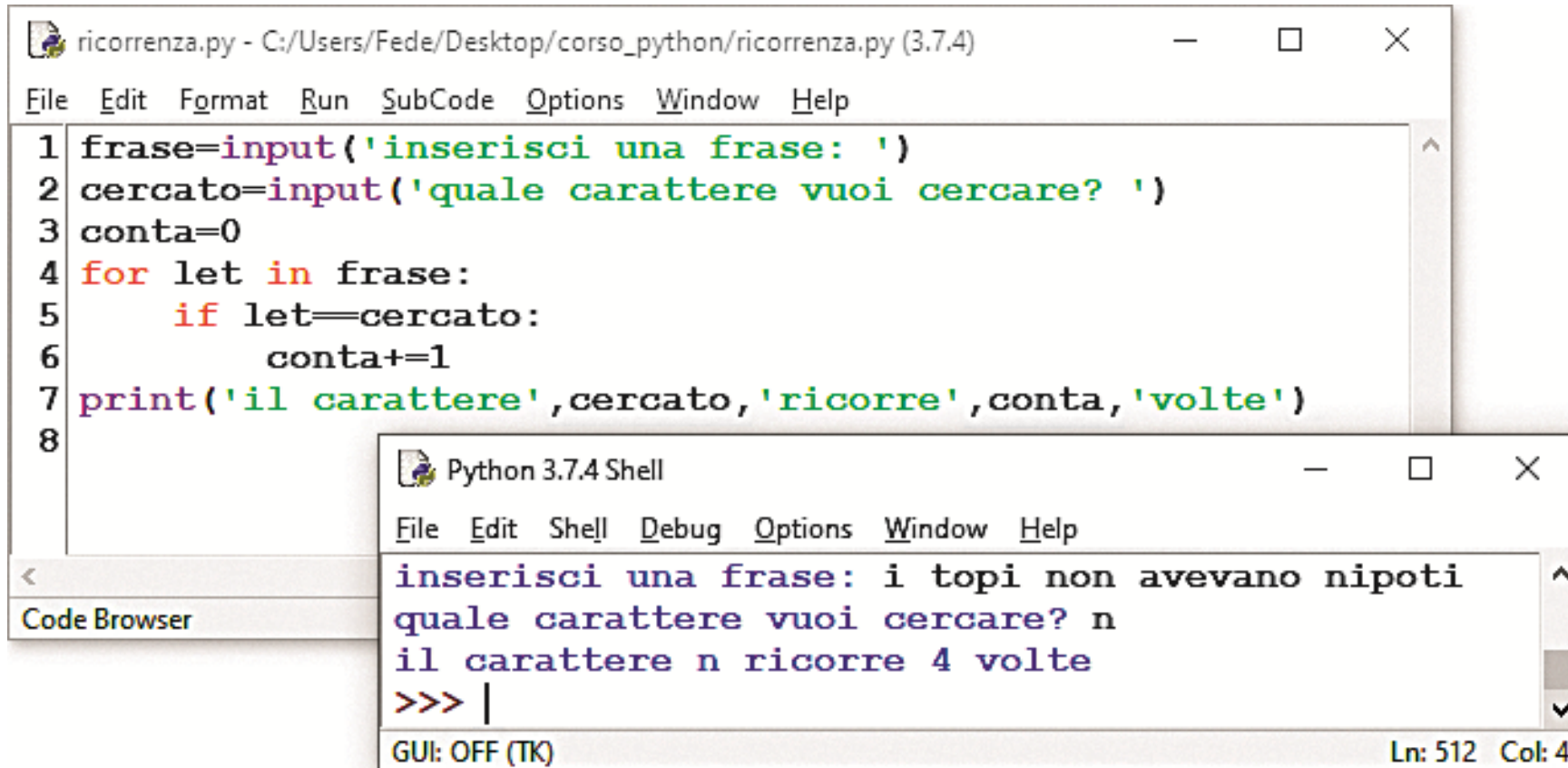
```
IFRAIA
FORMA
INFORMATICA
INFO
>>> |
```

The status bar at the bottom of the shell window indicates 'GUI: OFF (TK)' and 'Ln: 507 Col: 4'.

2.1 Lavorare con le stringhe

Indice

Per cercare ricorrenze o sottostringhe si usa la parola chiave **in**.



The image shows a Python IDE window titled "ricorrenza.py - C:/Users/Fede/Desktop/corso_python/ricorrenza.py (3.7.4)". The code in the editor is as follows:

```
1 frase=input('inserisci una frase: ')
2 cercato=input('quale carattere vuoi cercare? ')
3 conta=0
4 for let in frase:
5     if let==cercato:
6         conta+=1
7 print('il carattere',cercato,'ricorre',conta,'volte')
8
```

Below the code editor is a "Code Browser" tab. Overlaid on the bottom right is a "Python 3.7.4 Shell" window showing the execution of the script:

```
inserisci una frase: i topi non avevano nipoti
quale carattere vuoi cercare? n
il carattere n ricorre 4 volte
>>> |
```

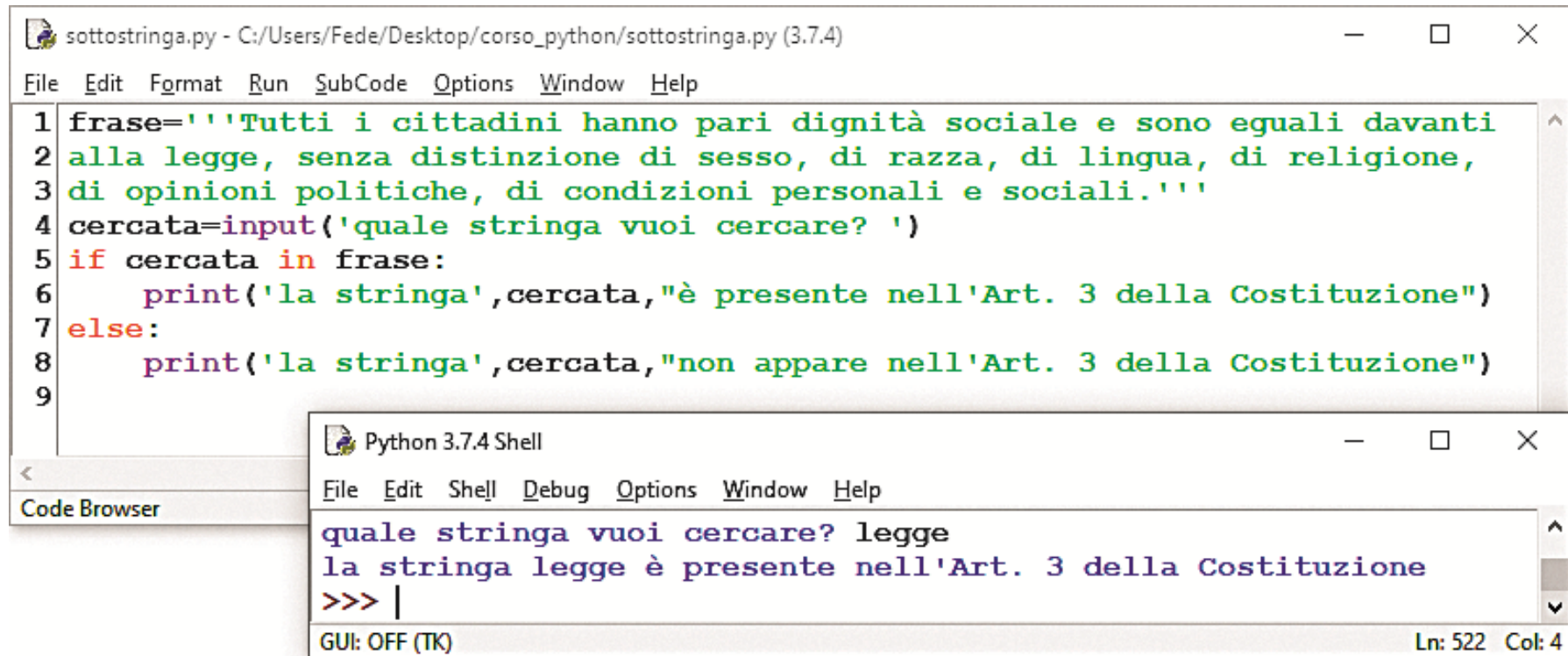
The shell window also displays "GUI: OFF (TK)" at the bottom left and "Ln: 512 Col: 4" at the bottom right.

Ricerca di un carattere

2.1 Lavorare con le stringhe

Indice

Per cercare ricorrenze o sottostringhe si usa la parola chiave **in**.



The screenshot displays a Python IDE with two windows. The top window, titled 'sottostringa.py - C:/Users/Fede/Desktop/corso_python/sottostringa.py (3.7.4)', contains the following Python code:

```
1 frase='''Tutti i cittadini hanno pari dignità sociale e sono eguali davanti
2 alla legge, senza distinzione di sesso, di razza, di lingua, di religione,
3 di opinioni politiche, di condizioni personali e sociali.'''
4 cercata=input('quale stringa vuoi cercare? ')
5 if cercata in frase:
6     print('la stringa',cercata,"è presente nell'Art. 3 della Costituzione")
7 else:
8     print('la stringa',cercata,"non appare nell'Art. 3 della Costituzione")
9
```

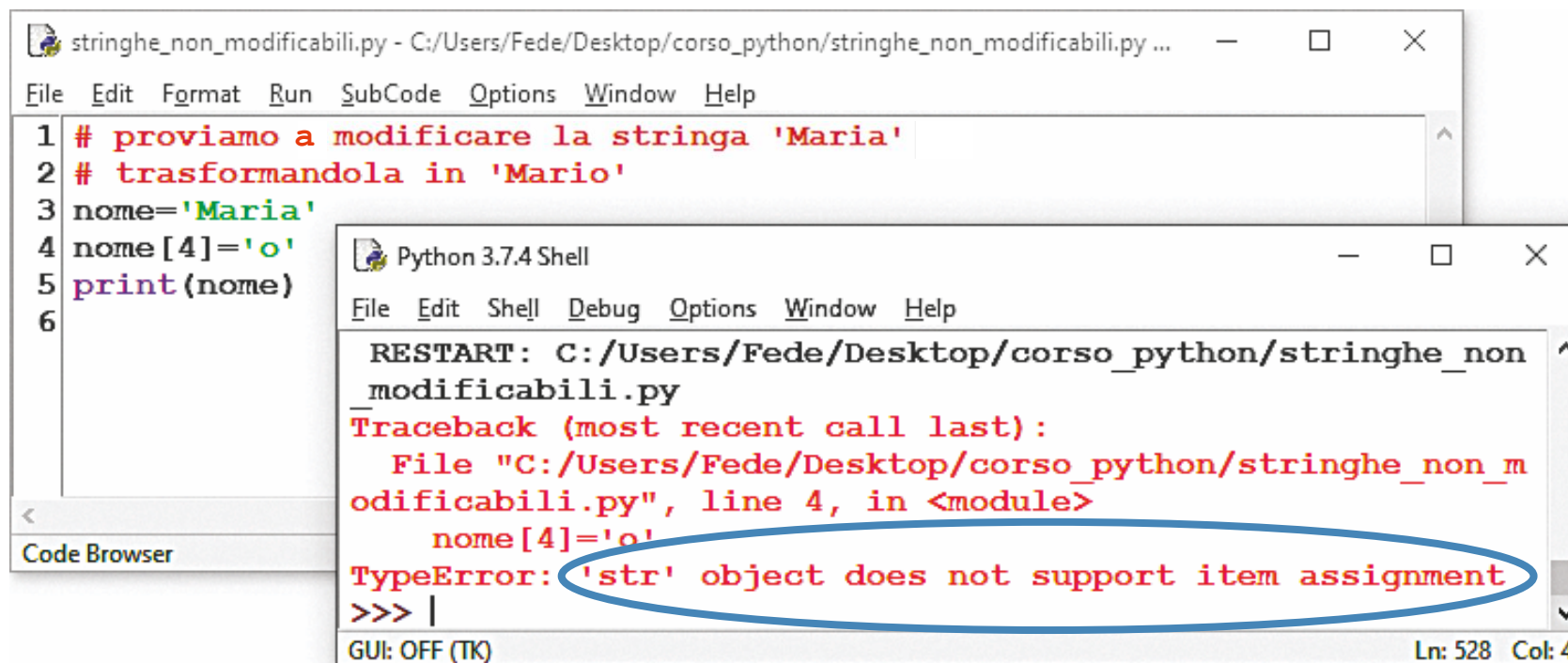
The bottom window, titled 'Python 3.7.4 Shell', shows the execution of the script. It displays the prompt 'quale stringa vuoi cercare?' followed by the user input 'legge'. The output is 'la stringa legge è presente nell'Art. 3 della Costituzione'. The shell also shows the prompt '>>>' and the status 'GUI: OFF (TK)' at the bottom.

Ricerca di una sottostringa

2.1 Lavorare con le stringhe

Indice

Le stringhe in Python **NON** sono modificabili.



```
stringhe_non_modificabili.py - C:/Users/Fede/Desktop/corso_python/stringhe_non_modificabili.py ...
File Edit Format Run SubCode Options Window Help
1 # proviamo a modificare la stringa 'Maria'
2 # trasformandola in 'Mario'
3 nome='Maria'
4 nome[4]='o'
5 print(nome)
6

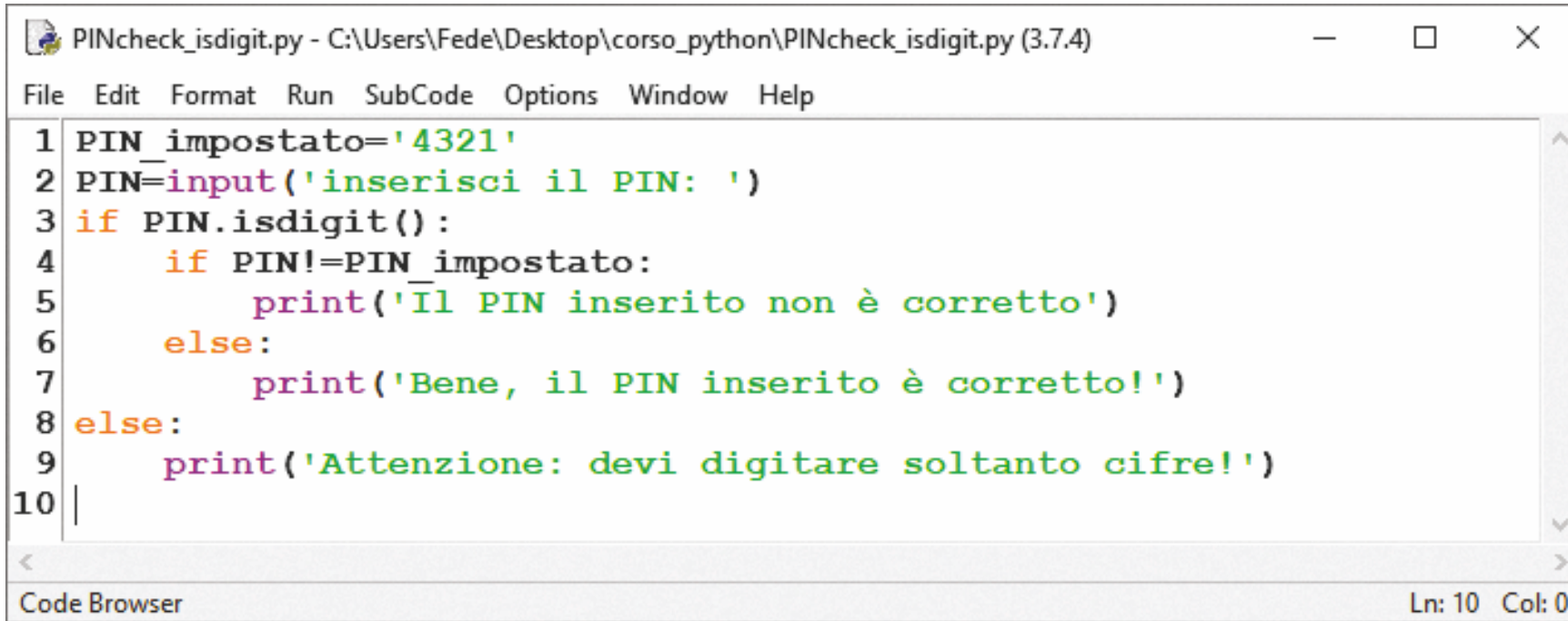
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
RESTART: C:/Users/Fede/Desktop/corso_python/stringhe_non_modificabili.py
Traceback (most recent call last):
  File "C:/Users/Fede/Desktop/corso_python/stringhe_non_modificabili.py", line 4, in <module>
    nome[4]='o'
TypeError: 'str' object does not support item assignment
>>> |
GUI: OFF (TK) Ln: 528 Col: 4
```

Il tentativo di cambiare il contenuto di una stringa fornisce come risultato un errore.

2.2 L'oggetto stringa e i suoi metodi

Indice

In Python le stringhe sono **oggetti** dotati di metodi specifici.



```
PINcheck_isdigit.py - C:\Users\Fede\Desktop\corso_python\PINcheck_isdigit.py (3.7.4)
File Edit Format Run SubCode Options Window Help
1 PIN_impostato='4321'
2 PIN=input('inserisci il PIN: ')
3 if PIN.isdigit():
4     if PIN!=PIN_impostato:
5         print('Il PIN inserito non è corretto')
6     else:
7         print('Bene, il PIN inserito è corretto!')
8 else:
9     print('Attenzione: devi digitare soltanto cifre!')
10 |
```

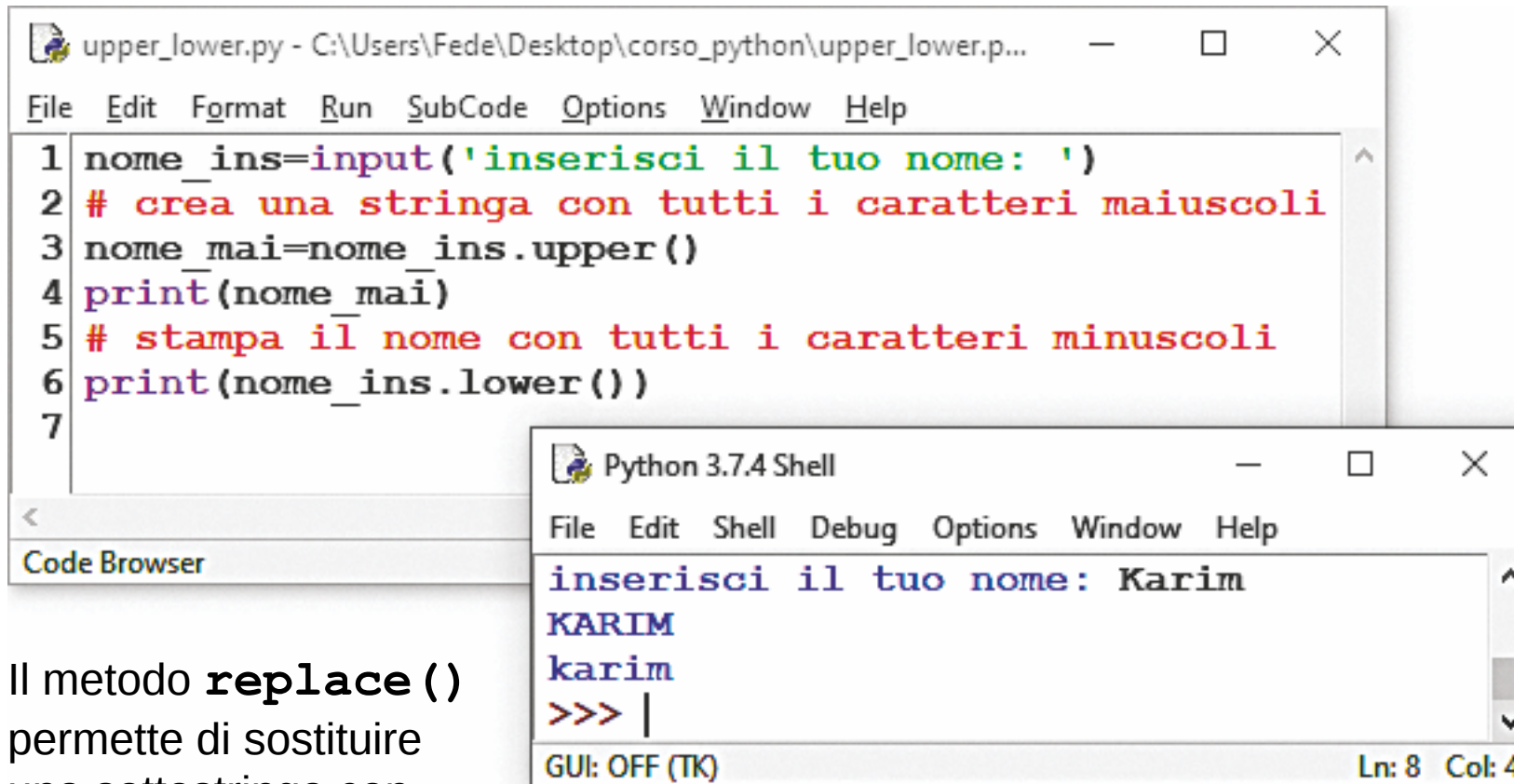
Code Browser Ln: 10 Col: 0

Metodi comuni per il trattamento delle stringhe sono **isalpha()**, **isalnum()**, **islower()**, **isupper()**.

2.2 L'oggetto stringa e i suoi metodi

Indice

Si possono creare stringhe a partire da altre stringhe.



The image shows a screenshot of a Python IDE. The main window, titled 'upper_lower.py', contains the following code:

```
1 nome_ins=input('inserisci il tuo nome: ')
2 # crea una stringa con tutti i caratteri maiuscoli
3 nome_mai=nome_ins.upper()
4 print(nome_mai)
5 # stampa il nome con tutti i caratteri minuscoli
6 print(nome_ins.lower())
7
```

Below the code editor is a 'Code Browser' panel. Overlaid on the bottom right is a 'Python 3.7.4 Shell' window. The shell shows the execution of the script:

```
inserisci il tuo nome: Karim
KARIM
karim
>>> |
```

The shell window also displays 'GUI: OFF (TK)' and the current cursor position 'Ln: 8 Col: 4'.

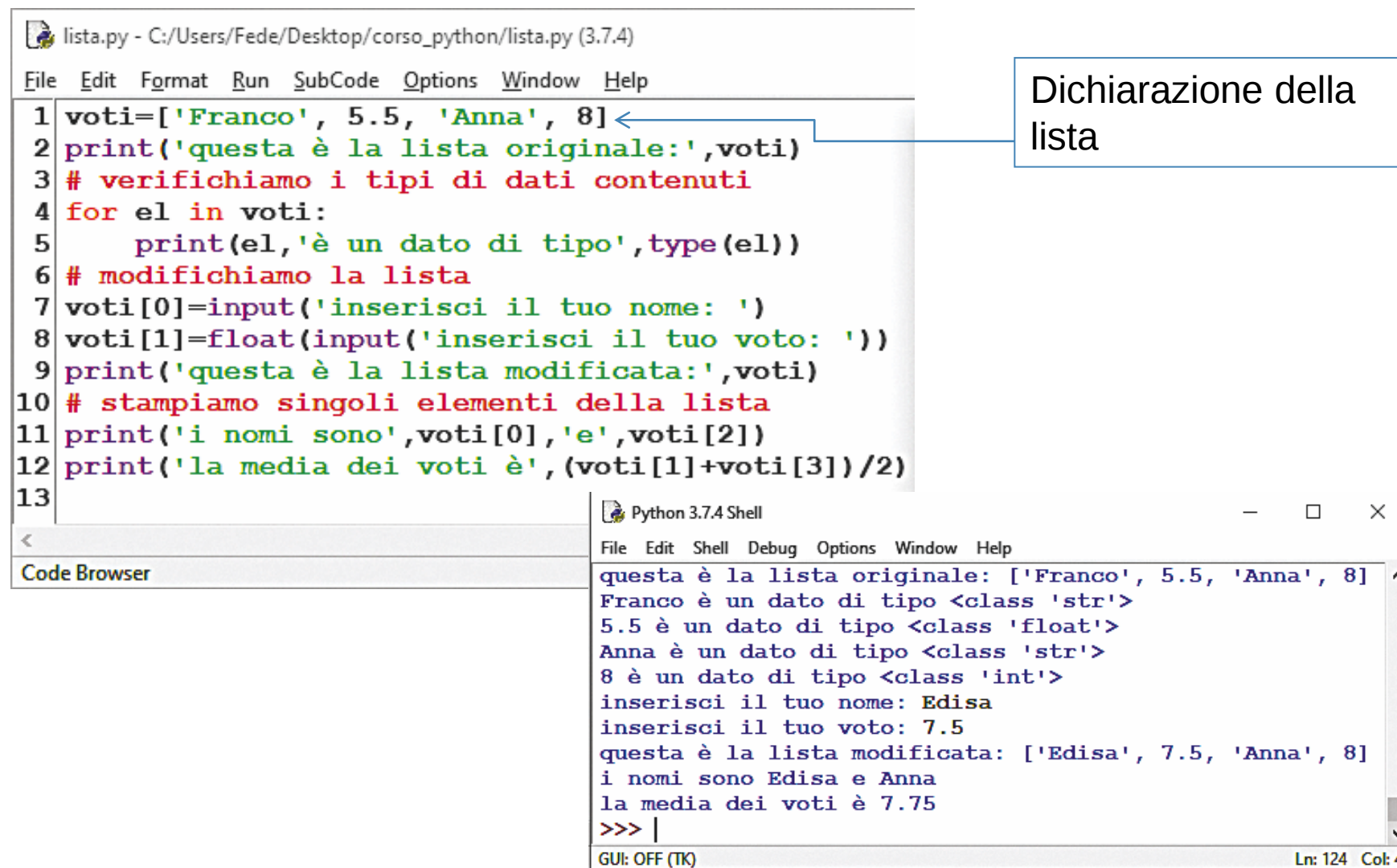
Il metodo **replace()** permette di sostituire una sottostringa con un'altra, anche di lunghezza diversa.

find() e **index()** restituiscono la posizione di una sottostringa, **count()** conta le ricorrenze.

2.3 Le liste e le matrici

Indice

Una lista è un oggetto che contiene una serie di dati, anche di tipo diverso.



The image shows a Python IDE window titled 'lista.py - C:/Users/Fede/Desktop/corso_python/lista.py (3.7.4)'. The code in the editor is as follows:

```
1 voti=['Franco', 5.5, 'Anna', 8]
2 print('questa è la lista originale:',voti)
3 # verifichiamo i tipi di dati contenuti
4 for el in voti:
5     print(el, 'è un dato di tipo', type(el))
6 # modifichiamo la lista
7 voti[0]=input('inserisci il tuo nome: ')
8 voti[1]=float(input('inserisci il tuo voto: '))
9 print('questa è la lista modificata:',voti)
10 # stampiamo singoli elementi della lista
11 print('i nomi sono',voti[0], 'e',voti[2])
12 print('la media dei voti è', (voti[1]+voti[3])/2)
13
```

A blue box with the text 'Dichiarazione della lista' has an arrow pointing to line 1 of the code.

Below the code editor is a 'Code Browser' tab. To the right is a 'Python 3.7.4 Shell' window showing the execution output:

```
questa è la lista originale: ['Franco', 5.5, 'Anna', 8]
Franco è un dato di tipo <class 'str'>
5.5 è un dato di tipo <class 'float'>
Anna è un dato di tipo <class 'str'>
8 è un dato di tipo <class 'int'>
inserisci il tuo nome: Edisa
inserisci il tuo voto: 7.5
questa è la lista modificata: ['Edisa', 7.5, 'Anna', 8]
i nomi sono Edisa e Anna
la media dei voti è 7.75
>>> |
```

The status bar at the bottom of the shell window shows 'GUI: OFF (TK)' and 'Ln: 124 Col: 4'.

2.3 Le liste e le matrici

Indice

Esistono diversi modi di creare una lista.

```
voti_str='Franco 5.5 Anna 8':
```

```
voti=voti_str.split() ➔ voti=['Franco','5.5','Anna','8']
```

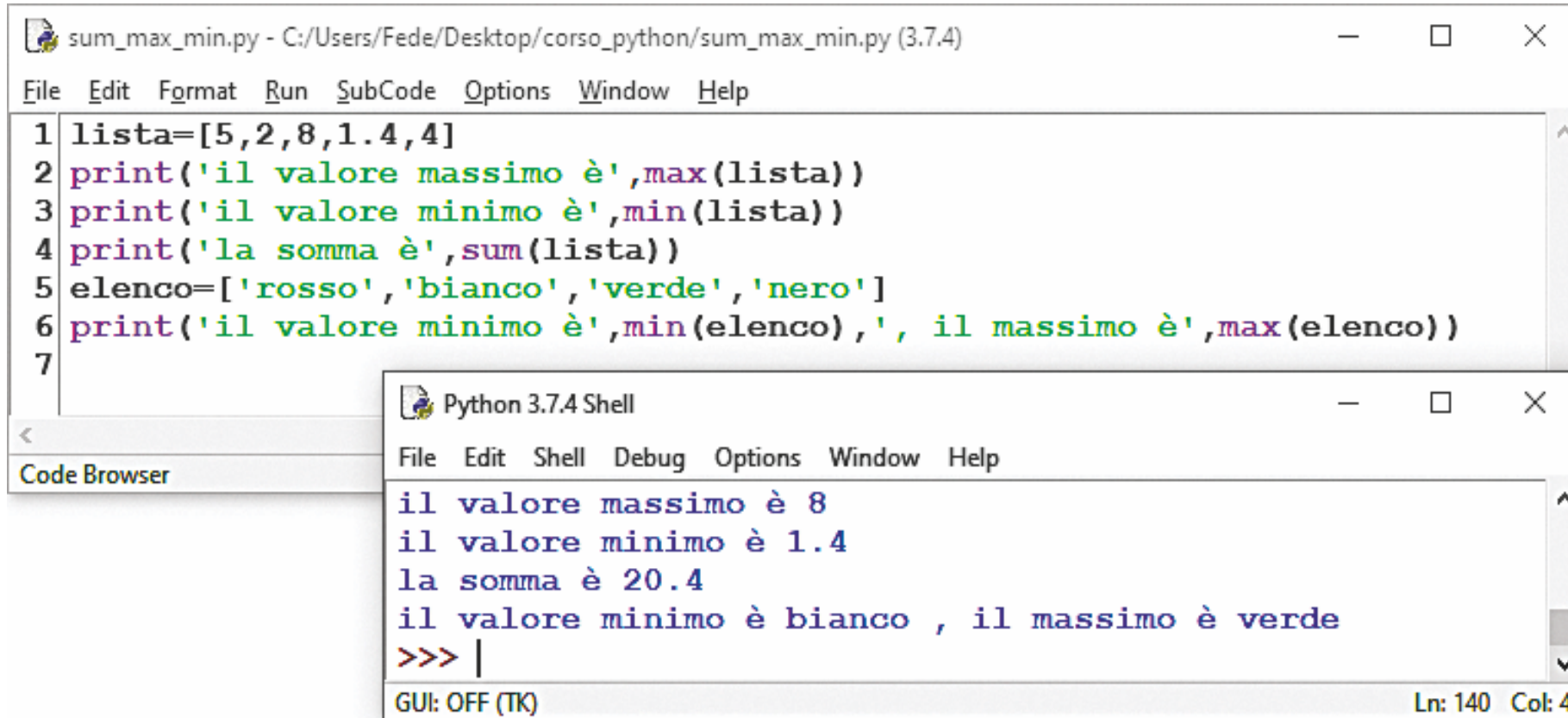
```
mia_lista=['nero']*3 ➔ ['nero','nero','nero'].
```

```
mia_lista=list(range(5))      Numeri tra 0 e 4  
mia_lista=list(range(5,51,5)) Tabellina del 5
```

2.3 Le liste e le matrici

Indice

Alle liste posso applicare le funzioni built-in.



The image shows a screenshot of a Python IDE with two windows. The top window, titled 'sum_max_min.py - C:/Users/Fede/Desktop/corso_python/sum_max_min.py (3.7.4)', contains the following Python code:

```
1 lista=[5,2,8,1.4,4]
2 print('il valore massimo è',max(lista))
3 print('il valore minimo è',min(lista))
4 print('la somma è',sum(lista))
5 elenco=['rosso','bianco','verde','nero']
6 print('il valore minimo è',min(elenco),', il massimo è',max(elenco))
7
```

The bottom window, titled 'Python 3.7.4 Shell', shows the output of the script:

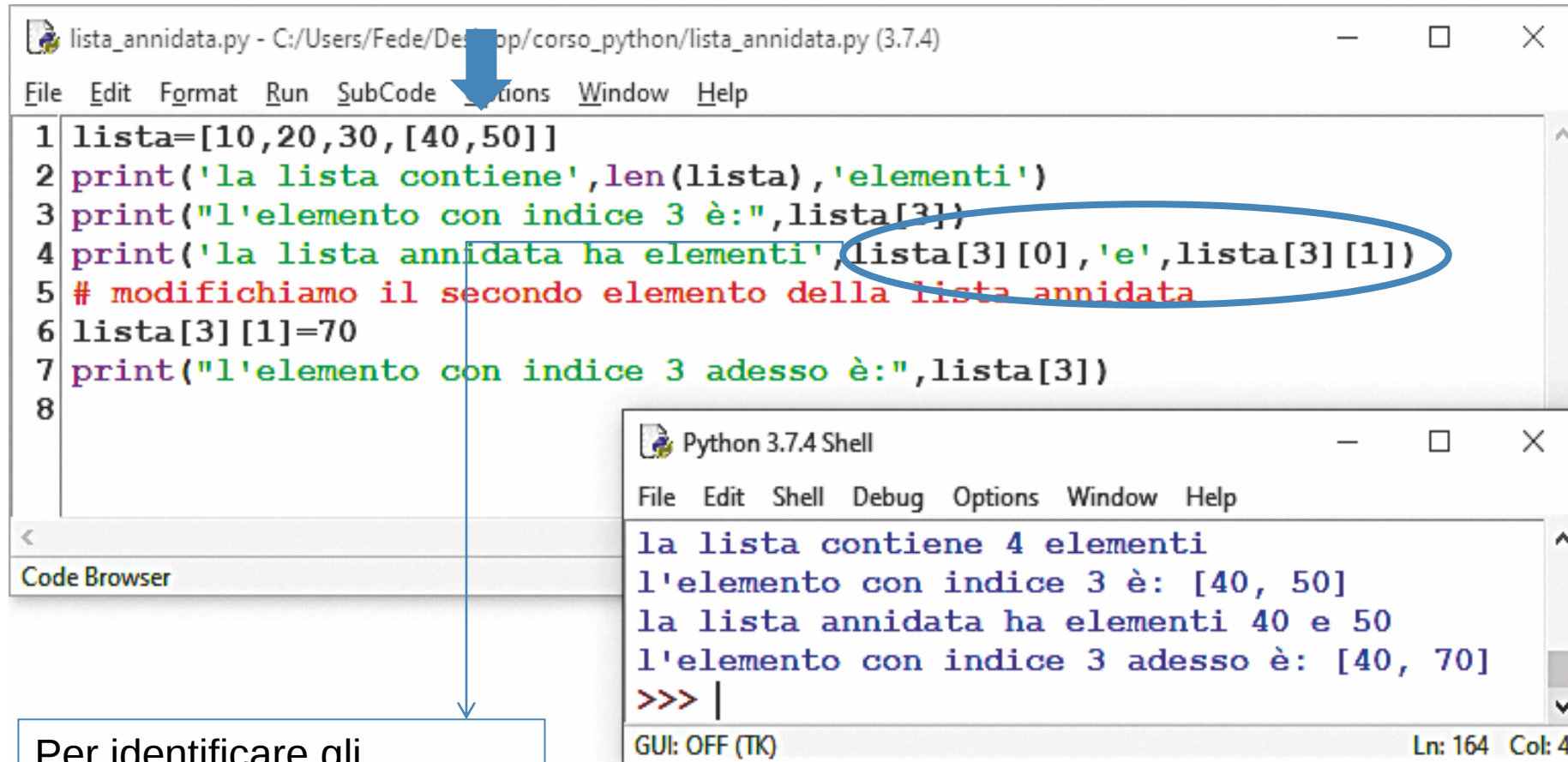
```
il valore massimo è 8
il valore minimo è 1.4
la somma è 20.4
il valore minimo è bianco , il massimo è verde
>>> |
```

The status bar at the bottom of the shell window indicates 'GUI: OFF (TK)' and 'Ln: 140 Col: 4'.

2.3 Le liste e le matrici

Indice

Una lista **annidata** è una lista contenuta come elemento in un'altra lista.



The image shows a Python IDE window titled 'lista_annidata.py - C:/Users/Fede/Desktop/corso_python/lista_annidata.py (3.7.4)'. The code in the editor is as follows:

```
1 lista=[10,20,30,[40,50]]
2 print('la lista contiene',len(lista),'elementi')
3 print("l'elemento con indice 3 è:",lista[3])
4 print('la lista annidata ha elementi',lista[3][0], 'e', lista[3][1])
5 # modifichiamo il secondo elemento della lista annidata
6 lista[3][1]=70
7 print("l'elemento con indice 3 adesso è:",lista[3])
8
```

A blue arrow points from the first parameter '10' in the list initialization to a text box at the bottom. Another blue arrow points from the expression 'lista[3][0]' in line 4 to the same text box. A third blue arrow points from the expression 'lista[3][1]' in line 6 to the same text box. A blue oval highlights the expression 'lista[3][0], 'e', lista[3][1]' in line 4.

Below the code editor, a text box contains the text: "Per identificare gli elementi si usano 2 indici".

To the right of the code editor is a 'Python 3.7.4 Shell' window showing the output of the script:

```
la lista contiene 4 elementi
l'elemento con indice 3 è: [40, 50]
la lista annidata ha elementi 40 e 50
l'elemento con indice 3 adesso è: [40, 70]
>>> |
```

The shell window also shows 'GUI: OFF (TK)' and 'Ln: 164 Col: 4' at the bottom.

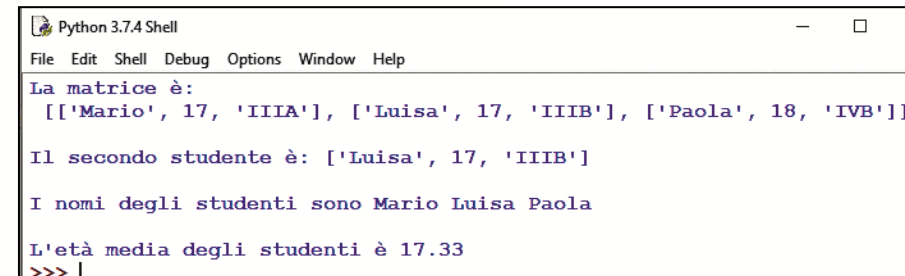
Per identificare gli
elementi si usano 2 indici

2.3 Le liste e le matrici

Indice

Una lista che contiene solo liste tutte della stessa lunghezza è detta **matrice**.

```
# definiamo la matrice studenti
studenti=[['Mario',17,'IIIA'], ['Luisa',17,'IIIB'], ['Paola', 18, 'IVB']]
indice1,elem=0,0
# stampiamo l'intera matrice
print('La matrice è:\n',studenti,'\n')
# stampiamo i dati relativi a un singolo studente
print('Il secondo studente è:',studenti[1],'\n')
# stampiamo i nomi di tutti gli studenti su una sola riga
print("I nomi degli studenti sono",end=' ')
while elem<len(studenti):
    print(studenti[indice1][0],end=' ')
    indice1+=1
    elem+=1
print('\n')
# stampiamo l'età media degli studenti
indice1,anni,elem=0,0,0
while elem<len(studenti):
    anni+=studenti[indice1][1]
    indice1+=1
    elem+=1
print("L'età media degli studenti è",format(anni/len(studenti),'.2f'))
```



```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
La matrice è:
[['Mario', 17, 'IIIA'], ['Luisa', 17, 'IIIB'], ['Paola', 18, 'IVB']]

Il secondo studente è: ['Luisa', 17, 'IIIB']

I nomi degli studenti sono Mario Luisa Paola

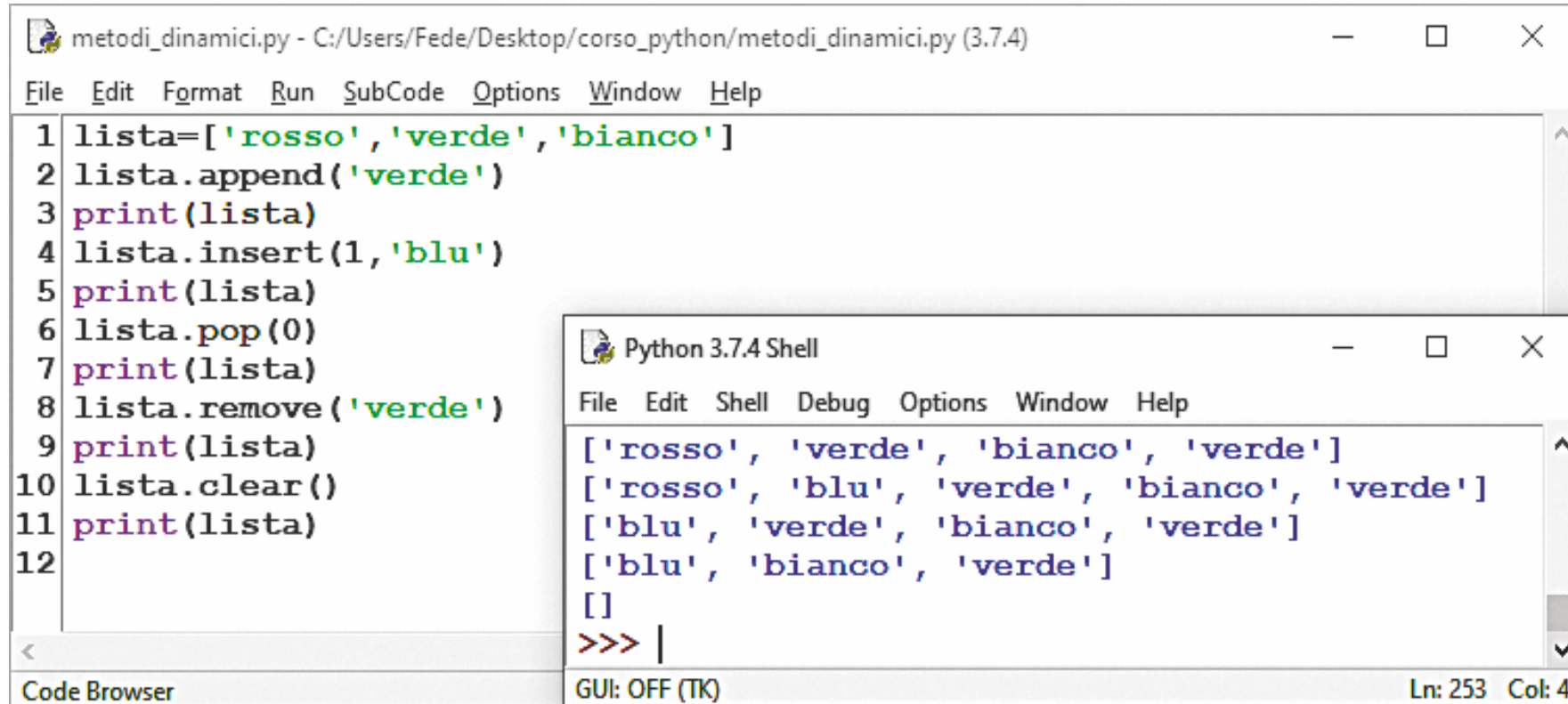
L'età media degli studenti è 17.33
>>> |
```

I dati sono trattati come se fossero contenuti in una tabella

2.3 Le liste e le matrici

Indice

Le liste sono **oggetti** che possono essere elaborate usando appositi metodi.



The screenshot displays a Python IDE window titled 'metodi_dinamici.py - C:/Users/Fede/Desktop/corso_python/metodi_dinamici.py (3.7.4)'. The code editor contains the following Python code:

```
1 lista=['rosso', 'verde', 'bianco']
2 lista.append('verde')
3 print(lista)
4 lista.insert(1, 'blu')
5 print(lista)
6 lista.pop(0)
7 print(lista)
8 lista.remove('verde')
9 print(lista)
10 lista.clear()
11 print(lista)
12
```

Below the code editor is a 'Python 3.7.4 Shell' window showing the output of the script:

```
['rosso', 'verde', 'bianco', 'verde']
['rosso', 'blu', 'verde', 'bianco', 'verde']
['blu', 'verde', 'bianco', 'verde']
['blu', 'bianco', 'verde']
[]
>>> |
```

The status bar at the bottom of the IDE shows 'Code Browser', 'GUI: OFF (TK)', and 'Ln: 253 Col: 4'.

Con i metodi **reverse()** e **sort()** si possono rispettivamente rovesciare e ordinare una lista.

2.4 Tuple, dizionari e set

Indice

Una tupla contiene una sequenza di dati ordinati in base a un indice, e **NON** può essere modificata.

Si può creare scrivendo:

```
tp='alfa',37,'tre'
```

O alternativamente:

```
tp=('alfa',37,'tre')
```

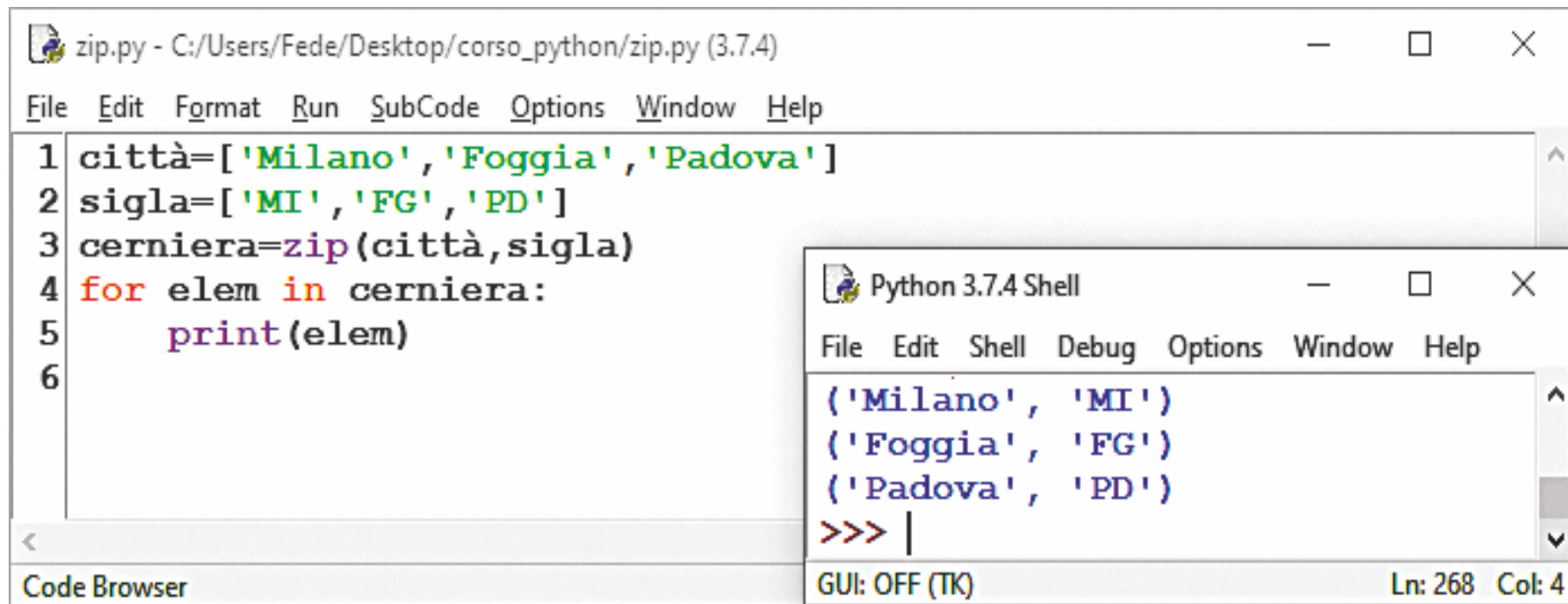
La funzione **tuple()** converte liste, stringhe o altri oggetti iterabili in tuple.

```
tp=tuple('abc')
```

2.4 Tuple, dizionari e set

Indice

La funzione `zip()` accetta come argomenti oggetti iterabili e restituisce gli elementi corrispondenti associati sotto forma di tuple.



The image shows a screenshot of a Python IDE window titled "zip.py - C:/Users/Fede/Desktop/corso_python/zip.py (3.7.4)". The code in the editor is as follows:

```
1 città=['Milano', 'Foggia', 'Padova']
2 sigla=['MI', 'FG', 'PD']
3 cerniera=zip(città, sigla)
4 for elem in cerniera:
5     print(elem)
6
```

Below the code editor, a "Python 3.7.4 Shell" window displays the output of the script:

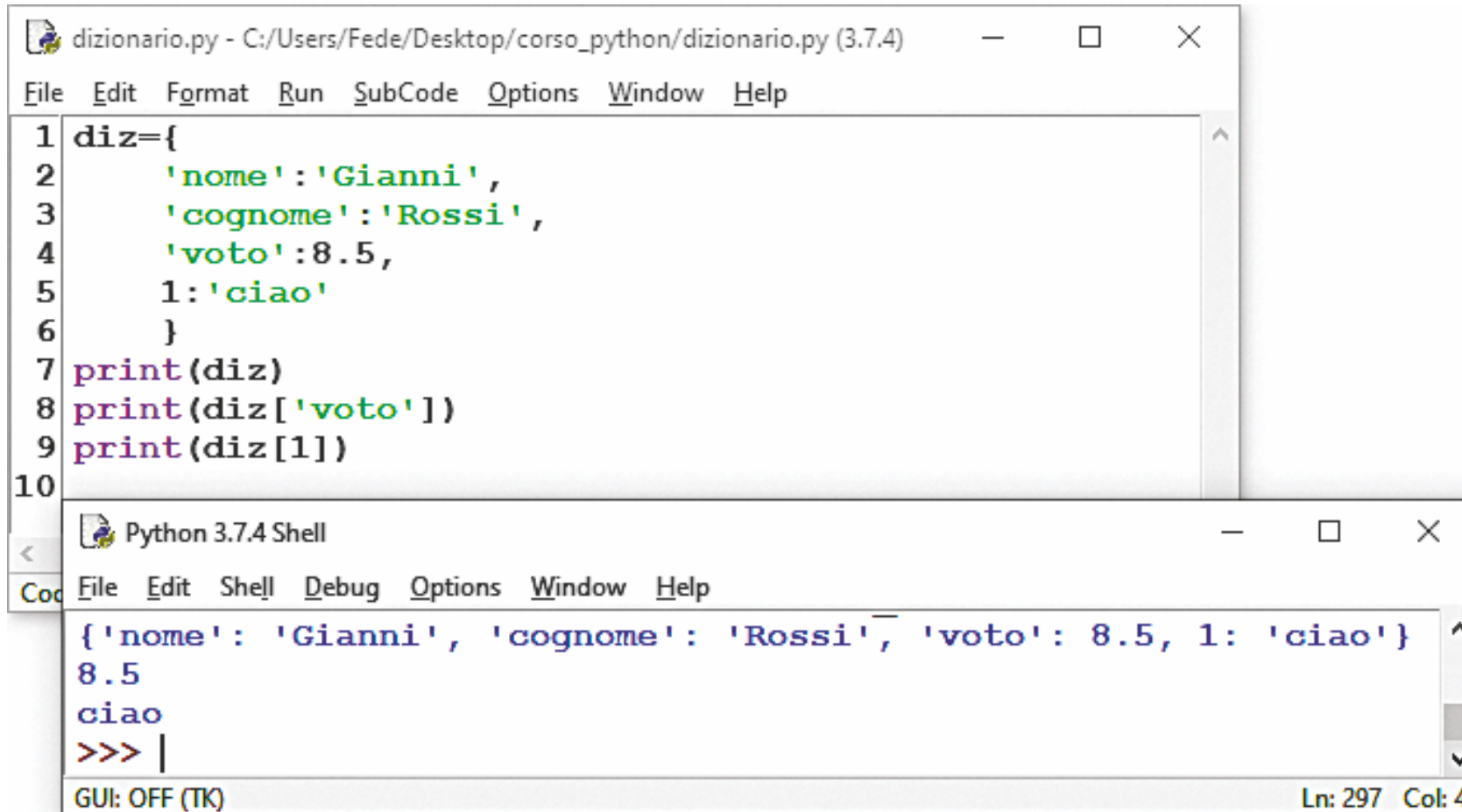
```
('Milano', 'MI')
('Foggia', 'FG')
('Padova', 'PD')
>>> |
```

The status bar at the bottom of the IDE shows "Code Browser" on the left and "GUI: OFF (TK) Ln: 268 Col: 4" on the right.

2.4 Tuple, dizionari e set

[Indice](#)

Un dizionario è un oggetto modificabile che contiene dati di vario tipo non ordinati ma indicizzati da **chiavi** scelte dal programmatore.



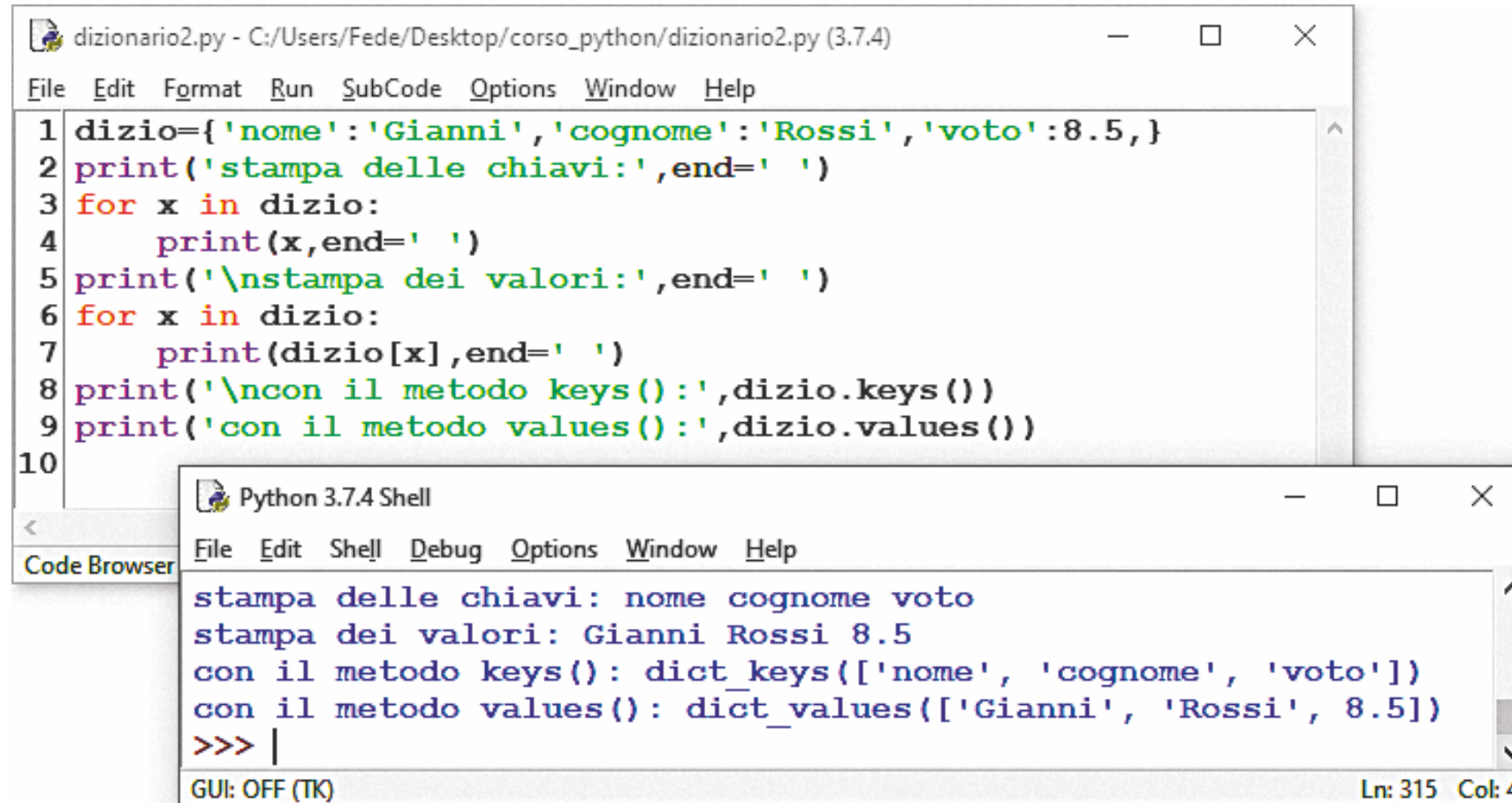
```
dizionario.py - C:/Users/Fede/Desktop/corso_python/dizionario.py (3.7.4)
File Edit Format Run SubCode Options Window Help
1 diz={
2     'nome': 'Gianni',
3     'cognome': 'Rossi',
4     'voto': 8.5,
5     1: 'ciao'
6 }
7 print(diz)
8 print(diz['voto'])
9 print(diz[1])
10

Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
{'nome': 'Gianni', 'cognome': 'Rossi', 'voto': 8.5, 1: 'ciao'}
8.5
ciao
>>> |
GUI: OFF (TK) Ln: 297 Col: 4
```

2.4 Tuple, dizionari e set

Indice

Un dizionario può essere elaborato per mezzo di un ciclo **for**.
I metodi **keys()** e **values()** selezionano le chiavi e i valori.



The image shows a screenshot of a Python IDE window titled 'dizionario2.py - C:/Users/Fede/Desktop/corso_python/dizionario2.py (3.7.4)'. The code in the editor is as follows:

```
1 dizio={'nome':'Gianni','cognome':'Rossi','voto':8.5,}  
2 print('stampa delle chiavi:',end=' ')  
3 for x in dizio:  
4     print(x,end=' ')  
5 print('\nstampa dei valori:',end=' ')  
6 for x in dizio:  
7     print(dizio[x],end=' ')  
8 print('\ncon il metodo keys():',dizio.keys())  
9 print('con il metodo values():',dizio.values())  
10
```

Below the code editor is a 'Python 3.7.4 Shell' window showing the output of the code:

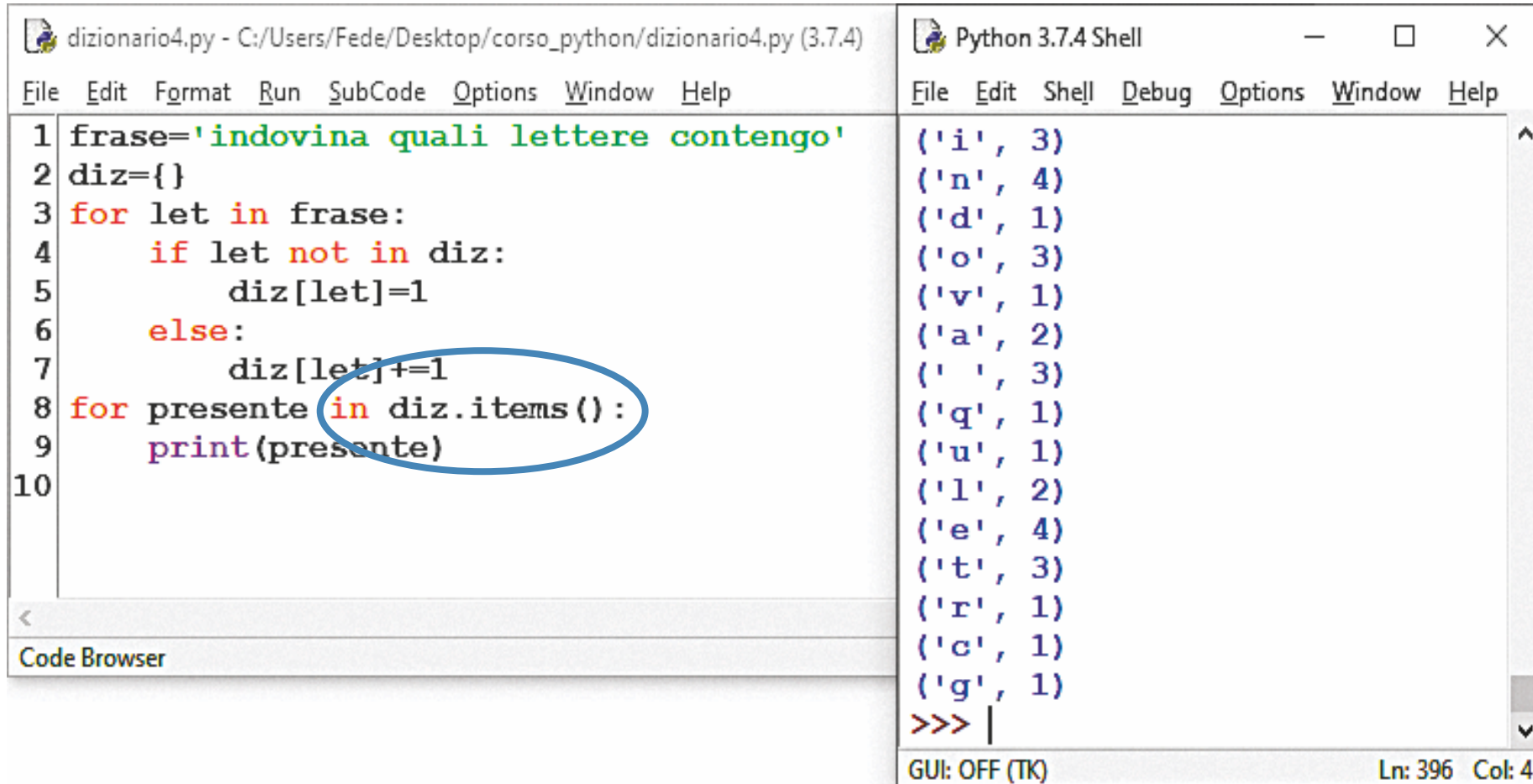
```
stampa delle chiavi: nome cognome voto  
stampa dei valori: Gianni Rossi 8.5  
con il metodo keys(): dict_keys(['nome', 'cognome', 'voto'])  
con il metodo values(): dict_values(['Gianni', 'Rossi', 8.5])  
>>> |
```

The shell window also shows 'GUI: OFF (TK)' and 'Ln: 315 Col: 4' at the bottom.

2.4 Tuple, dizionari e set

[Indice](#)

A un dizionario è possibile applicare dei metodi. Risulta molto utile il metodo `items()`.



The image shows a screenshot of a Python IDE with two windows. The left window, titled 'dizionario4.py - C:/Users/Fede/Desktop/corso_python/dizionario4.py (3.7.4)', contains the following Python code:

```
1 frase='indovina quali lettere contengo'
2 diz={}
3 for let in frase:
4     if let not in diz:
5         diz[let]=1
6     else:
7         diz[let]+=1
8 for presente in diz.items():
9     print(presente)
10
```

The `in` keyword in line 8 is circled in blue. The right window, titled 'Python 3.7.4 Shell', shows the output of the script as a list of tuples representing the items of the dictionary:

```
('i', 3)
('n', 4)
('d', 1)
('o', 3)
('v', 1)
('a', 2)
(' ', 3)
('q', 1)
('u', 1)
('l', 2)
('e', 4)
('t', 3)
('r', 1)
('c', 1)
('g', 1)
>>> |
```

The status bar at the bottom of the shell window indicates 'GUI: OFF (TK)' and 'Ln: 396 Col: 4'.

2.4 Tuple, dizionari e set

Indice

Un **set** è un oggetto modificabile che contiene dati di vario tipo, non indicizzati né ordinati con valori **tutti diversi tra loro**. Può essere manipolato utilizzando gli appositi metodi.

```
colori={'bianco','rosso','verde','blu'}
colori2={'viola','blu','rosa','verde'}
print(colori)
colori.update(colori2)
print(colori)
colori.add('giallo')
print(colori)
colori.remove('verde')
print(colori)
print('colore eliminato casualmente:',colori.pop())
print(colori)
colori.clear()
print('dopo clear:',colori)
```

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
{'bianco', 'rosso', 'blu', 'verde'}
{'bianco', 'verde', 'viola', 'rosa', 'rosso', 'blu'}
{'bianco', 'verde', 'viola', 'giallo', 'rosa', 'rosso', 'blu'}
{'bianco', 'viola', 'giallo', 'rosa', 'rosso', 'blu'}
colore eliminato casualmente: bianco
{'viola', 'giallo', 'rosa', 'rosso', 'blu'}
dopo clear: set()
>>> |
GUI: OFF (TK) Ln: 468 Col: 4
```

Esistono metodi che calcolano le operazioni tipiche degli insiemi:
unione, intersezione, differenza, differenza simmetrica e verifica di inclusione.