



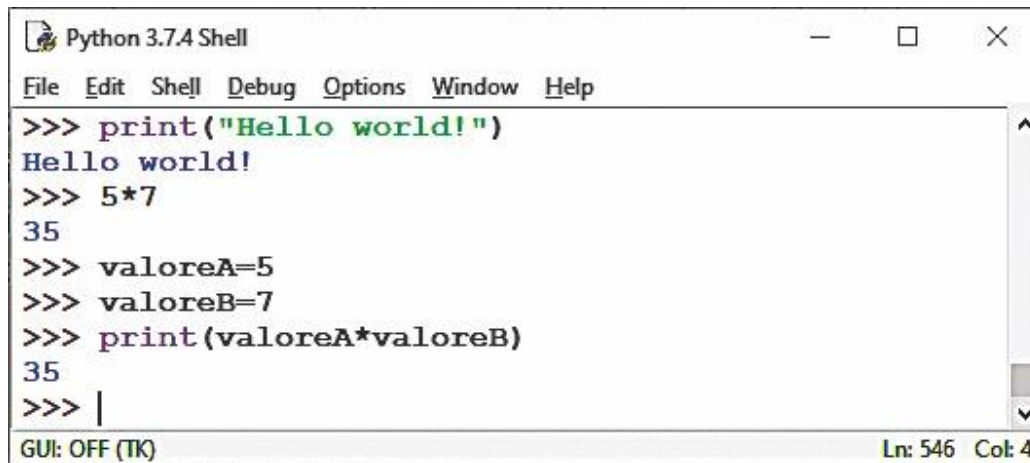
LEZIONE 1

1. Un linguaggio di programmazione per tutti
2. Gli elementi di base del linguaggio
3. Le strutture condizionali
4. I cicli iterativi

1.1 Un linguaggio di programmazione per tutti [Indice](#)

Python è un linguaggio intuitivo, facile da imparare e **open source**

Python è un linguaggio di scripting interpretato di alto livello, **multi-piattaforma** e **multi-paradigma**.



```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
>>> print("Hello world!")
Hello world!
>>> 5*7
35
>>> valoreA=5
>>> valoreB=7
>>> print(valoreA*valoreB)
35
>>> |
GUI: OFF (TK) Ln: 546 Col: 4
```



L'interprete di Python è stato scritto nel 1989 da **Guido van Rossum**.

Il **codice** si può scrivere con un normale **editor di testo non formattato**, oppure usando un ambiente di sviluppo come **IDLE**.

1.2 Gli elementi di base del linguaggio

Indice

Le **variabili** e i **tipi di dato** in Python

I principali tipi di dato sono **int**, **float**, **str** e **bool**.

Il linguaggio è caratterizzato da **tipizzazione debole**: una stessa variabile può assumere diversi tipi di dato nel corso della stessa elaborazione.

Python è **case-sensitive**: distingue le lettere minuscole dalle maiuscole.

```
nome='Anna'  
età=17  
print("Hello world! Sono", nome, "e ho", età, "anni.")
```

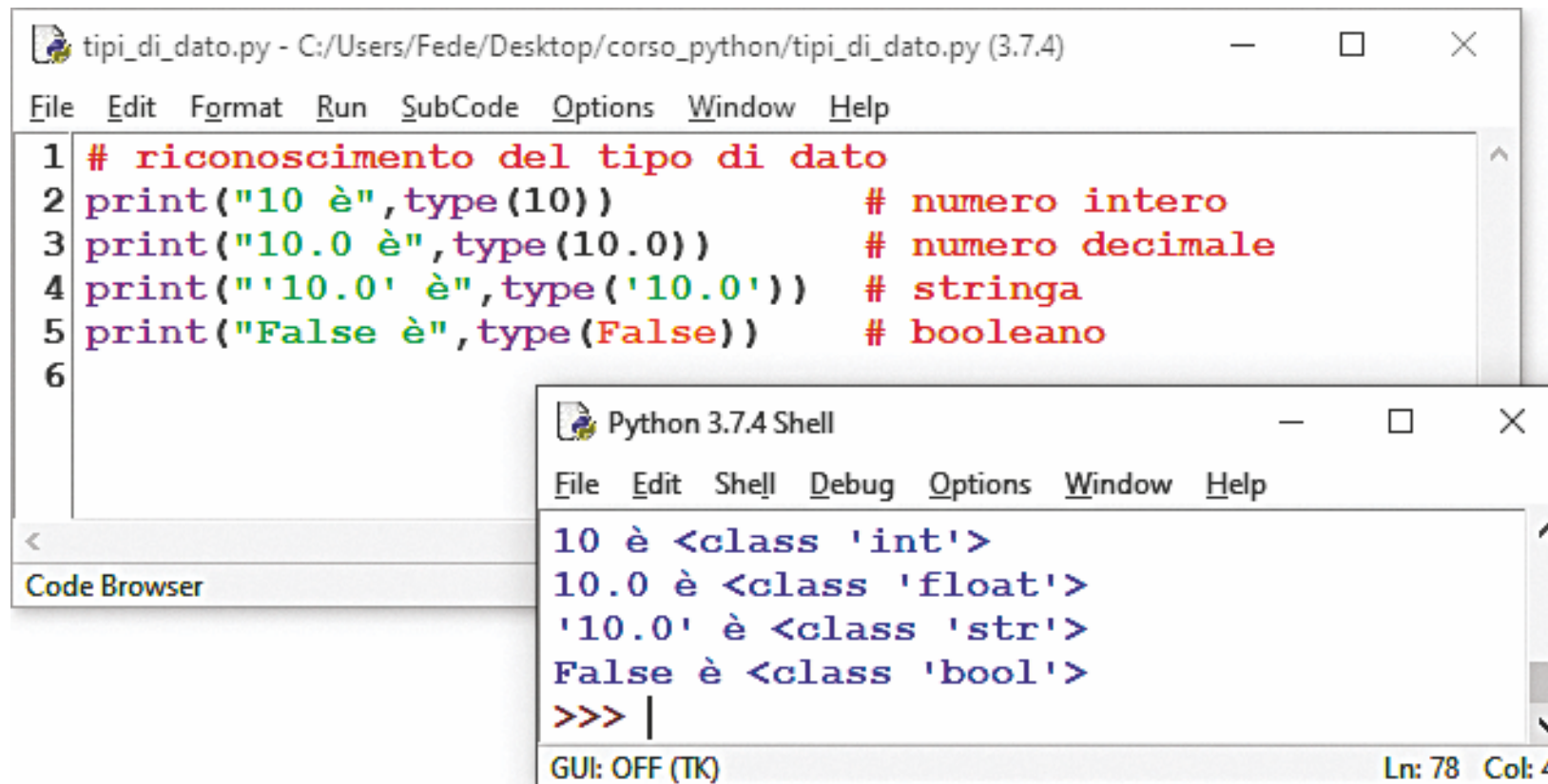
```
Hello world! Sono Anna e ho 17 anni.
```

print() è una funzione built-in che visualizza a schermo l'argomento.

1.2 Gli elementi di base del linguaggio

Indice

L'interprete di Python **riconosce il tipo di dato a *runtime***,
cioè al momento dell'esecuzione del codice



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

```
1 # riconoscimento del tipo di dato
2 print("10 è", type(10))           # numero intero
3 print("10.0 è", type(10.0))       # numero decimale
4 print("'10.0' è", type('10.0'))   # stringa
5 print("False è", type(False))     # booleano
6
```

Below the code 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:

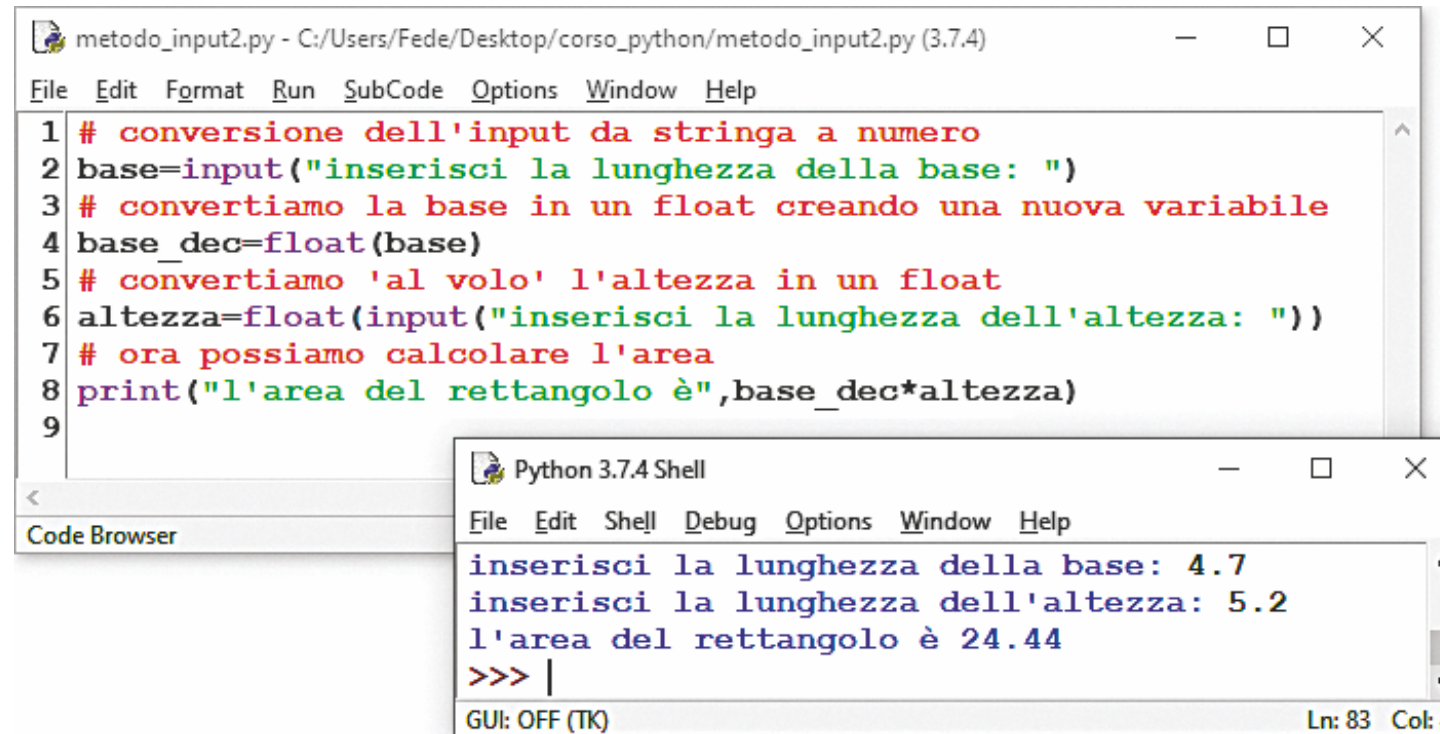
```
10 è <class 'int'>
10.0 è <class 'float'>
'10.0' è <class 'str'>
False è <class 'bool'>
>>> |
```

The shell window also shows 'GUI: OFF (TK)' and 'Ln: 78 Col: 4' at the bottom.

1.2 Gli elementi di base del linguaggio

Indice

La funzione `input()` di Python e la conversione dei dati



The image shows a screenshot of a Python IDE with two windows. The top window, titled 'metodo_input2.py - C:/Users/Fede/Desktop/corso_python/metodo_input2.py (3.7.4)', contains the following Python code:

```
1 # conversione dell'input da stringa a numero
2 base=input("inserisci la lunghezza della base: ")
3 # convertiamo la base in un float creando una nuova variabile
4 base_dec=float(base)
5 # convertiamo 'al volo' l'altezza in un float
6 altezza=float(input("inserisci la lunghezza dell'altezza: "))
7 # ora possiamo calcolare l'area
8 print("l'area del rettangolo è",base_dec*altezza)
9
```

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

```
inserisci la lunghezza della base: 4.7
inserisci la lunghezza dell'altezza: 5.2
l'area del rettangolo è 24.44
>>> |
```

The status bar at the bottom of the shell window indicates 'GUI: OFF (TK)' and 'Ln: 83 Col: 4'.

La funzione `input()` memorizza come **stringa** tutti i dati immessi da tastiera: per utilizzarli correttamente, perciò, occorre convertirli.

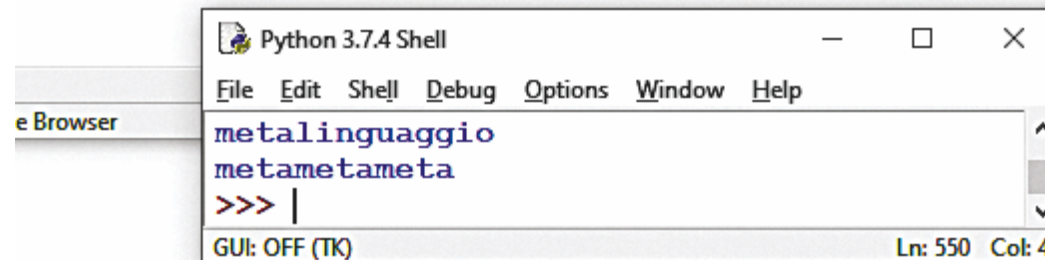
1.2 Gli elementi di base del linguaggio

Indice

Gli operatori aritmetici e gli operatori di concatenamento in Python

simbolo	operazione
+	addizione
-	sottrazione
*	moltiplicazione
/	divisione
//	divisione intera
%	modulo
**	elevazione a potenza

```
# usiamo l'operatore di concatenamento
pre='meta'
post='linguaggio'
print(pre+post)
# usiamo l'operatore di ripetizione
print(pre*3)
```



```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
metalinguaggio
metametameta
>>> |
GUI: OFF (TK) Ln: 550 Col: 4
```

Ogni operazione tra interi ha come risultato un intero, tra decimali ha come risultato un decimale.

Un'operazione tra un intero e un decimale restituisce un decimale.

per le **stringhe**:

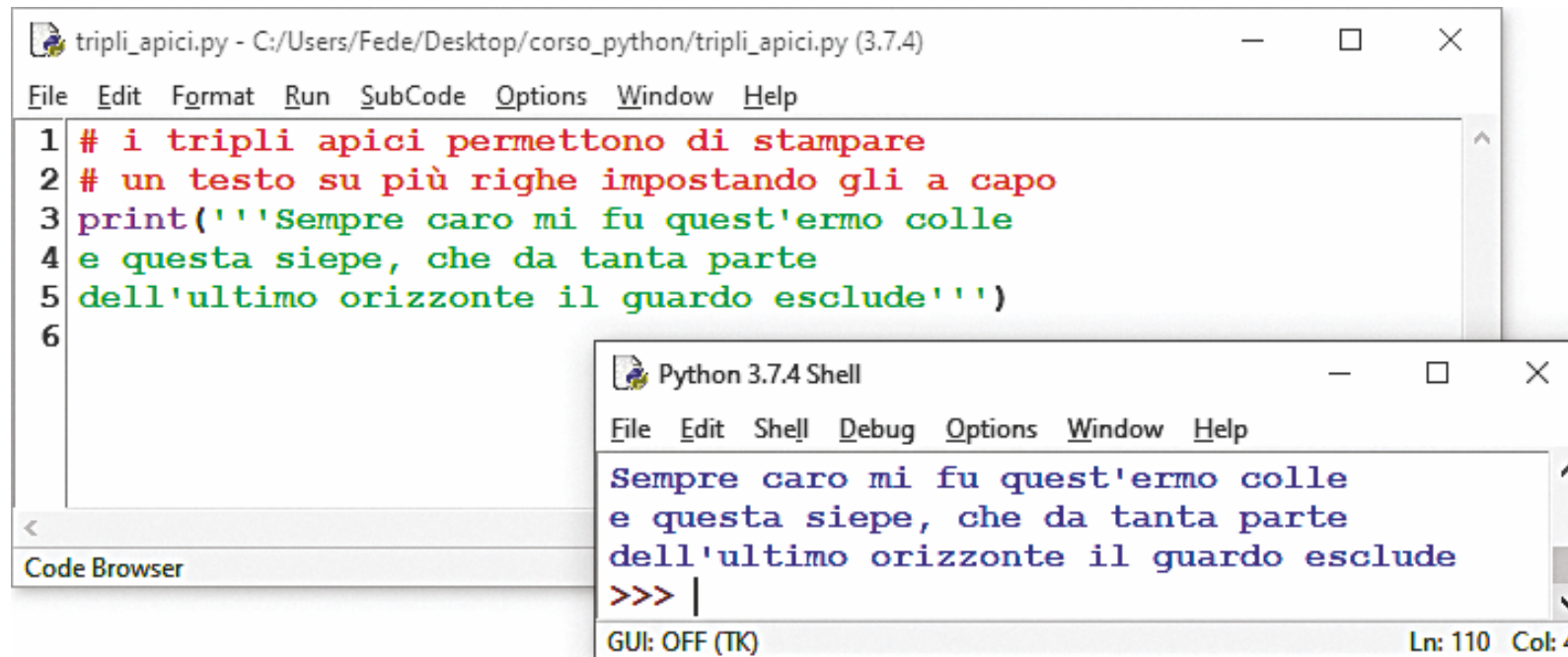
+ è l'operatore di **concatenamento**

* è l'operatore di **ripetizione**.

1.2 Gli elementi di base del linguaggio

Indice

In Python si può **formattare l'output** in vari modi



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

```
1 # i tripli apici permettono di stampare
2 # un testo su più righe impostando gli a capo
3 print('''Sempre caro mi fu quest'ermo colle
4 e questa siepe, che da tanta parte
5 dell'ultimo orizzonte il guardo esclude''')
6
```

Below the editor, a 'Python 3.7.4 Shell' window displays the output of the code, which is the text from the triple-quoted string, formatted with line breaks and indentation as specified in the code:

```
Sempre caro mi fu quest'ermo colle
e questa siepe, che da tanta parte
dell'ultimo orizzonte il guardo esclude
>>> |
```

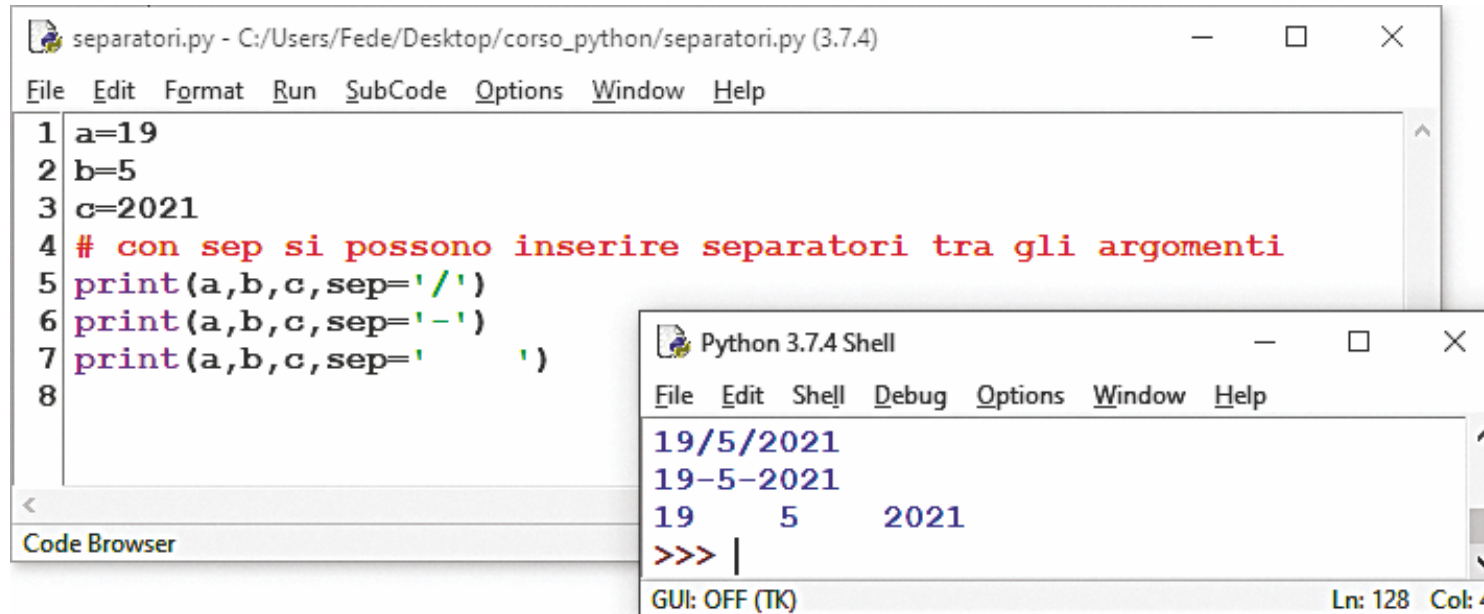
The status bar at the bottom of the shell window indicates 'GUI: OFF (TK)' and 'Ln: 110 Col: 4'.

Il testo contenuto tra **tripli apici** viene visualizzato esattamente come è scritto nel codice (*a capo* compresi).

1.2 Gli elementi di base del linguaggio

Indice

In Python si può **formattare l'output** in vari modi



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

```
1 a=19
2 b=5
3 c=2021
4 # con sep si possono inserire separatori tra gli argomenti
5 print(a,b,c,sep='/')
6 print(a,b,c,sep='-')
7 print(a,b,c,sep=' ')
8
```

Below the code editor is a 'Code Browser' tab. Overlaid on the bottom right is a 'Python 3.7.4 Shell' window. The shell displays the output of the three print statements:

```
19/5/2021
19-5-2021
19    5    2021
>>> |
```

The shell window also shows a status bar at the bottom with 'GUI: OFF (TK)' and 'Ln: 128 Col: 4'.

Si possono definire **caratteri separatori**, usare i **caratteri di escape** (come `\n` e `\t`) e **formattare i numeri** con la funzione `format()`.

1.3 Le strutture condizionali

Indice

La struttura condizionale più usata è `if...else`

```
dividendo=float(input('inserisci il dividendo: '))
divisore=float(input('inserisci il divisore: '))
#codice eseguito se il divisore è diverso da zero
if divisore!=0:
    print('il quoziente è',dividendo/divisore)
#codice eseguito se il divisore è zero
else:
    print('errore! il divisore non può essere 0')
    print('cambia il valore del divisore e riprova')
print('[fine del programma]')
```

Python riconosce i blocchi di codice in base alla loro **indentazione**:

`if` ed `else` devono avere lo stesso allineamento

e i blocchi di codice contenuti devono avere lo stesso rientro.

1.3 Le strutture condizionali

[Indice](#)

Le **condizioni** si scrivono usando gli **operatori relazionali** e **logici**

Operatori relazionali:

<code>==</code>	uguale a	<code>!=</code>	diverso da
<code>></code>	maggiore di	<code>>=</code>	maggiore o uguale a
<code><</code>	minore di	<code><=</code>	minore o uguale a

Operatori logici:

and

or

not

1.3 Le strutture condizionali

Indice

Si possono **annidare** le strutture condizionali o si può usare **elif**

```
num1=float(input('inserisci il primo numero: '))
num2=float(input('inserisci il secondo numero: '))
scelta=int(input('''quale operazione vuoi effettuare?
1 - somma
2 - sottrai
3 - dividi
4 - moltiplica
\tla tua scelta: '''))
if scelta==1:
    print('la somma dei due numeri è',num1+num2)
elif scelta==2:
    print('la differenza tra i due numeri è',num1-num2)
elif scelta==3:
    print('il quoziente dei due numeri è',num1/num2)
elif scelta==4:
    print('il prodotto dei due numeri è',num1*num2)
else:
    print('Attenzione: scelta non valida!')
```

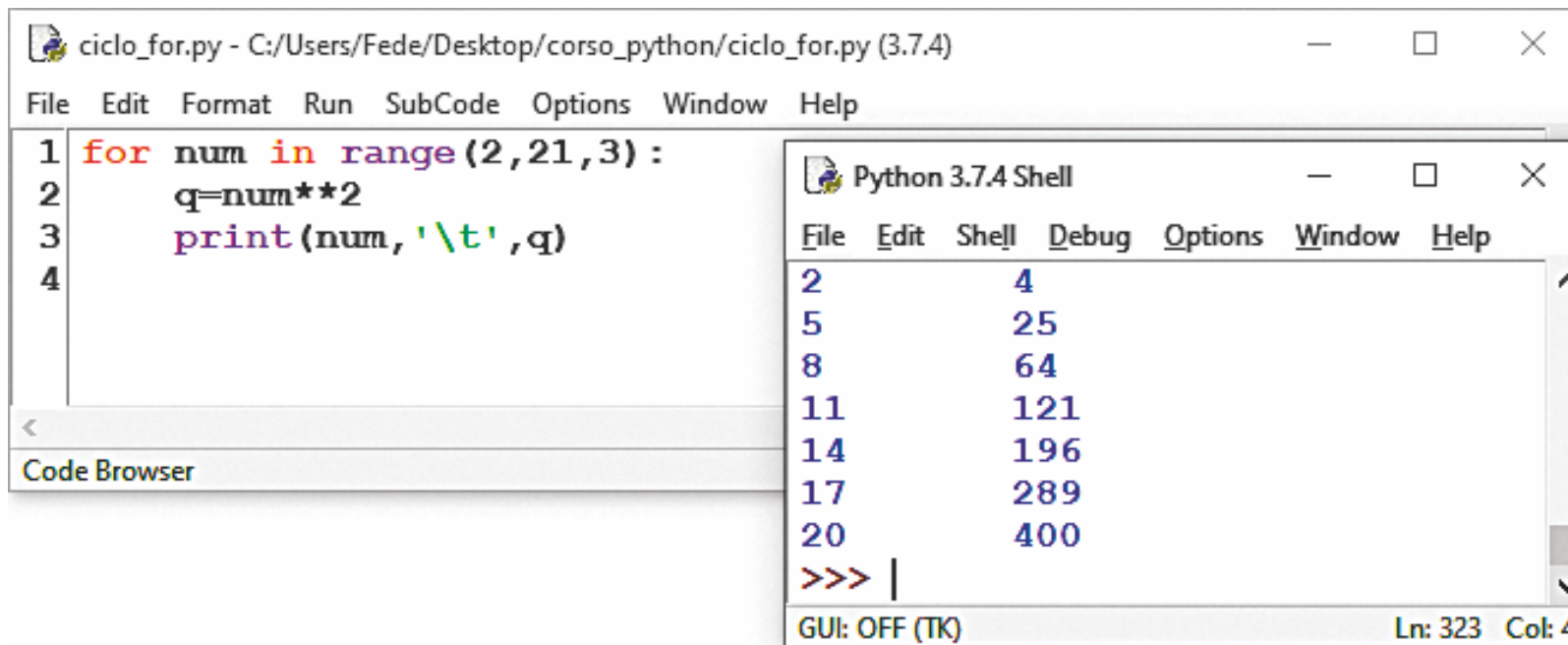
Per poter usare **elif** le **condizioni** devono essere **mutuamente esclusive**.

1.4 I cicli iterativi

Indice

La sintassi del ciclo `for` in Python

`for nome_variabile in range(valore iniziale, limite esterno, passo) :`
istruzioni da eseguire a ogni iterazione



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

```
1 for num in range(2,21,3):
2     q=num**2
3     print(num, '\t', q)
4
```

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

2	4
5	25
8	64
11	121
14	196
17	289
20	400

The shell window also shows the prompt `>>> |` and the status bar at the bottom indicates 'GUI: OFF (TK)' and 'Ln: 323 Col: 4'.

Se in `range()` si specifica **un solo argomento**, questo è interpretato come il **limite esterno** del contatore, con valore iniziale 0 e passo 1.

1.4 I cicli iterativi

Indice

Il **ciclo for** è utile anche per l'elaborazione delle **stringhe**

Una **stringa** in Python è una **sequenza di caratteri** ai quali è associato un **indice intero che parte da 0**.

```
for lettera in 'ciao':  
    print(lettera)
```

fornisce come output:

c

i

a

o

```
saluto='ciao'
```

```
print(saluto[1],saluto[3],sep='')
```

visualizza:

io

1.4 I cicli iterativi

[Indice](#)

Gli **operatori composti** permettono di scrivere codice più compatto

`a += b`

equivale a:

`a = a+b`

`c -= d`

equivale a:

`c = c-d`

`e *= f`

equivale a:

`e = e*f`

`g /= h`

equivale a:

`g = g/h`

`i %= j`

equivale a:

`i = i%j`

1.4 I cicli iterativi

Indice

I cicli possono essere annidati.

x=10

```
for i in range(x):
    print('\t',end='')
    for j in range(1,x+1-i):
        print('*',end='')
    print()
```

A ogni iterazione del ciclo esterno viene eseguito il ciclo interno.



The screenshot shows a Python 3.7.4 Shell window with a menu bar (File, Edit, Shell, Debug, Options, Window, Help) and a text area containing a star pattern. The pattern consists of 10 lines of stars, with the number of stars per line decreasing from 10 to 1. The prompt is >>> |. The status bar at the bottom shows 'GUI: OFF (TK)' and 'Ln: 89 Col: 4'.

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
*****
*****
*****
*****
*****
*****
*****
***
**
*
>>> |
GUI: OFF (TK) Ln: 89 Col: 4
```

1.4 I cicli iterativi

Indice

Il ciclo **while** si usa quando il numero delle iterazioni non è noto

while condizione:

istruzioni da eseguire se la condizione è vera

```
giorni=1
totale=float(input('inserisci la cifra spesa il primo giorno: '))
media=totale
while media<=50 and giorni<10:
    spesa=float(input('inserisci la spesa del giorno successivo: '))
    totale+=spesa
    giorni+=1
    media=totale/giorni
if media>50:
    print('\tAttenzione, hai superato il budget di 50 €/giorno!
\tOgni giorno hai speso in media'',format(media, '.2f'), '€')
else:
    print('bene, non hai superato il budget')
```

1.4 I cicli iterativi

Indice

Le istruzioni `break` e `continue` in Python

`break` fa uscire dal ciclo ed esegue la prima istruzione successiva:

codice:	<pre>for n in range(1,10): if n==5: break print(n,end=' ')</pre>	<pre>for n in range(1,10): if n==5: break print(n,end=' ')</pre>
output:	1 2 3 4	5

`continue` fa passare direttamente all'iterazione successiva del ciclo:

```
for n in range(1,10):  
    if n%3==0:  
        continue  
    print(n,end=' ')
```

1 2 4 5 7 8