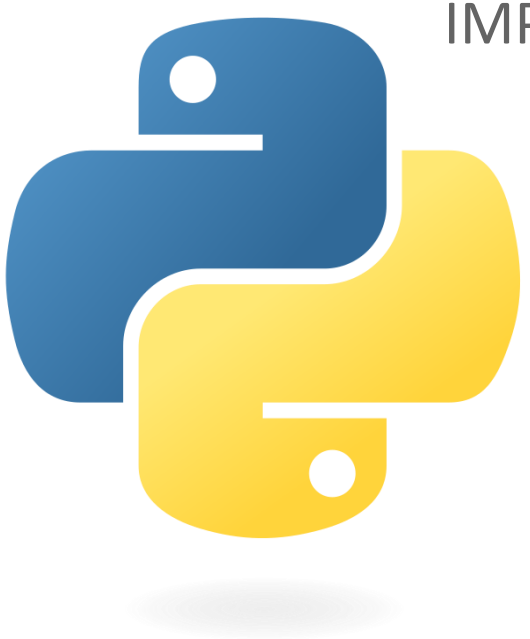


IMPARIAMO A PROGRAMMARE IN

python™

LEZIONE 1

Capitolo 1 – Introduzione a Python

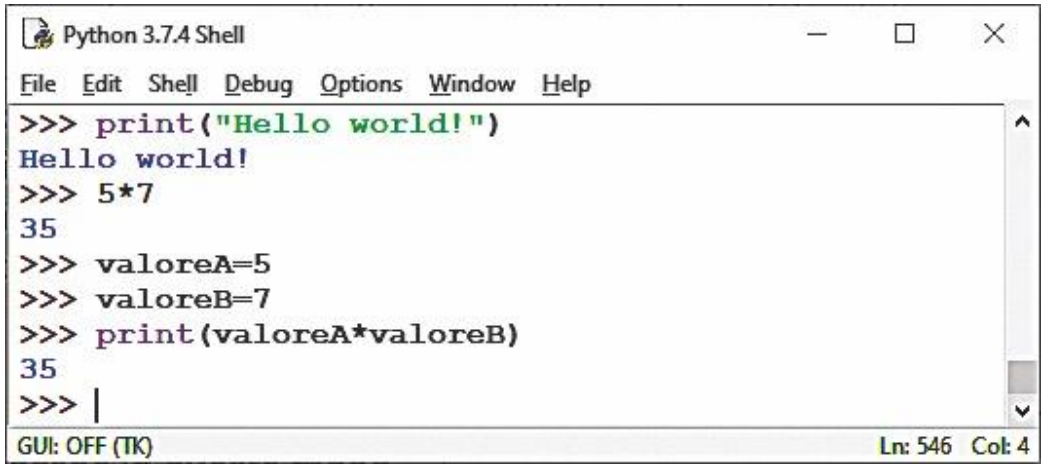
Indice

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.



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.")
```

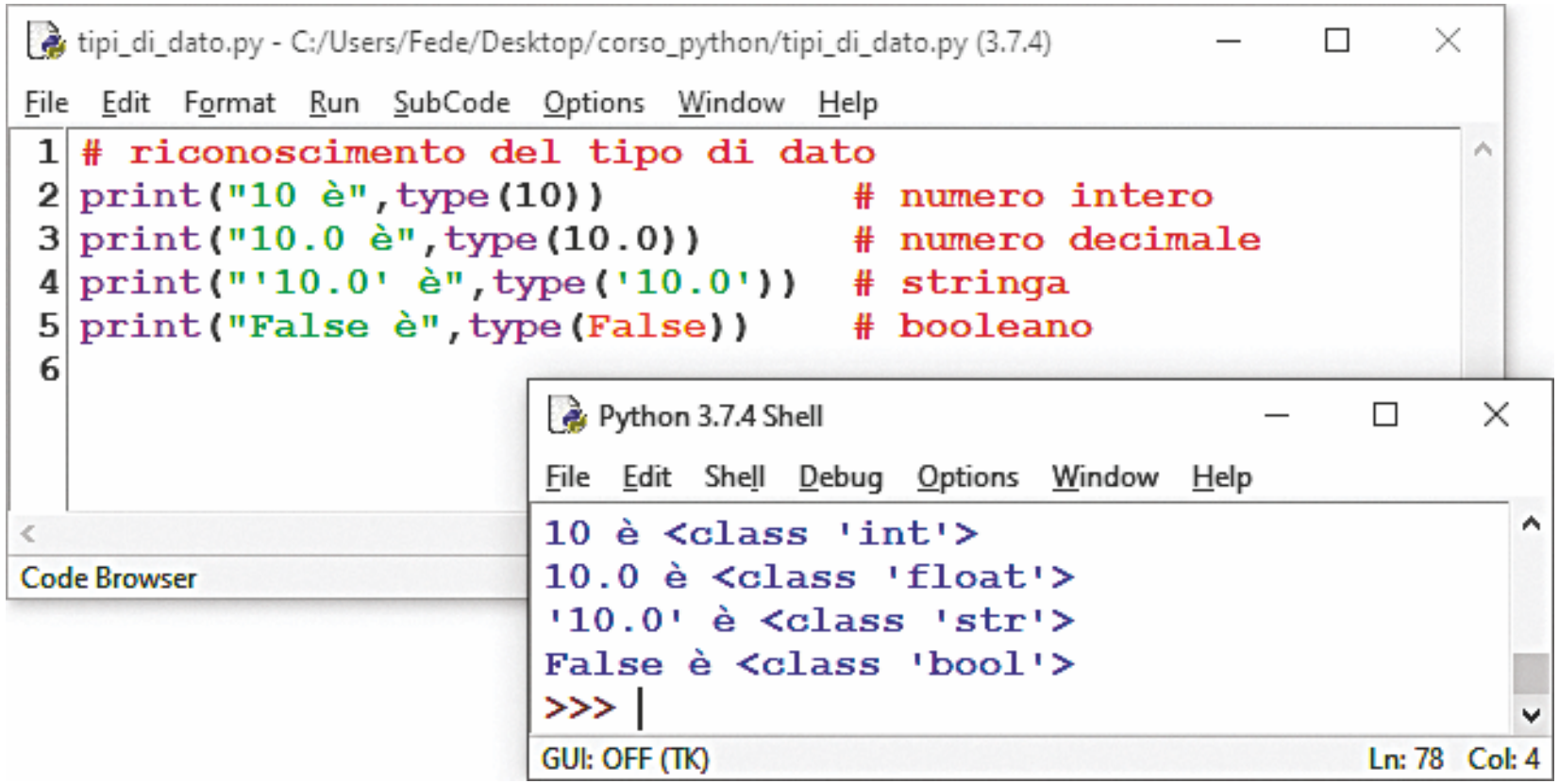
```
Hello world! Sono Anna e ho 17 anni.
```

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

1.2 Gli elementi di base del linguaggio

Indice

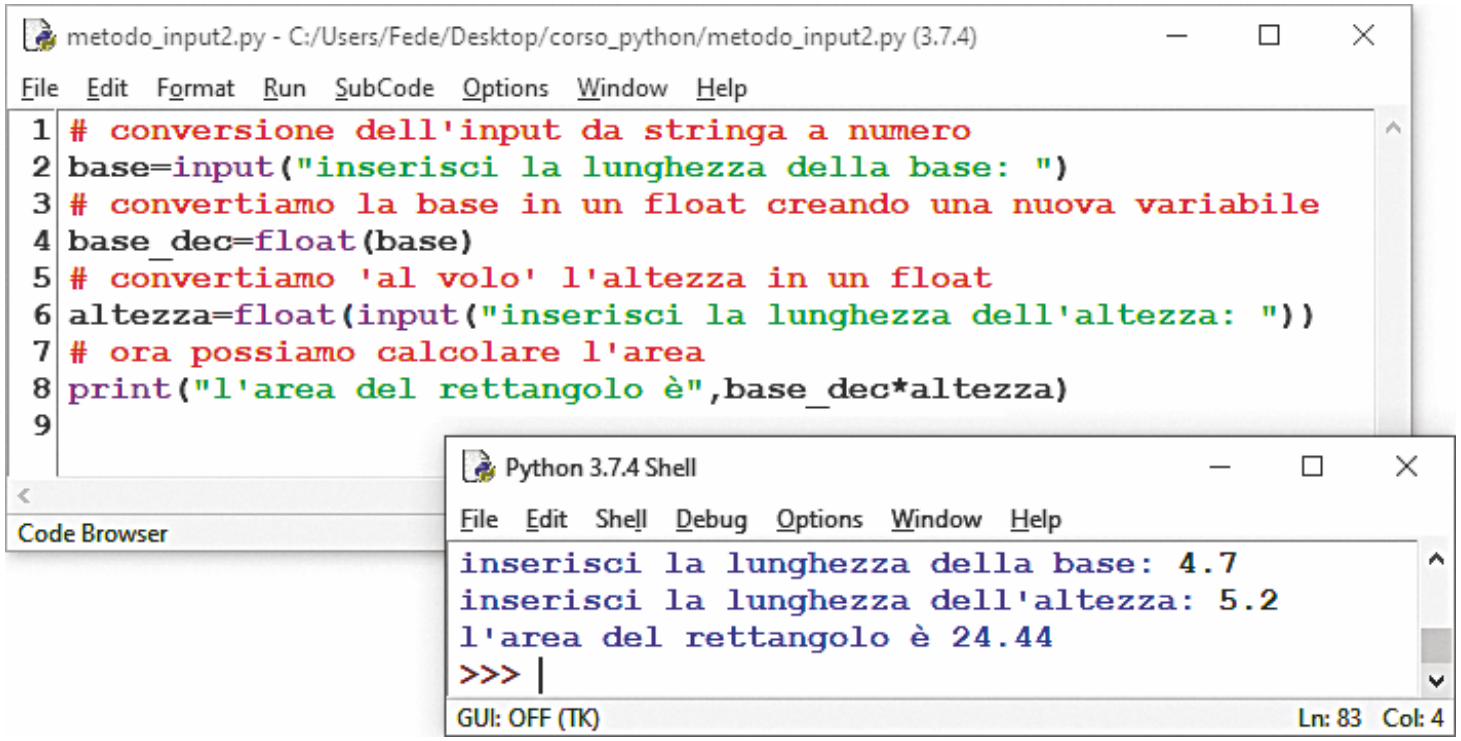
L'interprete di Python riconosce il tipo di dato a *runtime*, cioè al momento dell'esecuzione del codice



1.2 Gli elementi di base del linguaggio

Indice

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



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)
```

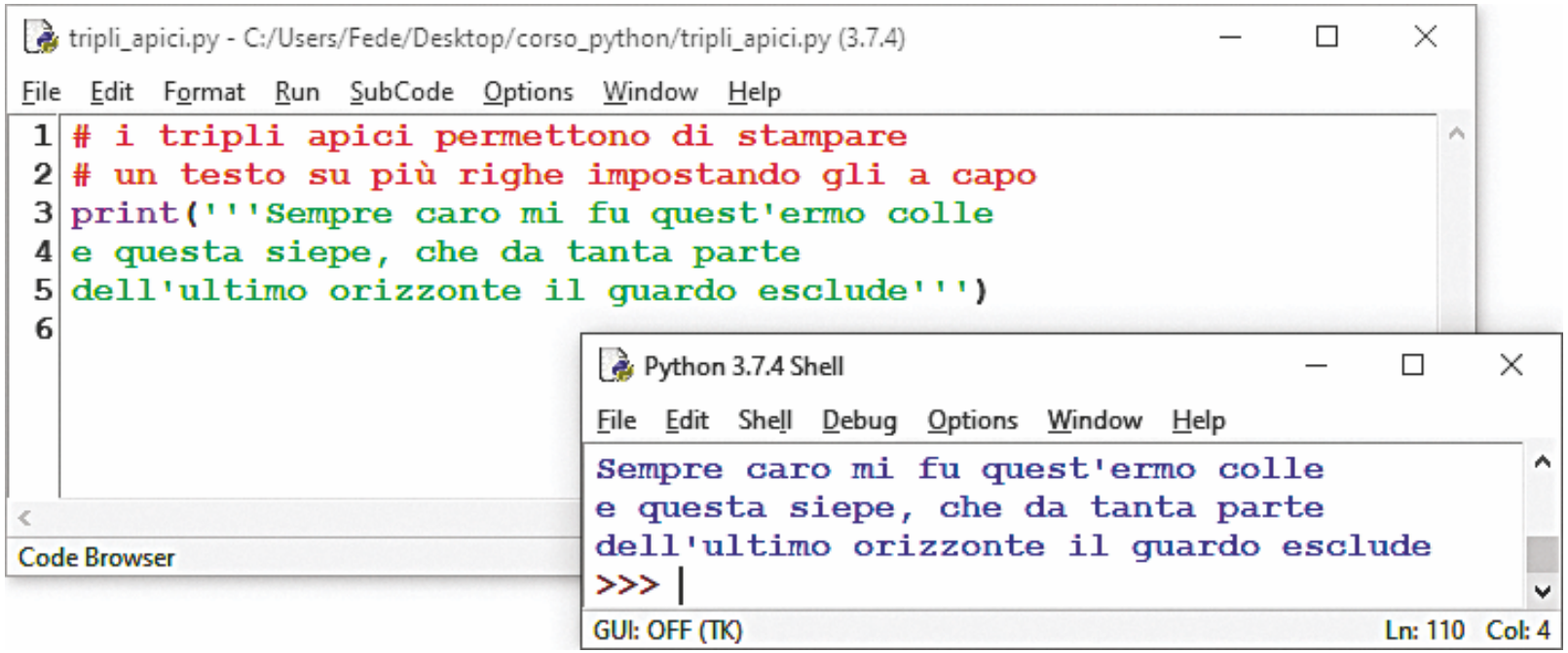
Ogni operazione tra interi ha come risultato un intero, tra decimali ha come risultato un decimale.
Un'operazione tra un intero e un decimale restituisce un decimale.

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

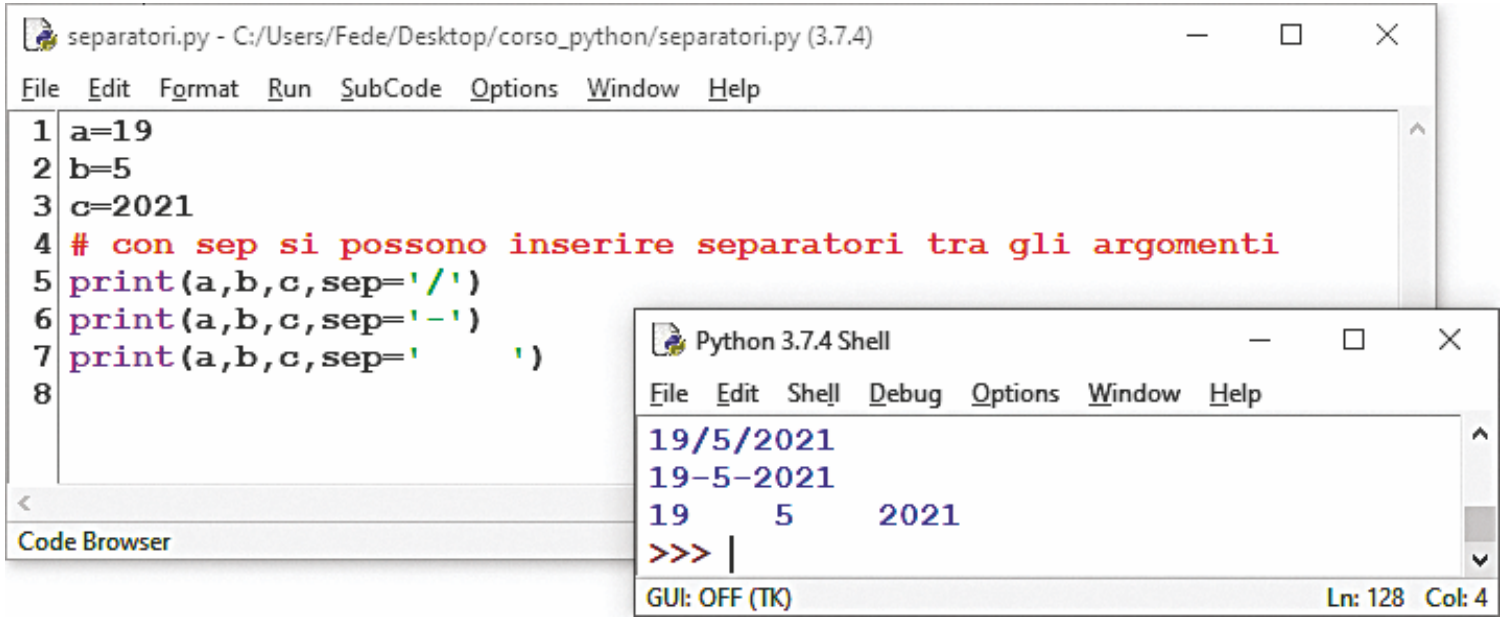


Il testo contenuto tra **tripli apici** viene visualizzato esattamente come è scritto nel codice (*a capo* compresi).

1.2 Gli elementi di base del linguaggio

Indice

In Python si può formattare l'output in vari modi



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:

==	uguale a	!=	diverso da
>	maggiore di	>=	maggiore o uguale a
<	minore di	<=	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!')
```

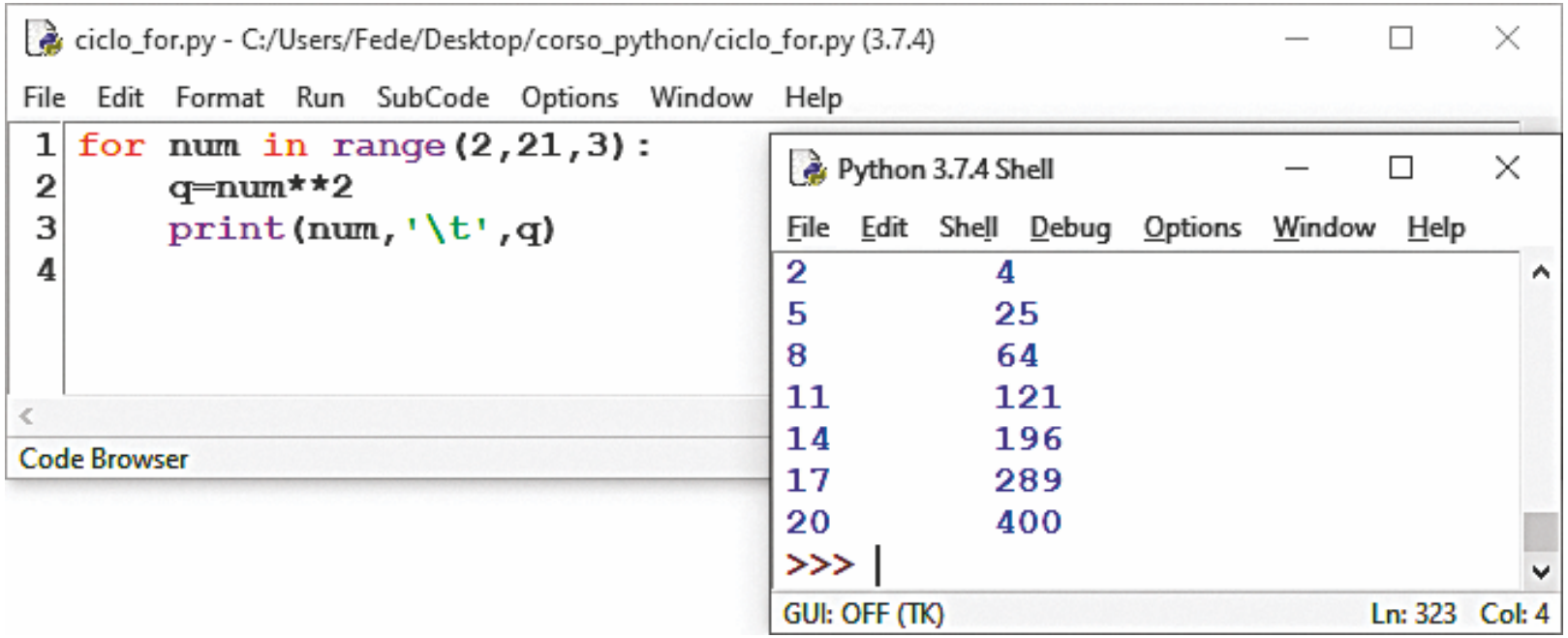
Per poter usare **`elif`** le **condizioni** devono essere **mutuamente esclusive**.

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



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

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`

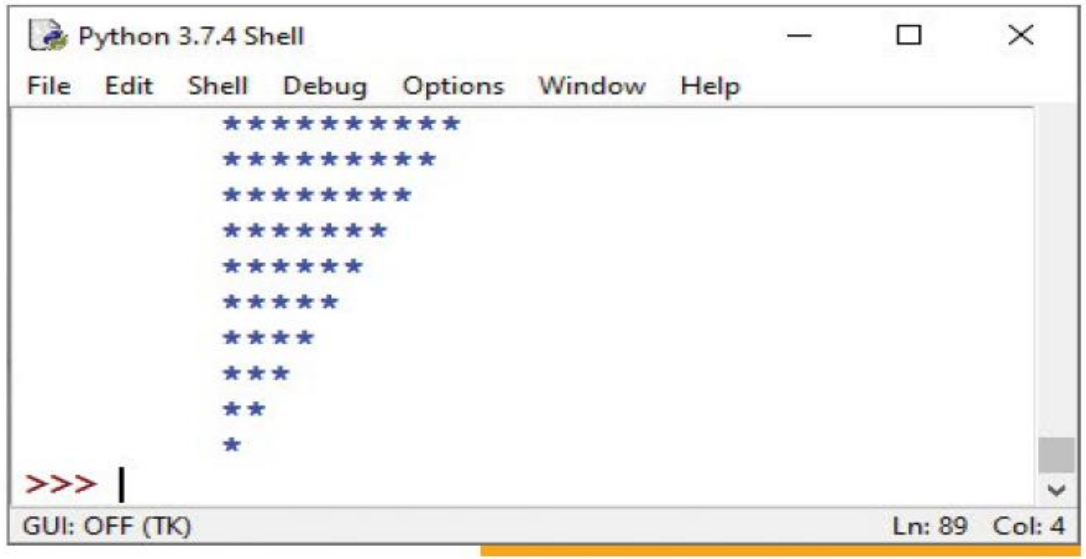
1.4 I cicli iterativi

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.



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:	`for n in range(1,10):`
	`  if n==5:`
	`  break`
	`  print(n,end=' ')`
		`for n in range(1,10):`
		`  if n==5:`
		`  break`
		`print(n,end=' ')`

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
```