



LEZIONE 3

Capitolo 3 – Le funzioni

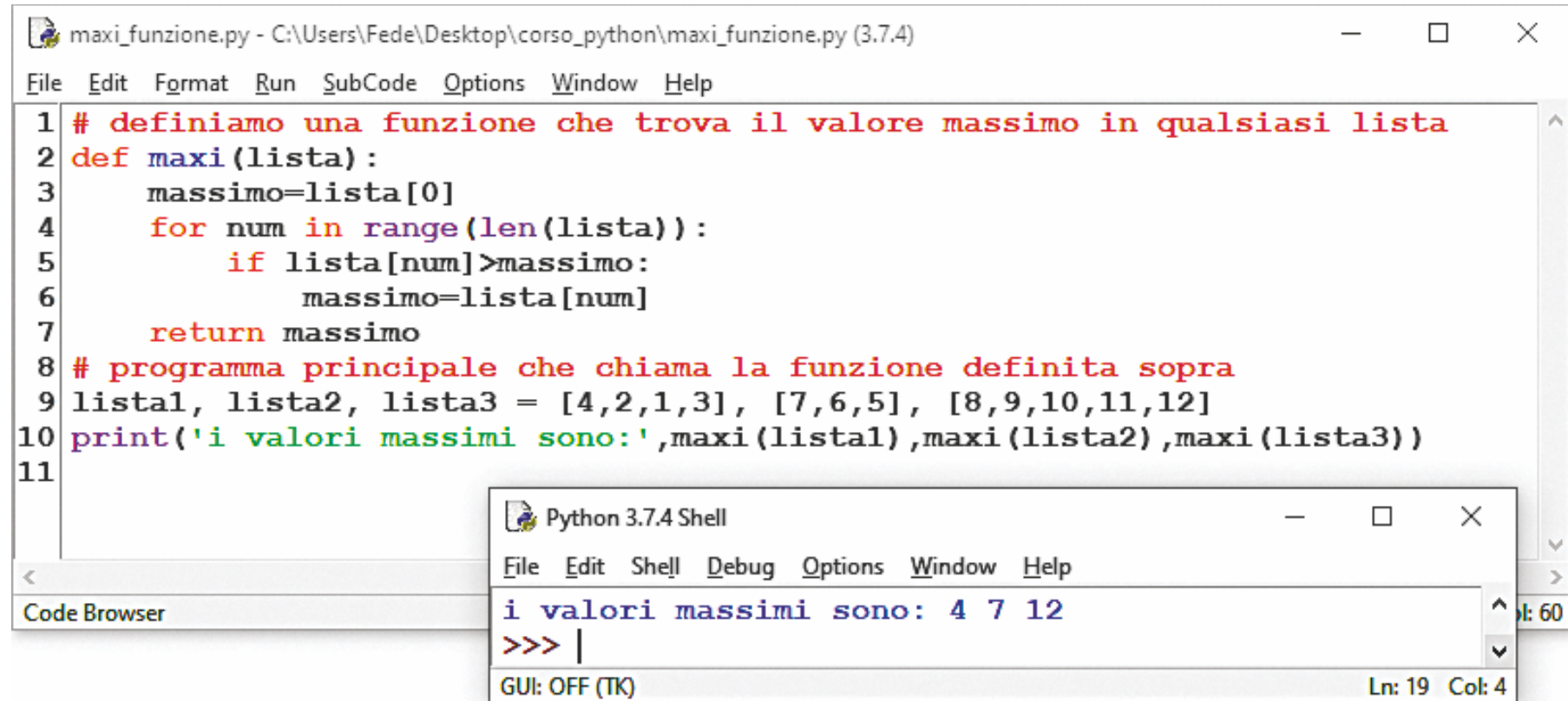
Indice

1. L'uso delle funzioni
2. Funzioni particolari
3. Gli ambiti di visibilità
4. I moduli
5. Disegnare con **turtle**

3.1 L'uso delle funzioni

Indice

Una **funzione** è un gruppo di istruzioni che si può eseguire ogni volta che serve, scrivendolo una volta sola.



The screenshot shows a Python IDE window titled 'maxi_funzione.py - C:\Users\Fede\Desktop\corso_python\maxi_funzione.py (3.7.4)'. The code defines a function 'maxi' that finds the maximum value in a list and then calls it on three different lists. Below the code editor, a 'Python 3.7.4 Shell' window displays the output of the script: 'i valori massimi sono: 4 7 12'. The status bar at the bottom of the shell indicates 'Ln: 19 Col: 4'.

```
1 # definiamo una funzione che trova il valore massimo in qualsiasi lista
2 def maxi(lista):
3     massimo=lista[0]
4     for num in range(len(lista)):
5         if lista[num]>massimo:
6             massimo=lista[num]
7     return massimo
8 # programma principale che chiama la funzione definita sopra
9 lista1, lista2, lista3 = [4,2,1,3], [7,6,5], [8,9,10,11,12]
10 print('i valori massimi sono:',maxi(lista1),maxi(lista2),maxi(lista3))
11
```

Python 3.7.4 Shell

```
i valori massimi sono: 4 7 12
>>> |
```

GUI: OFF (TK) Ln: 19 Col: 4

3.1 L'uso delle funzioni

Indice

In Python una funzione è composta da un'intestazione, seguita da un blocco di **istruzioni indentate**.

def nomeFunzione(argomento1,argomento2,...,argomentoN):

Di norma, una funzione riceve come input alcuni dati dal programma principale e li elabora calcolando un valore; poi, alla fine della sua esecuzione, restituisce questo dato di output al programma principale che l'ha chiamata attraverso un'operazione di **return**.

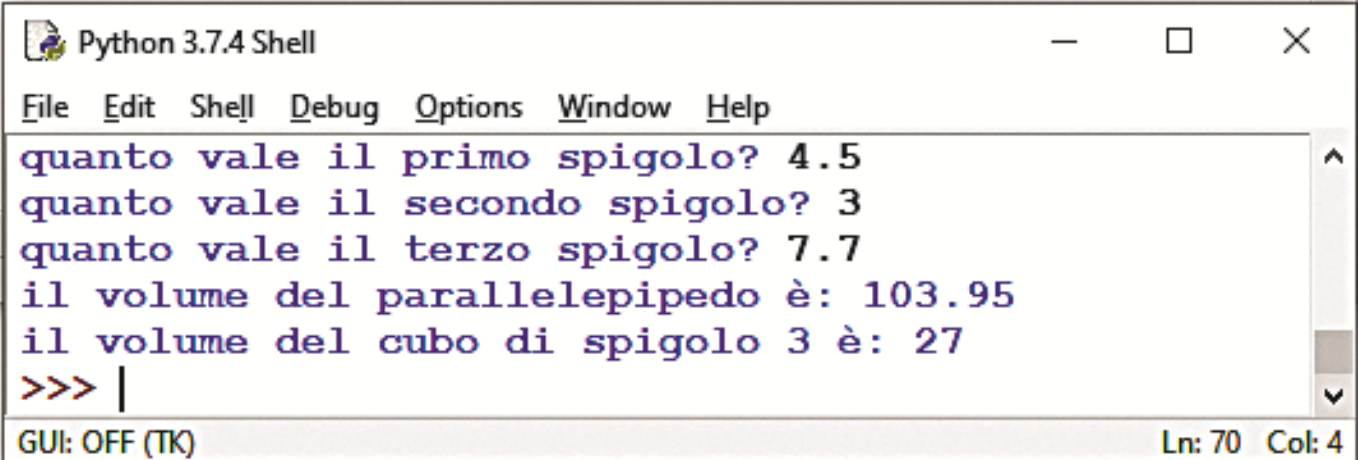
```
def maxi(lista):  
    massimo=lista[0]  
    for num in range(len(lista)):  
        if lista[num]>massimo:  
            massimo=lista[num]  
    return massimo
```

3.1 L'uso delle funzioni

Indice

Una funzione si chiama passandole dei valori per gli argomenti. Essa li elabora e restituisce al programma principale uno o più valori di ritorno.

```
6 volume=vol_para(inp1,inp2,inp3)
7 print('il volume del parallelepipedo è:',volume)
8 print('il volume del cubo di spigolo 3 è:',vol_para(3,3,3))
9
```



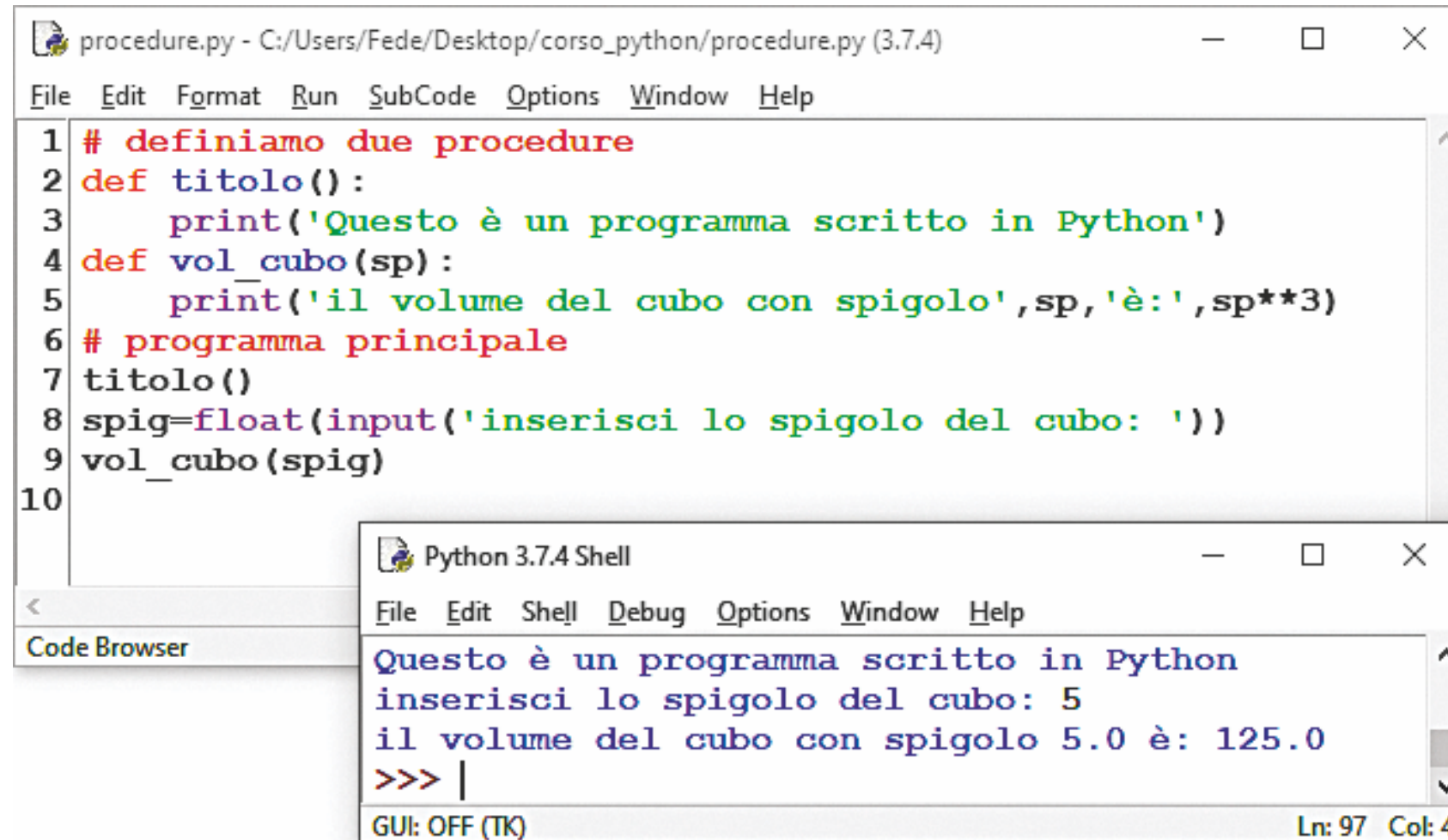
```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
quanto vale il primo spigolo? 4.5
quanto vale il secondo spigolo? 3
quanto vale il terzo spigolo? 7.7
il volume del parallelepipedo è: 103.95
il volume del cubo di spigolo 3 è: 27
>>> |
GUI: OFF (TK) Ln: 70 Col: 4
```

Nel caso in cui i valori siano più di uno, vengono restituiti sotto forma di tupla.

3.2 Funzioni particolari

Indice

Una **procedura** è una funzione che non produce valori.



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

```
1 # definiamo due procedure
2 def titolo():
3     print('Questo è un programma scritto in Python')
4 def vol_cubo(sp):
5     print('il volume del cubo con spigolo', sp, 'è:', sp**3)
6 # programma principale
7 titolo()
8 spig=float(input('inserisci lo spigolo del cubo: '))
9 vol_cubo(spig)
10
```

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

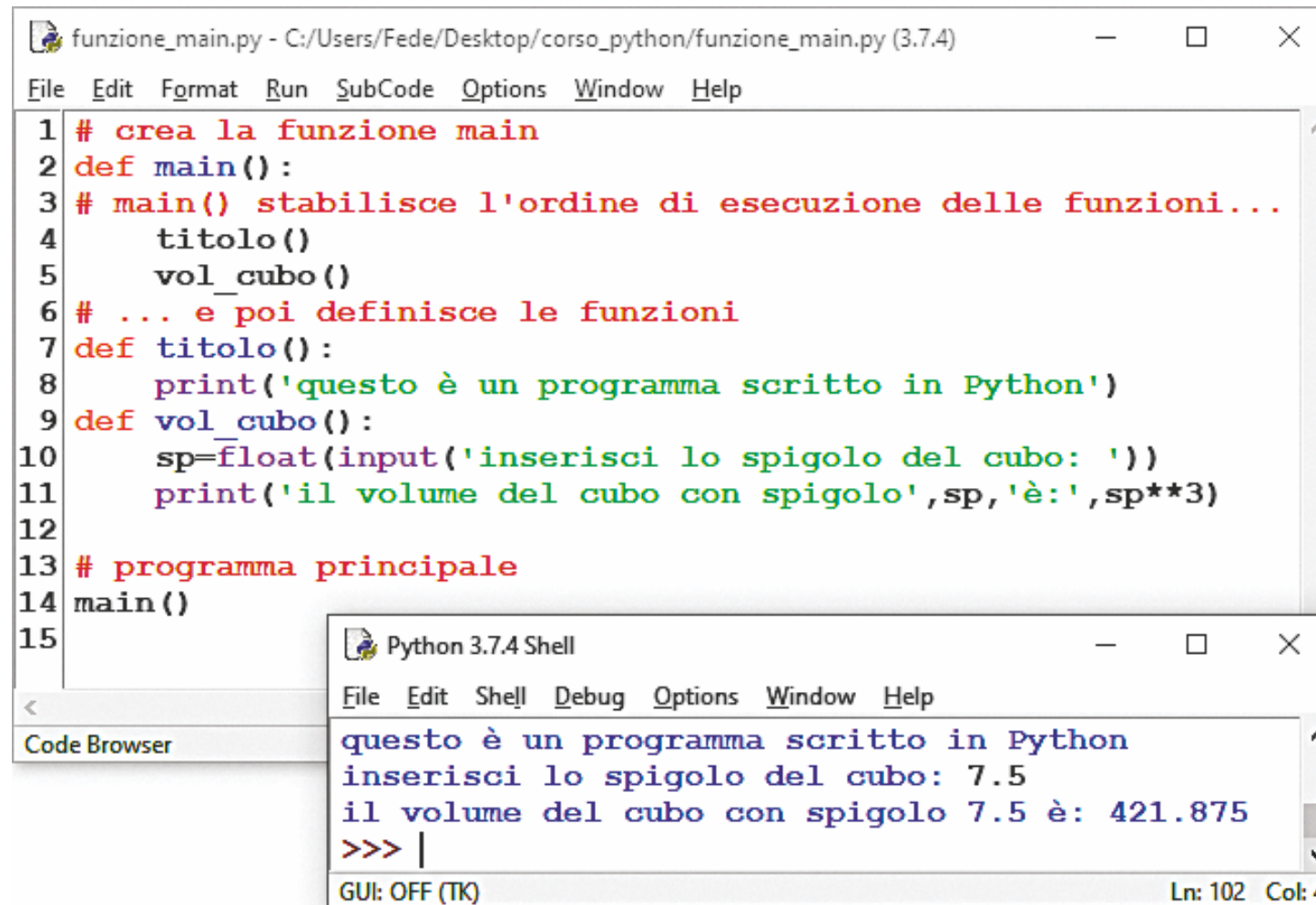
```
Questo è un programma scritto in Python
inserisci lo spigolo del cubo: 5
il volume del cubo con spigolo 5.0 è: 125.0
>>> |
```

The status bar at the bottom of the shell window indicates 'GUI: OFF (TK)' and 'Ln: 97 Col: 4'.

3.2 Funzioni particolari

Indice

In Python spesso si crea una funzione principale `main()` che chiama le altre funzioni del programma.



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

```
1 # crea la funzione main
2 def main():
3     # main() stabilisce l'ordine di esecuzione delle funzioni...
4     titolo()
5     vol_cubo()
6     # ... e poi definisce le funzioni
7 def titolo():
8     print('questo è un programma scritto in Python')
9 def vol_cubo():
10    sp=float(input('inserisci lo spigolo del cubo: '))
11    print('il volume del cubo con spigolo', sp, 'è:', sp**3)
12
13 # programma principale
14 main()
15
```

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

```
questo è un programma scritto in Python
inserisci lo spigolo del cubo: 7.5
il volume del cubo con spigolo 7.5 è: 421.875
>>> |
```

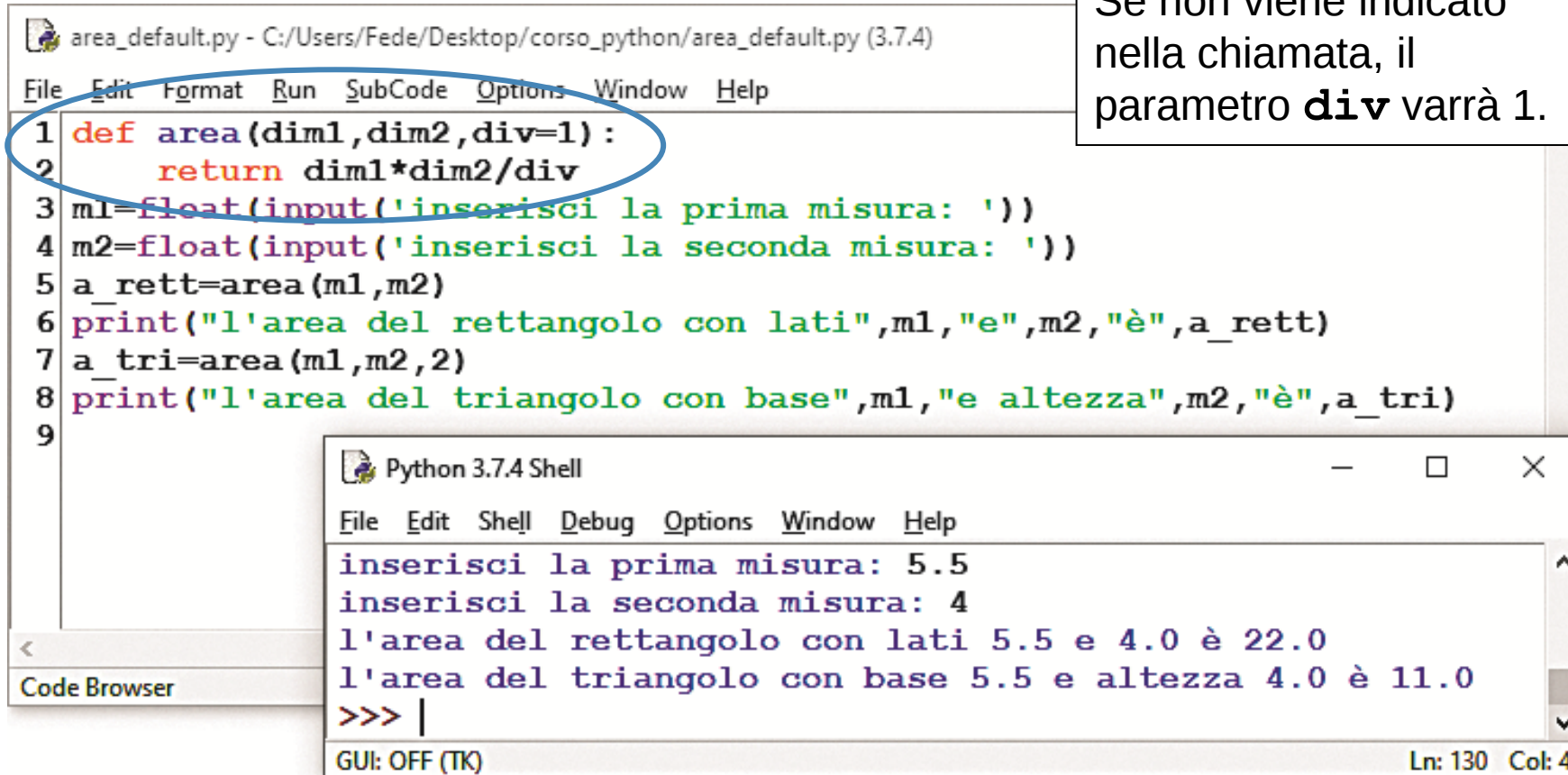
The status bar at the bottom of the shell window indicates 'GUI: OFF (TK)' and 'Ln: 102 Col: 4'.

3.2 Funzioni particolari

Indice

Nell'intestazione di una funzione si possono indicare valori di default per i parametri.

Se non viene indicato nella chiamata, il parametro **div** varrà 1.



The image shows a Python IDE window titled 'area_default.py - C:/Users/Fede/Desktop/corso_python/area_default.py (3.7.4)'. The code defines a function 'area' with parameters 'dim1', 'dim2', and a default parameter 'div=1'. The function returns 'dim1*dim2/div'. Below the function definition, there are two calls to the 'area' function: one for a rectangle (a_rett) and one for a triangle (a_tri). The triangle call uses '2' for the 'div' parameter. The IDE also shows a 'Python 3.7.4 Shell' window with the output of the program: 'inserisci la prima misura: 5.5', 'inserisci la seconda misura: 4', 'l'area del rettangolo con lati 5.5 e 4.0 è 22.0', and 'l'area del triangolo con base 5.5 e altezza 4.0 è 11.0'. The status bar at the bottom indicates 'Ln: 130 Col: 4'.

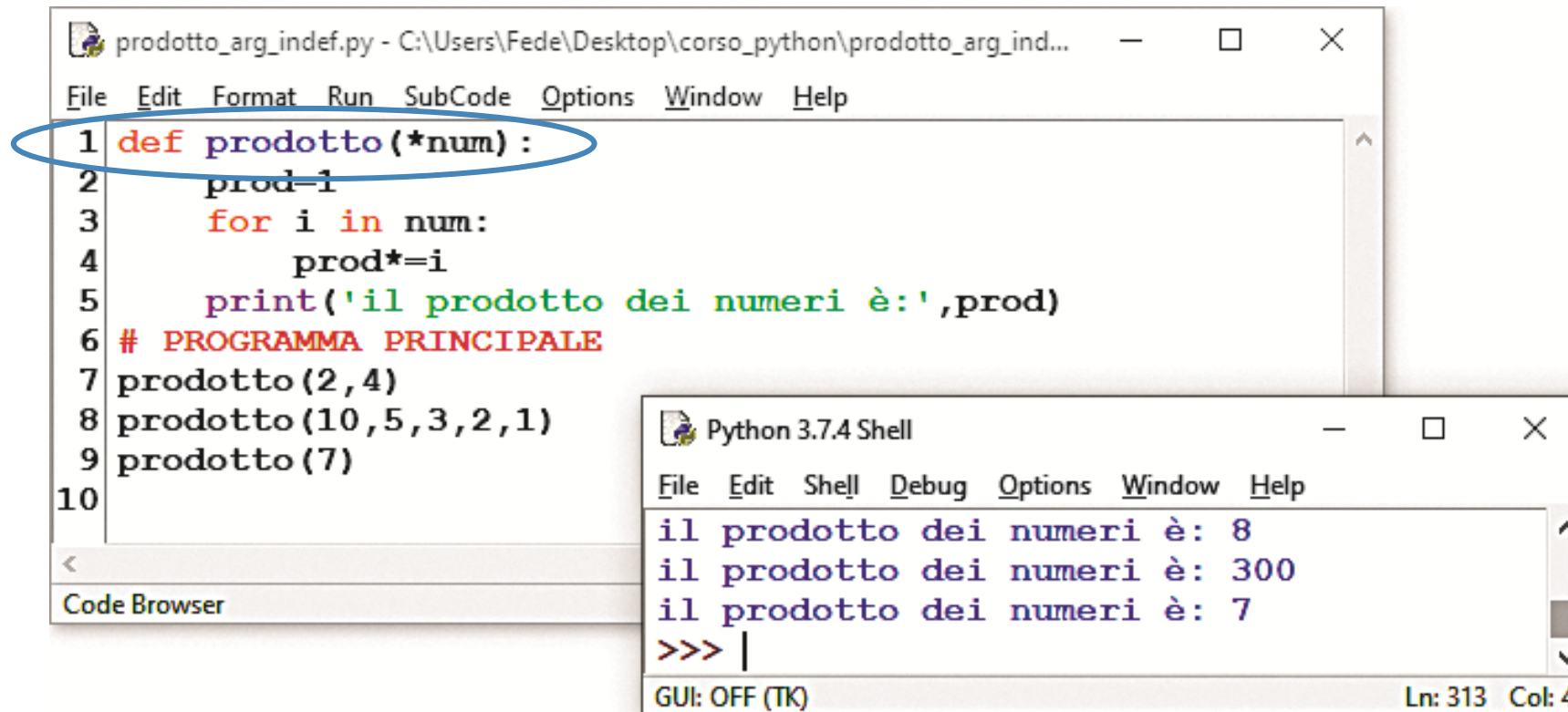
```
1 def area(dim1, dim2, div=1):
2     return dim1*dim2/div
3 m1=float(input('inserisci la prima misura: '))
4 m2=float(input('inserisci la seconda misura: '))
5 a_rett=area(m1,m2)
6 print("l'area del rettangolo con lati",m1,"e",m2,"è",a_rett)
7 a_tri=area(m1,m2,2)
8 print("l'area del triangolo con base",m1,"e altezza",m2,"è",a_tri)
9
```

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
inserisci la prima misura: 5.5
inserisci la seconda misura: 4
l'area del rettangolo con lati 5.5 e 4.0 è 22.0
l'area del triangolo con base 5.5 e altezza 4.0 è 11.0
>>> |
GUI: OFF (TK) Ln: 130 Col: 4
```

3.2 Funzioni particolari

Indice

Una funzione può ricevere un numero variabile di parametri.



The image shows a Python IDE window titled "prodotto_arg_indef.py" with a menu bar (File, Edit, Format, Run, SubCode, Options, Window, Help). The code in the editor is as follows:

```
1 def prodotto(*num) :
2     prod=1
3     for i in num:
4         prod*=i
5     print('il prodotto dei numeri è:',prod)
6 # PROGRAMMA PRINCIPALE
7 prodotto(2,4)
8 prodotto(10,5,3,2,1)
9 prodotto(7)
10
```

The first line of code, `def prodotto(*num) :`, is circled in blue. Below the editor is a "Code Browser" pane. Overlaid on the bottom right is a "Python 3.7.4 Shell" window with a menu bar (File, Edit, Shell, Debug, Options, Window, Help). The shell displays the output of the function calls:

```
il prodotto dei numeri è: 8
il prodotto dei numeri è: 300
il prodotto dei numeri è: 7
>>> |
```

The shell window also shows "GUI: OFF (TK)" and "Ln: 313 Col: 4" at the bottom.

3.3 Gli ambiti di visibilità

Indice

Ogni variabile ha un ambito di visibilità (**scope**) all'interno del quale è visibile o utilizzabile.

VARIABILI LOCALI

Dichiarate nelle funzioni.

Sono viste nella funzione e nelle eventuali funzioni annidate di essa

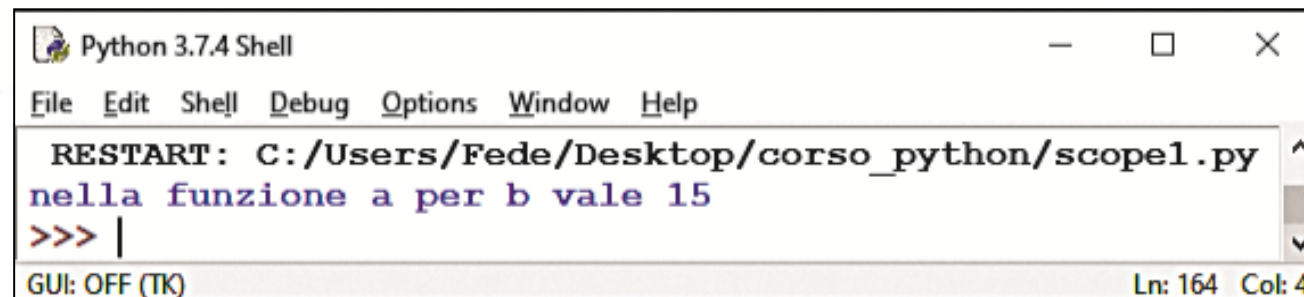
VARIABILI GLOBALI

Dichiarate nel programma principale.

Visibili nell'intero file.

```
def funz() :  
    print("nella funzione a per b vale",a*b)  
# programma principale  
a=5  
b=3  
funz()
```

La funzione può utilizzare le variabili globali.

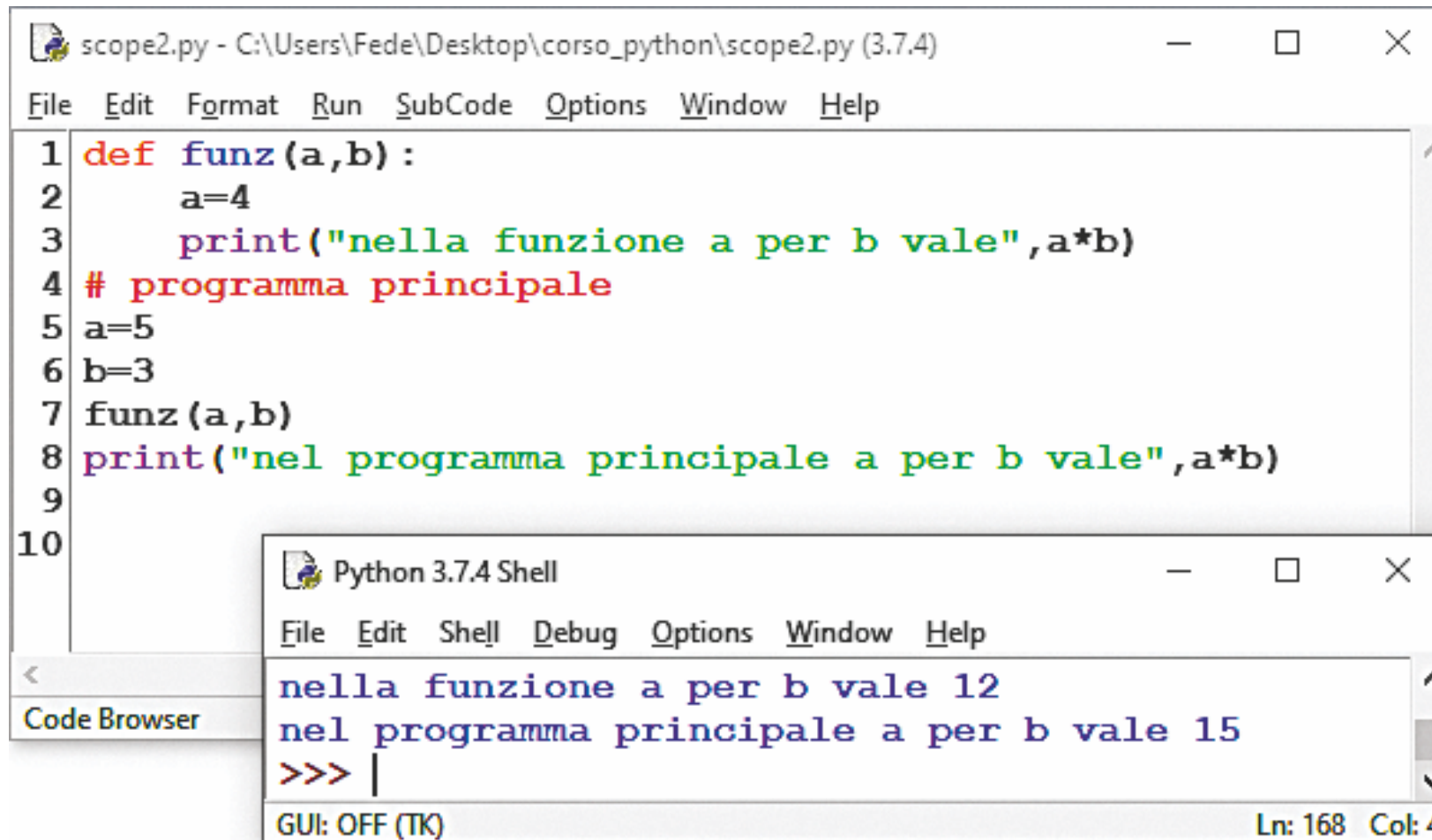


```
Python 3.7.4 Shell  
File Edit Shell Debug Options Window Help  
RESTART: C:/Users/Fede/Desktop/corso_python/scope1.py  
nella funzione a per b vale 15  
>>> |  
GUI: OFF (TK) Ln: 164 Col: 4
```

3.3 Gli ambiti di visibilità

Indice

Nel programma principale non sono visibili le variabili della funzione, che sono locali.



The image shows a screenshot of a Python IDE window titled "scope2.py - C:\Users\Fede\Desktop\corso_python\scope2.py (3.7.4)". The code in the editor is as follows:

```
1 def funz(a,b):  
2     a=4  
3     print("nella funzione a per b vale",a*b)  
4 # programma principale  
5 a=5  
6 b=3  
7 funz(a,b)  
8 print("nel programma principale a per b vale",a*b)  
9  
10
```

Below the code editor, there is a "Python 3.7.4 Shell" window showing the output of the program:

```
nella funzione a per b vale 12  
nel programma principale a per b vale 15  
>>> |
```

The status bar at the bottom of the shell window indicates "GUI: OFF (TK)" and "Ln: 168 Col: 4".

3.3 Gli ambiti di visibilità

Indice

Quando una funzione all'interno del suo blocco di istruzioni definisce un'altra funzione, questa viene detta **funzione annidata** e non è visibile dal programma principale.

```
def esterna():  
    a=5  
    def interna():  
        print("per la funzione interna a vale",a)  
        b=3  
    interna()  
    #print("per la funzione esterna b vale",b)
```

```
esterna()  
interna()
```

Questa istruzione causa un'eccezione, perché la funzione annidata non è riconosciuta dal programma principale.

3.4 I moduli

Indice

I **moduli** sono raccolte di funzioni pronte per l'uso.

Per utilizzare le funzioni contenute in un modulo bisogna importarlo con l'istruzione **import** nomeModulo.

Per chiamare una funzione appartenente a un modulo si usa **nomeModulo.nomeFunzione(argomenti)**.

```
# proviamo a usare il modulo math
import math
# attribuiamo il risultato della funzione a una variabile
var=math.sqrt(5.5)
```

Da un modulo si possono importare singole funzioni.
In questo caso la funzione viene richiamata senza il nome del modulo.

```
from math import hypot

hypot(cateto1,cateto2)
```

3.4 I moduli

Indice

Per abbreviare il codice si possono usare degli **alias** per i nomi delle funzioni e dei moduli.

```
# definiamo un alias  
rr=random.randint  
print('numero a caso tra 1 e 10:',rr(1,10))
```

```
Import random as r
```

Definisce **r** come alias di tutto il modulo **random**.

```
From random import randint as r
```

Definisce **r** come alias della funzione **randint** appartenente al modulo **random**.

3.4 I moduli

Indice

Si possono creare moduli personalizzati da salvare nella cartella che contiene il programma o in una cartella di sistema di Python.

```
# NUOVO MODULO CREATO DAL PROGRAMMATORE
# prima funzione
def inserisci(stringa):
    print(stringa)
    voto = float(input('inserisci il dato: '))
    return voto
# seconda funzione
def media(o,s):
    return (o+s)/2
```

Il modulo potrà essere importato normalmente nel programma principale.

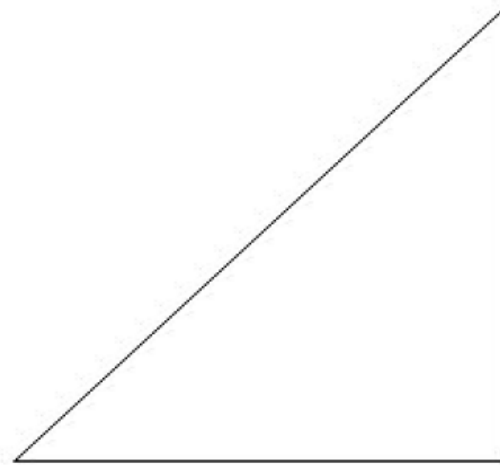
```
print('\t\tTEST! La media tra 4.5 e 7.5 è',mio_modulo.media(4.5,7.5))
```

3.5 Disegnare con `turtle`

Indice

Il modulo `turtle` mette a disposizione del programmatore funzioni per disegnare.

```
import turtle as t
t.showturtle()
t.forward(200)
t.left(90)
t.forward(200)
t.left(135)
import math
t.forward(math.hypot(200,200))
t.hideturtle()
```

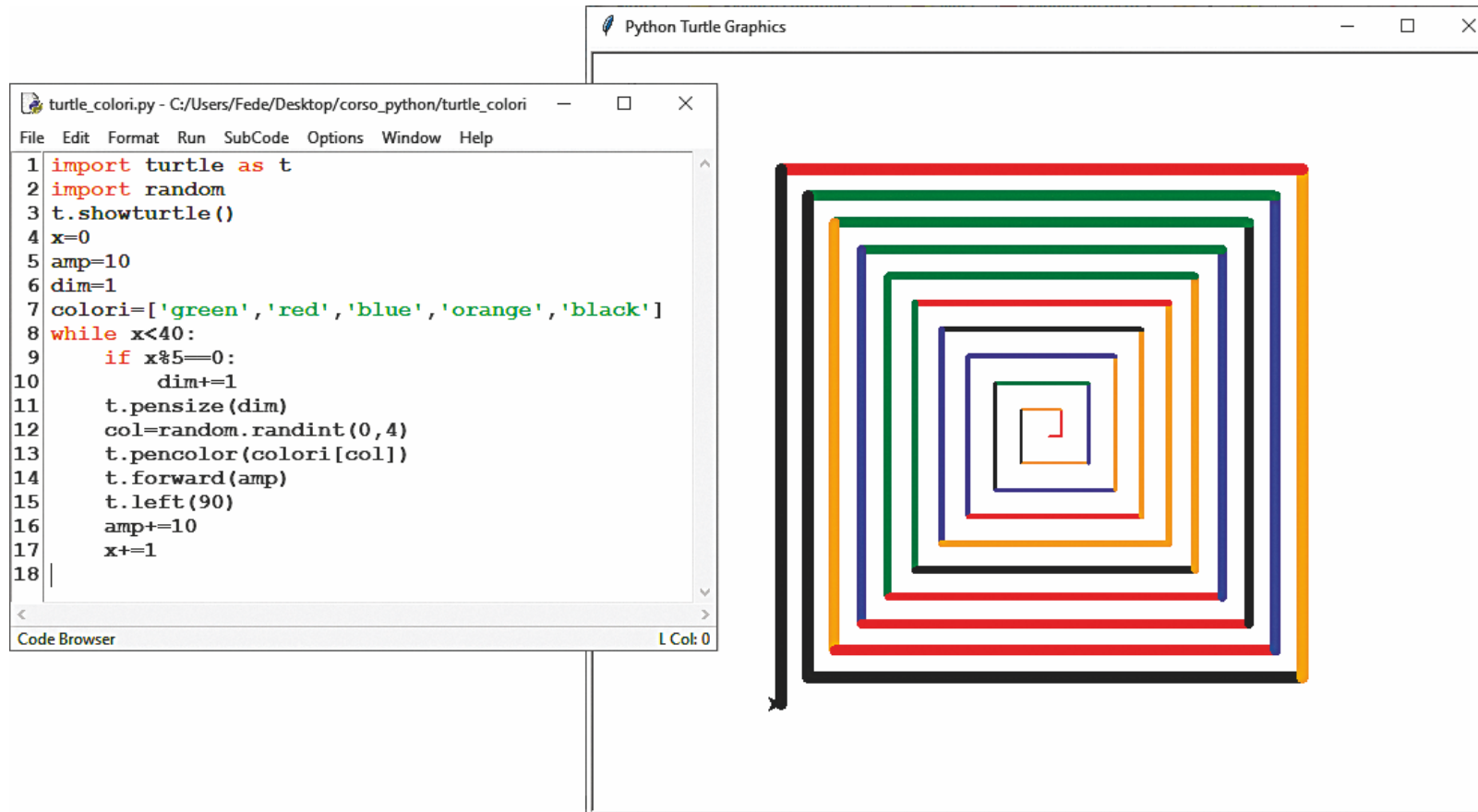


La «tartaruga» rappresenta l'oggetto che scrive

3.5 Disegnare con turtle

Indice

La «tartaruga» ha una penna che può scrivere in vari colori.



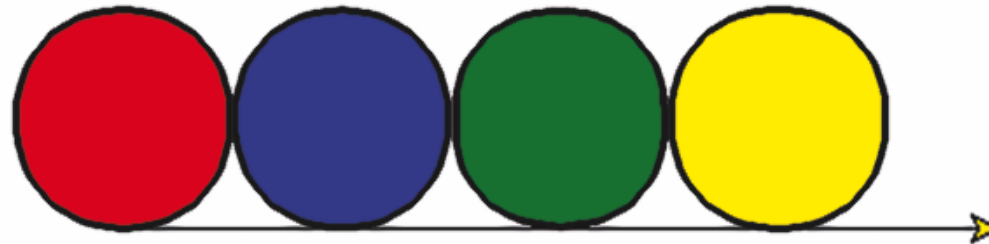
Le funzioni **pensize()** e **pencolor()** permettono di gestire lo spessore del tratto e il suo colore.

3.5 Disegnare con turtle

Indice

E' possibile anche riempire le forme disegnate.

```
import turtle as t
t.showturtle()
r=40
colori=['red', 'blue', 'green', 'yellow']
for col in colori:
    t.fillcolor(col)
    t.begin_fill()
    t.circle(r)
    t.end_fill()
    t.forward(r*2)
```



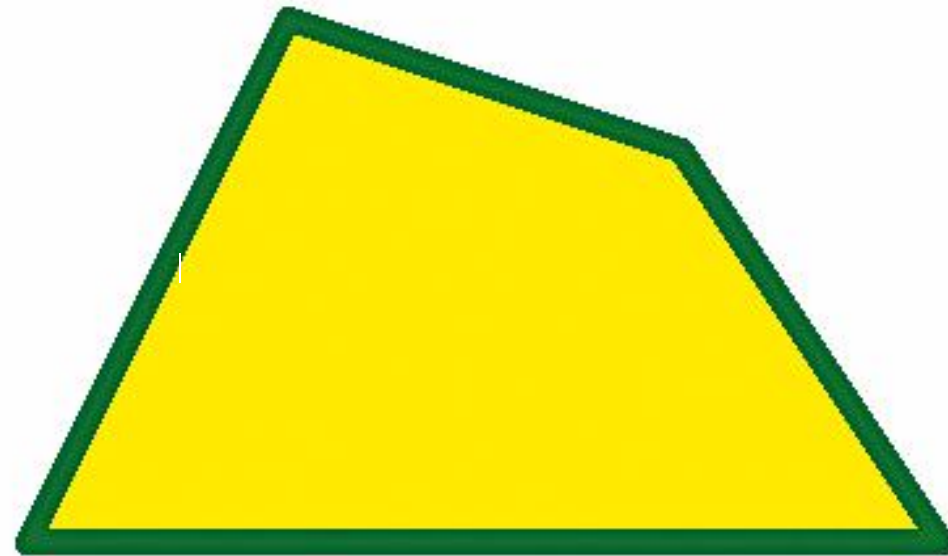
La funzione **fillcolor()** definisce il colore di riempimento, **begin_fill()** specifica che la figura creata in seguito deve essere riempita.

3.5 Disegnare con turtle

Indice

La tartaruga disegna su di un piano cartesiano la cui origine è nel centro della finestra grafica.

```
import turtle as t
t.showturtle()
t.pencolor('green')
t.pensize(10)
t.fillcolor('yellow')
t.begin_fill()
t.penup()
t.goto(100,100)
t.pendown()
t.goto(-50,150)
t.goto(-150,-50)
t.goto(200,-50)
t.goto(100,100)
t.end_fill()
t.hideturtle()
```



La funzione `goto()` permette di portare la tartaruga nella posizione desiderata.