anno 2014-15

Introduzione all'Algoritmica per i Licei

5 – Funzioni.

Elio Giovannetti
Dipartimento di Informatica
Università di Torino

versione 23 febbraio 2015

Riprendiamo gli esempi del massimo e della somma

Il pezzo di programma:

```
max = spese[0]
for elem in spese[1:]: if elem > max: max = elem
print(max)
```

calcola il massimo della particolare lista di nome **spese**.

Se all'interno di un programma più grande dovessimo trovare il massimo di un'altra lista, ad esempio una lista **pesi** di pesi di bagagli da spedire, dovremmo ricopiare il pezzo precedente di programma sostituendo **spese** con **pesi**.

```
max = pesi[0]
for elem in pesi[1:]: if elem > max: max = elem
print(max)
```

Definizione di funzione

Per evitare una tale duplicazione di istruzioni si può definire una funzione, che viene poi applicata alle diverse liste.

```
def listMax(lista):
  max = lista[0]
  for elem in lista[1:]:
    if elem > max: max = elem
  return max
```

...

```
print(listMax(spese))
print(listMax(pesi))
```

...

Che cosa succede se si applica la funzione listMax a una lista di stringhe, ad esempio di parole?

Dà come risultato l'ultima stringa in ordine alfabetico!

Funzione somma degli elementi di una lista.

Analogamente il programma

```
somma = 0
for spesa in spese: somma += spesa
print somma
```

si può immediatamente trasformare in funzione:

```
def somma(lista):
  som = 0
  for elem in lista: som += elem
  return som

 ...
print(somma(spese))
print(somma(pesi))

...
```

Parametri formali e argomenti

Nella definizione di funzione

```
def somma(lista):
  som = 0
  for elem in lista: som += elem
  return som
```

Il nome `lista`, che compare fra le parentesi nella prima riga, si chiama parametro formale della funzione. Il suo nome non ha importanza, si sarebbe potuto usare `lis` o `elementi` o qualunque altro nome. Nelle successive "invocazioni"

```
print(somma(spese))
print(somma(pesi))
```

il parametro formale viene "sostituito" con l'argomento effettivo `spese` oppure `pesi`.

Funzioni e matematica.

Le definizioni di funzione nei linguaggi di programmazione sono analoghe a definizioni di funzione in matematica.

Esempio. La definizione seguente, in matematica, di una funzione f:

$$f(x) = 2x^3 - 5x^2 + 3$$

in Python si traduce:

```
def f(x): return 2*x**3 - 5*x**2 + 3
```

Si possono definire funzioni con più parametri. Esempio:

$$r(x, y) = \sqrt{x^2 + y^2}$$

In Python si scrive:

```
def r(x, y): return sqrt(x**2 + y**2)
```

In realtà in Python la funzione $\sqrt{x^2 + y^2}$ è predefinita e si chiama `hypot(x, y)` (da *hypotenuse*, ὑποτείνουσα)

Funzioni: un esempio grafico.

Definiamo una funzione che, usando la grafica della tartaruga, disegna un poligono regolare di n lati, con lunghezza del lato e colore del perimetro dati come argomenti oltre a n:

```
from turtle import *
def poligono(n, lato, colore):
  color(colore)
  for i in range(n):
    forward(lato)
    right(360/n)
```

Disegniamo un esagono rosso di 100 pixel di lato:

```
poligono(6, 100, 'red')
exitonclick()
```

Passaggio parametri

Il meccanismo del passaggio parametri, cioè la "sostituzione" dei parametri formali con gli argomenti effettivi, è in realtà un meccanismo abbastanza delicato, dipendente dallo specifico linguaggio di programmazione, e che in generale può avvenire in modi diversi a seconda del tipo degli argomenti.

In Python, quando si passa come argomento un numero, la funzione si copia quel numero in una propria cella privata (o, come si dice meglio, locale); quando invece si passa come argomento una lista, la funzione NON si fa una copia locale della lista, ma memorizza solo l'indirizzo della lista (cioè, dove si trova la lista).

Ma al programmatore che cosa interessa tutto ciò?

Lo vediamo nelle prossime slides.

Passaggio di parametri numerici

Supponiamo che nel programma che stiamo scrivendo ci capiti di dover molte volte scambiare il contenuto di due variabili se la seconda variabile è maggiore della prima, e poi in ogni caso di incrementarle entrambe di 1.

Siano ad esempio a e b due variabili. Come si fa?

In Python si può scrivere:

```
if b > a : a, b = b, a
a += 1
b += 1
```

Perché non definire una funzione che esegua tale compito?

```
def scambiaIncr(x, y):
if y > x : x, y = y, x
x += 1
y += 1
```

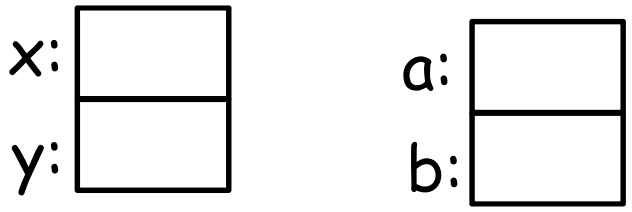
Passaggio di parametri numerici

Ora usiamo la nostra funzione tutte le volte che ci serve:

```
...
scambiaIncr(a, b)
...
scambiaIncr(primo, secondo)
...
```

Non funziona!

La funzione scambiaIncr scambia e incrementa le proprie copie private degli argomenti, non opera sulle celle da cui sono stati presi gli argomenti passati come parametri!



Passaggio di parametri liste

Una funzione che riceve per argomento una lista può cambiare gli elementi della lista. Esempio:

```
def celsiusFahre(tLista):
for i in range(len(tLista)):
  tLista[i] = tLista[i]*1.8 + 32
```

Ora si può applicare tale funzione a liste diverse:

```
tempTorino = [5, 7, -2, 0, 6, 4]
tempPalermo = [12, 11, 13, 17, 15]
celsiusFahre(tempTorino)
celsiusFahre(tempPalermo)
```

se poi si fa print delle due liste, si vede che esse risultano modificate.

Attenzione

Qualcuno potrebbe essere tentato di scrivere:

```
def celsiusFahre(tLista):
for elem in tLista:
  elem = elem*1.8 + 32
```

Non funziona! Perché?

Va a rileggere il Nota Bene al fondo della slide 16 del blocco A3.

Le funzioni sono valori !

Esempio

```
def applyToAll(f, list):
  newlist = [None]*len(list)
  for i in range(len(list)):
    newlist[i] = f(list[i])
  return newlist

applyToAll(sqrt, [2, 9, 25, 64])
```

dà come risultato

```
[1.4142, 3.0, 5.0, 8.0]
```

```
def h(x): return 2*x + 1

applyToAll(h, [2, 9, 25, 64])
```

dà come risultato

```
[5, 19, 51, 129]
```

Rifare gli esercizi sulle liste definendo delle funzioni.

Esercizio 1 (molto facile).
Definire una funzione la quale, data una lista di parole, dà come risultato (NON "scrive") quella di lunghezza massima.

```
def piuLungo(listaNomi):
  …
  return …
```

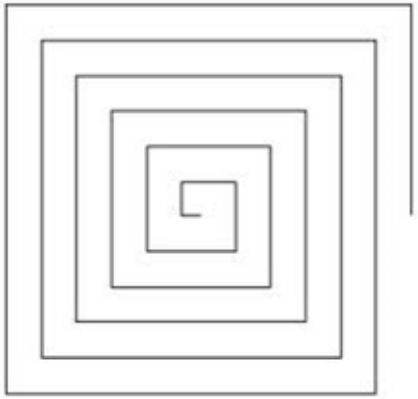
Esercizio 2 (molto facile)
Definire una funzione la quale, data una lista, dà come risultato (NON "scrive") il minimo elemento della lista:

```
def minimo(lis):
  …
 return …
```

Esercizio 3 (elementare)
Definire una funzione la quale, data una lista numerica, dà come risultato il prodotto degli elementi della lista.

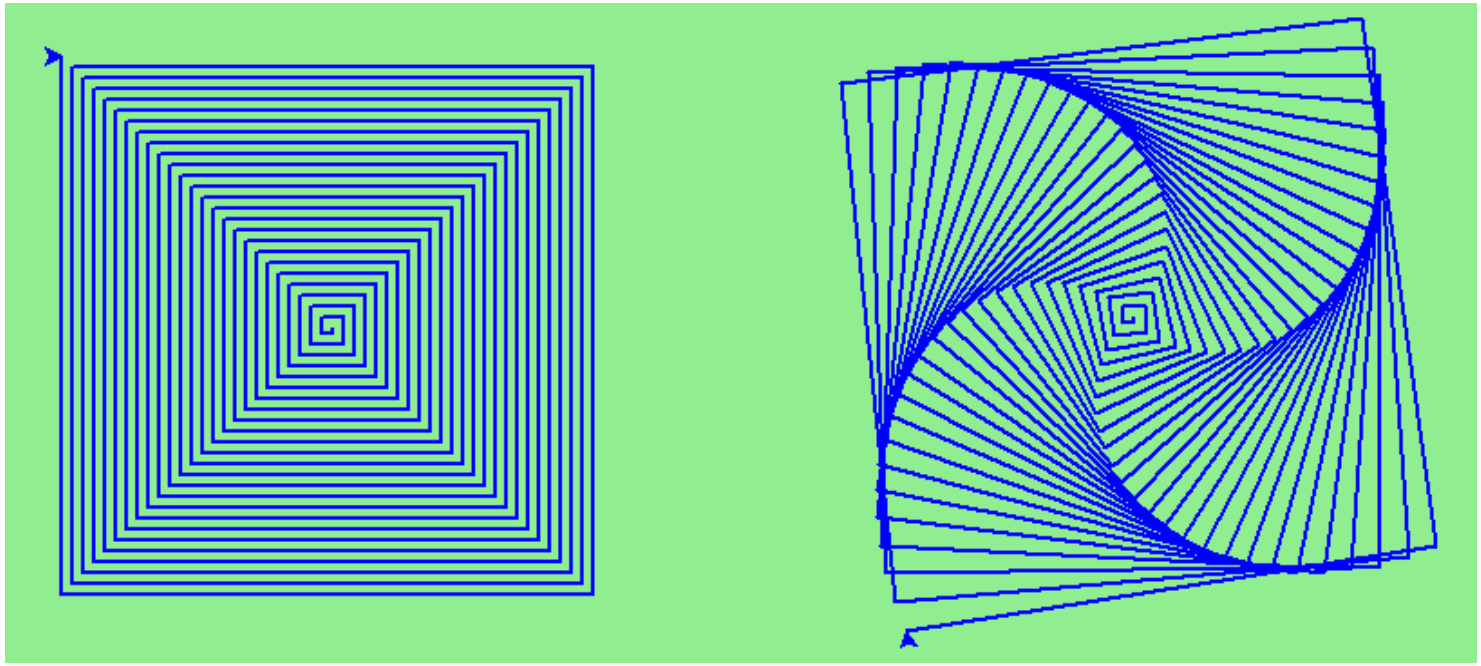
Esercizi

4. Definire una funzione che fa disegnare alla tartaruga una spirale quadrata.



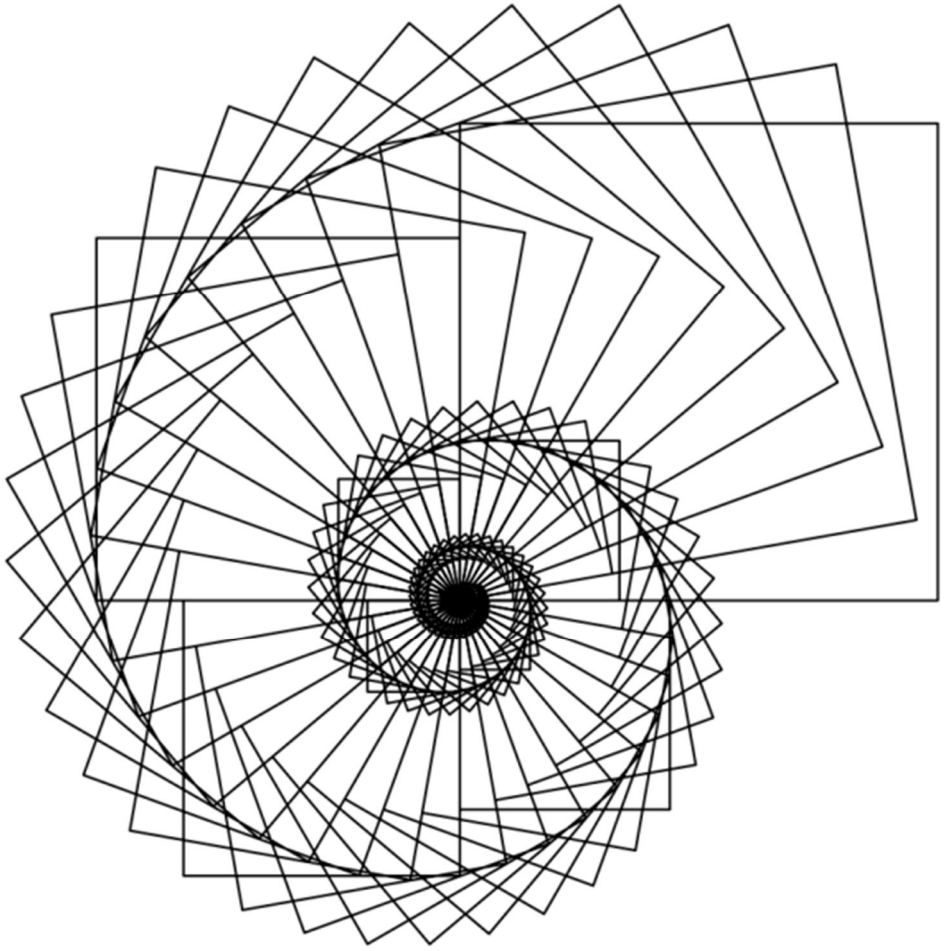
Altri esercizi con la tartaruga.

5. Definire una funzione in grado di disegnare, variando un parametro angolo, le due figure sottostanti.



Altri esercizi con la tartaruga.

6. Definire una funzione in grado di disegnare la figura sottostante (nautilus).



Esercizi

7. Definire una funzione èOrdinata(lista) che dà come risultato `True` se la lista è ordinata, `False` se non lo è. Esempi:

```
print(èOrdinata([3, 5, 7, 11, 17, 17, 23, 25, 25]))
```
deve scrivere sullo schermo `True`.

```
print(èOrdinata([3, 5, 7, 11, 17, 17, 10, 23, 25]))
```
deve scrivere sullo schermo `False`.

```
print(èOrdinata([23]))
```
deve scrivere sullo schermo `True`

```
print(èOrdinata([]))
```
deve scrivere sullo schermo `True`.

```
print(èOrdinata(["abate","abbazia","rio","rione"]))
```
deve scrivere sullo schermo `True`