Strumenti 5.1

Grafica "della tartaruga" (*turtle graphics*)

Nel Paragrafo 2.6 vi è stato presentato il modulo `ezgraphics`, una versione semplificata del più complesso modulo della libreria di Python dedicato alla grafica. Python, però, contiene anche un pacchetto di grafica elementare che può essere utilizzato per creare disegni semplici: il pacchetto Turtle Graphics (*grafica della tartaruga*, vedremo perché), una tecnica di programmazione grafica risalente al 1966, originariamente sviluppata come ausilio dell'insegnamento della programmazione ai bambini. Il pacchetto crea una tartaruga, che rappresenta allegoricamente un oggetto meccanico o cursore, che può essere controllato mediante comandi molto semplici. Mentre la tartaruga si sposta, disegna sullo schermo usando la penna che trasporta. In questa sezione di Strumenti, imparerete i comandi di base per controllare la tartaruga e disegnare semplici figure.

I comandi di base del pacchetto Turtle Graphics

Per creare un disegno, per prima cosa si importa nel programma il pacchetto dedicato alla Turtle Graphics:

```
import turtle
```

La tartaruga trasporta una penna che può essere sollevata oppure abbassata: quando la penna è abbassata la tartaruga disegna sullo schermo mentre si muove. Inizialmente la penna è sollevata, quindi, per iniziare a disegnare, bisogna abbassarla:

```
turtle.pendown()
```

La tartaruga parte dal centro della finestra grafica, rivolta verso est (cioè verso destra), e può essere spostata in avanti (*forward*) o all'indietro (*backward*) rispetto alla posizione che occupa. Ecco come spostare la tartaruga in avanti di 100 pixel:

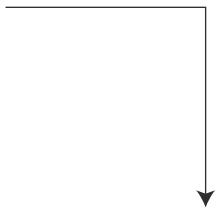
```
turtle.forward(100)
```

Questo spostamento disegna un segmento orizzontale lungo 100 pixel, come si vede nella figura.

La tartaruga si sposta sempre nella direzione verso cui è rivolta. Per modificarla, si può ruotare la tartaruga verso sinistra (cioè in senso antiorario) o verso destra (in senso orario), specificando l'angolo di rotazione, in gradi. Ora ruotiamo la tartaruga di 90 gradi verso destra, in modo che sia diretta verso il basso, dopodiché tracciamo un segmento verticale a partire dalla sua posizione attuale (che è identificata dalla freccia):

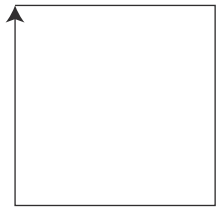
```
turtle.right(90)
turtle.forward(100)
```

Fin qui, come si vede nella figura, abbiamo disegnato due lati consecutivi di un quadrato.



Dando gli stessi comandi altre due volte, otteniamo un quadrato completo:

```
turtle.right(90)
turtle.forward(100)
turtle.right(90)
turtle.forward(100)
```



Terminato il disegno, il programma deve essere sospeso, in attesa che l'utente vi ponga termine (azione che si può ottenere, ad esempio, usando la funzione `input` e ignorando il dato introdotto dall'utente):

```
response = input("Press ENTER to quit.")
```

Se vi dimenticate di scrivere nel programma questa azione di attesa, il programma terminerà immediatamente dopo aver disegnato e il disegno scomparirà, senza lasciare all'utente il tempo di vederlo. In pratica, questa procedura svolge lo stesso compito dell'invocazione del metodo `win.wait()` che usiamo con il pacchetto `ezgraphics`.

Caratteristiche della penna

Il colore e la dimensione della penna posso essere modificati. Per cambiare il colore della penna si può invocare una di queste due funzioni:

```
turtle.color(nomeDiUnColore)
turtle.color(rosso, verde, blu)
```

Le stringhe utilizzabili come nomi dei colori sono le stesse che abbiamo utilizzato con il modulo `ezgraphics` (e riportati nella Tabella 11 del Capitolo 2), mentre, come al solito, i valori dei componenti rosso, verde e blu di un colore sono numeri interi compresi tra 0 e 255, estremi inclusi, secondo il modello RGB.

Immaginiamo, ora, di voler disegnare un segmento verticale di spessore maggiorato e rosso, alla destra del quadrato, come nella Figura 9. Ecco come fare:

1. Modifichiamo il colore e la dimensione della penna

```
turtle.pensize(3)
turtle.color("red")
```

2. Solleviamo la penna. Ricordate, infatti, che se la tartaruga si sposta con la penna abbassata, disegna. In questo caso, invece, abbiamo bisogno di spostare la tartaruga fino al punto iniziale (superiore) del segmento verticale senza che lo spostamento lasci una traccia sullo schermo. Per sollevare la penna si usa questo comando:

```
turtle.penup()
```

3. Ruotiamo la tartaruga verso est, perché poi si possa spostare verso destra. Al momento la tartaruga si trova nel vertice superiore sinistro del quadrato che ha appena disegnato, orientata verso nord, quindi, prima di spostarla, dobbiamo ruotarla verso est.

```
turtle.right(90)
```

4. Spostiamo la tartaruga in modo che, alla fine, si trovi 100 pixel a destra del lato destro del quadrato. Dato che si trova nel vertice superiore sinistro del quadrato, il cui lato è di 100 pixel, lo spostamento verso destra dovrà essere di 200 pixel.

```
turtle.forward(200)
```

5. Ruotiamo ancora a destra la tartaruga (in modo che punti verso il basso), abbassiamo la penna e disegniamo il segmento voluto.

```
turtle.right(90)
turtle.pendown()
turtle.forward(100)
```

Figura 9
La tartaruga disegna una linea di spessore maggiorato

Ecco il programma completo che disegna l'immagine presentata nella Figura 9.

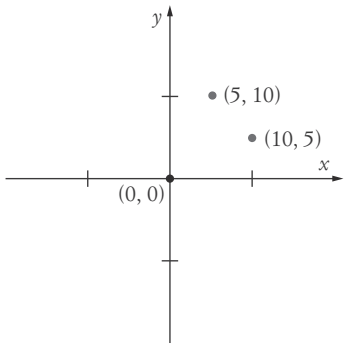
File ch05/toolbox_1/turtlebox.py

```
1  ##
2  #  Questo programma disegna un quadrato e un segmento verticale
3  #  usando il pacchetto Turtle Graphics di Python.
4  #
5  import turtle
6
7  # Disegna un quadrato con il colore e la dimensione predefiniti.
8  turtle.pendown()
9  turtle.forward(100)
10 turtle.right(90)
11 turtle.forward(100)
```

```
12  turtle.right(90)
13  turtle.forward(100)
14  turtle.right(90)
15  turtle.forward(100)
16
17  # Disegna un segmento a destra del quadrato.
18  turtle.pensize(3)
19  turtle.pencolor("red")
20  turtle.penup()
21  turtle.right(90)
22  turtle.forward(200)
23  turtle.right(90)
24  turtle.pendown()
25  turtle.forward(100)
26
27  # Attende che l'utente decida di terminare il programma.
28  response = input("Press ENTER to quit.")
```

Comandi avanzati

La tartaruga si sposta all'interno di un sistema bidimensionale di coordinate cartesiane discrete, assai simile a quello utilizzato in geometria analitica. L'origine del sistema è il punto centrale della finestra grafica e la direzione positiva dell'asse *y* punta verso l'alto (attenzione: è un sistema cartesiano diverso da quello utilizzato dal pacchetto `ezgraphics`!)



Il pacchetto Turtle Graphics mette anche a disposizione comandi che fanno direttamente riferimento al sistema cartesiano. Ad esempio, per portare la tartaruga in un punto di coordinate (x, y), indipendentemente dalla posizione e dalla direzione attuale della tartaruga stessa, si usa il comando:

```
turtle.goto(x, y)
```

mentre il comando seguente porta la tartaruga nell'origine delle coordinate ("a casa", *home*):

```
turtle.home()
```

Naturalmente, se la penna è abbassata, la tartaruga disegna mentre si sposta, come al solito. Per cancellare completamente il contenuto della finestra grafica, si può usare uno di questi due comandi:

```
turtle.clear()
turtle.reset()
```

La funzione `clear` cancella la finestra senza spostare la tartaruga, mentre la funzione `reset` cancella la finestra e riporta la tartaruga nell'origine delle coordinate, ripristinando tutti i parametri della penna al loro stato iniziale.

Sono molti i comandi disponibili nel pacchetto Turtle Graphics: alcuni di quelli più utilizzati sono elencati nella Tabella 1.

Utilizzo di funzioni

Se volete generare un disegno contenente più forme tra loro correlate, è decisamente utile progettare una funzione per ciascun tipo di forma. Ad esempio, questa funzione disegna un quadrato e riporta la tartaruga al suo stato originale (posizione e orientazione):

```
def square(width) :
   turtle.pendown()
   turtle.forward(width)
   turtle.right(90)
   turtle.forward(width)
```

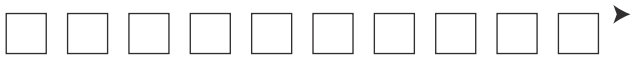
Tabella 1
Funzioni di Turtle Graphics

Funzione	Descrizione
`turtle.backward(`*distanza*`)`	Sposta la tartaruga all'indietro per una certa *distanza*.
`turtle.clear()`	Cancella quanto disegnato nella finestra, senza spostare la tartaruga.
`turtle.forward(`*distanza*`)`	Sposta la tartaruga in avanti per una certa *distanza*.
`turtle.goto(`*x*`, `*y*`)`	Sposta la tartaruga nella posizione (*x*, *y*).
`turtle.heading()`	Restituisce la direzione attuale della tartaruga, in gradi.
`turtle.hideturtle()`	Nasconde la tartaruga.
`turtle.home()`	Sposta la tartaruga nella posizione (0, 0) e la pone in direzione est.
`turtle.left(`*angolo*`)`	Ruota la tartaruga verso sinistra (cioè in senso antiorario) dell'*angolo* indicato.
`turtle.pencolor(`*nomeDiColore*`)` `turtle.pencolor(`*rosso*`, `*verde*`, `*blu*`)`	Imposta il colore della penna. Il colore può essere specificato da un *nomeDiColore* o dai valori delle tre componenti, *rosso*, *verde* e *blu*, nell'intervallo [0 … 255].
`turtle.pendown()`	Abbassa la penna, in modo che disegni mentre la tartaruga si sposta.
`turtle.pensize(`*larghezza*`)`	Imposta a *larghezza* lo spessore della penna.
`turtle.penup()`	Solleva la penna in modo che non disegni mentre la tartaruga si sposta.
`turtle.reset()`	Cancella quanto disegnato, porta la tartaruga nella posizione (0, 0) e la ruota verso est.
`turtle.right(`*angolo*`)`	Ruota la tartaruga verso destra (cioè in senso orario) dell'*angolo* indicato.
`turtle.showturtle()`	Rende visibile la tartaruga.

```
   turtle.right(90)
   turtle.forward(width)
   turtle.right(90)
   turtle.forward(width)
   turtle.right(90)
   turtle.penup()
```

A questo punto è semplice disegnare un numero di quadrati qualsiasi:

```
for i in range(0, 10) :
   square(20)
   turtle.forward(30)
```



Una più semplice implementazione della funzione `square` si ottiene ripetendo per quattro volte le azioni di disegno e di rotazione:

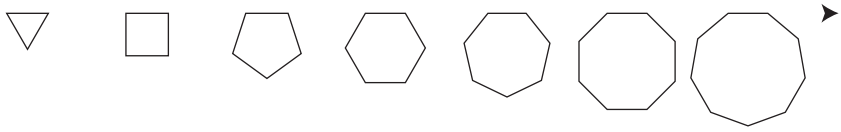
```
def square(width) :
   turtle.pendown()
   for i in range(0, 4) :
      turtle.forward(width)
      turtle.right(90)
   turtle.penup()
```

Questa nuova versione della funzione è facilmente generalizzabile per disegnare pentagoni, esagoni e così via:

```
def regularPolygon(n, width) :
   turtle.pendown()
   for i in range(0, n) :
      turtle.forward(width)
      turtle.right(360 / n)
   turtle.penup()
```

Vediamo un esempio con alcuni poligoni:

```
for n in range(3, 10) :
   regularPolygon(n, 20)
   turtle.forward(60)
```



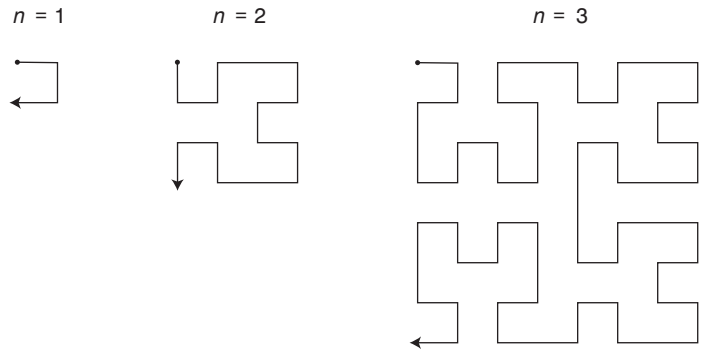
Con questa strategia è ovviamente possibile disegnare qualunque forma che sia composta di segmenti. L'Esercizio P5.42 vi chiederà di scrivere una funzione che disegni una casa, per poi invocarla ripetutamente per disegnare una scena urbana.

Disegno di curve frattali

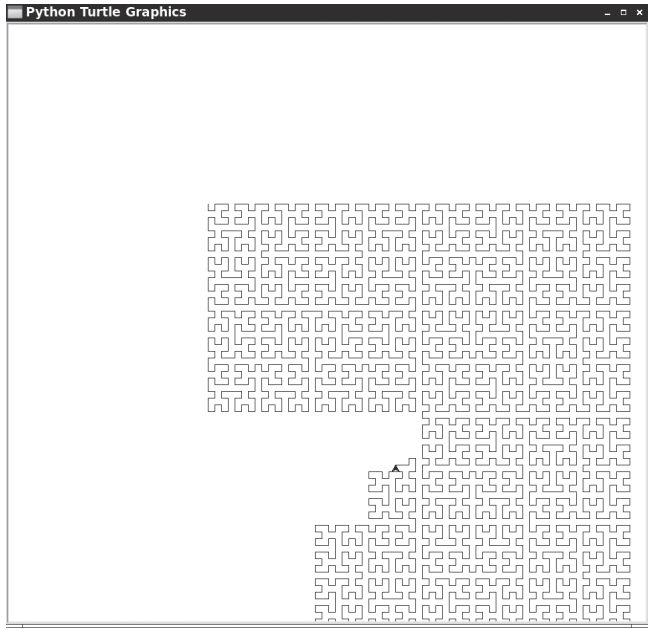
Una "curva frattale" è una curva i cui più piccoli dettagli ricordano la sua forma più ampia. Ad esempio, se osservate la linea costiera di un'isola e, poi, vi concentrare su una sua piccola parte, vedrete che anch'essa ha una forma con rientranze e sporgenze, proprio come la sua forma più estesa.

Faremo in modo che la nostra tartaruga disegni una di queste speciali curve, dotate di "auto-somiglianza": la curva di Hilbert, la cui interessante proprietà è quella di svilupparsi in modo da visitare, all'interno di un quadrato, tutti i punti aventi numeri interi come coordinate. La Figura 10 mostra le curve di Hilbert per $n = 1, 2$ e 3.

Figura 10
Alcune curve di Hilbert



La figura successiva mostra la finestra disegnata dal programma che stiamo per progettare, mentre sta costruendo la curva di Hilbert con $n = 6$ (queste curve prendono il nome dal famoso matematico David Hilbert che le ha inventate con un obiettivo completamente diverso: se si scalano queste curve in modo che si trovino sempre all'interno di un quadrato 1×1, al crescere del valore di n verso infinito si ottiene una curva che "riempie lo spazio", visitando tutti i punti del quadrato).



Questa parte della sezione Strumenti necessita della ricorsione, cioè ci servirà una funzione che invoca se stessa per risolvere un problema più semplice. Se non avete studiato il Paragrafo 5.10, che è facoltativo, dovreste evitare anche questa parte; oppure continuate ed eseguite il programma: è veramente piacevole osservare la tartaruga mentre riempie con cura e metodo un grande quadrato.

Progetteremo, ora, la funzione `hilbert`, che fa in modo che la tartaruga disegni queste curve. Quando $n = 1$, è molto facile:

```
def hilbert(n, turn, distance) :
   if n == 1 :
      turtle.forward(distance)
      turtle.right(turn)
      turtle.forward(distance)
      turtle.right(turn)
      turtle.forward(distance)
   else :
      . . .
```

Invocheremo questa funzione con `turn = 90` oppure, se vogliamo disegnare una curva speculare, con `turn = -90`.

La Figura 11 mostra come si disegna la *n*-esima generazione della curva, a patto di saper disegnare la generazione precedente. Osservate che la generazione *n*-esima contiene quattro curve della generazione precedente, due delle quali sono immagini speculari.

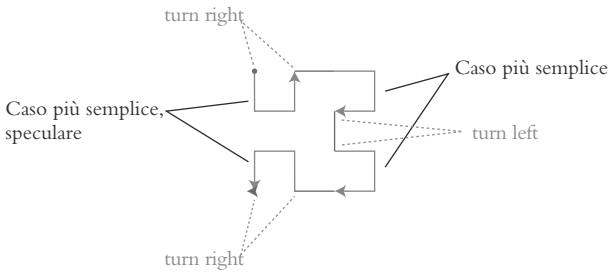


Figura 11
Come si disegna la *n*-esima generazione della curva di Hilbert

Come si vede nella Figura 11, la tartaruga deve anche effettuare alcune rotazioni prima e dopo i passi ricorsivi: ecco il completamento della funzione `hilbert`:

```
else :
   turtle.right(turn)
   hilbert(n - 1, -turn, distance)
   turtle.right(turn)

   turtle.forward(distance)

   hilbert(n - 1, turn, distance)

   turtle.left(turn)
   turtle.forward(distance)
   turtle.left(turn)

   hilbert(n - 1, turn, distance)
```

```
turtle.forward(distance)

turtle.right(turn)
hilbert(n - 1, -turn, distance)
turtle.right(turn)
```

Questo esempio è un'ottima applicazione per il disegno mediante la tartaruga, perché la curva di Hilbert si produce con una sequenza di movimenti della penna e di rotazioni della tartaruga. Ecco il programma completo.

File ch05/toolbox_1/hilbert.py

```
##
# Questo programma disegna una curva di Hilbert usando il
# pacchetto Turtle Graphics.
import turtle

def main() :
   turtle.reset()
   turtle.penup()
   n = 6
   turtle.goto(-n ** 2 * 10 / 2, n ** 2 * 10 / 2)
   turtle.pendown()
   hilbert(n, 90, 10)
   response = input("Press ENTER to quit.")

##  Disegna una generazione della curva di Hilbert.
#  @param n un numero intero che indica la generazione
#  @param turn l'angolo di rotazione della tartaruga
#  @param distance l'avanzamento della tartaruga
#
def hilbert(n, turn, distance) :
   if n == 1 :
      turtle.forward(distance)
      turtle.right(turn)
      turtle.forward(distance)
      turtle.right(turn)
      turtle.forward(distance)
      # Oppure, più elegantemente
      # turtle.right(2 * turn)
   else :
      turtle.right(turn)
      hilbert(n - 1, -turn, distance)
      turtle.right(turn)
      turtle.forward(distance)
      hilbert(n - 1, turn, distance)
      turtle.left(turn)
      turtle.forward(distance)
      turtle.left(turn)
      hilbert(n - 1, turn, distance)
      turtle.forward(distance)
      turtle.right(turn)
      hilbert(n - 1, -turn, distance)
      turtle.right(turn)

main()
```

Riepilogo del capitolo

Funzione, argomento e valore restituito

- Una funzione è una sequenza di istruzioni a cui viene dato un nome.
- Quando una funzione viene invocata, le vengono forniti i suoi argomenti.
- Il valore restituito da una funzione è il risultato che ha calcolato.
- Il valore restituito da una funzione non ha nulla a che vedere con la visualizzazione dei risultati di un programma.

Realizzazione di funzioni

- Nella definizione di una funzione, se ne specifica il nome e si indica una variabile per ciascuno dei suoi argomenti.
- I commenti delle funzioni ne descrivono lo scopo, il significato dei parametri e del valore restituito, oltre a eventuali requisiti specifici.

Passaggio dei parametri

- Le variabili parametro contengono gli argomenti forniti nell'invocazione della funzione.

Valore restituito da una funzione

- L'enunciato `return` termina l'esecuzione dell'invocazione di una funzione e ne trasmette il risultato.
- Trasformate in funzioni le elaborazione che possono essere eseguite più volte.

Funzioni che non restituiscono un valore

- Può darsi che alcune funzioni non restituiscano un valore, ma che visualizzino informazioni.

Progettare funzioni che si possano riutilizzare in più problemi

- La definizione di una funzione consente di eliminare duplicazioni di codice o pseudocodice.
- Progettate le vostre funzioni in modo che siano riutilizzabili, inserendo variabili parametro per quei valori che possono legittimamente cambiare quando la funzione viene usata in un diverso contesto.

Schema progettuale dei miglioramenti successivi

- Per scomporre un problema complesso in problemi più semplici si può procedere per miglioramenti successivi.
- Quando capite che vi serve una funzione, scrivete una descrizione delle sue variabili parametro e del valore restituito.
- Può darsi che una funzione necessiti di altre funzioni più semplici per svolgere il proprio compito.

Ambito di visibilità delle variabili

- L'ambito di visibilità di una variabile è la porzione di programma nella quale essa è visibile.
- Una variabile locale è definita all'interno di una funzione o, più in generale, di un blocco di codice.
- Una variabile globale è definita al di fuori di tutte le funzioni.

Progettare e realizzare un toolkit di funzioni

- Un toolkit è una raccolta di funzioni o classi tra loro correlate, aventi l'obiettivo di risolvere un problema specifico.
- Le funzioni definite in un toolkit, per essere utilizzate in un programma devono essere importate.

Invocazione di funzioni ricorsive e loro progettazione

- Un'elaborazione ricorsiva risolve un problema usando la soluzione del medesimo problema con ingressi più semplici.
- Perché una ricorsione termini, devono esistere casi speciali per i valori di ingresso più semplici.
- L'azione chiave nella ricerca di una soluzione ricorsiva è la scomposizione dei dati di ingresso del problema in insiemi più semplici che possano ancora essere dati di ingresso del medesimo problema.
- Quando progettate una soluzione ricorsiva, non pensate alle molte invocazioni annidate: concentratevi semplicemente sul modo in cui il problema può essere ricondotto a uno un po' più semplice.

Esercizi di ripasso

* **R5.1.** Considerate l'invocazione di funzione `len("black boxes")`. Quanti argomenti riceve la funzione? Qual è il valore restituito?

* **R5.2.** In quale sequenza vengono eseguite le righe del programma `cubes.py` analizzato nel Paragrafo 5.2, a partire dalla prima riga della funzione `main`?

* **R5.3.** Scrivete intestazioni di funzioni, con i relativi commenti, per i problemi seguenti.

a. Calcolare il maggiore tra due numeri interi.
b. Calcolare il minore tra tre numeri in virgola mobile.
c. Verificare se un numero intero è primo, restituendo `True` se lo è e `False` in caso contrario.
d. Verificare se una stringa è contenuta all'interno di un'altra stringa.
e. Calcolare il saldo di un conto corrente dopo un dato numero di anni in cui si accumulano interessi, dato il suo saldo iniziale e il tasso di interesse annuo.
f. Visualizzare il saldo di un conto corrente dopo un dato numero di anni in cui si accumulano interessi, dato il suo saldo iniziale e il tasso di interesse annuo.
g. Visualizzare il calendario di un dato mese e anno.
h. Calcolare il giorno della settimana (sotto forma di stringa, in inglese, come `"Monday"`) corrispondente a un data espressa come giorno, mese e anno.
i. Generare un numero intero casuale compreso tra 1 e *n*.

* **R5.4.** Vero o falso?

a. Una funzione ha esattamente un enunciato `return`.
b. Una funzione ha almeno un enunciato `return`.
c. Una funzione restituisce al massimo un valore.
d. Una funzione che non restituisce un valore non ha un enunciato `return`.
e. Quando viene eseguito un enunciato `return`, la funzione termina immediatamente.
f. Una funzione che non restituisce un valore deve visualizzare un risultato.
g. Una funzione priva di variabili parametro restituisce sempre lo stesso valore.

** **R5.5.** Considerate queste funzioni:

```
def f(x) :
   return g(x) + math.sqrt(h(x))

def g(x) :
   return 4 * h(x)

def h(x) :
   return x * x + k(x) - 1
```

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

Senza compilare ed eseguire un programma, determinare il risultato delle seguenti invocazioni di funzioni.

a. `x1 = f(2)`
b. `x2 = g(h(2))`
c. `x3 = k(g(2) + h(2))`
d. `x4 = f(0) + f(1) + f(2)`
e. `x5 = f(-1) + g(-1) + h(-1) + k(-1)`

★ **R5.6.** Qual è la differenza tra argomento e valore restituito? Quanti argomenti può avere un'invocazione di funzione? E quanti valori restituiti?

★★ **R5.7.** Progettate una funzione che visualizzi un numero in virgola mobile come valore valutario (cioè con un simbolo di $ e due cifre decimali).

a. Indicare come si devono modificare i programmi `ch02/sec05/volume2.py` e `ch04/sec06/investment.py` perché possano usare la funzione qui progettata.
b. Se i programmi dovessero visualizzare una valuta diversa, come l'euro, quali modifiche sarebbero necessarie?

★★ **R5.8 (economia).** Scrivete lo pseudocodice per una funzione che traduca un numero di telefono contenente anche lettere (come spesso si usa negli Stati Uniti per i "numeri verdi", ad esempio 1-800-FLOWERS) in un numero di telefono effettivo, usando le lettere standard che sono presenti nelle tastiere numeriche dei telefoni.

★★ **R5.9.** Descrivete l'ambito di visibilità di ciascuna variabile definita nel programma seguente. Successivamente, determinate cosa viene visualizzato dal programma, senza eseguirlo.

```
 1  a = 0
 2  b = 5
 3
 4  def main() :
 5     global a
 6     global b
 7     i = 10
 8     g = g(i)
 9     print(a + b + i)
10
11  def f(i) :
12     n = 0
13     while n * n <= i :
14        n = n + 1
15     return n - 1
16
17  def g(a) :
18     b = 0
19     for n in range(a) :
20        i = f(n)
21        b = b + 1
22     return b
23
24  main()
```

** **R5.10.** In Python abbiamo visto tre tipi di variabili: variabili globali, variabili parametro e variabili locali. Assegnate alla categoria corretta le variabili utilizzate nell'esercizio precedente.

** **R5.11.** Descrivete la procedura per cucinare le uova strapazzate usando miglioramenti successivi. Decidete cosa fare se non ci sono uova nel frigorifero.

* **R5.12.** Analizzate a mano, passo dopo passo, l'esecuzione della funzione `intName` quando viene invocata con i seguenti argomenti:

a. `5`
b. `12`
c. `21`
d. `301`
e. `324`
f. `0`
g. `-2`

** **R5.13.** Considerate la funzione seguente:

```
def f(a) :
   if a < 0 : return -1
   n = a
   while n > 0 :
      if n % 2 == 0 :  # n è pari
         n = n // 2
      elif n == 1 :
         return 1
      else :
         n = 3 * n + 1
   return 0
```

Analizzate a mano, passo dopo passo, l'esecuzione delle invocazioni `f(-1)`, `f(0)`, `f(1)`, `f(2)`, `f(10)` e `f(100)`.

*** **R5.14.** Considerate la seguente funzione `falseSwap`, che ha l'obiettivo di scambiare il valore di due variabili di tipo intero:

```
def main() :
   x = 3
   y = 4
   falseSwap(x, y)
   print(x, y)

def falseSwap(a, b) :
   temp = a
   a = b
   b = temp

main()
```

Perché la funzione `falseSwap` non scambia i contenuti di `x` e `y`?

*** **R5.15.** Descrivete, mediante pseudocodice, una funzione ricorsiva che visualizzi tutte le sottostringhe di una stringa data. Ad esempio, le sottostringhe della stringa `"rum"` sono la stessa `"rum"`, la stringa vuota e: `"ru"`,`"um"`,`"r"`,`"u"`,`"m"`. Potete fare l'ipotesi che le lettere della stringa siano tutte diverse tra loro.

*** **R5.16.** Descrivete, mediante pseudocodice, una funzione ricorsiva che ordini tutte le lettere di una stringa. Ad esempio, la stringa `"goodbye"` deve essere così ordinata: `"bdegooy"`.

Esercizi di programmazione

* **P5.1.** Scrivete le seguenti funzioni e progettate un programma che le collaudi.

a.
```
def smallest(x, y, z)  # restituisce l'argomento minore
```
b.
```
def average(x, y, z)   # restituisce la media degli argomenti
```

** **P5.2.** Scrivete le seguenti funzioni e progettate un programma che le collaudi.

a.
```
def allTheSame(x, y, z)  # restituisce True se gli argomenti sono tutti uguali
```
b.
```
def allDifferent(x, y, z)  # restituisce True se gli argomenti sono tutti
                           # diversi
```
c.
```
def sorted(x, y, z)  # restituisce True se gli argomenti sono in ordine
                     # crescente
```

** **P5.3.** Scrivete le seguenti funzioni.

a.
```
def firstDigit(n)   # restituisce la prima cifra dell'argomento
```
b.
```
def lastDigit(n)   # restituisce l'ultima cifra dell'argomento
```
c.
```
def digits(n)      # restituisce il numero di cifre dell'argomento
```

Ad esempio, `firstDigit(1729)` restituisce 1, `lastDigit(1729)` restituisce 9 e `digits(1729)` restituisce 4. Progettate un programma che collaudi le funzioni.

* **P5.4.** Scrivete la funzione:

```
def middle(string)
```

che restituisca una stringa contenente il carattere centrale di `string` se la lunghezza di `string` è dispari, oppure i due caratteri centrali se la lunghezza è pari. Ad esempio, `middle("middle")` restituisce `"dd"`.

* **P5.5.** Scrivete la funzione:

```
def repeat(string, n, delim)
```

che restituisca la stringa `string` ripetuta `n` volte, con le ripetizioni separate dalla stringa `delim`. Ad esempio, `repeat("ho", 3, ", ")` restituisce `"ho, ho, ho"`.

** **P5.6.** Scrivete la funzione:

```
def countVowel(string)
```

che restituisca il numero di vocali presenti nella stringa `string`. Le vocali sono le lettere a, e, i, o e u, oltre alle rispettive versioni maiuscole.

** **P5.7.** Scrivete la funzione:

```
def countWords(string)
```

che restituisca il numero di parole presenti nella stringa `string`. Le parole sono sequenze di caratteri separate da spazi. Ad esempio, `countWords("Mary had a little lamb")` restituisce 5.

** **P5.8.** Parliamo di un fenomeno ben conosciuto: la maggior parte delle persone è in grado di leggere un testo le cui parole hanno due caratteri scambiati, a condizione che la prima e l'ultima lettera di ciascuna parola si trovino al loro posto. Ad esempio:

> I dn'ot gvie a dman for a man taht can olny sepll a wrod one way. (Mrak Taiwn)

Scrivete una funzione, `scramble(word)`, che restituisca una versione "mescolata" di una data parola, scambiando a caso due caratteri tra loro, evitando che il primo e l'ultimo carattere della parola siano coinvolti nello scambio. Poi, scrivete un programma che legga parole e le visualizzi dopo averle sottoposte a "mescolamento" tramite tale funzione.

* **P5.9.** Scrivete le funzioni:

```
def sphereVolume(r)
def sphereSurface(r)
def cylinderVolume(r, h)
def cylinderSurface(r, h)
def coneVolume(r, h)
def coneSurface(r, h)
```

che calcolino, rispettivamente, il volume e l'area superficiale di una sfera di raggio `r`, di un cilindro a base circolare di raggio `r` e altezza `h` e di un cono a base circolare di raggio `r` e altezza `h`. Poi, scrivete un programma che chieda all'utente di fornire i valori di `r` e `h`, invochi le sei funzioni e visualizzi i risultati.

** **P5.10.** Scrivete la funzione:

```
def readFloat(prompt)
```

che visualizzi la stringa `prompt`, seguita da uno spazio, poi acquisisca un numero in virgola mobile e lo restituisca, secondo il seguente utilizzo tipico:

```
salary = readFloat("Please enter your salary:")
percentageRaise = readFloat("What percentage raise would you like?")
```

** **P5.11.** Migliorate la funzione `intName` in modo che funzioni correttamente per valori minori di un miliardo.

** **P5.12.** Migliorate la funzione `intName` in modo che funzioni correttamente anche per valori negativi e per lo zero. *Attenzione:* verificate che la funzione "migliorata" non trasformi 20 in `"twenty zero"`.

*** **P5.13.** Per alcuni valori (ad esempio, 20) la funzione `intName` restituisce una stringa contenente uno spazio iniziale (`" twenty"`). Correggete questo difetto e accertatevi che gli spazi vengano inseriti soltanto quando questo è necessario. *Suggerimento.* Questo problema può essere risolto in due modi: si può fare in modo che non vengano mai inseriti spazi iniziali, oppure si possono rimuovere eventuali spazi iniziali dal risultato prima di restituirlo.

*** **P5.14.** Scrivete una funzione, `getTimeName(hours, minutes)`, che restituisca il nome, in inglese, di un istante di tempo, come `"ten minutes past two"`, `"half past three"`, `"a quarter to four"` o `"five o'clock"`. Ipotizzate che il valore di `hours` sia compreso tra 1 e 12.

** **P5.15.** Scrivete una funzione ricorsiva:

```
def reverse(string)
```

che calcoli la stringa inversa di `string`. Ad esempio, `reverse("flow")` deve restituire `"wolf"`. *Suggerimento.* Invertite la sottostringa che inizia dal secondo carattere, poi aggiungete alla fine il primo carattere. Ad esempio, per invertire `"flow"`, per prima cosa invertite `"low"`, ottenendo `"wol"`, poi aggiungete `"f"` alla fine.

** **P5.16.** Scrivete una funzione ricorsiva:

```
def isPalindrome(string)
```

che restituisca `True` se la stringa `string` è un palindromo, cioè è una stringa identica alla propria inversa. Le stringhe `"deed"`, `"rotor"` e `"aibohphobia"` sono esempi di palindromo. *Suggerimento:* una parola è un palindromo se la prima e l'ultima lettera sono uguali e la stringa rimanente è un palindromo.

** **P5.17.** Usate la ricorsione per realizzare una funzione, `find(string, match)`, che verifichi se la stringa `match` è contenuta nella stringa `string`:

```
b = find("Mississippi", "sip")  # Assegna a b il valore True
```

Suggerimento. Se `string` inizia con `match`, la verifica ha successo; altrimenti, considerate la stringa che si ottiene eliminando il primo carattere da `string`.

* **P5.18.** Usate la ricorsione per determinare il numero di cifre che compongono il numero intero `n`. *Suggerimento*. Se `n` è minore di 10, allora ha una sola cifra; altrimenti, ha una cifra in più di `n // 10`.

* **P5.19.** Usate la ricorsione per calcolare a^n, dove n è un numero intero positivo. *Suggerimento*. Se `n` è uguale a 1, allora $a^n = a$; altrimenti, se n è pari, allora $a^n = (a^{n/2})^2$, altrimenti $a^n = a \times a^{n-1}$.

** **P5.20.** *Anni bisestili*. Scrivete la funzione:

```
def isLeapYear(year)
```

che verifichi se un anno è bisestile (*leap year*), cioè se ha 366 giorni, un problema che è già stato descritto nell'Esercizio P3.27. In questo caso, dovete usare enunciati `return` e `if` multipli in modo da restituire il risultato non appena vi è noto.

** **P5.21.** Nell'Esercizio P3.28 vi è stato chiesto di scrivere un programma per convertire un numero nella sua rappresentazione numerica romana. A quel punto del vostro apprendimento, non sapevate come eliminare le porzioni di codice duplicate e, in conseguenza di ciò, il programma era piuttosto lungo. Riscrivete quel programma realizzando e utilizzando la funzione seguente:

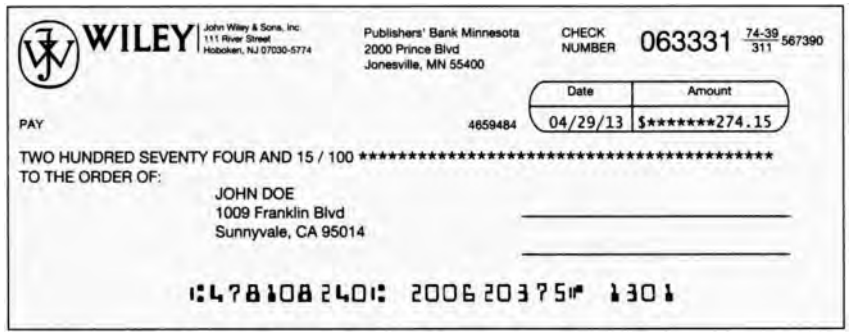
```
def romanDigit(n, one, five, ten)
```

che traduca una singola cifra, usando, come valori per uno, cinque e dieci, le stringhe fornite come parametri. La funzione si può invocare, ad esempio, in questo modo:

```
romanOnes = romanDigit(n % 10, "I", "V", "X")
n = n // 10
romanTens = romanDigit(n % 10, "X", "L", "C")
. . .
```

** **P5.22 (economia).** Scrivete una funzione che calcoli il saldo di un conto bancario dopo che siano trascorsi un dato numeri di anni, a partire da un dato saldo iniziale e con un dato tasso di interesse annuo, accreditando gli interesse annualmente.

** **P5.23 (economia).** Scrivete un programma che visualizzi un assegno di pagamento dello stipendio di un dipendente, simile a quello qui raffigurato. Chiedete all'utente di fornire il nome del dipendente, la paga oraria e il numero di ore lavorate. Se il numero di ore supera 40, le ore in eccesso vengono pagate "una volta e mezza", cioè il 150% della paga oraria. Usate nomi falsi per la banca e per la persona che paga lo stipendio. Procedete per miglioramenti successivi e decomponete la soluzione in più funzioni. Per visualizzare la quantità di dollari in lettere, usate la funzione `intName`.


WILEY John Wiley & Sons, Inc. 111 River Street Hoboken, NJ 07030-5774
Publishers' Bank Minnesota
2000 Prince Blvd
Jonesville, MN 55400
CHECK NUMBER 063331 74-39/311 567390
Date 04/29/13 Amount $*******274.15
PAY 4659484
TWO HUNDRED SEVENTY FOUR AND 15 / 100 **********
TO THE ORDER OF:
JOHN DOE
1009 Franklin Blvd
Sunnyvale, CA 95014
⑆478108240⑆ 2006 203 75⑈ 1301

** **P5.24 (economia).** Scrivete un programma che visualizzi le istruzioni per bere un caffè, facendo domande opportune all'utente ogni volta che c'è qualcosa da decidere. Assegnate a una funzione ciascuno dei singoli compiti in cui avete decomposto il problema, ad esempio in questo modo:

```
def brewCoffee() :
   print("Add water to the coffee maker.")
   print("Put a filter in the coffee maker.")
   grindCoffee()
   print("Put the coffee in the filter.")
   . . .
```

** **P5.25 (economia).** *Codici postali a barre.* Per velocizzare lo smistamento della posta, il servizio postale statunitense (USPS, United States Postal Service) suggerisce alle aziende che inviano grandi volumi di posta di usare un codice a barre che rappresenti il codice postale (*zip code*, un esempio del quale è visibile nella Figura 12).

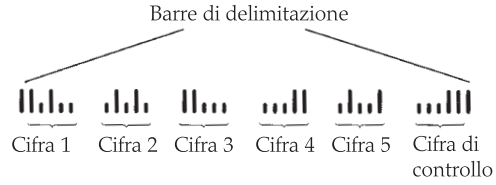
La Figura 13 mostra lo schema di codifica di un codice postale a cinque cifre. A ciascun lato, iniziale e finale, è presente una barra di delimitazione ad altezza piena. Le cinque cifre codificate sono seguite da una cifra di controllo, che viene calcolata in questo modo: si sommano le cinque cifre e si sceglie la cifra di controllo in modo che, sommata alla somma ottenuta, dia come risultato un multiplo di 10. Ad esempio, nel codice postale 95014 la somma delle cifre è 19, per cui la cifra di controllo è 1, per rendere la somma uguale a 20.

Ciascuna cifra del codice postale, così come la cifra di controllo, viene codificata seguendo la tabella qui riportata, dove 0 indica una barra di metà altezza e 1 indica una barra ad altezza piena.

Figura 12
Un codice postale a barre


********** ECRLOT ** CO57
CODE C671RTS2
JOHN DOE CO57
1009 FRANKLIN BLVD
SUNNYVALE CA 95014 – 5143

Figura 13
Codifica a barre con cinque cifre



Cifra	Barra 1 (peso 7)	Barra 2 (peso 4)	Barra 3 (peso 2)	Barra 4 (peso 1)	Barra 5 (peso 0)
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	1	0	0	0	1
8	1	0	0	1	0
9	1	0	1	0	0
0	1	1	0	0	0

Dal codice a barre si può facilmente ricostruire la cifra usando i pesi delle colonne: 7, 4, 2, 1, 0. Ad esempio, la codifica 01100 rappresenta la cifra $0 \times 7 + 1 \times 4 + 1 \times 2 + 0 \times 1 + 0 \times 0 = 6$. L'unica eccezione è la cifra 0, per la quale la formula dei pesi darebbe come risultato 11.

Scrivete un programma che chieda all'utente di fornire un codice postale a cinque cifre e visualizzi il corrispondente codice a barre, usando il carattere : per le barre di metà altezza e ¦ per le barre ad altezza piena. Ad esempio, il codice 95014 diventa:

```
¦¦:¦:::¦:¦:¦¦:::::::¦¦:¦::¦:::¦¦¦
```

Progettate queste funzioni:

```
def printDigit(d)
def printBarCode(zipCode)
```

★★★ **P5.26 (economia).** Scrivete un programma che legga un codice a barre (come descritto nell'esercizio precedente) e visualizzi il codice postale numerico corrispondente. Visualizzate un messaggio d'errore se il codice a barre non è corretto.

★★ **P5.27 (economia).** Scrivete un programma che converta un numero romano, come MCMLXXVIII, nella sua rappresentazione decimale. *Suggerimento*. Per prima cosa, scrivete una funzione che restituisce il valore numerico di ciascuna singola lettera, poi usate l'algoritmo seguente:

```
totale = 0
Finché la stringa s contenente il numero romano non è vuota
   Se valore(primo carattere di s) è almeno uguale a
         valore(secondo carattere di s) oppure s ha lunghezza 1
      Aggiungi valore(primo carattere di s) a totale.
      Elimina il primo carattere di s.
   Altrimenti
      Aggiungi a totale la differenza
         valore(secondo carattere di s) – valore(primo carattere di s)
      Elimina il primo e il secondo carattere di s.
```

★★ **P5.28 (economia).** Un'organizzazione non governativa ha bisogno di un programma per calcolare la quota di sussidio economico da assegnare a ciascuna famiglia bisognosa di assistenza. La formula è la seguente:

- Se il reddito annuo della famiglia è compreso tra $ 30 000 e $ 40 000 e la famiglia ha almeno tre figli, il sussidio è pari a $ 1000 per ogni figlio.
- Se il reddito annuo della famiglia è compreso tra $ 20 000 e $ 30 000 e la famiglia ha almeno due figli, il sussidio è pari a $ 1500 per ogni figlio.
- Se il reddito annuo della famiglia è minore di $ 20 000, il sussidio è pari a $ 2000 per ogni figlio.

Scrivete una funzione che faccia questi calcoli, poi scrivete un programma che chieda all'utente di fornire il reddito annuo e il numero di figli di ciascuna famiglia richiedente il sussidio, visualizzando il valore corrispondentemente restituito dalla funzione. Usate –1 come valore sentinella per terminare l'immissione dei dati.

★★★ **P5.29 (economia).** In un *social network* di internet, un utente ha amici, i quali hanno altri amici, e così via. Vogliamo sapere quante persone possono essere raggiunte da un utente seguendo un dato numero di relazioni di amicizia. Questo numero viene detto, in sociologia, "grado di separazione" e vale: uno per gli amici, due per amici di amici, e così via. Non avendo a disposizione i dati effettivamente rilevati in un social network, useremo semplicemente un numero medio di amici per utente.

Scrivete la funzione ricorsiva:

```
def reachablePeople(degree, averageFriendsPerUser)
```

e usatela in un programma che chieda all'utente il grado desiderato e il numero medio di amici per utente della rete, visualizzando, poi, il numero di persone raggiungibili. Tale numero deve comprendere anche l'utente di partenza.

★★ **P5.30 (economia).** Dal momento che gran parte delle vostre informazioni è oggi memorizzata in internet, è particolarmente importante avere una password sicura. Scrivete un programma che determini la validità di una password, dal punto di vista della sicurezza, secondo queste regole:

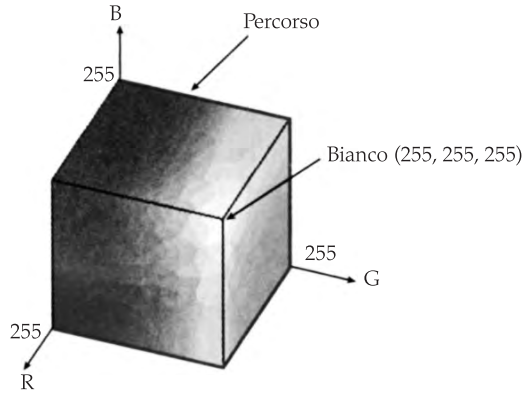
- La password deve contenere almeno 8 caratteri.
- La password deve contenere almeno una lettera maiuscola e almeno una lettera minuscola.
- La password deve contenere almeno una cifra.

Scrivete un programma che chieda all'utente di fornire una password, chiedendo anche di ripeterla per conferma. Se le password non coincidono oppure se le regole non sono soddisfatte, chiedetela di nuovo. Il programma deve contenere una funzione che verifichi se una password è valida.

★★★ **P5.31 (scienze).** State progettando un elemento grafico di un pannello di controllo che visualizza un valore di temperatura compreso tra 0 e 100. Il colore dell'elemento deve variare in modo continuo dal blu (quando la temperatura è 0) al rosso (quando la temperatura è 100). Scrivete la funzione `colorForValue(temperature)` che restituisca il valore che identifica il colore corrispondente alla temperatura data. I colori sono codificati mediante le loro tre componenti, rosso/verde/blu, ciascuna delle quali è un numero compreso tre 0 e 255 (usiamo, rispettivamente, le lettere R, G e B, dove G sta per *green*, cioè verde in inglese). I tre colori si combinano per generare un unico valore di tipo intero, usando la formula seguente:

```
color = 65536 * red + 256 * green + blue
```

Ognuno dei colori intermedi deve essere completamente saturo, cioè deve trovarsi sulla superficie del cubo che rappresenta tutti i colori, lungo il percorso che va dal blu al rosso, passando per i colori ciano (o turchese), verde e giallo.



Dovete sapere come si effettua l'*interpolazione* tra i valori. In generale, se un risultato y deve variare tra c e d al variare del dato di ingresso x tra a e b, si calcola y in questo modo:

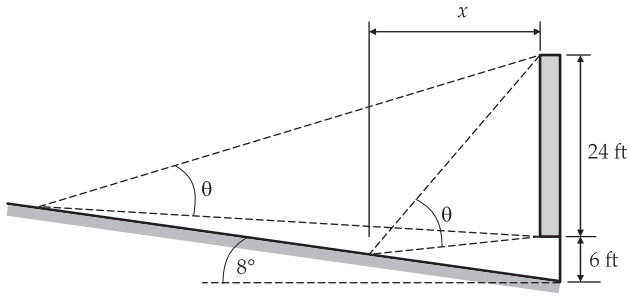
$$z = (x - a)/(b - a)$$
$$y = c(1 - z) + dz$$

Se la temperatura è compresa tra 0 e 25 gradi, interpolate tra i colori blu e turchese, le cui componenti (rosso, verde, blu) sono (0, 0, 255) e (0, 255, 255). Per temperature comprese tra 25 e 50, interpolate tra (0, 255, 255) e (0, 255, 0), che rappresenta ovviamente il colore verde. Usate regole analoghe per gli altri due segmenti del percorso.

Dovete interpolare ciascun colore separatamente dagli altri, per poi combinare i colori interpolati e generare un unico numero intero.

Risolvete il problema usando appropriate funzioni ausiliarie.

★★ **P5.32 (scienze).** In un teatro cinematografico, l'angolo θ secondo il quale uno spettatore vede le immagini sullo schermo dipende dalla sua distanza x dallo schermo stesso. Per un teatro avente le dimensioni annotate nella figura (misurate in piedi, ft.), scrivete una funzione che calcoli l'angolo di visuale per una data distanza.

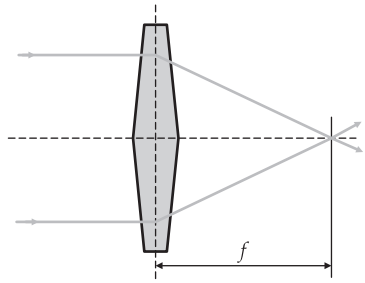


Poi, scrivete una versione più generale della funzione, che sia valida per teatri di dimensioni arbitrarie.

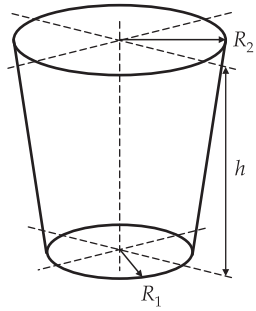
★★ **P5.33 (scienze).** La lunghezza focale efficace f di una lente di spessore d, le cui superfici abbiano raggio di curvatura R_1 e R_2, è data da:

$$\frac{1}{f} = (n-1)\left[\frac{1}{R_1} - \frac{1}{R_2} + \frac{(n-1)d}{nR_1R_2}\right]$$

dove n è l'indice di rifrazione del materiale con cui è costruita la lente. Scrivete una funzione che calcoli f sulla base degli altri parametri forniti come argomenti.



** **P5.34 (scienze).** Un contenitore da laboratorio ha la forma di un tronco di cono:



Scrivete funzioni che ne calcolino il volume e l'area superficiale, usando queste formule:

$$V = \frac{1}{3}\pi h\left(R_1^2 + R_2^2 + R_1R_2\right)$$
$$S = \pi(R_1 + R_2)\sqrt{(R_2 - R_1)^2 + h^2} + \pi R_1^2$$

** **P5.35 (scienze).** Un filo elettrico è un conduttore cilindrico ricoperto da materiale isolante. La resistenza di un tratto di filo è data dalla formula:

$$R = \frac{\rho L}{A} = \frac{4\rho L}{\pi d^2}$$

dove ρ è la resistività del conduttore e L, A e d sono, rispettivamente, la lunghezza, l'area della sezione e il diametro del filo. La resistività del rame è 1.678×10^{-8} Ωm. Il diametro d del filo è solitamente specificato, negli Stati Uniti, da una sigla del tipo AWG n, dove n è un numero intero e AWG è l'acronimo di American Wire Gauge, un sistema standard per la misura del diametro dei fili. Il diametro di un filo AWG n è dato dalla formula:

$$d = 0.127 \times 92^{\frac{36-n}{39}} \text{ mm}$$

Scrivete la funzione:

```
def diameter(wireGauge)
```

che riceva l'indice AWG (il numero intero *n* nella formula) e restituisca il corrispondente diametro del filo. Scrivete, poi, un'altra funzione:

```
def copperWireResistance(length, wireGauge)
```

che riceva la lunghezza e l'indice AWG di un filo di rame e ne restituisca la resistenza. Sapendo che la resistività dell'alluminio è, invece, 2.82×10^{-8} Ωm, scrivete una terza funzione:

```
def aluminumWireResistance(length, wireGauge)
```

che riceva la lunghezza e l'indice AWG di un filo di alluminio e ne restituisca la resistenza.

Infine, scrivete un programma che collaudi queste tre funzioni.

** **P5.36 (scienze).** La spinta aerodinamica (*drag force*) che agisce su un'automobile in movimento è data dalla formula seguente:

$$F_D = \frac{1}{2}\rho v^2 A C_D$$

dove ρ è la densità dell'aria (1.23 kg/m^3), v è la velocità in m/s, A è la proiezione verticale della superficie dell'automobile (circa 2.5 m^2) e C_D è il coefficiente aerodinamico (circa 0.2).

La potenza (misurata in watt, W) necessaria per vincere tale forza è $P = F_D v$, che si trasforma in "cavalli di potenza" (*horse power*) usando l'equivalenza Hp = P/746. Scrivete un programma che acquisisca la velocità dell'automobile (in miglia orarie, mph) e calcoli la potenza necessaria per vincere la spinta aerodinamica, in watt e in cavalli. *Nota*: 1 mph = 0.447 m/s.

*** **P5.37 (grafica).** Aggiungete al *toolkit* per l'elaborazione di immagini visto nel Paragrafo 5.9 una funzione che aggiunga a un'immagine una cornice di un colore assegnato. Aggiornate il programma `processimg.py` in modo che collaudi anche la nuova funzione.

*** **P5.38 (grafica).** Aggiungete al *toolkit* per l'elaborazione di immagini visto nel Paragrafo 5.9 una funzione che dimezzi la dimensione dell'immagine, eliminando i pixel che hanno almeno una coordinata dispari. Aggiornate il programma `processimg.py` in modo che collaudi anche la nuova funzione.

*** **P5.39 (grafica).** Aggiungete al *toolkit* per l'elaborazione di immagini visto nel Paragrafo 5.9 una funzione che raddoppi la dimensione dell'immagine, duplicando ciascun pixel sia orizzontalmente sia verticalmente. Aggiornate il programma `processimg.py` in modo che collaudi anche la nuova funzione.

*** **P5.40 (grafica).** Aggiungete al *toolkit* per l'elaborazione di immagini visto nel Paragrafo 5.9 una funzione che posizioni una accanto all'altra orizzontalmente due copie di un'immagine, e un'altra che lo faccia verticalmente. Aggiornate il programma `processimg.py` in modo che collaudi anche le nuove funzioni.

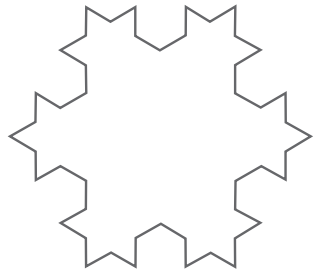
*** **P5.41 (grafica).** Aggiungete al *toolkit* per l'elaborazione di immagini visto nel Paragrafo 5.9 una funzione che trasformi un'immagine in toni di grigio (*grayscale*), seguendo l'approccio visto nell'Esercizio P4.52.

** **P5.42 (strumenti).** Progettate una funzione che usi la Turtle Graphics per disegnare una casa con un dato numero di piani e un dato numero di finestre per ciascun piano. La casa deve anche avere un tetto e una porta d'entrata. Quindi, disegnate una scena urbana invocando più volte la funzione progettata.

*** **P5.43 (strumenti).** Il *fiocco di neve di Koch* (*Koch snowflake*) è una forma definita ricorsivamente. Si parte con un triangolo equilatero:

Poi, si aumenta la dimensione della figura di un fattore tre e si sostituisce ciascun segmento con quattro segmenti, in questo modo:

Ripetendo il procedimento, si ottiene:



Scrivete un programma che usi la Turtle Graphics per disegnare, una dopo l'altra, le iterazioni previste per la costruzione del fiocco di neve di Koch. Dotate la finestra di un pulsante che, quando premuto, produce l'iterazione successiva.