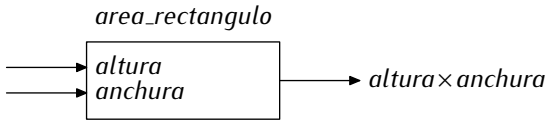
6.2.2. Definición y uso de funciones con varios parámetros

No todas las funciones tienen un sólo parámetro. Vamos a definir ahora una con dos parámetros: una función que devuelve el valor del área de un rectángulo dadas su altura y su anchura:



Importaciones, definiciones de función y programa principal

Los programas que diseñes a partir de ahora tendrán tres «tipos de línea»: importación de módulos (o funciones y variables de módulos), definición de funciones y sentencias del programa principal. En principio puedes alternar líneas de los tres tipos. Mira este programa, por ejemplo,

```
def cuadrado(x):
    return x**2

vector = []
for i in range(3):
    vector.append(float(raw_input('Dame un número: ')))

def suma_cuadrados(v):
    s = 0
    for e in v:
        s += cuadrado(e)
    return s

y = suma_cuadrados(vector)

from math import sqrt

print 'Distancia al origen:', sqrt(y)
```

En él se alternan definiciones de función, importaciones de funciones y sentencias del programa principal, así que resulta difícil hacerse una idea clara de qué hace el programa. No diseñes así tus programas.

Esta otra versión del programa anterior pone en primer lugar las importaciones, a continuación, las funciones y, al final, de un tirón, las sentencias que conforman el programa principal:

```
from math import sqrt

def cuadrado(x):
    return x**2

def suma_cuadrados(v):
    s = 0
    for e in v:
        s += cuadrado(e)
    return s

# Programa principal
vector = []
for i in range(3):
    vector.append(float(raw_input('Dame un número: ')))
y = suma_cuadrados(vector)
print 'Distancia al origen:', sqrt(y)
```

Es mucho más legible. Te recomendamos que sigas siempre esta organización en tus programas. Recuerda que la legibilidad de los programas es uno de los objetivos del programador.

rectangulo.py

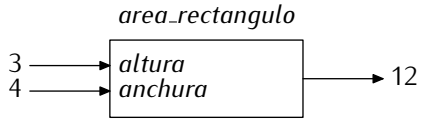
```
def area_rectangulo(altura, anchura):
    return altura * anchura
```

Observa que los diferentes parámetros de una función deben separarse por comas. Al usar la función, los argumentos también deben separarse por comas:

rectangulo_2.py

rectangulo.py

```
1 def area_rectangulo(altura, anchura):
2     return altura * anchura
3
4 print area_rectangulo(3, 4)
```



EJERCICIOS

▶ 290 Define una función que, dado el valor de los tres lados de un triángulo, devuelva la longitud de su perímetro.

▶ 291 Define una función que, dados dos parámetros b y x, devuelva el valor de $\log_b(x)$, es decir, el logaritmo en base b de x.

▶ 292 Diseña una función que devuelva la solución de la ecuación lineal $ax + b = 0$ dados a y b. Si la ecuación tiene infinitas soluciones o no tiene solución alguna, la función lo detectará y devolverá el valor *None*.

▶ 293 Diseña una función que calcule $\sum_{i=a}^{b} i$ dados a y b. Si a es mayor que b, la función devolverá el valor 0.

▶ 294 Diseña una función que calcule $\prod_{i=a}^{b} i$ dados a y b. Si a es mayor que b, la función devolverá el valor 0. Si 0 se encuentra entre a y b, la función devolverá también el valor cero, pero sin necesidad de iterar en un bucle.

▶ 295 Define una función llamada *raiz_n_esima* que devuelva el valor de $\sqrt[n]{x}$. (Nota: recuerda que $\sqrt[n]{x}$ es $x^{1/n}$).

▶ 296 Haz una función que reciba un número de DNI y una letra. La función devolverá *True* si la letra corresponde a ese número de DNI, y *False* en caso contrario. La función debe llamarse *comprueba_letra_dni*.

Si lo deseas, puedes llamar a la función *letra_dni*, desarrollada en el ejercicio 270, desde esta nueva función.

▶ 297 Diseña una función que diga (mediante la devolución de *True* o *False*) si dos números son *amigos*. Dos números son amigos si la suma de los divisores del primero (excluído él) es igual al segundo y viceversa.