6.5. El mecanismo de las llamadas a función

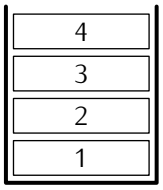
Hemos visto que desde una función podemos llamar a otra función. Desde esta última función podríamos llamar a otra, y desde ésta aún a otra... Cada vez que se produce una llamada, la ejecución del programa principal o de la función «actual» queda suspendida a la espera de que finalice la llamada realizada y prosigue cuando ésta finaliza. ¿Cómo recuerda Python qué funciones están «suspendidas» y en qué orden deben reanudarse?

Por otra parte, hemos visto que si una variable local a una función tiene el mismo nombre que una variable global, durante la ejecución de la función la variable local oculta a la global y su valor es inaccesible. ¿Cómo es posible que al finalizar la ejecución de una función se restaure el valor original? ¿Dónde se había almacenado éste mientras la variable era invisible?

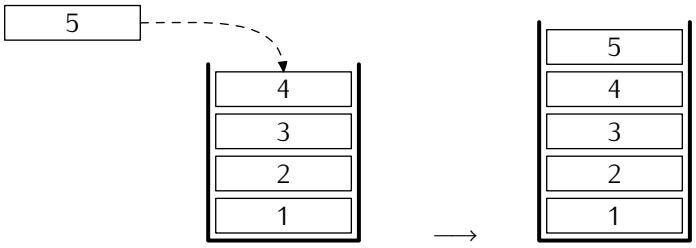
6.5.1. La pila de llamadas a función y el paso de parámetros

Python utiliza internamente una estructura especial de memoria para recordar la información asociada a cada invocación de función: la *pila de llamadas a función*. Una pila es una serie de elementos a la que sólo podemos añadir y eliminar componentes por uno de sus dos extremos: el que denominamos la *cima*.

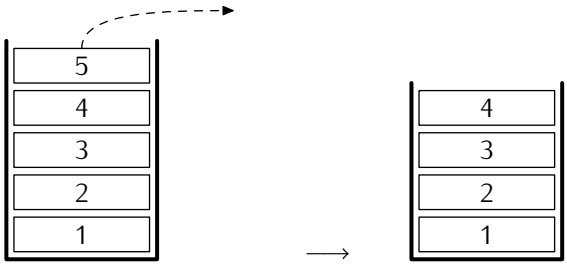
Un montón de platos, por ejemplo, es una pila: sólo puedes añadir un plato poniéndolo encima de la pila (*apilar*) y sólo puedes quitar el plato que está encima (*desapilar*). Aquí tienes una representación gráfica de una pila con cuatro elementos (cada uno de ellos es un número entero).



Sólo podemos añadir nuevos elementos (apilar) por el extremo superior:



Y sólo podemos eliminar el elemento de la cima (desapilar):



Cada activación de una función apila un nuevo componente en la pila de llamadas a función. Dicho componente, que recibe el nombre de *trama de activación*, es una zona de memoria en la que Python dispondrá espacio para los punteros asociados a parámetros, variables locales y otra información que se ha de recordar, como el punto exacto desde el que se efectuó la llamada a la función. Cuando iniciamos la ejecución de un programa, Python reserva una trama especial para las variables globales, así que empezamos con un elemento en la pila. Estudiemos un ejemplo: una ejecución particular del programa *area_y_angulo.py* que reproducimos aquí:

area_y_angulo_4.py — area_y_angulo.py

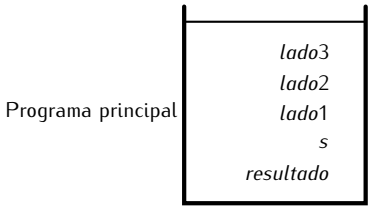
```
1  from math import sqrt, asin, pi
2
3  def area_triangulo(a, b, c):
4      s = (a + b + c) / 2.0
5      return sqrt(s * (s-a) * (s-b) * (s-c))
6
7  def angulo_alfa(a, b, c):
8      s = area_triangulo(a, b, c)
9      return 180 / pi * asin(2.0 * s / (b*c))
10
11 def menu():
12     opcion = 0
13     while opcion != 1 and opcion != 2:
14         print '1) Calcular área del triángulo'
15         print '2) Calcular ángulo opuesto al primer lado'
16         opcion = int(raw_input('Escoge opción: '))
17     return opcion
18
19 lado1 = float(raw_input('Dame lado a: '))
20 lado2 = float(raw_input('Dame lado b: '))
```

```
21 lado3 = float(raw_input('Dame␣lado␣c:␣'))
22
23 s = menu()
24
25 if s == 1:
26     resultado = area_triangulo(lado1, lado2, lado3)
27 else:
28     resultado = angulo_alfa(lado1, lado2, lado3)
29
30 print 'Escogiste␣la␣opción', s
31 print 'El␣resultado␣es:', resultado
```

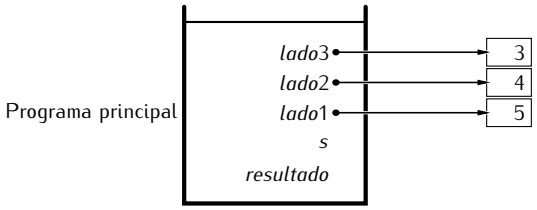
Aquí tienes un pantallazo con el resultado de dicha ejecución:

```
Dame lado a: 5
Dame lado b: 4
Dame lado c: 3
1) Calcular área del triángulo
2) Calcular ángulo opuesto al primer lado
Escoge opción: 2
Escogiste la opción 2
El resultado es: 90.0
```

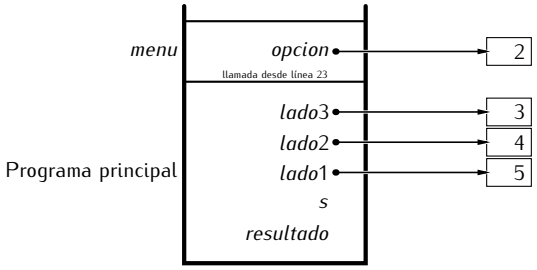
Cuando el programa arranca, Python prepara en la pila el espacio necesario para las variables globales:



El usuario introduce a continuación el valor de *lado*1, *lado*2 y *lado*3. La memoria queda así:



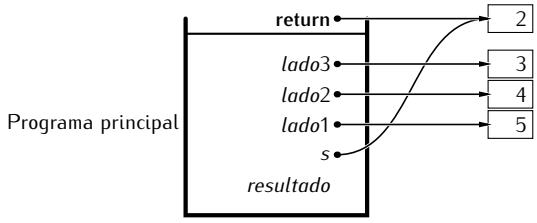
Se produce entonces la llamada a la función *menu*. Python crea una trama de activación para la llamada y la dispone en la cima de la pila. En dicha trama se almacena el valor de *opcion* y el punto desde el que se efectuó la llamada a *menu*. Aquí tienes una representación de la pila cuando el usuario acaba de introducir por teclado la opción seleccionada:



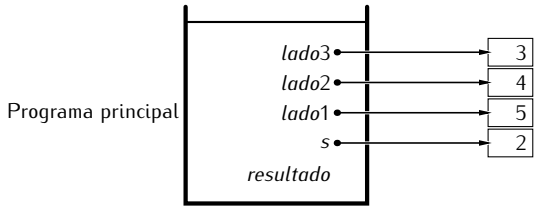
¿Qué ocurre cuando finaliza la ejecución de la función *menu*? Ya no hace falta la trama de activación, así que se desapila, es decir, se elimina. Momentáneamente, no obstante, se mantiene una referencia al objeto devuelto, en este caso, el contenido de la variable *opcion*. Python recuerda en qué línea del programa principal debe continuar (línea 23) porque se había memorizado en la trama de activación. La línea 23 dice:

```
23  s = menu()
```

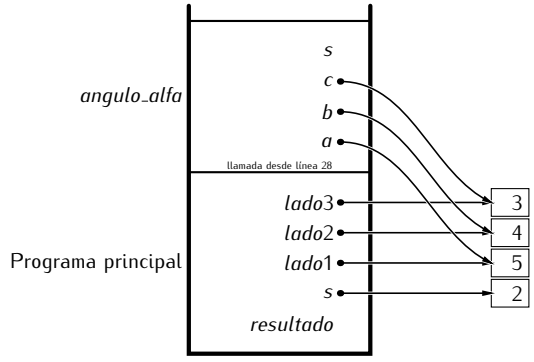
así que la referencia devuelta por *menu* con la sentencia **return** es apuntada ahora por la variable *s*:



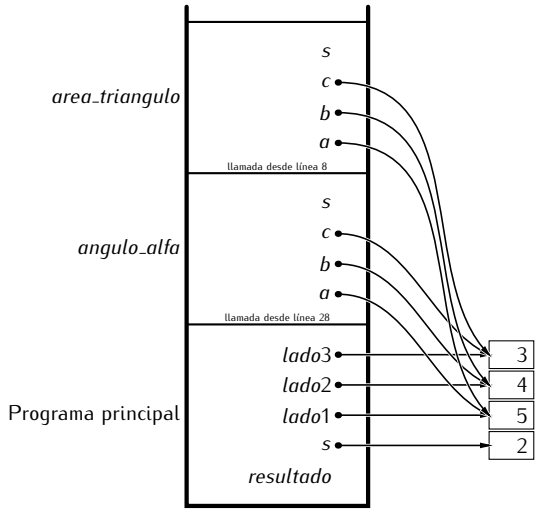
Y ahora que ha desaparecido completamente la trama de activación de *menu*, podemos reorganizar gráficamente los objetos apuntados por cada variable:



La ejecución prosigue y, en la línea 28, se produce una llamada a la función *angulo_alfa*. Se crea entonces una nueva trama de activación en la cima de la pila con espacio para los punteros de los tres parámetros y el de la variable local *s*. A continuación, cada parámetro apunta al correspondiente valor: el parámetro *a* apunta adonde apunta *lado*1, el parámetro *b* adonde *lado*2 y el parámetro *c* adonde *lado*3. Esta acción se denomina *paso de parámetros*.

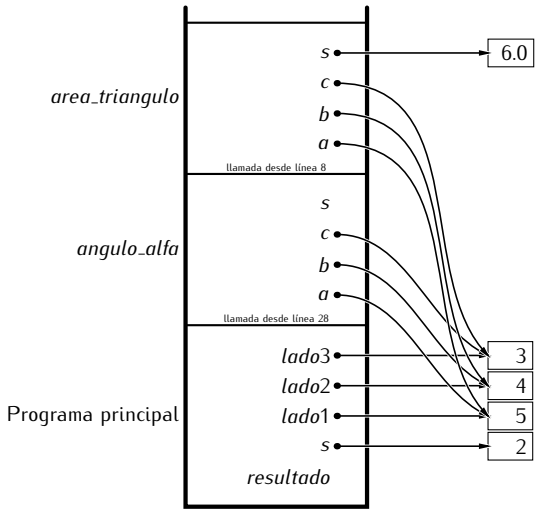


Desde el cuerpo de la función *angulo_alfa* se efectúa una llamada a la función *area_triangulo*, así que se crea una nueva trama de activación. Fíjate en que los identificadores de los parámetros y las variables locales de las dos tramas superiores tienen los mismos nombres, pero residen en espacios de memoria diferentes. En esta nueva imagen puedes ver el estado de la pila en el instante preciso en que se efectúa la llamada a *area_triangulo* y se ha producido el paso de parámetros:

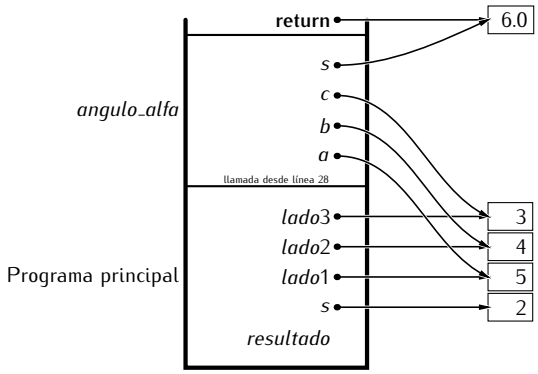


Como puedes comprobar, los parámetros *a*, *b* y *c* de *area_triangulo* apuntan al mismo lugar que los parámetros del mismo nombre de *angulo_alfa*.

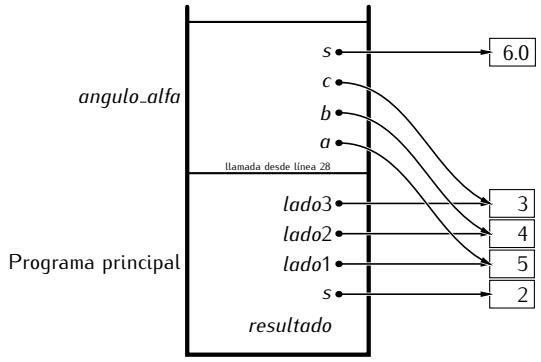
Cuando *area_triangulo* ejecuta su primera línea, la variable local *s* recibe el valor 6.0:



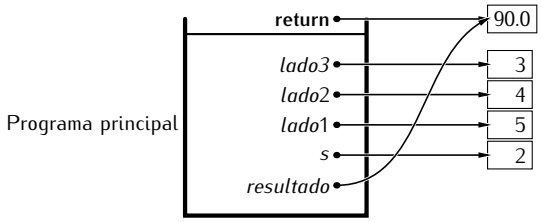
La ejecución de *area_triangulo* finaliza devolviendo el valor del área, que resulta ser 6.0. La variable *s* local a *angulo_alfa* apunta a dicho valor, pues hay una asignación al resultado de la función en la línea 8:



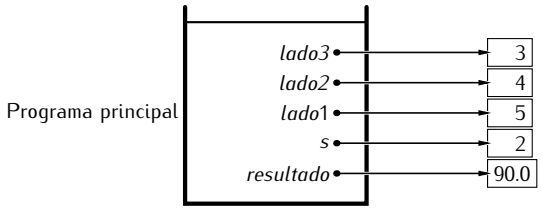
Nuevamente podemos simplificar la figura así:



Y, ahora, una vez finaliza la ejecución de *angulo_alfa*, el valor devuelto (90.0) se almacena en la variable global *resultado*:



El estado final de la pila es, pues, éste:



Observa que la variable *s* de la trama de activación del programa principal siempre ha valido 2, aunque las variables locales del mismo nombre han almacenado diferentes valores a lo largo de la ejecución del programa.