**[Volver al índice](http://www.mclibre.org/consultar/python/index.html)Funciones (1)**

En esta lección se introducen las funciones. Otros conceptos relativos a las funciones se tratan en la lección [Funciones (2)](http://www.mclibre.org/consultar/python/lecciones/python_funciones_2.html):

* [Subrutinas y bibliotecas en Programación](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#L56)
* [Funciones en Python](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#L447)
* [Variables en funciones](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#L3191)
  + [Conflictos de nombres de variables](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#L735)
  + [Conflictos de nombres de variables en Python](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#L770)
  + [Variables locales](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#L807)
  + [Variables libres globales o no locales](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#L1728)
  + [Variables declaradas global o nonlocal](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#L2952)

**Subrutinas y bibliotecas en Programación**

A veces, en un determinado programa es necesario efectuar una misma tarea en distintos puntos del programa y, a menudo, en diferentes programas es necesario efectuar una misma tarea. Para evitar tener que volver a escribir una y otra vez las mismas instrucciones, casi todos los lenguajes de programación permiten agrupar porciones de programa y reutilizarlos en un mismo programa o en diferentes programas.

Estas agrupaciones de código fueron inventadas en 1951 por [Maurice Wilkes](http://en.wikipedia.org/wiki/Maurice_Wilkes), matemático y físico británico que construyó en 1949 el EDSAC, el primer ordenador que almacenaba los programas en la propia circuitería del ordenador. Por sus contribuciones a la informática, Wilkes recibió en 1967 el premio Turing, considerado el premio Nobel de la Informática. Wilkes bautizó a las agrupaciones de código con el nombre de **subrutinas**, pero posteriores lenguajes de programación les han ido dando distintos nombres (subrutinas, procedimientos, funciones, métodos, etc.) y dándoles distintas características.

Las subrutinas tienen muchas ventajas evidentes:

* ahorrar trabajo: si sabemos cómo realizar una tarea, no es necesario volver a programarla una y otra vez.
* facilitar el mantenimiento: si descubrimos errores en una subrutina, sólo hay que corregirlos en un sitio, sin tener que recordar en cuántos sitios utilizamos el algoritmo.
* simplificar el código: la longitud y complejidad del programa se reducen porque las tareas repetitivas ya no aparecen en el cuerpo del programa.
* facilitar la creación de programas: el trabajo se puede dividir entre varios programadores (unos escriben las subrutinas, otros el cuerpo principal del programa, etc.).

Si una subrutina se va a utilizar en un único programa, se suele definir en el propio programa. Pero cuando una subrutina se va a utilizar en varios programas, no es necesario repetirla en cada programa (se perderían algunas de las ventajas comentadas anteriormente), sino que se puede incluir en un fichero aparte al que los programas pueden acceder para recuperar las subrutinas. Estos ficheros reciben el nombre de **bibliotecas** (en español se suele decir mucho **librería**, traduciendo incorrectamente el término inglés **library**).

En Python se utiliza el término **función** para referirse a las subrutinas y el término **módulo** para referirse a las bibliotecas.

[Volver al principio de la página](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html)

**Funciones en Python**

En Python las subrutinas reciben el nombre de funciones.

Las funciones se pueden definir en cualquier punto de un programa, aunque siempre antes de ser utilizadas.

La primera línea de la definición de una función contiene:

* la palabra reservada def
* el nombre de la función (la guía de estilo de Python recomienda escribir todos los caracteres en minúsculas separando las plabras por guiones bajos)
* paréntesis (que pueden incluir los argumentos de la función, como se explica más adelante)

Las instrucciones que forman la función se escriben con sangría con respecto a la primera línea.

Por comodidad, se puede indicar el final de la función con la palabra reservada return (más adelante se explica el uso de esta palabra reservada), aunque no es obligatorio.

El ejemplo siguiente muestra a la izquierda un programa que contiene una función y a la derecha el resultado de la ejecución de ese programa.

**Ejemplo de función**

|  |  |
| --- | --- |
| def licencia():  print("Copyright 2013 Bartolomé Sintes Marco")  print("Licencia CC-BY-SA 3.0")  return  print("Este programa no hace nada interesante.")  licencia()  print("Programa terminado.") | Este programa no hace nada interesante.  Copyright 2013 Bartolomé Sintes Marco  Licencia CC-BY-SA 3.0  Programa terminado. |

Puede ver la ejecución paso a paso de este programa utilizando los iconos de avance y retroceso situados abajo a la derecha.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso11)

**Ejemplo de función - Paso 1**

|  |  |
| --- | --- |
| def licencia():  print("Copyright 2013 Bartolomé Sintes Marco")  print("Licencia CC-BY-SA 3.0")  return  print("Este programa no hace nada interesante.")  licencia()  print("Programa terminado.") |  |

La definición de la función no provoca ninguna salida del programa. Las instrucciones que forman parte de la función se ejecutan únicamente cuando se llama a la función en el programa.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso10)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso12)

**Ejemplo de función - Paso 2**

|  |  |
| --- | --- |
| def licencia():  print("Copyright 2013 Bartolomé Sintes Marco")  print("Licencia CC-BY-SA 3.0")  return  print("Este programa no hace nada interesante.")  licencia()  print("Programa terminado.") | Este programa no hace nada interesante. |

La primera instrucción que se ejecuta es la que está después de la definición de la función.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso11)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso13)

**Ejemplo de función - Paso 3**

|  |  |
| --- | --- |
| def licencia():  print("Copyright 2013 Bartolomé Sintes Marco")  print("Licencia CC-BY-SA 3.0")  return  print("Este programa no hace nada interesante.")  licencia()  print("Programa terminado.") | Este programa no hace nada interesante. |

A continuación se encuentra la llamada a la función. Como tal, la llamada no produce ninguna salida del programa, pero hace que a continuación se ejecuten las instrucciones incluidas en la función.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso12)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso14)

**Ejemplo de función - Paso 4**

|  |  |
| --- | --- |
| def licencia():  print("Copyright 2013 Bartolomé Sintes Marco")  print("Licencia CC-BY-SA 3.0")  return  print("Este programa no hace nada interesante.")  licencia()  print("Programa terminado.") | Este programa no hace nada interesante.  Copyright 2013 Bartolomé Sintes Marco |

Se ejecuta la primera instrucción de la función.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso13)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso15)

**Ejemplo de función - Paso 5**

|  |  |
| --- | --- |
| def licencia():  print("Copyright 2013 Bartolomé Sintes Marco")  print("Licencia CC-BY-SA 3.0")  return  print("Este programa no hace nada interesante.")  licencia()  print("Programa terminado.") | Este programa no hace nada interesante.  Copyright 2013 Bartolomé Sintes Marco  Licencia CC-BY-SA 3.0 |

Se ejecuta la segunda instrucción de la función.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso14)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso16)

**Ejemplo de función - Paso 6**

|  |  |
| --- | --- |
| def licencia():  print("Copyright 2013 Bartolomé Sintes Marco")  print("Licencia CC-BY-SA 3.0")  return  print("Este programa no hace nada interesante.")  licencia()  print("Programa terminado.") | Este programa no hace nada interesante.  Copyright 2013 Bartolomé Sintes Marco  Licencia CC-BY-SA 3.0 |

La instrucción return indica el final de la función y continúa la ejecución del programa tras la llamada a la función.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso15)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso17)

**Ejemplo de función - Paso 7**

|  |  |
| --- | --- |
| def licencia():  print("Copyright 2013 Bartolomé Sintes Marco")  print("Licencia CC-BY-SA 3.0")  return  print("Este programa no hace nada interesante.")  licencia()  print("Programa terminado.") | Este programa no hace nada interesante.  Copyright 2013 Bartolomé Sintes Marco  Licencia CC-BY-SA 3.0  Programa terminado. |

Se ejecuta la última instrucción del programa y el programa termina.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso16)

[Volver al principio de la página](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html)

**Variables en funciones**

**Conflictos de nombres de variables**

Como se ha comentado antes, una de las principales ventajas de las subrutinas es que permiten reutilizar el código. Pero copiar y pegar subrutinas de un programa a otro puede producir lo que se llama un **conflicto de nombres de variables**. En efecto, si la subrutina que pegamos en un programa utiliza alguna variable auxiliar para algún cálculo intermedio y resulta que el programa ya utilizaba una variable con el mismo nombre que esa variable auxiliar, los cambios en la variable que se hagan en la subrutina podrían afectar al resto del programa de forma imprevista.

Para resolver el problema de los conflictos de nombres, los lenguajes de programación limitan lo que se llama el alcance o el ámbito de las variables. Es decir, que los lenguajes de programación permiten que una variable exista únicamente en el interior de una subrutina y no afecte a otras variables de mismo nombre situadas fuera de esa subrutina. Como las subrutinas pueden contener a su vez subrutinas, se suele hablar de niveles: el nivel más alto sería el programa principal, el siguiente nivel serían las subrutinas incluidas en el programa principal y cada vez que hay una subrutina incluida dentro de otra estaríamos bajando un nivel.

El problema es más complicado de lo que parece a primera vista, porque a menudo también nos interesará que una subrutina pueda modificar variables que estén definidas en otros puntos del programa. Así que los lenguajes de programación tienen que establecer mecanismos para aislar las variables y evitar los conflictos de nombres, pero al mismo tiempo deben permitir el acceso a las variables en los casos que así lo quiera el programador.

Aunque cada lenguaje tiene sus particularidades, el mecanismo más habitual se basa en los siguientes principios:

* cada variable pertenece a un ámbito determinado: al programa principal o a una subrutina.
* las variables son completamente inaccesibles en los ámbitos superiores al ámbito al que pertenecen
* las variables pueden ser accesibles o no en ámbitos inferiores al ámbito al que pertenecen (el lenguaje puede permitir al programador elegir o no esa accesibilidad)
* en cada subrutina, las variables que se utilizan pueden ser entonces:
  + variables locales: las que pertenecen al ámbito de la subrutina (y que pueden ser accesibles a niveles inferiores)
  + variables libres: las que pertenecen a ámbitos superiores pero son accesibles en la subrutina

**Conflictos de nombres de variables en Python**

Python sigue estos principios generales, pero con algunas particularidades:

* En los lenguajes tipificados, como se debe declarar las variables que se utilizan, la declaración se aprovecha para indicar si la variable pertenece a la subrutina o procede de un ámbito superior. Pero como Python no es un lenguaje tipificado, el ámbito de pertenencia de la variable debe deducirse del programa siguiendo unas reglas que se detallan más adelante (aunque Python también permite declarar explícitamente el ámbito en los casos en que se quiera un ámbito distinto al determinado por las reglas).
* Python distingue tres tipos de variables: las variables locales y dos tipos de variables libres (globales y no locales):
  + variables locales: las que pertenecen al ámbito de la subrutina (y que pueden ser accesibles a niveles inferiores)
  + variables globales: las que pertenecen al ámbito del programa principal.
  + variables no locales: las que pertenecen a un ámbito superior al de la subrutina, pero que no son globales.

Si el programa contiene solamente funciones que no contienen a su vez funciones, todas las variables libres son variables globales. Pero si el programa contiene una función que a su vez contiene una función, las variables libres de esas "subfunciones" pueden ser globales (si pertenecen al programa principal) o no locales (si pertenecen a la función).

* Para identificar explícitamente las variables globales y no locales se utilizan las palabras reservadas global y nonlocal. Las variables locales no necesitan identificación. La palabra reservada nonlocal se introdujo en Python 3 ([PEP 3104](http://www.python.org/dev/peps/pep-3104/)).

A continuación, se detallan las reglas y situaciones posibles, acompañadas de ejemplos.

**Variables locales**

* Si no se han declarado como globales o no locales, las variables **a las que se asigna valor** en una función se consideran variables **locales**, es decir, sólo existen en la propia función, incluso cuando en el programa exista una variable con el mismo nombre, como muestra el siguiente ejemplo:

**Ejemplo de variable local 1**

|  |  |
| --- | --- |
| def subrutina():  a = 2  print(a)  return  a = 5  subrutina()  print(a) | 2  5 |

Puede ver la ejecución paso a paso de este programa utilizando los iconos de avance y retroceso situados abajo a la derecha.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso31)

**Ejemplo de variable local 1 - Paso 1**

|  |  |
| --- | --- |
| def subrutina():  a = 2  print(a)  return  a = 5  subrutina()  print(a) |  |

La primera instrucción del programa asigna el valor 5 a la variable "a".

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso30)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso32)

**Ejemplo de variable local 1 - Paso 2**

|  |  |
| --- | --- |
| def subrutina():  a = 2  print(a)  return  a = 5  subrutina()  print(a) |  |

A continuación se llama a la función subrutina().

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso31)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso33)

**Ejemplo de variable local 1 - Paso 3**

|  |  |
| --- | --- |
| def subrutina():  a = 2  print(a)  return  a = 5  subrutina()  print(a) |  |

La primera instrucción de la función asigna el valor 2 a la variable "a".

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso32)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso34)

**Ejemplo de variable local 1 - Paso 4**

|  |  |
| --- | --- |
| def subrutina():  a = 2  print(a)  return  a = 5  subrutina()  print(a) | 2 |

La segunda instrucción escribe el valor de la variable "a", es decir, 2.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso33)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso35)

**Ejemplo de variable local 1 - Paso 5**

|  |  |
| --- | --- |
| def subrutina():  a = 2  print(a)  return  a = 5  subrutina()  print(a) | 2 |

La instrucción return indica el final de la función y continúa la ejecución del programa tras la llamada a la función.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso34)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso36)

**Ejemplo de variable local 1 - Paso 6**

|  |  |
| --- | --- |
| def subrutina():  a = 2  print(a)  return  a = 5  subrutina()  print(a) | 2  5 |

La última instrucción del programa escribe de nuevo el valor de "a", pero el valor que se escribe no es 2, sino 5, ya que la variable "a" del programa es distinta de la variable "a" de la función, aunque se llamen igual.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso35)

* Las variables **locales** sólo existen en la propia función y no son accesibles desde niveles superiores, como puede verse en el siguiente ejemplo:

**Ejemplo de variable local 2**

|  |  |
| --- | --- |
| def subrutina():  a = 2  print(a)  return  subrutina()  print(a) | 2  Traceback (most recent call last):  File "ejemplo.py", line 7, in <module>  print(a)  NameError: name 'a' is not defined |

Puede ver la ejecución paso a paso de este programa utilizando los iconos de avance y retroceso situados abajo a la derecha.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso21)

**Ejemplo de variable local 2 - Paso 1**

|  |  |
| --- | --- |
| def subrutina():  a = 2  print(a)  return  subrutina()  print(a) |  |

La primera instrucción del programa llama a la función subrutina().

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso20)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso22)

**Ejemplo de variable local 2 - Paso 2**

|  |  |
| --- | --- |
| def subrutina():  a = 2  print(a)  return  subrutina()  print(a) |  |

La primera instrucción de la función asigna el valor 2 a la variable "a".

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso21)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso23)

**Ejemplo de variable local 2 - Paso 3**

|  |  |
| --- | --- |
| def subrutina():  a = 2  print(a)  return  subrutina()  print(a) | 2 |

La segunda instrucción escribe el valor de la variable "a", es decir, 2.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso22)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso24)

**Ejemplo de variable local 2 - Paso 4**

|  |  |
| --- | --- |
| def subrutina():  a = 2  print(a)  return  subrutina()  print(a) | 2 |

La instrucción return indica el final de la función y continúa la ejecución del programa tras la llamada a la función.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso23)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso25)

**Ejemplo de variable local 2 - Paso 5**

|  |  |
| --- | --- |
| def subrutina():  a = 2  print(a)  return  subrutina()  print(a) | 2  Traceback (most recent call last):  File "ejemplo.py", line 7, in <module>  print(a)  NameError: name 'a' is not defined |

La última instrucción del programa da error porque la variable "a" no existe para el programa, ya que se creó en la función.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso24)

* Si en el interior de una función **se asigna valor** a una variable que no se ha declarado como global o no local, esa variable es **local** a todos los efectos. Por ello el siguiente programa da error:

**Ejemplo de variable local 3**

|  |  |
| --- | --- |
| def subrutina():  print(a)  a = 2  print(a)  return  a = 5  subrutina()  print(a) | Traceback (most recent call last):  File "ejemplo.py", line 8, in <module>  subrutina()  File "ejemplo.py", line 2, in subrutina  print(a)  UnboundLocalError: local variable 'a' referenced before assignment |

Puede ver la ejecución paso a paso de este programa utilizando los iconos de avance y retroceso situados abajo a la derecha.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso41)

**Ejemplo de variable local 3 - Paso 1**

|  |  |
| --- | --- |
| def subrutina():  print(a)  a = 2  print(a)  return  a = 5  subrutina()  print(a) |  |

La primera instrucción del programa asigna el valor 5 a la variable "a".

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso40)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso42)

**Ejemplo de variable local 3 - Paso 2**

|  |  |
| --- | --- |
| def subrutina():  print(a)  a = 2  print(a)  return  a = 5  subrutina()  print(a) |  |

A continuación se llama a la función subrutina().

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso41)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso43)

**Ejemplo de variable local 3 - Paso 3**

|  |  |
| --- | --- |
| def subrutina():  print(a)  a = 2  print(a)  return  a = 5  subrutina()  print(a) | Traceback (most recent call last):  File "ejemplo.py", line 8, in <module>  subrutina()  File "ejemplo.py", line 2, in subrutina  print(a)  UnboundLocalError: local variable 'a' referenced before assignment |

La primera instrucción de la función produce un mensaje de error. El motivo es que en la función se asigna valor a la variable "a" (concretamente en la segunda instrucción) y por tanto Python la considera variable local. Como la primera instrucción quiere imprimir el valor de "a", pero a esa variable todavía no se le ha dado valor en la función (el valor de la variable "a" del programa principal no cuenta pues se trata de variables distintas, aunque se llamen igual), se produce el mensaje de error.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso42)

**Variables libres globales o no locales**

* Si a una variable **no se le asigna valor** en una función, Python la considera **libre** y busca su valor en los niveles superiores de esa función, empezando por el inmediatamente superior y continuando hasta el programa principal. Si a la variable se le asigna valor en algún nivel intermedio la variable se considera **no local** y si se le asigna en el programa principal la variable se considera **global**, como muestran los siguientes ejemplos:
  + En el ejemplo siguiente, la variable libre "a" de la función subrutina() se considera global porque obtiene su valor del programa principal:

**Ejemplo de variable libre global**

|  |  |
| --- | --- |
| def subrutina():  print(a)  return  a = 5  subrutina()  print(a) | 5  5 |

Puede ver la ejecución paso a paso de este programa utilizando los iconos de avance y retroceso situados abajo a la derecha.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso51)

**Ejemplo de variable libre global - Paso 1**

|  |  |
| --- | --- |
| def subrutina():  print(a)  return  a = 5  subrutina()  print(a) |  |

La primera instrucción del programa asigna el valor 5 a la variable "a".

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso50)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso52)

**Ejemplo de variable libre global - Paso 2**

|  |  |
| --- | --- |
| def subrutina():  print(a)  return  a = 5  subrutina()  print(a) |  |

A continuación se llama a la función subrutina().

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso51)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso53)

**Ejemplo de variable libre global - Paso 3**

|  |  |
| --- | --- |
| def subrutina():  print(a)  return  a = 5  subrutina()  print(a) | 5 |

En la función subrutina(), la variable "a" es libre puesto que no se le asigna valor. Su valor se busca en los niveles superiores, por orden. En este caso, el nivel inmediatamente superior es el programa principal. Como en él hay una variable que también se llama "a", Python coge de ella el valor (en este caso, 5) y lo imprime. Para la función subrutina(), la variable "a" es una variable global, porque su valor proviene del programa principal.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso52)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso54)

**Ejemplo de variable libre global - Paso 4**

|  |  |
| --- | --- |
| def subrutina():  print(a)  return  a = 5  subrutina()  print(a) | 5 |

La instrucción return indica el final de la función y continúa la ejecución del programa tras la llamada a la función.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso53)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso55)

**Ejemplo de variable libre global - Paso 5**

|  |  |
| --- | --- |
| def subrutina():  print(a)  return  a = 5  subrutina()  print(a) | 5  5 |

La última instrucción del programa escribe de nuevo el valor de "a", que sigue siendo 5 pues la función no ha modificado el valor de la variable.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso54)

* + En el ejemplo siguiente, la variable libre "a" de la función sub\_subrutina() se considera no local porque obtiene su valor de una función intermedia:

**Ejemplo de variable libre no local**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  print(a)  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) | 3  3  4 |

Puede ver la ejecución paso a paso de este programa utilizando los iconos de avance y retroceso situados abajo a la derecha.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso101)

**Ejemplo de variable libre no local - Paso 1**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  print(a)  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) |  |

La primera instrucción del programa asigna el valor 4 a la variable "a".

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso100)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso102)

**Ejemplo de variable libre no local - Paso 2**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  print(a)  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) |  |

A continuación se llama a la función subrutina().

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso101)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso103)

**Ejemplo de variable libre no local - Paso 3**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  print(a)  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) |  |

La primera instrucción de la función subrutina() asigna el valor 3 a la variable "a", así que esta variable "a" será una variable local.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso102)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso104)

**Ejemplo de variable libre no local - Paso 4**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  print(a)  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) |  |

A continuación se llama a la función sub\_subrutina().

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso103)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso105)

**Ejemplo de variable libre no local - Paso 5**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  print(a)  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) | 3 |

En la función sub\_subrutina(), la variable "a" es libre puesto que no se le asigna valor. Su valor se busca en los niveles superiores, por orden. En este caso, el nivel inmediatamente superior es la función subrutina(). Como en ella hay una variable local que también se llama "a", Python coge de ella el valor (en este caso, 3) y lo imprime. Para la función sub\_subrutina(), la variable "a" es una variable no local, porque su valor proviene de una función intermedia.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso104)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso106)

**Ejemplo de variable libre no local - Paso 6**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  print(a)  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) | 3 |

La instrucción return indica el final de la función y continúa la ejecución del programa tras la llamada a la función.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso105)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso107)

**Ejemplo de variable libre no local - Paso 7**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  print(a)  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) | 3  3 |

La tercera instrucción de la función subrutina() escribe el valor de la variable local "a", es decir, 3.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso106)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso108)

**Ejemplo de variable libre no local - Paso 8**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  print(a)  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) | 3  3 |

La instrucción return indica el final de la función y continúa la ejecución del programa tras la llamada a la función.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso107)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso109)

**Ejemplo de variable libre no local - Paso 9**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  print(a)  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) | 3  3  4 |

La última instrucción del programa escribe el valor de la variable global "a", 4.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso108)

* + Si a una variable que Python considera libre (porque no se le asigna valor en la función) tampoco se le asigna valor en niveles superiores, Python dará un mensaje de error, como muestra el programa siguiente:

**Ejemplo de variable libre no definida**

|  |  |
| --- | --- |
| def subrutina():  print(a)  return  subrutina()  print(a) | Traceback (most recent call last):  File "ejemplo.py", line 5, in <module>  subrutina()  File "ejemplo.py", line 2, in subrutina  print(a)  NameError: global name 'a' is not defined |

Puede ver la ejecución paso a paso de este programa utilizando los iconos de avance y retroceso situados abajo a la derecha.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso61)

**Ejemplo de variable libre no definida - Paso 1**

|  |  |
| --- | --- |
| def subrutina():  print(a)  return  subrutina()  print(a) |  |

La primera instrucción del programa llama a la función subrutina().

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso60)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso62)

**Ejemplo de variable libre no definida - Paso 2**

|  |  |
| --- | --- |
| def subrutina():  print(a)  return  subrutina()  print(a) | Traceback (most recent call last):  File "ejemplo.py", line 5, in <module>  subrutina()  File "ejemplo.py", line 2, in subrutina  print(a)  NameError: global name 'a' is not defined |

Como la función no asigna valor a la variable "a", Python la considera variable libre y busca su valor en niveles superiores, en este caso el programa principal. Pero como en el programa principal tampoco se asigna valor a la variable "a", se produce un error.

**Nota**: El programa hubiera dado error de todas formas en la última instrucción del programa, que también imprimía el valor de la variable "a" no definida.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso61)

**Variables declaradas global o nonlocal**

* Si queremos asignar valor a una variable en una subrutina, pero no queremos que Python la considere local, debemos declararla en la función como global o nonlocal, como muestran los ejemplos siguientes:
  + En el ejemplo siguiente la variable se declara como global, para que su valor sea el del programa principal:

**Ejemplo de variable declarada global**

|  |  |
| --- | --- |
| def subrutina():  global a  print(a)  a = 1  return  a = 5  subrutina()  print(a) | 5  1 |

Puede ver la ejecución paso a paso de este programa utilizando los iconos de avance y retroceso situados abajo a la derecha.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso71)

**Ejemplo de variable declarada global - Paso 1**

|  |  |
| --- | --- |
| def subrutina():  global a  print(a)  a = 1  return  a = 5  subrutina()  print(a) |  |

La primera instrucción del programa asigna el valor 5 a la variable "a".

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso70)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso72)

**Ejemplo de variable declarada global- Paso 2**

|  |  |
| --- | --- |
| def subrutina():  global a  print(a)  a = 1  return  a = 5  subrutina()  print(a) |  |

A continuación se llama a la función subrutina().

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso71)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso73)

**Ejemplo de variable declarada global - Paso 3**

|  |  |
| --- | --- |
| def subrutina():  global a  print(a)  a = 1  return  a = 5  subrutina()  print(a) |  |

Al declarar global la variable "a", la variable "a" de la función y la variable "a" del programa principal son la misma variable.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso72)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso74)

**Ejemplo de variable declarada global - Paso 4**

|  |  |
| --- | --- |
| def subrutina():  global a  print(a)  a = 1  return  a = 5  subrutina()  print(a) | 5 |

La segunda instrucción de la función escribe el valor de la variable "a", es decir, 5, puesto que se trata de la misma variable "a" del programa principal.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso73)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso75)

**Ejemplo de variable declarada global - Paso 5**

|  |  |
| --- | --- |
| def subrutina():  global a  print(a)  a = 1  return  a = 5  subrutina()  print(a) | 5 |

La siguiente instrucción cambia el valor de la variable "a" a 1.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso74)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso76)

**Ejemplo de variable declarada global - Paso 6**

|  |  |
| --- | --- |
| def subrutina():  global a  print(a)  a = 1  return  a = 5  subrutina()  print(a) | 5 |

La instrucción return indica el final de la función y continúa la ejecución del programa tras la llamada a la función.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso75)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso77)

**Ejemplo de variable declarada global - Paso 7**

|  |  |
| --- | --- |
| def subrutina():  global a  print(a)  a = 1  return  a = 5  subrutina()  print(a) | 5  1 |

La última instrucción del programa escribe el valor de la variable global "a", que es ahora 1, puesto que la función ha modificado el valor de la variable.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso76)

* + En el ejemplo siguiente la variable se declara como nonlocal, para que su valor sea el de la función intermedia:

**Ejemplo de variable declarada nonlocal**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  nonlocal a  print(a)  a = 1  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) | 3  1  4 |

Puede ver la ejecución paso a paso de este programa utilizando los iconos de avance y retroceso situados abajo a la derecha.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso901)

**Ejemplo de variable declarada nonlocal - Paso 1**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  nonlocal a  print(a)  a = 1  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) |  |

La primera instrucción del programa asigna el valor 4 a la variable "a".

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso900)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso902)

**Ejemplo de variable declarada nonlocal - Paso 2**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  nonlocal a  print(a)  a = 1  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) |  |

A continuación se llama a la función subrutina().

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso901)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso903)

**Ejemplo de variable declarada nonlocal - Paso 3**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  nonlocal a  print(a)  a = 1  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) |  |

La primera instrucción de la función subrutina() asigna el valor 3 a la variable "a", así que esta variable "a" será una variable local.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso902)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso904)

**Ejemplo de variable declarada nonlocal - Paso 4**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  nonlocal a  print(a)  a = 1  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) |  |

A continuación se llama a la función sub\_subrutina().

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso903)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso905)

**Ejemplo de variable declarada nonlocal - Paso 5**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  nonlocal a  print(a)  a = 1  return    a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) |  |

Al declarar nonlocal la variable "a", Python busca en los niveles superiores, por orden, una variable que también se llame "a", que en este caso se encuentra en la función subrutina(). Python toma el valor de esa variable, es decir, 3.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso904)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso906)

**Ejemplo de variable declarada nonlocal - Paso 6**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  nonlocal a  print(a)  a = 1  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) | 3 |

La primera instrucción de la función sub\_subrutina() escribe el valor de la variable no local "a", es decir, 3.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso905)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso907)

**Ejemplo de variable declarada nonlocal - Paso 7**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  nonlocal a  print(a)  a = 1  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) | 3 |

La segunda instrucción de la función sub\_subrutina() cambia el valor de la variable "a" a 1.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso906)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso908)

**Ejemplo de variable declarada nonlocal - Paso 8**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  nonlocal a  print(a)  a = 1  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) | 3 |

La instrucción return indica el final de la función y continúa la ejecución del programa tras la llamada a la función.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso907)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso909)

**Ejemplo de variable declarada nonlocal - Paso 9**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  nonlocal a  print(a)  a = 1  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) | 3  1 |

La tercera instrucción de la función subrutina() escribe el valor de la variable local "a", es decir, 1.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso908)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso910)

**Ejemplo de variable declarada nonlocal - Paso 10**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  nonlocal a  print(a)  a = 1  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) | 3  1 |

La instrucción return indica el final de la función y continúa la ejecución del programa tras la llamada a la función.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso909)[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso911)

**Ejemplo de variable declarada nonlocal - Paso 11**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  nonlocal a  print(a)  a = 1  return  a = 3  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) | 3  1  4 |

La última instrucción del programa escribe el valor de la variable global "a", que sigue siendo 4, puesto que ninguna función la ha modificado.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso910)

* + Si a una variable declarada global o nonlocal en una función no se le asigna valor en el nivel superior correspondiente, Python dará un error de sintaxis, como muestra el programa siguiente:

**Ejemplo de variable declarada nonlocal no definida**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  nonlocal a  print(a)  a = 1  return  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) | Python - Error de sintaxis |

Puede ver la ejecución paso a paso de este programa utilizando los iconos de avance y retroceso situados abajo a la derecha.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso1001)

**Ejemplo de variable declarada nonlocal no definida - Paso 1**

|  |  |
| --- | --- |
| def subrutina():  def sub\_subrutina():  nonlocal a  print(a)  a = 1  return  sub\_subrutina()  print(a)  return  a = 4  subrutina()  print(a) | Python - Error de sintaxis |

El programa no llega a ejecutarse puesto que Python detecta un error de sintaxis.

* + - La variable "a" de la función subrutina() es una variable libre global. Es una variable libre porque no está definida en la función. Python busca su valor en un nivel superior, que para esa función es el programa principal. En el programa principal sí que hay una variable "a", por lo que la variable "a" de la función subrutina() se considera global. Python tomaría su valor de la variable "a" del programa principal.
    - La variable "a" de la función sub\_subrutina() está declarada como nonlocal, por lo que Python busca su valor en una función superior que no sea el programa principal. El problema es que la variable "a" de la función subrutina() es global, así que Python no la puede tomar para dar valor a la variable "a" de la función sub\_subrutina().
    - Ante la imposibilidad de dar valor a la variable "a" de la función sub\_subrutina() se produce un error de sintaxis.

[](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html#Paso1000)

[Volver al principio de la página](http://www.mclibre.org/consultar/python/lecciones/python_funciones_1.html)

[Licencia Creative Commons](http://creativecommons.org/licenses/by-sa/4.0/)  
**Introducción a la programación con Python** por [Bartolomé Sintes Marco](http://www.mclibre.org/)  
se distribuye bajo una [Licencia Creative Commons Atribución-CompartirIgual 4.0 Internacional](http://creativecommons.org/licenses/by-sa/4.0/).  
Última modificación de esta página: 26 de marzo de 2013