INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones.

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones.

Las **funciones** son un elemento muy utilizado en la programación. Empaquetan y 'aíslan' del resto del programa, una parte de código que realiza alguna **tarea específica** y puede **retornar un valor**.

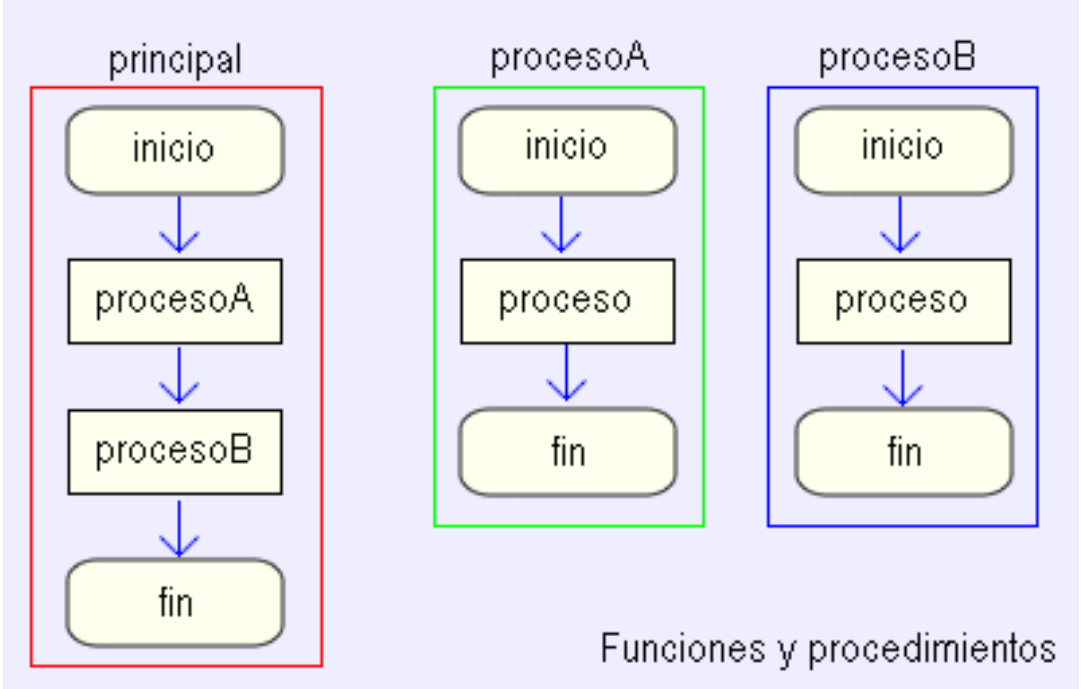
Son por tanto un **conjunto de instrucciones que ejecutan una tarea determinada** y que hemos encapsulado en un formato estándar para que nos sea muy sencillo de manipular y reutilizar.

Una función es una parte de un programa (**subrutina**) con un nombre, que puede ser invocada **(llamada a ejecución**) desde otras partes tantas veces como se desee.

INTRODUCCIÓN A LA PROGRAMACIÓN

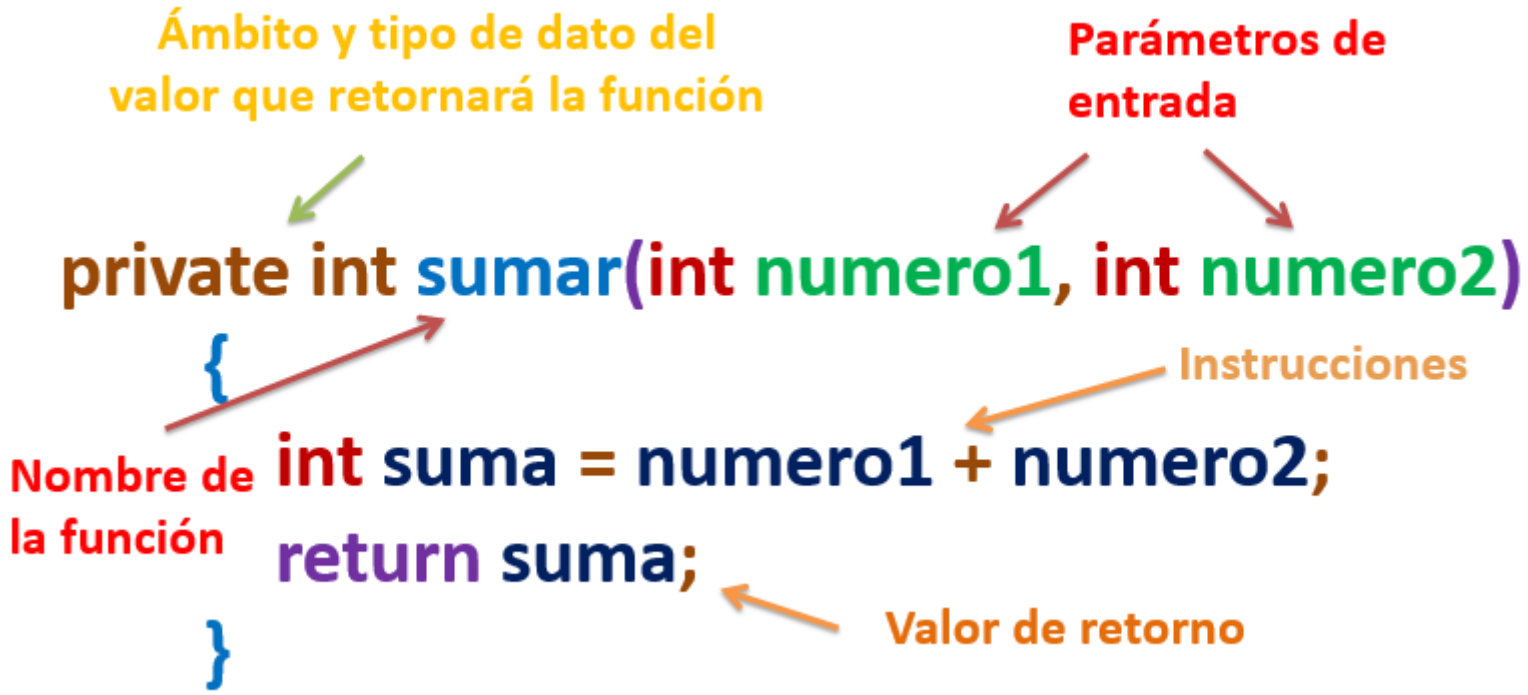
Funciones

Las funciones son utilizadas para **descomponer** grandes problemas en tareas simples y para implementar operaciones que son comúnmente utilizadas durante un programa y de esta manera reducir la cantidad de código. Cuando una función es invocada se le pasa el control a la misma, una vez que esta finalizó con su tarea el control es devuelto al punto desde el cual la función fue llamada.



INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones



Las funciones son la parte central de la programación. Algunos lenguajes, como Pascal, distinguen entre procedimientos y funciones.
Un procedimiento es una función que no devuelve resultado alguno.

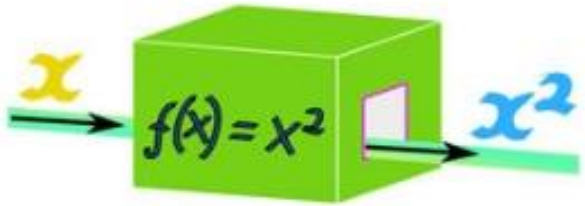
INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

Función

Definición

- Subalgoritmo, parte de un algoritmo principal que **permite resolver una tarea específica**
- Puede ser invocada desde otras partes del algoritmo cuantas veces se necesite
- Puede recibir valores (parámetros o argumentos)
- Puede devolver valores



INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

En Python, la definición de funciones se realiza mediante la instrucción **def** más un **nombre de función** descriptivo -para el cuál, aplican las mismas reglas que para el nombre de las variables- seguido de **paréntesis de apertura y cierre**. Como toda estructura de control en Python, la definición de la función finaliza con dos puntos (:) y el algoritmo que la compone, irá identado con 4 espacios:

```
def mi_funcion():
    print "Hola Mundo"
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

Es recomendable **añadir comentarios al definir una nueva función**. Esto ayuda a entender su propósito.

En el caso de las funciones , Python toma como documentación o comentario a la función una cadena de texto si esta aparece como primera sentencia del cuerpo de la función. Es habitual enmarcar esta descripción de la función entre tres dobles o simples comillas, y se pueden usar varias líneas.

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

```
def mayor(a,b) :
    """Esta función devuelve el mayor de dos valores"""
    if a>b :
        return a
    return b
```

Al documentar así la definición de una función, podemos acceder a su documentación mediante la propiedad **_doc_**

print(mayor._doc_) #Devuelve: Esta función devuelve el mayor de dos valores

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

Para poder utilizar una función en un programa se tiene que haber definido antes. Por ello, normalmente las definiciones de las funciones se suelen escribir al principio de los programas

Una función, no es ejecutada hasta tanto no sea invocada. Para invocar una función, simplemente se la llama por su nombre:

```
funcion()
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

Cuando una función, haga un retorno de datos, éstos, pueden ser asignados a una variable:

```
frase = funcion()
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

Un **parámetro** es un valor que la función espera recibir cuando sea llamada (invocada), a fin de ejecutar acciones en base al mismo. Una función puede esperar uno o más parámetros (que irán separados por una coma) o ninguno.

Los parámetros que una función espera, serán utilizados por ésta, dentro de su algoritmo, a modo de variables de ámbito local. Es decir, que los parámetros serán **variables locales**, a las cuáles **solo la función** podrá **acceder**:

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

```
def calcula_media(x, y):
    resultado = (x + y) / 2
    return resultado
```

Al llamar a una función, siempre se le deben pasar sus argumentos en el mismo orden en el que los espera.

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

En Python, también es posible, asignar **valores por defecto** a los **parámetros** de las funciones. Esto significa, que la función podrá ser llamada con menos argumentos de los que espera:

```
def pagar(importe, dto_aplicado = 5):
    ''' La función aplica descuentos '''
    return importe - (importe * dto_aplicado / 100)
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

Es posible que una función, espere recibir un número arbitrario -desconocido- de argumentos. Estos argumentos, llegarán a la función en forma de tupla.

Para definir argumentos arbitrarios en una función, se antecede al parámetro un asterisco (*):

```
def calcula_media(*args):
    total = 0
    for i in args:
        total += i
    resultado = total / len(args)
    return resultado
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

```
def calcula_media(*args):
    total = 0
    for i in args:
        total += i
    resultado = total / len(args)
    return resultado

a, b, c = 3, 5, 10
media = calcula_media(a, b, c)
print(f"La media de {a}, {b} y {c} es: {media}")
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

Puede ocurrir que la función espere una lista fija de parámetros, pero que éstos, en vez de estar disponibles de forma separada, se encuentren contenidos en una **lista** o **tupla**. En este caso, el signo asterisco (*) deberá preceder al nombre de la lista o tupla que es pasada como parámetro durante la llamada a la función.

Cuando los valores a ser pasados como parámetros a una función se encuentren disponibles en un **diccionario** deberán pasarse a la función precedidos de dos asteriscos (**).

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

```
def calcular(importe, descuento):
    return importe - (importe * descuento / 100)
```

Lista o Tupla

```
datos = [1500, 10]
print (calcular(*datos))
```

Diccionario

```
datos = {"descuento": 10, "importe": 1500}
print calcular(**datos)
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

Las funciones pueden devolver varios valores simultáneamente:

```
def desguazar(dividendo, divisor):
    cociente = dividendo//divisor
    resto = dividendo%divisor
    return cociente,resto
```

```
x,y = desguazar(14,4)    #Devuelve x=3, y=2
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

Ámbito de las variables.

La zona del programa en la que una variable puede ser utilizada es lo que se conoce como su ámbito.

El ámbito de una variable es el contexto en el que existe esa variable. Así, una variable existe en dicho ámbito a partir del momento en que se crea y deja de existir cuando desaparece su ámbito. Además, las variables son accesibles solo desde su propio ámbito.

Los principales tipos de ámbitos en Python son dos:

- ❑ Ámbito local.
- ❑ Ámbito global.

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

Ámbito de las variables.

Ámbito local: corresponde con el ámbito de una función, que existe desde que se invoca a una función hasta que termina su ejecución.
En un programa, el ámbito local corresponde con las líneas de código de una función. Dicho ámbito se crea cada vez que se invoca a la función. Cada función tiene su ámbito local. No se puede acceder a las variables de una función desde fuera de esa función o desde otra función.

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

Ámbito de las variables.

Ámbito global: corresponde con el ámbito que existe desde el comienzo de la ejecución de un programa.

Todas las variables definidas fuera de cualquier función corresponden al ámbito global, que es accesible desde cualquier punto del programa, incluidas las funciones. Si desde un módulo A importamos un módulo B mediante un **import**, desde A podremos acceder a las variables globales de B.

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

Ámbito de las variables.

En Python las **variables locales** son aquellas definidas dentro de una función. Solamente son accesibles desde la propia función y dejan de existir cuando esta termina su ejecución. Los parámetros de una función también son considerados como variables locales. La diferencia entre los parámetros de una función y las variables locales definidas dentro de una función radica en que los parámetros nos permiten comunicarnos con la función para introducir valores de entrada a través de ellos.

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

Ámbito de las variables.

Cualquier **variable local**, incluidos los parámetros, deja de existir en cuanto termina de ejecutarse la función en la que está definida.

Desde el cuerpo principal del programa no se puede acceder a las variables locales de ninguna función. Tampoco es posible acceder a las variables locales de una función desde otra.

Si intentamos acceder al valor de una variable local desde el cuerpo principal del programa o, en general, a una variable que no ha sido definida obtendremos un error típico de Python: NameError.

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

Ámbito de las variables.

```
def subrutina():
    a = 2
    print(a)
    return

a = 5
subrutina() # 2
print(a) # 5
```

```
def subrutina():
    a = 2
    print(a)
    return

subrutina() #2
print(a) # NameError: name 'a' is not defined
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

Ámbito de las variables.

Las **variables globales** en Python son aquellas definidas en el cuerpo principal del programa fuera de cualquier función. Son accesibles desde cualquier punto del programa, incluso desde dentro de funciones. También se puede acceder a las variables globales de otros programas o módulos importados. Si en un módulo o programa determinado hay definida alguna variable global podremos acceder a ella importando dicho módulo.

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

Ámbito de las variables.

Las **variables globales** se pueden leer desde cualquier línea del programa simplemente haciendo referencia a ellas a través de su nombre. Pero **no** podemos modificar una variable global desde dentro de una función. Esto sucede así no por capricho, sino porque es un mecanismo de protección para evitar modificar sin querer una variable global que posiblemente terminaría alterando el funcionamiento normal de un programa.

Para modificarla es necesario utilizar el modificador **global**. Con esta declaración le estamos diciendo a Python que sabemos que vamos a utilizar una variable global y que queremos modificarla.

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

Ámbito de las variables.

```
contador = 10

def reiniciar_contador():
        global contador
        contador = 0



print(f'Contador antes es {contador}’)  # Contador antes es 10
reiniciar_contador()
print(f'Contador después es {contador}’) # Contador después es 0
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

Ámbito de las variables.

A veces puede suceder que dentro de una función existe una variable local con el mismo nombre que una variable global, es decir, sus nombres entran en conflicto.

Cuando hacemos referencia a una variable Python busca primero en el ámbito local actual para ver si encuentra dicha variable y si no, la busca en el ámbito global. Es decir, en cierta forma, el ámbito local tiene preferencia sobre el global. Se puede decir que la variable local le hace sombra a la variable global.

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

Ámbito de las variables.

El **ámbito de una variable** es la parte del programa donde esta variable puede ser utilizado, porque se conoce su existencia.

Variables globales: se definen en un lugar concreto del programa que no pertenezca a ninguna de las funciones.

Variables locales: deben estar definidas dentro de una función del programa y será en esta función donde tendrán su ámbito de validez.

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

La finalidad de una función, debe ser realizar una única acción, reutilizable y por lo tanto, tan genérica como sea posible.

INTRODUCCIÓN A LA PROGRAMACIÓN

Funciones

“Fin del tema”