Python
para no programadores

Módulo 3

Argumentos

Argumentos

Cambiemos el ejemplo y consideremos ahora el siguiente código:

```
alumnos = ["Pablo", "Juan", "Matias", "Martin"]
notas = [5.5, 9, 6.25, 8]

print("Alumnos:")
for alumno in alumnos:
    print(alumno)

print("Notas:")
for nota in notas:
    print(nota)
```

En este caso también tenemos cierta repitencia: los bucles “for” son muy similares, aunque cambia el nombre de la lista sobre la que se aplican. No obstante, el método es siempre el mismo: recorrer cada uno de los elementos de la lista e imprimirlos en pantalla vía print() (a partir de ahora, cuando indicamos paréntesis luego del nombre de una instrucción en este material, querremos decir que se trata de una función).

Argumentos

Si tuviésemos que hacer una versión genérica de ese bucle “for”, podríamos expresarla así:

```
for elemento in lista:
    print(elemento)
```

Donde lista puede ser alumnos o notas, o bien cualquier otra lista de la cual queramos imprimir sus elementos. Esto bien podemos hacerlo en una función:

```
def imprimir_elementos(lista):
    for elemento in lista:
        print(elemento)
```

Lo que hacemos aquí es indicar entre paréntesis aquellas variables (que llevan el nombre de argumentos de una función) que serán “llenadas” por el usuario al momento de llamar a la función. Como en nuestro caso aún no sabemos cuál es la lista que se querrá imprimir, simplemente ponemos un nombre genérico lista.

Argumentos

Ahora nuestro código se vería así:

```
def imprimir_elementos(lista):
    for elemento in lista:
        print(elemento)

alumnos = ["Pablo", "Juan", "Matias", "Martin"]
notas = [5.5, 9, 6.25, 8]

print("Alumnos:")
imprimir_elementos(alumnos)

print("Notas:")
imprimir_elementos(notas)
```

Nótese que en el primer caso llamamos `imprimir_elementos(alumnos)`, de modo que el código que se ejecutará será el siguiente:

```
for elemento in alumnos:
    print(elemento)
```

En el código de la función se “reemplaza” lista por lo que indicamos entre paréntesis al llamarla; en este caso, la lista alumnos. Del mismo modo ocurre en el segundo caso para `imprimir_elementos(notas)`.

Argumentos

Incluso podemos usar dicha función para imprimir cualquier lista, aunque no la hayamos guardado en una variable:

```
imprimir_elementos([1, 2, 3, 4, 5])
```

Ahora bien, de nuestra función `imprimir_elementos()` decimos que requiere un argumento, que es el nombre de la lista.

Una función puede tener múltiples argumentos, la cantidad que nosotros precisemos. Para indicar más de uno, separamos sus nombres por comas.

Por ejemplo:

```
def imprimir_suma(a, b):
    print("Resultado:")
    print(a + b)

imprimir_suma(7, 5)
imprimir_suma(-5, 3.5)
```

La función `imprimir_suma()` requiere dos argumentos, al primero de ellos le dimos el nombre `a` y, al segundo, `b`. Al llamarla, indicamos en orden los valores que queremos otorgarle, separados por comas. (Aunque en este caso es indistinto, por cuanto en una suma el orden de los sumandos no altera el resultado).

¡Muchas gracias!

¡Sigamos trabajando!