ThePower business school

Funciones

En Python, una función se define con la palabra clave `def`, seguida por el nombre de la función, paréntesis que pueden incluir parámetros y dos puntos. El cuerpo de la función comienza en la siguiente línea, y debe estar indentado.

```
def saludar():
    print("¡Hola Mundo!")
```

Para llamar a la función, simplemente se utiliza su nombre seguido de paréntesis:

```
saludar()  # Llama a la función y muestra: ¡Hola Mundo!
```

Funciones con parámetros

Las funciones pueden tomar parámetros, que son valores que se pasan a la función cuando se llama. Dentro de la función, estos valores pueden ser utilizados como variables.

```
def saludar(nombre):
    print(f"¡Hola {nombre}!")

saludar("María")  # Llama a la función y muestra: ¡Hola María!
```

Funciones que devuelven valores

Las funciones también pueden devolver valores. Para esto, se usa la palabra clave `return`. Cuando Python encuentra una declaración `return`, sale de la función inmediatamente y pasa el valor a la derecha de la declaración de retorno al contexto que llamó a la función.

```
def suma(a, b):
    return a + b

resultado = suma(5, 3)  # Llama a la función y almacena el resultado en la variable re
sultado
print(resultado)  # Muestra: 8
```

Ejemplos de funciones

Función que suma todos los números en una lista:

```
def suma_lista(numeros):
    suma = 0
    for numero in numeros:
        suma += numero
    return suma

mi_lista = [1, 2, 3, 4, 5]
print(suma_lista(mi_lista))  # Muestra: 15
```

Función que encuentra el número más grande en una lista:

```
def maximo_lista(numeros):
    maximo = numeros[0]
    for numero in numeros:
        if numero > maximo:
            maximo = numero
    return maximo

mi_lista = [1, 2, 3, 4, 5]
print(maximo_lista(mi_lista))  # Muestra: 5
```

Función que cuenta cuántas veces aparece un carácter específico en una cadena:

```
def cuenta_caracteres(cadena, caracter):
    conteo = 0
    for c in cadena:
        if c == caracter:
```

```
            conteo += 1
    return conteo

mi_cadena = "hola mundo"
print(cuenta_caracteres(mi_cadena, 'o'))  # Muestra: 2
```

Para entender el siguiente ejemplos vamos a tener que aprender un par de conceptos, `[::-1]` es un tipo de operación de corte (slicing) que se aplica a secuencias, como listas o cadenas de texto. Esta operación invierte el orden de los elementos de la secuencia.

- El primer `:` indica que la operación de corte debe comenzar desde el principio de la secuencia.
- El segundo `:` indica que la operación de corte debe continuar hasta el final de la secuencia.
- El `1` es el paso, que indica que la operación de corte debe avanzar en dirección opuesta, es decir, de derecha a izquierda, en lugar de la dirección predeterminada de izquierda a derecha.

Por lo tanto, `[::-1]` toma todos los elementos de la secuencia en orden inverso.

Aquí tienes un ejemplo con una cadena de texto y una lista:

```
cadena = "hola"
cadena_invertida = cadena[::-1]
print(cadena_invertida)  # Salida: aloh

lista = [1, 2, 3, 4, 5]
lista_invertida = lista[::-1]
print(lista_invertida)  # Salida: [5, 4, 3, 2, 1]
```

Como puedes ver, `cadena[::-1]` devuelve una nueva cadena que contiene los caracteres de la cadena original en orden inverso, y `lista[::-1]` devuelve una nueva lista con los elementos de la lista original en orden inverso.

Función que verifica si una palabra es un palíndromo (una palabra que se lee igual hacia adelante y hacia atrás):

```
def es_palindromo(palabra):
    palabra = palabra.lower()
```

```
    return palabra == palabra[::-1]

print(es_palindromo("radar"))  # Muestra: True
print(es_palindromo("python"))  # Muestra: False
```

Función que devuelve una lista con todos los números pares desde 0 hasta un número dado:

```
def lista_pares(n):
    return [i for i in range(n+1) if i % 2 == 0]

print(lista_pares(10))  # Muestra: [0, 2, 4, 6, 8, 10]
```