05_funciones_python

February 3, 2025

1 Lenguanjes de Programación

1.0.1 Universidad Politecnica Salesiana

1.0.2 Ingeniería en Ciencias de la Computación

1.0.3 Programación

1.1 Autores del Material

- **Instructor**: Ing. Pablo Torres
- **Contacto**: ptorresp@ups.edu.ec

1.2 Practica Autonma

Funciones en Python

1.2.1 Descripción General

Requisitos del Sistema

El sistema simula una aplicación de perfil ambiental en la que los usuarios ingresan datos sobre sus hábitos para obtener un "perfil ecológico." Según la frecuencia de ciertas actividades y prácticas, los usuarios recibirán recomendaciones personalizadas para mejorar su impacto ambiental. Los usuarios ingresarán su nombre, edad, ciudad, y luego se les pedirá información adicional basada en su categoría de hábitos ecológicos. Esta categoría se define en función de tres datos de entrada clave:

Frecuencia de reciclaje (reciclaje_frecuencia): un número entero que representa la frecuencia en días en la que reciclan. Tipo de transporte habitual (tipo_transporte): una cadena de texto que especifica su medio de transporte principal, como "publico", "auto", o "bici". Tipo de dieta (tipo_dieta): una cadena de texto que describe su dieta (por ejemplo, "carnivora", "vegetariana", o "vegana").

1.2.2 Objetivos de Aprendizaje

- Crear funciones en Python que permitan categorizar a una persona en función de sus respuestas sobre prácticas ecológicas y generar un perfil con recomendaciones personalizadas.

El perfil se muestra al final de la interacción.

1.3 1. Introducción a las Funciones

Las funciones son bloques de código que realizan tareas específicas y pueden ser reutilizados en diferentes partes de un programa. Ayudan a organizar el código, mejorar la legibilidad y facilitar el mantenimiento.

1.3.1 1.1. Definición y Propósito

- **Definición**: Una función es un bloque de código que se ejecuta solo cuando es llamada. Puede recibir entradas, conocidas como parámetros, y puede devolver una salida.
- **Propósito**: Facilitar la reutilización del código, mejorar la organización y reducir la redundancia.

1.3.2 1.2. Sintaxis de una Función

La sintaxis básica para definir una función en Python utiliza la palabra clave `def`, seguida del nombre de la función y paréntesis que pueden contener parámetros.

```
def nombre_de_la_funcion(parametros):
    """
    Docstring opcional que describe la función.
    """
    # Bloque de código
    return valor_de_retorno
```

1.3.3 1.3. Parámetros y Argumentos

- **Parámetros**: Variables que se definen dentro de los paréntesis en la definición de la función.
- **Argumentos**: Valores que se pasan a la función cuando se llama.

Ejemplo:

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

saludar("Ana")  # Llamada a la función con el argumento "Ana"
```

[1]:
```
def saludar(nombre):
    print(f"Hola {nombre}")


saludar("Pablo")
```

```
Hola Pablo
```

[28]:
```
def saludar(nombre):
    print(f"Hola {nombre}")
```

```
    print(f"Hola {nombre}")

for i in range(3):
    if i %2 ==0:
        nombre_tres = "Pablo"
        saludar(nombre_tres)
    else:
        nombre_tres = "Pablo" + str(i)
        saludar(nombre_tres)
```

```
Hola True
Hola True
```

[ ]:
```
saludar("Deste otro bloque")
```

```
Hola Deste otro bloque
Hola Deste otro bloque
```

[ ]:
```
def saludar():
    print(f"Hola ")

saludar()
```

1.3.4 1.4. Valor de Retorno

Las funciones pueden devolver valores utilizando la palabra clave `return`. Si una función no tiene una declaración `return`, devuelve `None` por defecto.

Ejemplo:

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

resultado = sumar(5, 3)
print(resultado)  # Output: 8
```

[36]:
```
a = 5
b =3
def sumar(a, b):
    res=a + b
    return res

resultado = sumar(a,a-1)
print(a+b)
print(resultado)
```

```
8
9
```

1.3.5 1.5. Ejemplos Básicos de Funciones

Función sin parámetros ni retorno:

```
def imprimir_mensaje():
    print("Esta es una función sin parámetros ni retorno.")

imprimir_mensaje()
```

Función con múltiples parámetros:

```
def multiplicar(a, b):
    return a * b

producto = multiplicar(4, 5)
print(producto)  # Output: 20
```

1.4 B. Ejercicios Prácticos

1.4.1 Ejercicio 1: Función para Calcular el Área de un Círculo

Descripción: Crea una función que calcule el área de un círculo dado su radio.

Instrucciones: 1. Define una función llamada `area_circulo` que reciba el radio como parámetro. 2. Calcula el área utilizando la fórmula: Área = * radio². 3. Devuelve el valor del área. 4. Llama a la función con un radio de tu elección y muestra el resultado.

```
import math

def area_circulo(radio):
    """
    Calcula el área de un círculo dado su radio.
    """
    area = math.pi * radio ** 2
    return area

# Ejemplo de uso
radio = 5
resultado = area_circulo(radio)
print(f"El área del círculo con radio {radio} es: {resultado:.2f}")
```

[40]:
```
def area_circulo(radio):
    return radio**2 * 3.14
```

[41]:
```
radio = int(input("Ingrese el radio"))
resultado = area_circulo(radio)
print(f"el resultado es = {resultado}")
```

```
el resultado es = 78.5
```

1.5 C. Resumen de Conceptos Clave

1. **Funciones**: Bloques de código reutilizables que realizan tareas específicas.
2. **Definición de Funciones**: Se utilizan la palabra clave `def` seguida del nombre de la función y paréntesis que pueden contener parámetros.
3. **Parámetros y Argumentos**: Parámetros son variables definidas en la función; argumentos son los valores pasados a la función al llamarla.
4. **Valor de Retorno**: Las funciones pueden devolver valores utilizando la palabra clave `return`.
5. **Funciones con Múltiples Parámetros**: Las funciones pueden aceptar múltiples parámetros para realizar operaciones más complejas.
6. **Funciones con Parámetros por Defecto**: Permiten asignar valores por defecto a los parámetros, haciendo opcional su especificación al llamar a la función.
7. **Funciones Lambda**: Funciones anónimas de una sola línea que se utilizan para operaciones simples.
8. **Documentación de Funciones**: Las docstrings (`""" """`) se utilizan para documentar la funcionalidad de las funciones.
9. **Alcance de las Variables (Scope)**: Las variables definidas dentro de una función son locales y no afectan al resto del programa.
10. **Modularización y Reutilización de Código**: Las funciones facilitan la división del código en módulos más manejables y reutilizables.
11. **Importación de Módulos y Funciones**: Permite utilizar funciones definidas en otros archivos o módulos, promoviendo la reutilización del código.

1.6 D. Recursos Adicionales

- Documentación Oficial de Python - Estructuras de Decisión
- Tutorial de Estructuras de Decisión en Real Python
- Ejemplos de Estructuras de Decisión en W3Schools

1.7 E. Referencias

1. Van Rossum, G. (1991). *Python Tutorial*. Python Software Foundation.
2. Lutz, M. (2013). *Learning Python*. O'Reilly Media.
3. Sweigart, A. (2015). *Automate the Boring Stuff with Python*. No Starch Press.
