Introducción a Python – Sesión 3

Cesar Garcia

Introducción a Python – Sesión 3

Objetivos de la sesión

- Aprender a definir y usar funciones.
- Comprender parámetros y valores de retorno.
- Entender el ámbito (scope) de las variables.
- Crear y manipular diccionarios.
- Construir una agenda de contactos como mini-proyecto.

Repaso rápido de la sesión anterior

- Condicionales (`if`, `elif`, `else`)
- Listas (indexación, slicing, métodos)
- Bucles (`for`, `while`)
- Mini-proyecto: Verificador de contraseñas

¿Qué es una función?

- Un bloque de código reutilizable.
- Evita duplicación.
- Mejora la organización del programa.

Ejemplo simple:

```
def saludar():
    print("Hola")
```

Definir y llamar funciones

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

saludar("Cesar")
saludar("Ana")
```

Parámetros y argumentos

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

multiplicar(3, 4)  # 12
```

Valores de retorno

```
def cuadrado(x):
    return x * x

resultado = cuadrado(5)
print(resultado)  # 25
```

Funciones con varios parámetros

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

print(sumar(10, 20))   # 30
```

Parámetros con valores por defecto

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

saludar()
saludar("Cesar")
```

Ámbito de variables (scope)

```
x = 10  # global

def ejemplo():
    y = 5  # local
    print(x, y)

ejemplo()
print(x)
# print(y)  # Error: y no existe fuera de la función
```

Ejercicio: Función para calcular el BMI

```
def bmi(peso, altura):
    return peso / (altura ** 2)

p = float(input("Peso (kg): "))
h = float(input("Altura (m): "))
print(f"Tu BMI es {bmi(p, h):.2f}")
```

Diccionarios en Python

```
persona = {
    "nombre": "Cesar",
    "edad": 59,
    "pais": "México/USA"
}

print(persona["nombre"])
print(persona["edad"])
```

Operaciones básicas con diccionarios

```python
persona = {"nombre": "Cesar", "edad": 59}

persona["edad"] = 60                    # modificar
persona["profesion"] = "Desarrollador"   # agregar

print(persona.keys())     # claves
print(persona.values())   # valores
print(persona.items())    # pares clave-valor
```

Recorrer diccionarios con for

```
for clave, valor in persona.items():
    print(clave, "→", valor)
```

Diccionarios dentro de listas

```
estudiantes = [
    {"nombre": "Ana", "nota": 89},
    {"nombre": "Luis", "nota": 92}
]

for est in estudiantes:
    print(est["nombre"], "tiene", est["nota"])
```

Comprehension de listas y Diccionarios

Si quieres crear una lista de cuadrados

```
nums = [1, 2, 3, 4, 5]
cuadrados = []
for x in nums:
    cuadrados.append(x*x)
```

Usando list comprehension

```
cuadrados = [x*x for x in nums]
```

Mini-proyecto: Agenda de contactos

Objetivo: - Registrar contactos con nombre, teléfono y email. - Almacenar en una lista de diccionarios. - Listar todos los contactos ingresados.

Estructura sugerida de la agenda

```
contactos = []

# Cada contacto es:
# {
#   "nombre": "Cesar",
#   "telefono": "555-1234",
#   "email": "cesar@example.com"
# }
```

Función para agregar un contacto

```
def agregar_contacto(contactos):
    nombre = input("Nombre: ")
    telefono = input("Teléfono: ")
    email = input("Email (opcional): ")

    contacto = {
        "nombre": nombre,
        "telefono": telefono,
        "email": email
    }

    contactos.append(contacto)
    print("Contacto agregado.")
```

Función para mostrar contactos

```
def mostrar_contactos(contactos):
    if not contactos:
        print("No hay contactos.")
        return

    for i, c in enumerate(contactos, start=1):
        print(f"{i}. {c['nombre']} - {c['telefono']} - {c['ema
```

Programa principal de la agenda

```
def main():
    contactos = []

    while True:
        print("\n1. Agregar contacto")
        print("2. Ver contactos")
        print("3. Salir")

        opcion = input("Opción: ")

        if opcion == "1":
            agregar_contacto(contactos)
        elif opcion == "2":
            mostrar_contactos(contactos)
        elif opcion == "3":
            print("Adiós")
            break
```

Errores comunes en funciones y diccionarios

- Olvidar `return` en funciones.
- Intentar acceder a claves que no existen.
- Confundir listas y diccionarios en la sintaxis.
- Modificar estructuras mientras se recorre sin cuidado.

Tarea

1. Escribir una función `es_primo(n)` que devuelva True o False.
2. Crear un diccionario con:
   - nombre
   - edad
   - país
   - profesión
   - hobbies (lista)

   Luego imprimir una biografía formateada.

Cierre de la sesión

- Funciones
- Diccionarios
- Agenda de contactos
- Ejercicios prácticos