Estructuras de Datos y Modularidad

Organizando información y reutilizando código

Probabilidad y Estadística

Aprendiendo con diversión

25 de noviembre de 2025

¿Qué vamos a aprender?

Nota importante

¡Hoy organizamos datos y creamos funciones reutilizables! 🎯

Objetivo 1: Almacenar múltiples valores en listas y arrays

Objetivo 2: Usar diccionarios para datos estructurados

Objetivo 3: Intercambiar datos con formato JSON

Objetivo 4: Crear funciones para no repetir código

Ejemplo

Como organizar un archivador: cada cajón tiene carpetas con documentos específicos

Listas y Arrays: Múltiples Valores

Nota importante

¡Una variable que guarda muchos valores! 

En vez de crear 10 variables separadas, usamos 1 lista

Ejemplo

```
# Forma incorrecta
transaccion1 = 15000
transaccion2 = 23000
transaccion3 = 8500

# Forma correcta
transacciones = [15000, 23000, 8500, 12000, 31000]
```

Las listas se indexan desde 0

Acceso a Elementos de una Lista

Operación	Código	Resultado
Primer elemento	transacciones[0]	15000
Segundo elemento	transacciones[1]	23000
Último elemento	transacciones[-1]	31000
Cantidad de elementos	len(transacciones)	5

Ejemplo

$primera_venta = transacciones[0]$

Nota importante

¡El índice 0 es el primero, no el 1!

Operaciones con Listas

Agregar elementos:

$$transacciones.append(18000) Agregaalfinal$$

Eliminar elementos:

$$transacciones.remove(8500) Eliminaesevalor$$

Recorrer la lista:

```
for transaccion in transacciones:
    print("Venta:", transaccion)
```

Ejemplo: Gestión de Empleados

Problema

Almacenar y procesar lista de empleados

Ejemplo

```
empleados = ["Ana", "Juan", "Maria", "Pedro"]

# Agregar nuevo empleado
empleados.append("Carlos")

# Pagar a todos
for empleado in empleados:
    print("Procesando pago de", empleado)

# Cantidad total
print("Total empleados:", len(empleados))
```

Diccionarios: Datos Estructurados

Nota importante

¡Como una ficha con múltiples campos!

Cada dato tiene una clave (nombre del campo) y un valor

Ejemplo

```
empresa = {
    "nombre": "Tech Solutions S.A.",
    "rut": "76.123.456-7",
    "saldo": 1500000,
    "activo": True
}
```

Se accede por clave, no por índice numérico

Acceso a Datos en Diccionarios

Operación	Código	Resultado
Obtener valor	empresa['nombre']	Tech Solutions S.A.
Modificar valor	empresa['saldo'] = 2000000	2000000
Agregar campo	empresa['ciudad'] = 'Santiago'	Santiago
Verificar clave	'rut' in empresa	True

Ejemplo

$nombre_empresa = empresa['nombre']$

Ejemplo: Registro de Cliente

Problema

Crear y gestionar información de un cliente

Ejemplo

```
cliente = {
    "id": 12345,
    "nombre": "Juan Prez",
    "email": "juan@email.com",
    "compras_totales": 0,
    "vip": False
}

# Registrar una compra
cliente["compras_totales"] += 50000

# Promover a VIP si compra ms de 100000
if cliente["compras_totales"] > 100000:
```

Listas de Diccionarios: Base de Datos Simple

Nota importante

¡Combinar listas y diccionarios para datos complejos!

Ejemplo

```
facturas = [
    {"numero": 1001, "cliente": "Empresa A", "monto": 150000},
    {"numero": 1002, "cliente": "Empresa B", "monto": 230000},
    {"numero": 1003, "cliente": "Empresa C", "monto": 180000}
]

# Calcular total facturado
total = 0
for factura in facturas:
    total += factura["monto"]
```

JSON: El Formato Universal

Nota importante

JavaScript Object Notation - Estándar para intercambiar datos

Es como un diccionario pero en formato de texto

Ejemplo

```
{
  "numero_factura": 1001,
  "cliente": "Empresa A",
  "items": [
    {"producto": "Laptop", "precio": 500000},
    {"producto": "Mouse", "precio": 15000}
  ],
  "total": 515000
}
```

Trabajar con JSON en Código

Convertir diccionario a JSON (texto):

```
import json

factura_dict = {"numero": 1001, "monto": 150000}
factura_json = json.dumps(factura_dict)
# Resultado: '{"numero": 1001, "monto": 150000}'
```

Convertir JSON (texto) a diccionario:

Funciones: No Repetir Código

Nota importante

Principio DRY: Don't Repeat Yourself

Si escribes el mismo código 2 veces, crea una función

Ejemplo

En vez de calcular IVA manualmente cada vez:

```
#     Repetitivo
total1 = precio1 * 1.19
total2 = precio2 * 1.19
total3 = precio3 * 1.19

#     Con funcin
def calcular_con_iva(precio):
    return precio * 1.19

total1 = calcular_con_iva(precio1)
```

Estructura de una Función

Parte	Descripción	Ejemplo
Nombre	Identifica la función	calcular_iva
Parámetros	Datos de entrada	(precio)
Cuerpo	Código a ejecutar	total = precio * 0.19
Retorno	Dato de salida	return total

Ejemplo

```
def calcular_iva(precio):
    iva = precio * 0.19
    return iva
```

Ejemplo: Función Validar RUT

Problema

Crear función que valide formato de RUT chileno

Ejemplo

```
def validar_rut(rut):
    # Eliminar puntos y guion
    rut_limpio = rut.replace(".", "").replace("-", "")

    # Verificar longitud
    if len(rut_limpio) < 8 or len(rut_limpio) > 9:
        return False

    # Validar que sean numeros (excepto ultimo digito)
    if not rut_limpio[:-1].isdigit():
        return False

    return True
```

Parámetros y Valores de Retorno

Nota importante

Input → Proceso → Output

Ejemplo

Función con múltiples parámetros:

```
def calcular_descuento(precio, porcentaje):
    descuento = precio * (porcentaje / 100)
    precio_final = precio - descuento
    return precio_final

# Uso
total = calcular_descuento(100000, 10)
print(total)  # 90000
```

Los parámetros son variables locales dentro de la función

Alcance de Variables (Scope)

Nota importante

¿Dónde viven las variables?

Tipo	Alcance	Uso
Variable Local	Dentro de la función	Solo existe en la función
Variable Global	Fuera de funciones	Accesible en todo el código

Ejemplo

```
saldo_global = 100000  # Global

def retirar(monto):
    saldo_local = saldo_global - monto  # Local
    return saldo_local

# saldo_local NO existe aqu (fuera de la funcin)
```

Ejemplo Completo: Sistema de Descuentos

Problema

Sistema que aplica descuentos según tipo de cliente

Ejemplo

```
def calcular_precio_final(precio, es_vip):
    if es_vip:
        descuento = precio * 0.15
    else:
        descuento = precio * 0.10

    precio_final = precio - descuento
    return precio_final

# Uso
cliente_vip = calcular_precio_final(100000, True)
cliente_normal = calcular_precio_final(100000, False)
```

¡Tu Turno de Practicar!

Problema

Crear función que procese lista de ventas

La función debe:

1. Recibir una lista de montos de ventas
2. Calcular el total
3. Calcular el promedio
4. Retornar ambos valores

Nota importante

Pista: Usa len() para contar elementos y sum() para sumar

¡Resumen de lo Aprendido! 🎓

- Listas almacenan múltiples valores en una sola variable
- Diccionarios organizan datos con claves y valores
- JSON es el formato estándar para intercambiar datos
- Combinar listas y diccionarios crea estructuras complejas
- Funciones evitan repetir código (principio DRY)
- Funciones tienen parámetros (input) y retorno (output)
- Variables locales existen solo dentro de la función

Nota importante

¡Ya puedes organizar datos y crear código modular! 🚀

¡Sigue aprendiendo!