



Gobierno del
CHACO

Ministerio
de la Producción y el Desarrollo
Económico Sostenible



INFORMATARIO

Informatorio Chaco | 2024 | Data Analytics

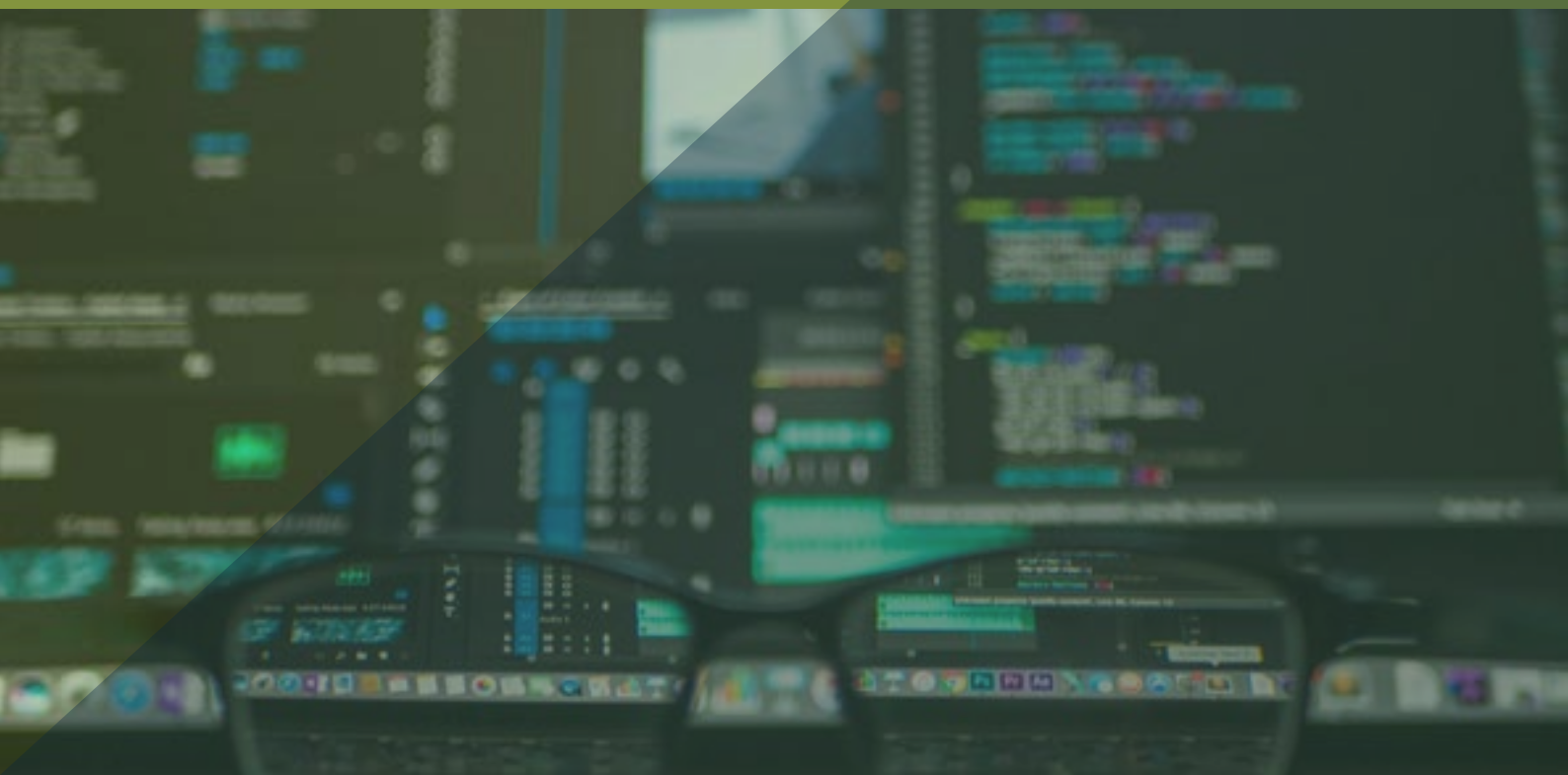
Clase 3

PROGRAMACIÓN ORIENTADA A OBJE- TOS, LABORATORIO PYTHON

*Programación Orientada a Objetos | Clases y
Objetos | Atributos y Métodos | Constructores
| Herencia y Polimorfismo | Encapsulamiento
| Laboratorio Python | Ejercicio CRUD.*

módulo

2



Programación Orientada a Objetos

Clases y Objetos

La programación orientada a objetos (POO) es un paradigma de programación basado en la creación de "objetos", que pueden contener datos y métodos.

Definición de una clase:

```
python Copiar código  
  
class Persona:  
    def __init__(self, nombre, edad):  
        self.nombre = nombre  
        self.edad = edad  
  
    def saludar(self):  
        print(f"Hola, mi nombre es {self.nombre} y tengo {self.edad} años")
```

Crear un objeto:

```
python Copiar código  
  
p = Persona('Ana', 25)  
p.saludar() # Salida: Hola, mi nombre es Ana y tengo 25 años
```

Atributos y Métodos

- **Atributos:** Variables que pertenecen a una clase.
- **Métodos:** Funciones que pertenecen a una clase.

Ejemplo:

```
python Copiar código

class Coche:
    def __init__(self, marca, modelo):
        self.marca = marca
        self.modelo = modelo

    def arrancar(self):
        print(f"El coche {self.marca} {self.modelo} está arrancado")
```

Constructores (`__init__`)

El método `__init__` es un constructor especial que se llama cuando se crea una instancia de la clase.

Ejemplo:

```
python Copiar código

class Animal:
    def __init__(self, nombre):
        self.nombre = nombre

    def hacer_sonido(self):
        print("El animal hace un sonido")
```

Herencia y Polimorfismo

- **Herencia:** Permite crear una nueva clase que es una modificación de una clase existente.
- **Polimorfismo:** Permite que diferentes clases implementen el mismo método de diferentes maneras.

Ejemplo de herencia:

python

 Copiar código


```
class Perro(Animal):  
    def hacer_sonido(self):  
        print("Guau")
```

Encapsulamiento

El encapsulamiento es la ocultación de los detalles de implementación de una clase, exponiendo solo lo necesario a través de una interfaz pública.

Ejemplo:

python

 Copiar código

```
class CuentaBancaria:  
    def __init__(self, saldo):  
        self.__saldo = saldo # Atributo privado  
  
    def depositar(self, monto):  
        self.__saldo += monto  
  
    def obtener_saldo(self):  
        return self.__saldo
```

Laboratorio Python

Ejercicio CRUD

Crear un script que permita Crear, Leer, Actualizar y Eliminar (CRUD) registros en un archivo de texto.

1. Definir la clase principal:

```
python Copiar código  
  
class Registro:  
    def __init__(self, nombre, edad):  
        self.nombre = nombre  
        self.edad = edad
```

2. Crear funciones para cada operación CRUD:

a). Crear un Registro:


```
python Copiar código  
  
def crear_registro(nombre, edad):  
    with open('registros.txt', 'a') as f:  
        f.write(f"{nombre},{edad}\n")
```

b). Leer Registros:

```
python Copiar código  
  
def leer_registros():  
    with open('registros.txt', 'r') as f:  
        registros = f.readlines()  
        for registro in registros:  
            print(registro.strip())
```

c). Actualizar un Registro:


python

 Copiar código

```
def actualizar_registro(nombre, nueva_edad):
    registros_actualizados = []
    with open('registros.txt', 'r') as f:
        registros = f.readlines()
        for registro in registros:
            nombre_actual, edad_actual = registro.strip().split(',')
            if nombre_actual == nombre:
                registros_actualizados.append(f"{nombre},{nueva_edad}\n")
            else:
                registros_actualizados.append(registro)
    with open('registros.txt', 'w') as f:
        f.writelines(registros_actualizados)
```

d). Eliminar un Registro:

python

 Copiar código

```
def eliminar_registro(nombre):
    registros_actualizados = []
    with open('registros.txt', 'r') as f:
        registros = f.readlines()
        for registro in registros:
            nombre_actual, edad_actual = registro.strip().split(',')
            if nombre_actual != nombre:
                registros_actualizados.append(registro)
    with open('registros.txt', 'w') as f:
        f.writelines(registros_actualizados)
```

Ejemplo de uso del sistema CRUD:

python

 Copiar código

```
crear_registro('Ana', 25)
leer_registros()
actualizar_registro('Ana', 26)
eliminar_registro('Ana')
```

Al finalizar este módulo, habrás revisado y consolidado los conceptos fundamentales de programación en Python, incluyendo el manejo de variables, operadores, estructuras de datos, control de flujo, funciones, manejo de archivos y programación orientada a objetos. Estos conocimientos son esenciales para avanzar en el análisis de datos y aplicarlos en situaciones prácticas.