Estructuras de datos , argoritmos y POO

Contenido

Algoritmos	Algoritmos de datos y estructura de datos secuencial
Estruturas de de datos	Lista, Tuplas, Diccionarios.
Funciones	Tipos de funciones y sus usos.
Bibliotecas	Estandares de Bibliotecas y sus usos.
Programacion POO	Metodos ,Clases,Herencias, Interfaces graficas.
Bases de datos.	MySQL,SQLite y PostgrestSQL.

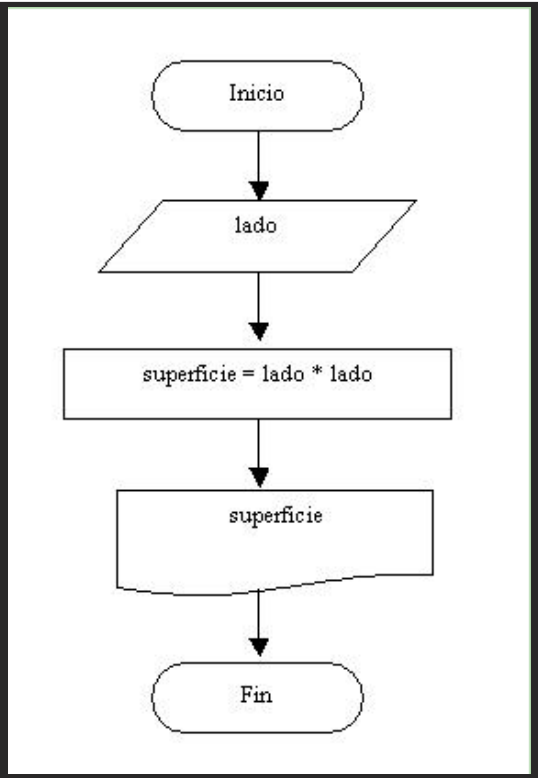
Infotep Ciberseguridad 2025

Instrutor Carlos Garcia

Introducción

Repaso de Programacion Secuencial.

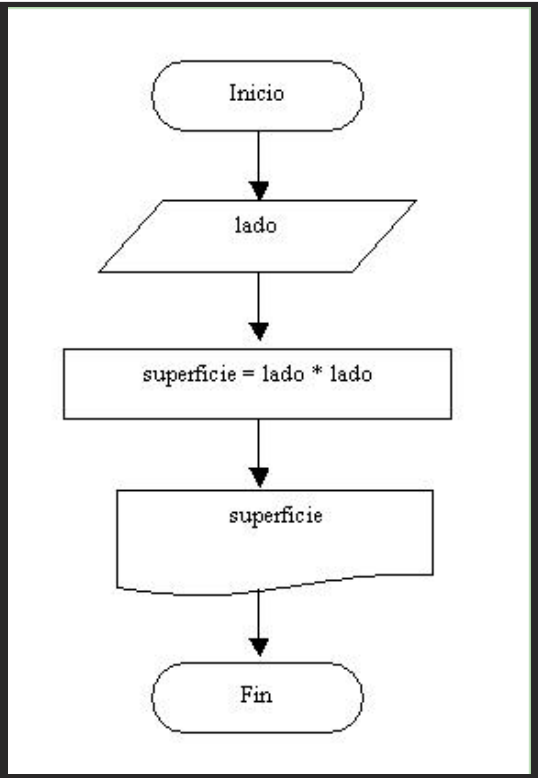
- **Como usar Vscode con Python.**
- **Instalacion del Laboratorio.**
  - **Anaconda**
  - **Visual studio Code**
  - **Git**
- **Tipos de datos**
- **Estrutura codicionales y repetitivas**



Introducción

Repaso de Programacion Secuencial.

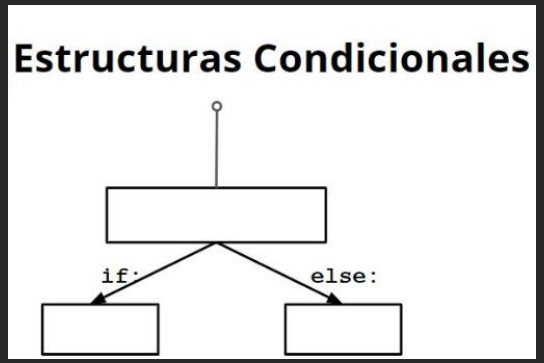
- **Como usar Vscode con Python.**
- **Instalacion del Laboratorio.**
  - **Anaconda**
  - **Visual studio Code**
  - **Git**
- **Tipos de datos**
- **Estrutura codicionales y repetitivas**

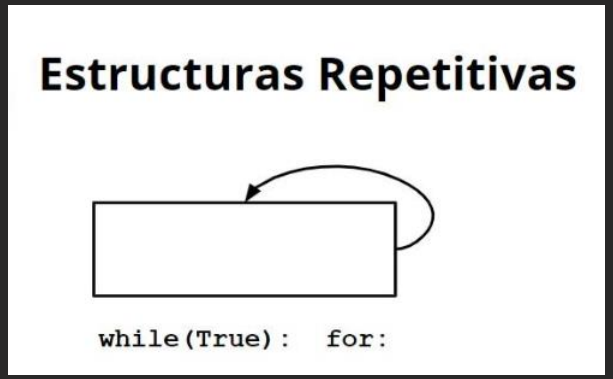


Tipos de datos

Tipo	Clase	Notas	Ejemplo
str	Cadena	Inmutable	"Hola"
unicode	Cadena	Versión Unicode de str	u"Hola"
list	Secuencia	Mutable, contiene objetos de diverso tipo	[4, "Hola", 3.14]
tuple	Secuencia	Inmutable, contiene objetos de diverso tipo	(4, "Hola", 3.14)
set	Conjunto	Mutable, sin orden y sin duplicados	set([4, "Hola", 3.14])
frozenset	Conjunto	Inmutable, sin orden, sin duplicados	frozenset([4, "Hola", 3.14])
dict	Diccionario	Pares de clave:valor	{"clave1": 4, "clave2": "Hola"}
int	Entero	Precisión fija, convierte a long si necesario	32
long	Entero	Precisión arbitraria	32L ó 12989182983989 23L
float	Decimal	Coma flotante de doble precisión	3.141592
complex	Complejo	Parte real e imaginaria.	(4.5 + 3j)
bool	Booleano	Valores verdadero o falso	True o False

Estructura condicionales y repetitivas.





1 0 0 1 0 1 0 1 0 1 0 1 0 0

1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0

Estruturas de Datos

1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0

Listas

Permite almacenar una colección ordenada y mutable de elementos.

Tuplas

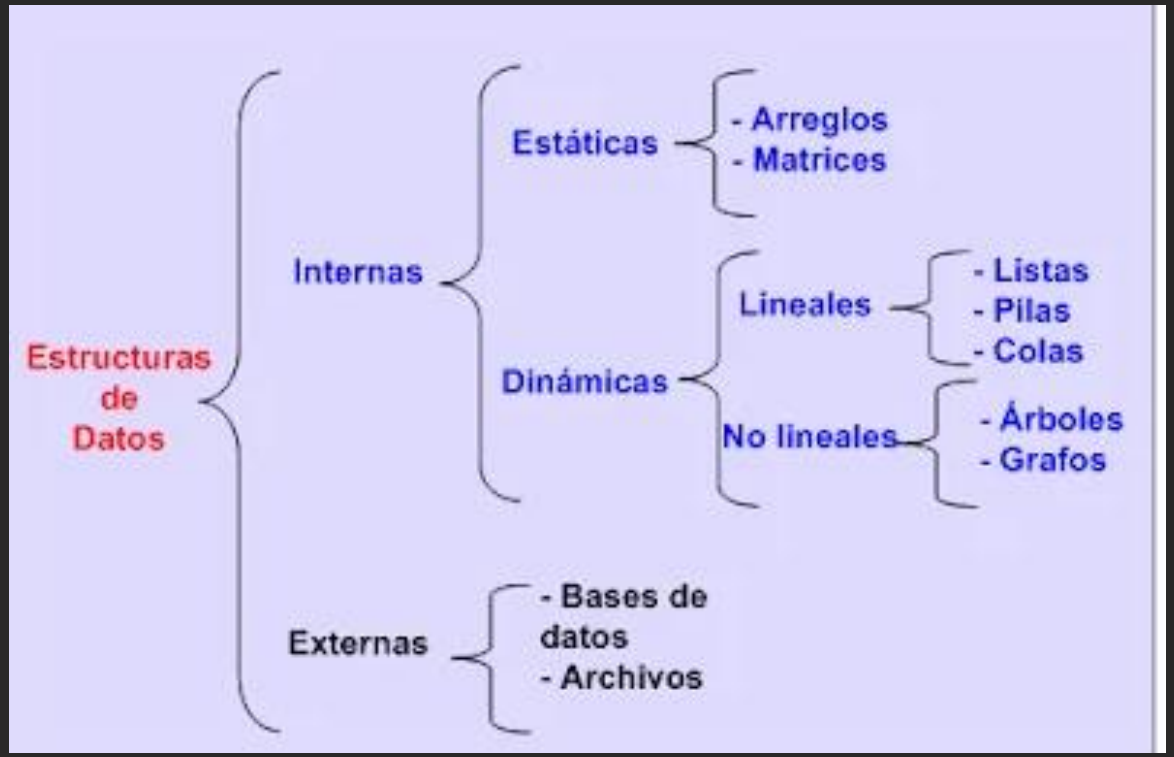
Es una **colección ordenada e inmutable de elementos.**

Diccionarios

Es una **colección no ordenada y mutable de pares clave-valor.**

1 0 1 0 0 1 0 0 1 0 1 1 0 1 1 1

Estructura de datos.



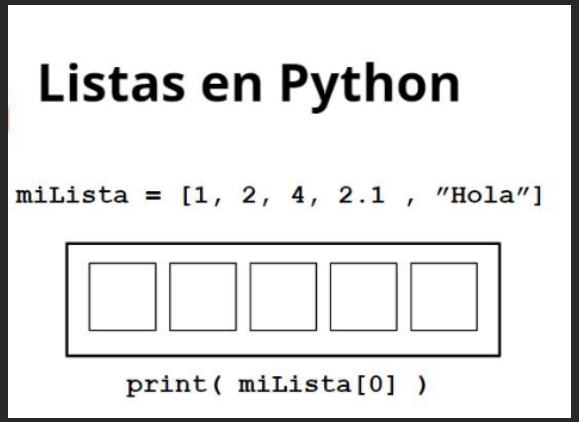
Estructura de datos tipo Lista

Que es una Lista?

Una **lista** es una **estructura de datos mutable y ordenada** que permite almacenar múltiples elementos en una sola variable. Los elementos de una lista pueden ser de cualquier tipo de datos, incluyendo números, cadenas, otras listas, objetos, etc.....

Problema:

Definir una lista que almacene 5 enteros. Sumar todos sus elementos y mostrar dicha suma.



Metodos útiles en las lista

Método	Descripción	Ejemplo
`append(x)`	Agrega un elemento al final de la lista	`lista.append(5)`
`insert(i, x)`	Inserta un elemento `x` en la posición `i`	`lista.insert(2, "dato")`
`remove(x)`	Elimina la primera aparición de `x`	`lista.remove(3)`
`pop(i)`	Elimina y devuelve el elemento en la posición `i`	`lista.pop(2)`
`index(x)`	Devuelve el índice de la primera aparición de `x`	`lista.index(4)`
`count(x)`	Cuenta cuántas veces aparece `x`	`lista.count(2)`
`sort()`	Ordena la lista en orden ascendente	`lista.sort()`
`reverse()`	Invierte el orden de la lista	`lista.reverse()`

Estructura de datos tipo Tuplas

```
tupla4 = (2,3,(4,5),[3,6], 'a','b')
```

Que es una Estrutura tipo tupla?

Una tupla permite almacenar una colección de datos no necesariamente del mismo tipo. Los datos de la tupla son inmutables a diferencia de las listas que son mutables.

Problema:

Definir varias tuplas e imprimir sus elementos.

Metodos útiles en las Tuplas

Método/Función	Descripción	Ejemplo	Salida
`count(x)`	Cuenta cuántas veces aparece `x` en la tupla	`(1,2,3,2,4,2).count(2)`	`3`
`index(x)`	Devuelve el índice de la primera aparición de `x`	`(10, 20, 30, 40).index(30)`	`2`
`len(tupla)`	Retorna el número de elementos	`len((5, 10, 15))`	`3`
`max(tupla)`	Retorna el valor máximo	`max((4, 7, 1, 9))`	`9`
`min(tupla)`	Retorna el valor mínimo	`min((4, 7, 1, 9))`	`1`
`sum(tupla)`	Retorna la suma de los elementos numéricos	`sum((1, 2, 3, 4))`	`10`
`sorted(tupla)`	Retorna una lista ordenada con los elementos	`sorted((5, 3, 8, 1))`	`[1, 3, 5, 8]`
`tuple(iterable)`	Convierte un iterable en tupla	`tuple([1, 2, 3])`	`(1, 2, 3)`

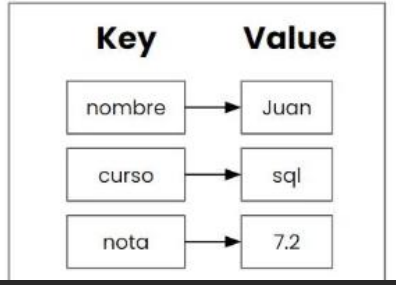
Estructura de datos tipo diccionario.

Que es una Estrutura tipo Diccionario?

Un **diccionario** es una estructura de datos que almacena pares de **clave-valor**. Es similar a una tabla de datos o un mapa, lo que permite acceder rápidamente a los valores mediante una clave única.

```
miDiccionario =

{ "nombre": "Juan",
  "curso": "sql",
  "nota": 7.2 }
```

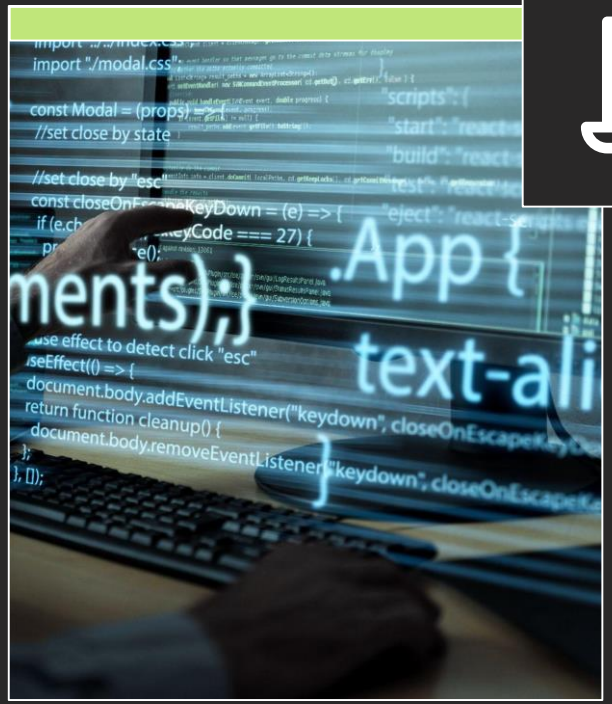


Problema:

Crear un diccionario donde puedas acceder a un valor, modificar un valor, agregar un nuevo valor,modificar un valor y eliminar un valor.

Metodos útiles en los diccionarios

Método	Descripción	Ejemplo	Salida
`dict.keys()`	Devuelve una vista con las claves	`persona.keys()`	`dict_keys(['nombre', 'edad', 'ciudad'])`
`dict.values()`	Devuelve una vista con los valores	`persona.values()`	`dict_values(['Juan', 30, 'Madrid'])`
`dict.items()`	Devuelve pares (clave, valor)	`persona.items()`	`dict_items([('nombre', 'Juan'), ('edad', 30), ('ciudad', 'Madrid')])`
`dict.get(clave, valor_defecto)`	Obtiene el valor de una clave (evita errores si no existe)	`persona.get("nombre")`	`'Juan'`
`dict.update(otro_dict)`	Agrega/modifica elementos desde otro diccionario	`persona.update({"pais": "España"})`	`{'nombre': 'Juan', 'edad': 30, 'ciudad': 'Madrid', 'pais': 'España'}`
`dict.pop(clave)`	Elimina un elemento y lo devuelve	`persona.pop("edad")`	`30`
`dict.clear()`	Borra todo el diccionario	`persona.clear()`	`{}`



02

Funciones

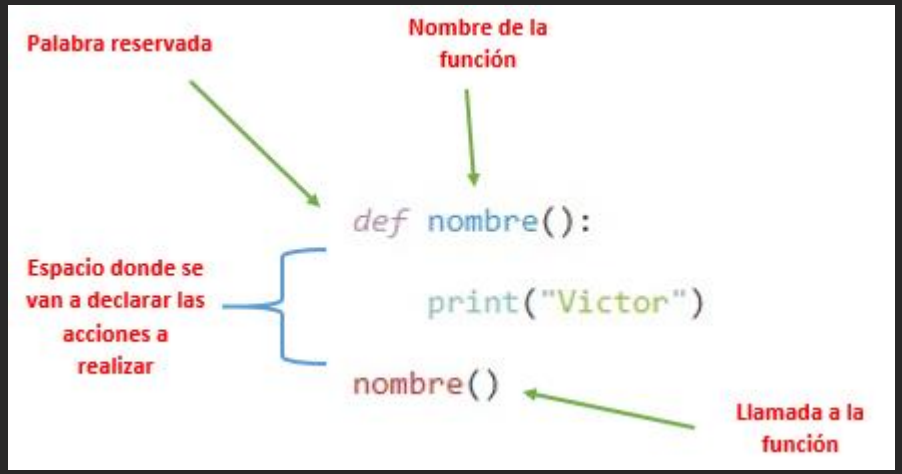
1 1 0 0 1 0 1 0 1 0 1 0 1 0 1 1 0

1 0 1 0 1 0 1 1 1 0 0 0 1 1 1 0

Que es una funcion?

1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0

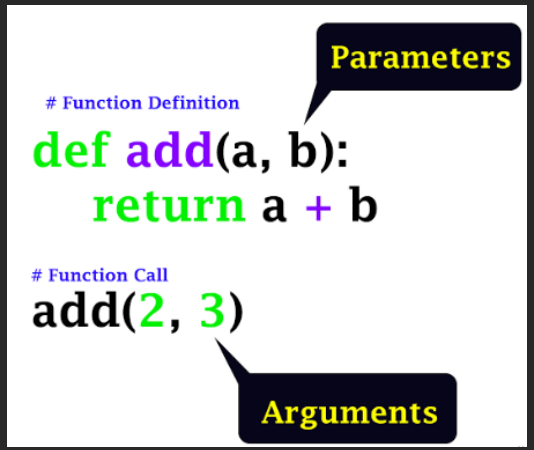
En Python, una función es un bloque de código reutilizable que realiza una tarea específica. Se define con la palabra clave def, puede recibir argumentos como entrada y devolver un valor como salida



1 0 1 0 0 1 0 0 1 0 1 1 0 1 1 1

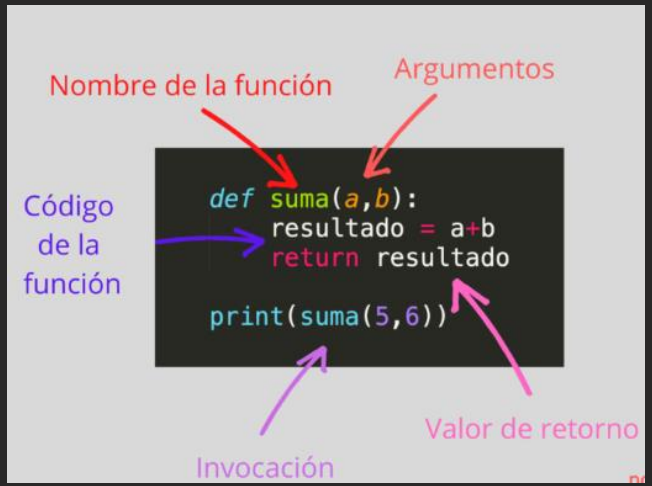
Tipo de funciones y sus uso

- Funciones con parametros: Una función con parámetros recibe valores para realizar una operación.



Tipo de funciones y sus uso

- Funciones de retorno de datos:Devuelven un valor después de procesar los datos.



Tipo de funciones y sus uso

- Funciones de parámetros de tipo lista: Reciben una lista como parámetro para operar sobre sus elementos.

Ejemplo:

```
def sumarizar(lista):
    suma=0
    for x in range(len(lista)):
        suma=suma+lista[x]
    return suma
```

Ejemplo: Función que suma los elementos de una lista

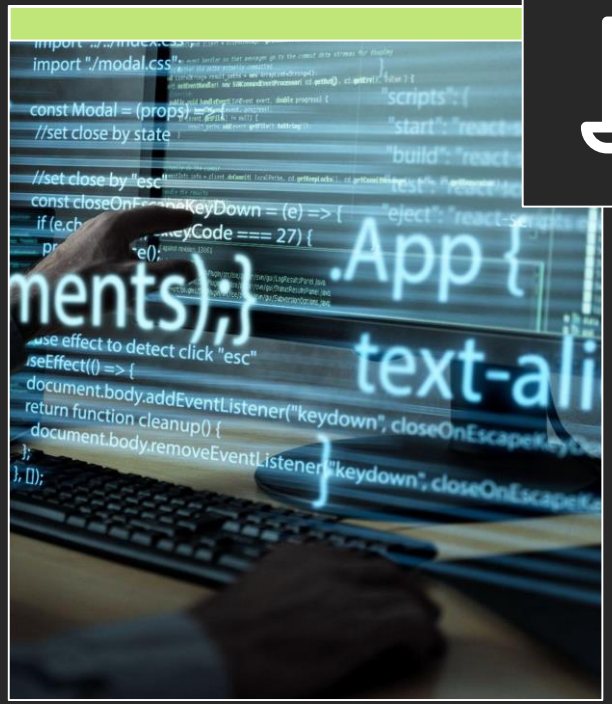
Tipo de funciones y sus uso

1. Funciones con parámetros con valor por defecto: Las funciones pueden tener **parámetros con valores por defecto**, lo que significa que si no se proporciona un argumento, se usa el valor predeterminado.

Ejemplo:

```
def titulo_subrayado(titulo,caracter="*"):
    print(titulo)
    print(caracter*len(titulo))
```

Ejercicio: Función que calcula el precio con impuestos



03

Bibliotecas

1 1 0 0 1 0 1 0 1 0 1 0 1 0 1 1 0

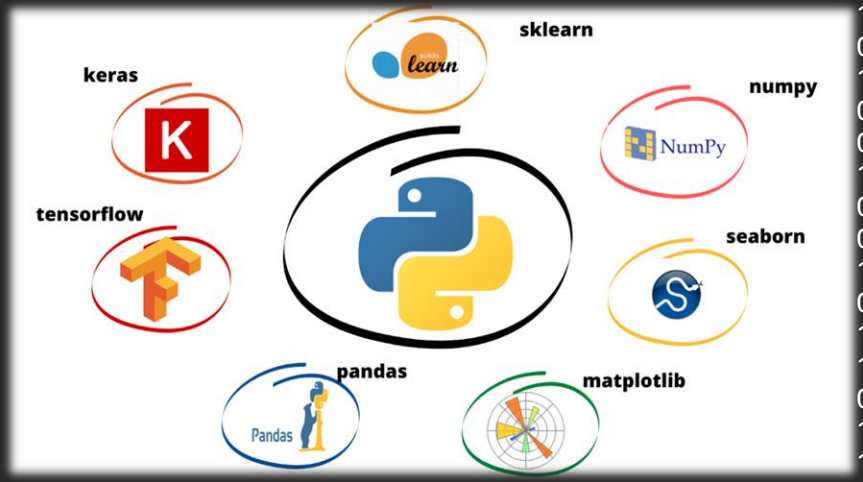
1 0 1 0 1 0 1 1 1 0 0 0 1 1 1 0

Que es una biblioteca ?

Una **biblioteca** o **librería** es un conjunto de **módulos y funciones** predefinidas que facilitan la programación al proporcionar herramientas listas para usar, evitando que los programadores tengan que escribir código desde cero.

Las bibliotecas pueden ser:

- Estándar: Vienen incluidas con Python.
- Externas: Se instalan con pip y amplían las capacidades de Python.



Diferentes biblioteca ?

1001010100100101010

Librería	Descripción	Instalación
math	Operaciones matemáticas avanzadas como trigonometría, logaritmos y raíces cuadradas.	Incluida en Python
random	Generación de números aleatorios y selección aleatoria de elementos.	Incluida en Python
datetime	Manejo de fechas y horas.	Incluida en Python
os	Interacción con el sistema operativo (archivos, directorios, variables de entorno).	Incluida en Python
sys	Permite acceder a funciones y variables del sistema, como argumentos de línea de comandos.	Incluida en Python
requests	Realiza solicitudes HTTP para consumir APIs y páginas web.	`pip install requests`
numpy	Operaciones matemáticas avanzadas, matrices y álgebra lineal.	`pip install numpy`
pandas	Manipulación y análisis de datos en estructuras como DataFrames.	`pip install pandas`
matplotlib	Creación de gráficos y visualización de datos.	`pip install matplotlib`
seaborn	Visualización avanzada de datos con gráficos estadísticos.	`pip install seaborn`
scipy	Funciones científicas y matemáticas avanzadas.	`pip install scipy`
scikit-learn	Algoritmos de aprendizaje automático (machine learning).	`pip install scikit-learn`
tensorflow	Creación y entrenamiento de modelos de inteligencia artificial.	`pip install tensorflow`

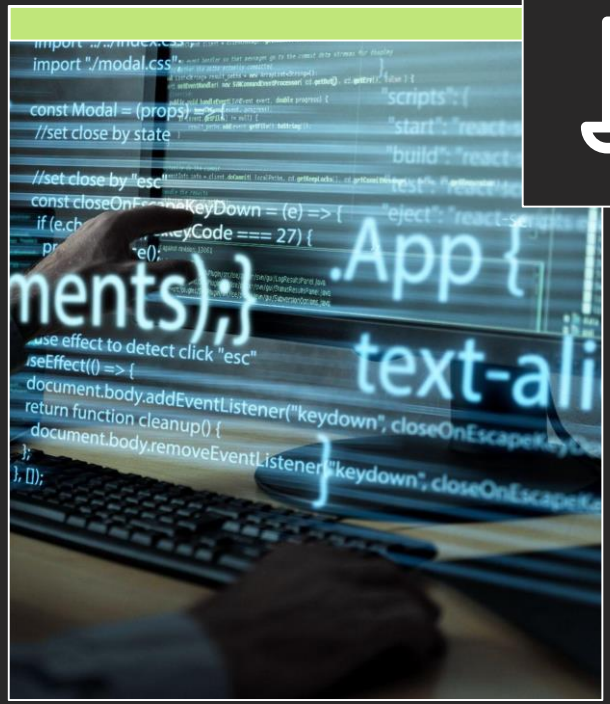
1010010010110111

Usando bibliotecas

1
0
0
1
0
1
0
1
0
0
1
0
1
0
1
0

- ✓ **Uso de math**
- ✓ **Uso de random**
- ✓ **Uso de datetime**
- ✓ **Uso de requests**

1
0
1
0
0
1
0
0
1
0
1
1
0
1
1
1



04

POO

Programacion Orientada a Objeto

1 1 0 0 1 0 1 0 1 0 1 0 1 0 1 1 0

Contenido

Algoritmos	Algoritmos de datos y estructura de datos secuencial
Estruturas de de datos	Lista, Tuplas, Diccionarios.
Funciones	Tipos de funciones y sus usos.
Bibliotecas	Estandares de Bibliotecas y sus usos.
Programacion POO	Metodos ,Clases,Herencias, Interfaces graficas.
Bases de datos.	MySQL,SQLite y PostgrestSQL.

Infotep Ciberseguridad 2025

Instrutor Carlos Garcia

Programacion Orientada a Objeto

1 0 0 1 0 1 0 1 0 1 0 1 0 0

1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0

Que es una objeto?

- Es un **paradigma de programación** que se basa en la idea de organizar el código en **objetos** que interactúan entre sí. Cada objeto es una instancia de una **clase**, que actúa como una plantilla con **atributos (datos)** y **métodos (acciones)**.

- Conceptos Claves de POO
- A.Clase
- B. Objeto
- C. Atributos
- D. Métodos
- E. Herencia
- F. Polimorfismo

Clases y objetos

Que es una Clase?

- Una **clase** es un molde o plantilla que define atributos (características) y métodos (acciones).
- Ejemplo:"Pensemos en una receta de cocina. La receta es la **clase**, porque define cómo hacer el platillo. Pero cada vez que alguien cocina, obtiene un plato diferente, que sería un **objeto** basado en esa receta.

Que es una Objeto?

- Un **objeto** es una instancia de una clase, con valores específicos.
- Ejemplo:"Si Persona es la clase, entonces Juan y Ana son objetos de esa clase."

Clases y objetos

```
class nombre_de_la_clase:
    código de la clase
```

```
class Classname:
    Objeto1 = valor
    Objeto2 = valor
    Objeto3 = valor
        .
        .
        .
```

Clases y objetos

```
class nombre_de_la_clase:
    código de la clase
```

```
class Classname:
    Objeto1 = valor
    Objeto2 = valor
    Objeto3 = valor
        .
        .
        .
```

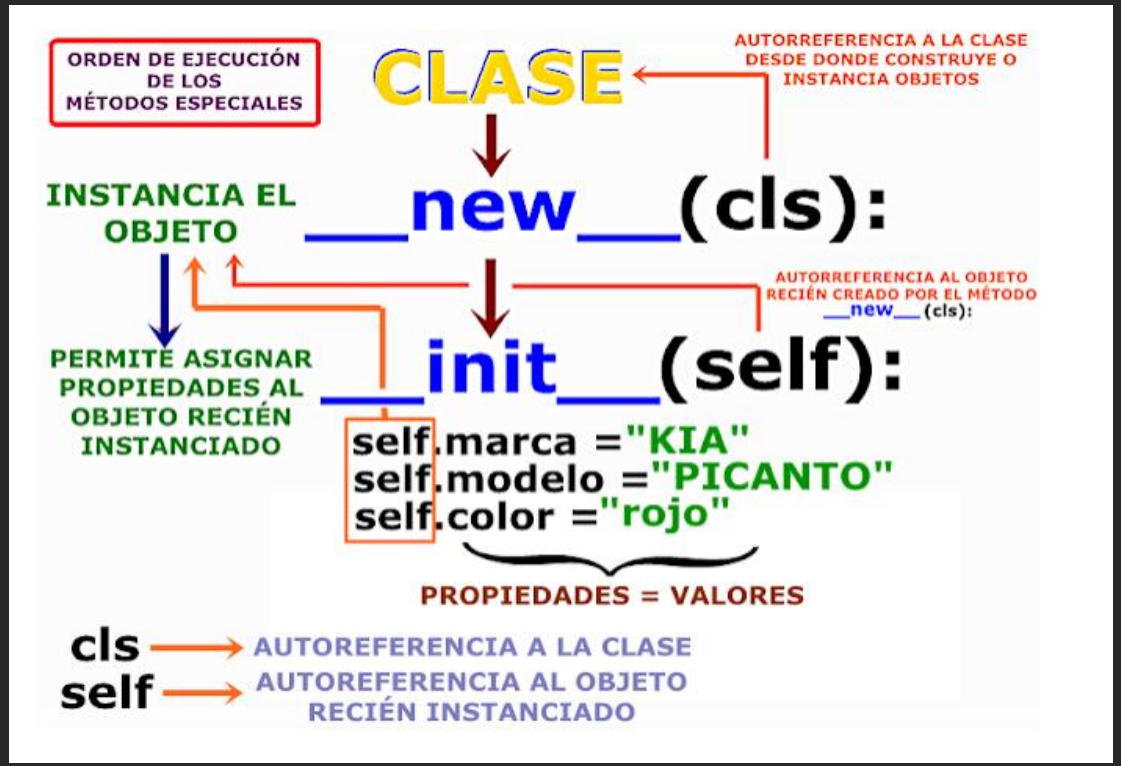
Clases y objetos

```
class nombre_de_la_clase:
    código de la clase
```

```
class Classname:
    Objeto1 = valor
    Objeto2 = valor
    Objeto3 = valor
    .
    .
    .
```

Ejemplo:

Implementaremos una clase llamada Persona que tendrá como atributo (variable) su nombre y dos métodos (funciones), uno de dichos métodos inicializará el atributo nombre y el siguiente método mostrará en la pantalla el contenido del mismo..


ORDEN DE EJECUCIÓN DE LOS MÉTODOS ESPECIALES
CLASE
AUTORREFERENCIA A LA CLASE DESDE DONDE CONSTRUYE O INSTANCIA OBJETOS
INSTANCIA EL OBJETO
__new__(cls):
AUTORREFERENCIA AL OBJETO RECIÉN CREADO POR EL MÉTODO __new__(cls):
PERMITE ASIGNAR PROPIEDADES AL OBJETO RECIÉN INSTANCIADO
__init__(self):
self.marca ="KIA"
self.modelo ="PICANTO"
self.color ="rojo"
PROPIEDADES = VALORES
cls → AUTOREFERENCIA A LA CLASE
self → AUTOREFERENCIA AL OBJETO RECIÉN INSTANCIADO

Atributos y Metodos

Que es una atributo?

- Son las características de un objeto, almacenadas como variables dentro de la clase.

"Si un coche tiene color y marca, estos son atributos.

Que es una Metodo?

- Son funciones dentro de una clase que definen comportamientos del objeto.

"Si un coche puede acelerar o frenar, esas acciones son métodos."

Ejercicio : Clase Automovil con sus métodos y atributos

Método __init__ de la clase

El método __init__ es un método especial de una clase en Python. El objetivo fundamental del método __init__ es inicializar los atributos del objeto que creamos

- El método __init__ es el primer método que se ejecuta cuando se crea un objeto.
- El método __init__ se llama automáticamente. Es decir es imposible de olvidarse de llamarlo ya que se llamará automáticamente.
- Quien utiliza POO en Python (Programación Orientada a Objetos) conoce el objetivo de este método.

Otras características del método __init__ son:

- Se ejecuta inmediatamente luego de crear un objeto.
- El método __init__ no puede retornar dato.
- el método __init__ puede recibir parámetros que se utilizan normalmente para inicializar atributos.
- El método __init__ es un método opcional, de todos modos es muy común declararlo.

1001010101010100

1010010101010010

Método __init__ de la clase

Las ventajas de implementar el método __init__ en lugar del método inicializar son:

1. El método __init__ es el primer método que se ejecuta cuando se crea un objeto.
2. El método __init__ se llama automáticamente. Es decir es imposible de olvidarse de llamarlo ya que se llamará automáticamente.
3. Quien utiliza POO en Python (Programación Orientada a Objetos) conoce el objetivo de este método.

Sintaxis del constructor:

```
def __init__([parámetros]):
        [algoritmo]
```

Ejemplo: Confeccionar una clase que represente un empleado. Definir como atributos su nombre y su sueldo. En el método __init__ cargar los atributos por teclado y luego en otro método imprimir sus datos y por último uno que imprima un mensaje si debe pagar impuestos (si el sueldo supera a 3000.

Herencia – Definición

- La **herencia** permite que una clase (llamada **subclase** o **clase hija**) herede atributos y métodos de otra clase (llamada **superclase** o **clase padre**). Esto promueve la reutilización del código.
- La herencia permite crear nuevas clases que reutilizan, extienden o modifican una clase existente.
- - Clase padre: atributos/métodos comunes
- - Clase hija: hereda y puede sobrescribir

1001010101010100

1010010101001010

Herencia — Ejemplo en Código

```
class Animal:
    def __init__(self, nombre):
        self.nombre = nombre
    def hablar(self):
        print(f"{self.nombre} hace un sonido.")

class Perro(Animal):
    def hablar(self):
        print(f"{self.nombre} dice: ¡Guau!")
```

Herencia – Uso

- perro = Perro("Firulais")
- gato = Gato("Michi")

- perro.hablar() # Firulais dice: ¡Guau!
- gato.hablar()   # Michi dice: ¡Miau!

Polimorfismo — Definición

- Permite que objetos de diferentes clases respondan de forma distinta al mismo método.
- - Mismo nombre de método
- - Comportamiento diferente

Polimorfismo – Ejemplo

- animales = [Perro("Bobby"), Gato("Luna")]

- for animal in animales:
- animal.hablar()

- # Salida:
- # Bobby dice: ¡Guau!
- # Luna dice: ¡Miau!

Diferencias Clave

- Herencia : Reutilización de código entre clases.

  Objetivo: Reutilizar código, evitar duplicación y organizar clases jerárquicamente.

- El **polimorfismo** permite usar un mismo método en múltiples clases, donde cada clase implementa el método de manera distinta.

  Permitir diferentes comportamientos con una misma interfaz (nombre de método), facilitando el mantenimiento y la escalabilidad.

1 0 0 1 0 1 0 1 0 1 0 1 0 0

1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0

¿Qué es una GUI?

- Una **GUI (Graphical User Interface)** o **Interfaz Gráfica de Usuario** es una forma visual de interactuar con un programa o sistema operativo usando elementos como: Ventana, Botones, cuadro de texto, menú icono.

- - Interfaz visual para interactuar con programas
  - Sustituye la línea de comandos
  - Ejemplos: ventanas, botones, campos de texto

- Bibliotecas populares:
  - Tkinter (estándar)
  - PyQt, Kivy, wxPython

1001010101010100

1010010101010010

Ejemplo Básico con Tkinter

```
import tkinter as tk
from tkinter import messagebox

def saludar():
    messagebox.showinfo("Saludo", "¡Hola, mundo!")

ventana = tk.Tk()
ventana.title("Mi Primera GUI")
ventana.geometry("300x150")

boton = tk.Button(ventana, text="Haz clic aquí", command=saludar)
boton.pack(pady=20)

ventana.mainloop()
```

¿Qué Hace Este Código?

- - Crea una ventana con `tk.Tk()`
- - Agrega un botón con `tk.Button`
- - Muestra mensaje con `messagebox.showinfo`
- - Ejecuta la interfaz con `mainloop()`

¿Para Qué Se Usa una GUI en Python?

- - Formularios de entrada
- - Herramientas de cálculo
- - Automatización con botones
- - Interfaces para scripts
- - Aplicaciones educativas y personales

Que es una API?

1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0

Es un conjunto de reglas y definiciones que permiten que diferentes programas o sistemas se comuniquen entre sí.
En términos simples, es como un **mensajero** que recibe solicitudes, obtiene datos o ejecuta acciones y devuelve una respuestas.

1 0 1 0 0 1 0 0 1 0 1 1 0 1 1 1

Trabajo final

- Desarrollar una aplicación visual con la librería tkinter que permita implementar los algoritmos de carga de artículos, consulta por código y listado completo.