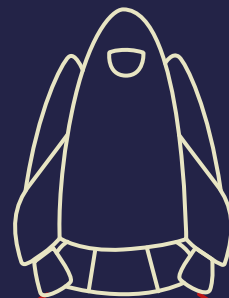


# Programa académico CAMPUS



Ciclo 1:  
Fundamentos de  
Programación



## Presentación Ciclo 1 – Fundamentos de Programación

**Temas**

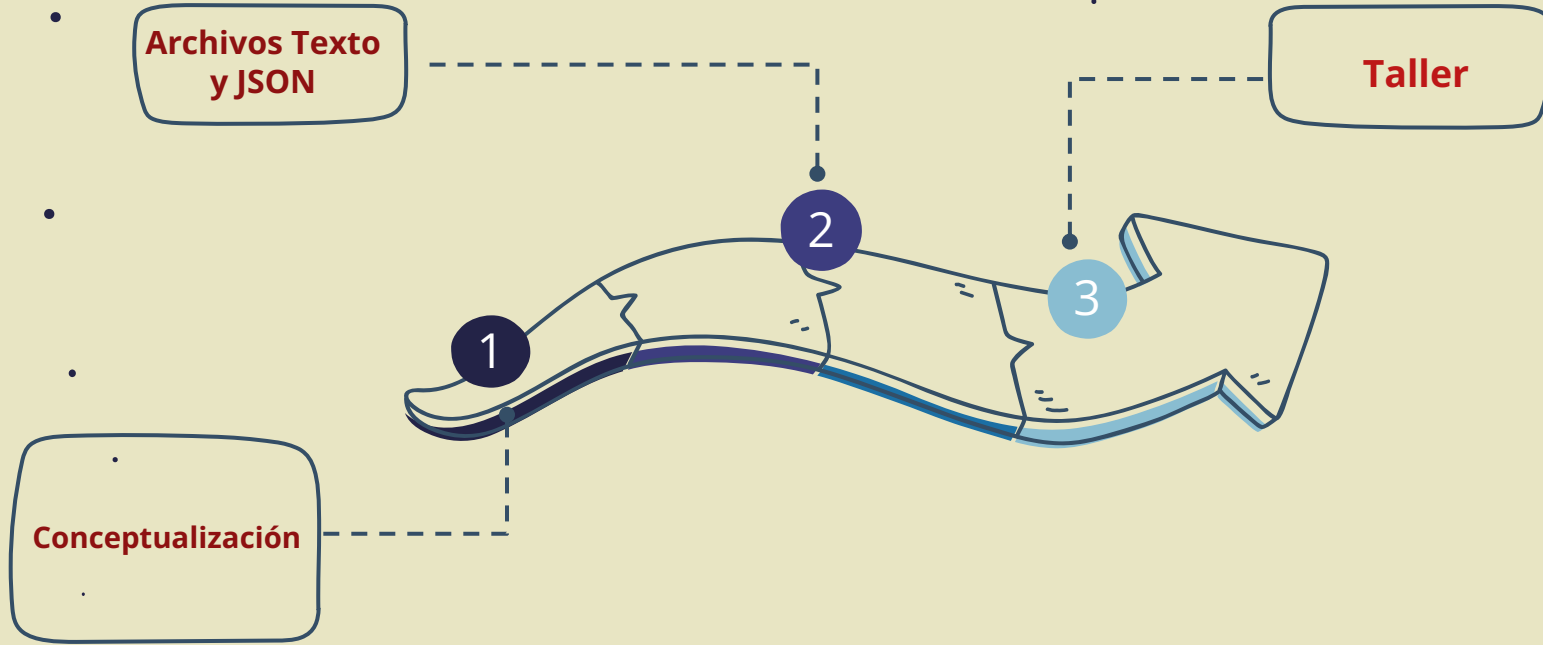


**Persistencia  
de Datos**



Archivos Texto
Archivos JSON

# Persistencia de Datos



# Persistencia de Datos

## Conceptualización

La **persistencia de datos** se refiere a la capacidad de un sistema para **almacenar datos** de manera **permanente**. Esto significa que los datos deben ser guardados de manera **segura y accesible** para su uso futuro, incluso después de que el sistema que los generó se haya cerrado.

Existen diferentes tecnologías y herramientas que permiten la persistencia de datos, como las bases de datos, sistemas de archivos, sistemas de almacenamiento en la nube, entre otros. Cada una de estas tecnologías tiene diferentes características en términos de velocidad, escalabilidad, capacidad de almacenamiento, seguridad y otras funcionalidades.

La **persistencia de datos** es un aspecto fundamental en el desarrollo de aplicaciones y sistemas informáticos. Es importante tener en cuenta aspectos como la **integridad** de los datos, la **escalabilidad** y el **rendimiento** del sistema, así como la **seguridad** y la **privacidad** de la información almacenada. También es importante considerar el uso de técnicas de respaldo y recuperación de datos para garantizar que los datos estén disponibles en caso de fallas del sistema o errores humanos.

# Persistencia de Datos

## Conceptualización



El concepto de archivo proviene del latín **archivum**, y se refiere al conjunto de **documentos producidos** por **personas naturales o jurídicas, públicas o privadas**, en ejercicio de su actividad

# Persistencia de Datos

## Conceptualización

Se llama “archivo” al **elemento de información compuesto por una suma de registros** (combinaciones de bytes). Llevan este nombre por ser los equivalentes digitalizados de los archivos antes descritos. Tanto es así que muchos de los archivos “en papel” se están actualmente digitalizando, para reducir su tamaño físico y facilitar su organización y búsqueda. Los archivos informáticos, en general, tienen algunas características en común:

# Persistencia de Datos

## Conceptualización

**Nombre.** Cada archivo es identificable con un nombre, que no puede coincidir con otro que esté en la misma ubicación.

**Extensión.** Los archivos llevan una extensión opcional, que muchas veces indica su formato

**Tamaño.** Como se dijo, están compuestos por una serie de bytes que determinan su tamaño. Puede alcanzar kilobytes, megabytes, gigabytes.

# Persistencia de Datos

## Conceptualización

### Archivos: De texto

Extensión	Tipo de Archivo
.txt	Texto plano
.xml	XML
.json	De intercambio de información
.props	De propiedades
.conf	De configuración
.sql	Script SQL
.srt	De subtítulo





# Persistencia de Datos

## Conceptualización

### Archivos: Binarios

Extensión	Tipo de Archivo
.pdf	PDF
.jpg, .png	De imagen
.doc, .docx	Microsoft Word
.avi	Video
.ppt, .pptx	Microsoft PowerPoint
.pdf	PDF
.exe	Programa ejecutable

# Persistencia de Datos

## Archivos Texto



Estos **archivos** están compuestos de **bytes** que representan caracteres ordinarios como *letras, números y signos de puntuación* (incluyendo espacios en blanco), también incluye algunos pocos **caracteres de control** como *tabulaciones, saltos de línea y retornos de carro*.

# Persistencia de Datos

## Archivos Texto

El manejo de archivos externos se realiza a través del módulo de Python io y su objetivo es

la persistencia de datos, es decir poder almacenar los datos que se utilizan en los programas, en este caso en archivos de texto plano.

# Persistencia de Datos

## Archivos: Pasos para interactuar

Existen tres (3) pasos básicos para interactuar con archivo:

### 1. Abrir el archivo

- En Python para abrir un archivo usamos la función `open()`

**`f = open("data.txt", "r")`**

f es un HANDLE (manejador)

f es un objeto de tipo `<class '_io.TextIOWrapper'>`

### 2. Manipular o realizar operaciones con el archivo

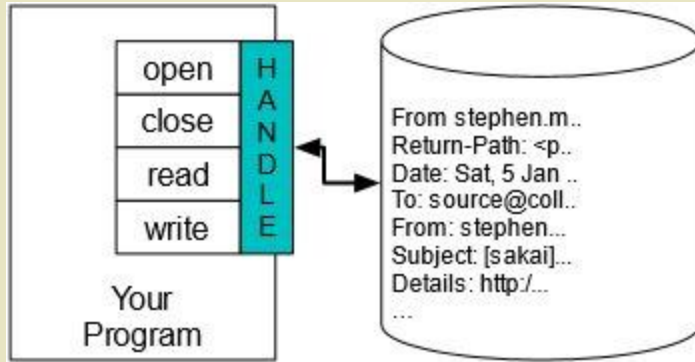
### 2. Cerrar el archivo

En Python para cerrar un archivo usamos el método `close` del objeto `'_io.TextIOWrapper'`

**`f.close()`**

# Persistencia de Datos

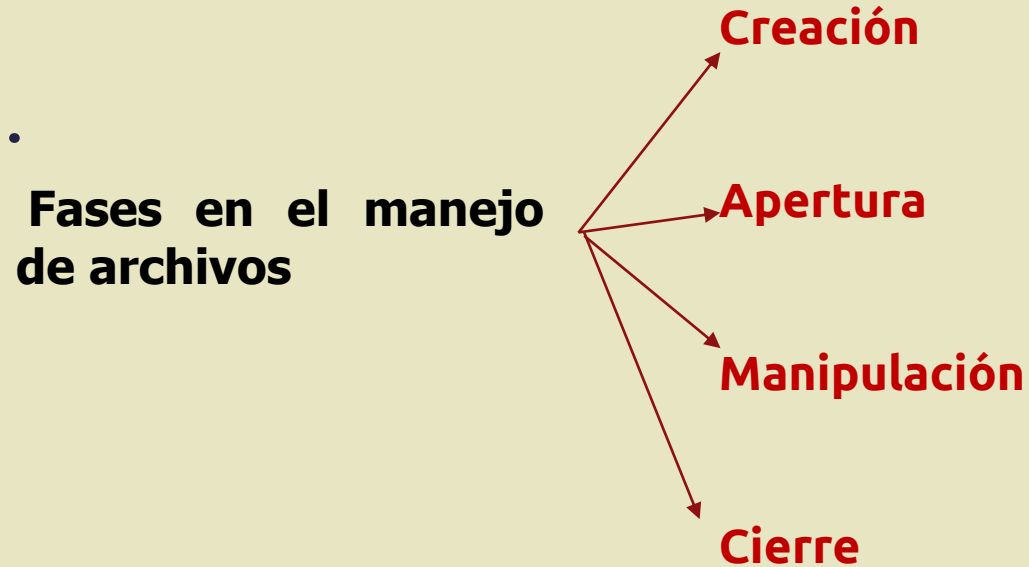
## Qué es un HANDLE?



El **Handle** del archivo no son los datos reales contenidos en el archivo, sino que es un "**identificador**" que podemos usar para leer los datos. Se le proporciona un identificador si el archivo solicitado existe y tiene los permisos adecuados para leer el archivo.

# Persistencia de Datos

Archivos Texto



# Persistencia de Datos

## Modos de manejo de un archivo

La función open recibe un parámetro opcional para indicar el modo en que se desea abrir el archivo. Los tres modos de apertura permitidos son:

1. De sólo lectura (**r**). No es posible realizar modificaciones sobre el archivo, solamente se puede leer su contenido.
2. De sólo escritura (**w**). El archivo es truncado (vaciado) si existe, y si no se crea.
3. Sólo escritura posicionándose al final del archivo (**a**). Se crea el archivo, si no existe, en caso de que exista se posiciona al final, manteniendo el contenido original

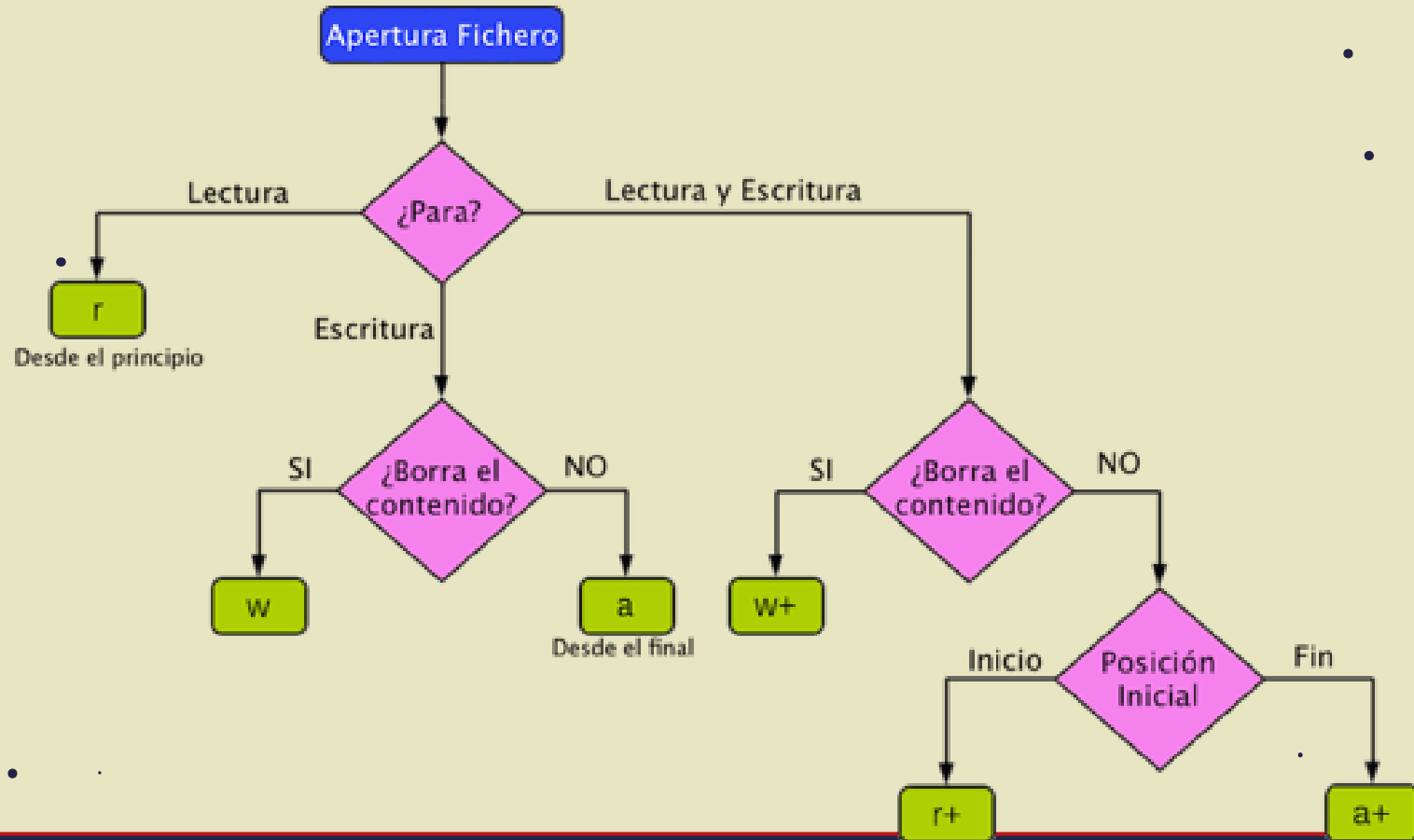
# Persistencia de Datos

## Modos de manejo de un archivo

Modo	Significado específico
'r'	Leer (predeterminado)
'w'	Escribir (truncar primero el contenido anterior)
'x'	Escribir, si el archivo ya existe generará una excepción
'a'	Agregar, escribir contenido al final de un archivo existente
'b'	Modo binario
't'	Modo de texto (predeterminado)
'+'	Actualización (tanto lectura como escritura)



# Persistencia de Datos



# Persistencia de Datos

## Manejo de excepciones

- FileNotFoundError
- InterruptedError
- LookupError
- UnicodeDecodeError
- IsADirectoryError
- NotADirectoryError
- PermissionError
- OSError
- ProcessLookupError
- TimeoutError

# Persistencia de Datos

## Archivos Texto

**open** => Abrir el archivo: Nombre y modo (lectura. (**r**), escritura (**w**), agregar (**a**))

**write** => Escribir en el archivo

**read** => Leer información del archivo (Total)

**readlines** => Leer la información del archivo línea a línea

**close** => Cerrar el archivo

# Persistencia de Datos

## Leer desde un archivo de texto

Existen varias formas para leer el contenido de un archivo:

1. Hacia una cadena con **read()**.
2. Leer una línea con **readline()**.
3. Obtener una lista conteniendo las líneas del archivo con **readlines()**.

Los tres métodos aceptan **un parámetro entero opcional** que define el *número máximo de bytes a leer del archivo*. Si este parámetro es negativo o simplemente se omite, **read** y **readlines** leerán todo el archivo y **readline** una línea completa sin importar su largo.



# Persistencia de Datos

## Leer desde un archivo de texto

Un archivo de texto se puede considerar como una secuencia de líneas

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
Return-Path: <postmaster@collab.sakaiproject.org>
Date: Sat, 5 Jan 2008 09:12:18 -0500
To: source@collab.sakaiproject.org
From: stephen.marquard@uct.ac.za
Subject: [sakai] svn commit: r39772 - content/branches/

Details:
http://source.sakaiproject.org/viewsvn/?view=rev&rev=39772
```

# Persistencia de Datos

## Leer desde un archivo de texto

Un archivo de texto tiene nuevas líneas al final de cada línea.

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008\nReturn-Path: <postmaster@collab.sakaiproject.org>\nDate: Sat, 5 Jan 2008 09:12:18 -0500\nTo: source@collab.sakaiproject.org\nFrom: stephen.marquard@uct.ac.za\nSubject: [sakai] svn commit: r39772 - content/branches/\n\nDetails: http://source.sakaiproject.org/viewsvn/?view=rev&rev=39772\n
```

# Persistencia de Datos

## Archivos Texto

Lectura de archivo texto en Python (Toda la información)

Fases:

Apertura, manipulación y cierre



```
>>>
= RESTART: D:/UNAB/Materiales/s
o.py
Sergio Medina
Luisa Ruiz
Mario Moreno
>>> |
```

```
#Importar el módulo io
import io

#Fase de apertura
archivo_texto=open("nombres.txt", "r")

#Fase de manipulación
texto=archivo_texto.read()

#Fase de cierre
archivo_texto.close()

print(texto)
```

# Persistencia de Datos

## Ejemplo 1: Contar líneas en un archivo

- Los archivos **mbox.txt** y **mbox-short.txt** están en un formato estándar para un archivo que contiene varios mensajes de correo
- Las líneas que comienzan con “**From**” separan los mensajes y las líneas que comienzan con “From:” son parte de los mensajes. Para obtener más información sobre el formato mbox, consulte <https://en.wikipedia.org/wiki/Mbox>.



# Persistencia de Datos

## Ejemplo 1: Contar líneas en un archivo

**From** stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

Return-Path: <postmaster@collab.sakaiproject.org>

Date: Sat, 5 Jan 2008 09:12:18 -0500

To: source@collab.sakaiproject.org

**From:** stephen.marquard@uct.ac.za

Subject: [sakai] svn commit: r39772 - content/branches/

Details: <http://source.sakaiproject.org/viewsvn/?view=rev&rev=3977>

# Persistencia de Datos

## Ejemplo 1: Contar líneas en un archivo

- Abra el **archivo** en solo lectura
- Use un ciclo **for** para leer cada línea
- **Cuente** las líneas que hay
- Los archivos **mbox.txt** y **mbox-short** están en un formato estándar para un archivo que contiene varios mensajes de correo



# Persistencia de Datos

## Ejemplo 1: Contar líneas en un archivo



# Persistencia de Datos

## Ejemplo 2: Buscar en el archivo


- Podemos poner una declaración **if** en nuestro bucle **for** para imprimir solo líneas que cumplan con algunos criterios
- **Busque** las líneas que empiezan con la palabra “**From:**” y muéstrelas por pantalla

```
fhand = open('mbox-short.txt')
for line in fhand:
    if line.startswith('From:'):
        print(line)
```

# Persistencia de Datos

## Ejemplo 2: envío de mensajes

- Diseñe un programa que simule el envío de mensajes a los correos que aparecen en el **From:** de mbox-short.
- El primer mensaje que se enviará es a la última dirección de correo que aparece en el mbox, el siguiente es el anterior y así sucesivamente.
- El mensaje que se muestra en la pantalla es el que aparece a la derecha:



```
cwen@iupui.edu enviado [ok]  
ray@media.berkeley.edu enviado [ok]
```

# Persistencia de Datos

## Archivos: Escribir en un archivo

Presentamos dos formas comunes de escribir a un archivo:

- Mediante cadenas: escribe una cadena al archivo.

**archivo.write(cadena)**

- Mediante listas de cadenas: recibe una lista de líneas para escribir.

**archivo.writelines(lista\_de\_cadenas)**

# Persistencia de Datos

## Archivos: Escribir en un archivo

### 1. Con write()

```
línea= "Linea 1"  
f = open("archivo.txt", "w")  
f.write(línea)  
f.close()
```

segunda línea\n


### 2. Con writelines()

```
lista = ["línea 1\n", "línea 2"]  
f = open("archivo.txt", "w")  
f.writelines(lista)  
f.close()
```

# Persistencia de Datos

## Ejemplo 1: Escribir en disco - envío de mensajes

- Diseñe un programa que simule el envío de mensajes a los correos que aparecen en el **From:** de mbox-short.
- El primer mensaje que se enviará es a la última dirección de correo que aparece en el mbox, el siguiente es el anterior y así sucesivamente.
- El mensaje que se muestra en la pantalla es el que aparece a la derecha.
- Adicional que cree un archivo que contenga los mensajes tal como se imprimieron en la consola.



```
cwen@iupui.edu enviado [ok]  
ray@media.berkeley.edu enviado [ok]
```



# Persistencia de Datos

## Archivos: Escribir en un archivo

Fases:

Apertura, creación,  
manipulación y  
cierre



nombres: Bloc de notas

Archivo Edición Formato V

Sergio Medina

Luisa Ruiz

Mario Moreno

```
#Importar el módulo io
import io

#Fase de creación y apertura
archivo_texto=open("nombres.txt", "w")

#Fase de manipulación
nom="Sergio Medina \nLuisa Ruiz \nMario Moreno"
archivo_texto.write(nom)

#Fase de cierre
archivo_texto.close()
```

# Persistencia de Datos

## Archivos Texto

### Creación de archivo texto en Python

Fases:

Apertura, creación,  
manipulación y  
cierre



nombres: Bloc de notas

Archivo Edición Formato V

Sergio Medina

Luisa Ruiz

Mario Moreno

```
#Importar el módulo io
import io

#Fase de creación y apertura
archivo_texto=open("nombres.txt", "w")

#Fase de manipulación
nom="Sergio Medina \nLuisa Ruiz \nMario Moreno"
archivo_texto.write(nom)

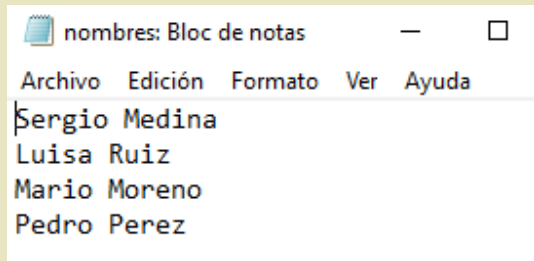
#Fase de cierre
archivo_texto.close()
```

# Persistencia de Datos

## Archivos Texto

Agregar información a un archivo texto en Python

Fases:  
Apertura, manipulación  
cierre



```
#Importar el módulo io
import io

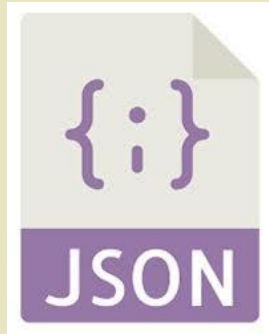
#Fase de apertura
archivo_texto=open("nombres.txt","a")

#Fase de manipulación
nom="\nPedro Perez"
archivo_texto.write(nom)

#Fase de cierre
archivo_texto.close()
```

# Persistencia de Datos

## Archivos JSON



Los **archivos JSON**, cuyo nombre corresponde a las siglas **JavaScript Object Notation** o Notación de Objetos de JavaScript, es un formato ligero de intercambio de datos, que resulta sencillo de leer y escribir para los programadores y simple de interpretar y generar para las máquinas.

Permiten almacenar estructuras de datos de Python, como listas o diccionarios para que se puedan utilizar en otro momento o en otro programa

# Persistencia de Datos

## Características JSON



- JSON es solo un formato de datos.
- Requiere usar comillas dobles para las cadenas y los nombres de propiedades. Las comillas simples no son válidas.
- Una coma o dos puntos mal ubicados pueden producir que un archivo JSON no funcione.
- Puede tomar la forma de cualquier tipo de datos que sea válido para ser incluido en un JSON, no solo arreglos u objetos. Así, por ejemplo, una cadena o un número único podrían ser objetos JSON válidos.
- A diferencia del código JavaScript, en el que las propiedades del objeto pueden no estar entre comillas, en JSON solo las cadenas entre comillas pueden ser utilizadas como propiedades.

# Persistencia de Datos

## Ejemplos y tipos de datos

A nivel granular, JSON está formado por tipos de datos.

1. Cadena
2. Número
3. Booleano
4. Nulo
5. Almacenamiento
6. Matriz



# Persistencia de Datos

## Ejemplos y tipos de datos



### Cadena

Una cadena en JSON se compone de caracteres Unicode, con escape de barra invertida (\).

### Ejemplo

```
{ "name" : "Jones" }
```

# Persistencia de Datos

## Ejemplos y tipos de datos



### Número

El número JSON sigue el formato de punto flotante de precisión doble de JavaScript.

### Ejemplo

```
{  
  "number_1" : 210,  
  "number_2" : 215,  
  "number_3" : 21.05,  
  "number_4" : 10.05  
}
```



# Persistencia de Datos

## Ejemplos y tipos de datos



### Booleano

Los valores booleanos se designan como `verdadero` o `falso`. Los valores booleanos no están delimitados por comillas y se tratan como valores de cadena.

### Ejemplo

```
{ "AllowPartialShipment" : false }
```

# Persistencia de Datos

## Ejemplos y tipos de datos



### Nulo

Nulo es un valor vacío. Cuando no hay ningún valor que asignar a una clave, se puede tratar como nulo.

### Ejemplo

```
{ "Special Instructions" : null }
```

# Persistencia de Datos

## Ejemplos y tipos de datos



### Almacenamiento

El tipo de datos de objeto JSON es un conjunto de pares de nombres o valores insertados entre {} (llaves). Las claves deben ser cadenas y deben ser únicas separadas por comas.

### Ejemplo

```
{  
  "Influencer" : { "name" : "Jaxon" , "age" : "42" , "city" , "New York" }  
}
```

# Persistencia de Datos

## Ejemplos y tipos de datos



### Matriz

Un tipo de datos de matriz es una recopilación ordenada de valores. En JSON, los valores de matriz deben ser cadena, número, objeto, matriz, booleano o **nulo**.

# Persistencia de Datos

## Ejemplos y tipos de datos



### Ejemplo

```
{  
  
  "Influencers" :  [  
    {  
      "name" : "Jaxon",  
      "age" : 42,  
      "Works At" : "Tech News"  
    },  
    {  
      "name" : "Miller",  
      "age" : 35,  
      "Works At" : "IT Day"  
    }  
  ]  
}
```

# Persistencia de Datos

## Archivos JSON

```
{
  "firstName": "John",
  "lastName": "Smith",
  "address": [{
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": 10021
  }, {
    "streetAddress": "577 Airport Blvd",
    "city": "Burlingame",
    "state": "CA",
    "postalCode": 94010
  }],
  "phoneNumbers": [
    "212-732-1234",
    "646-123-4567"
  ]
}
```



# Persistencia de Datos

## Archivos JSON

- Utiliza la librería o módulo json: `import json`
- Se realizar su apertura con el comando `open` igual que los archivos texto.
- El método para almacenar en el archivo json es `dump`
- El método para leer o cargar la información del archivo json es `load`
- El método para cerrar el archivo json es `close`

# Persistencia de Datos

## Archivos JSON

### Archivos json en Python

Crear archivo json con una estructura de datos lista

```
import json

# Estructura de datos (Lista)
lista=[10,20,30,40,60]

# Fase de apertura y creación
with open("lista.json","w") as archivo:
    json.dump(lista,archivo)

# Fase de cierre
archivo.close()
```



# Persistencia de Datos

## Archivos JSON

Archivos json en Python

•

**Crear archivo json  
con una  
estructura de  
datos diccionario**

•

```
import json

# Estructura de datos (Diccionario)
diccionario={'1':'Lapiz','2':'Borrador','3':'Cuaderno','4':'Lapicero'}

# Fase de apertura y creación
with open("diccionario.json","w") as archivo:
    json.dump(diccionario,archivo)

#Fase de Cierre
archivo.close()
```

# Persistencia de Datos

## Archivos JSON

Archivos json en Python

Leer un archivo  
json para  
recuperar la  
estructura de  
datos lista

```
import json

with open("lista.json", "r") as archivo:
    lista=json.load(archivo)

archivo.close()

print("Lista: ", lista)

for i in range(len(lista)):
    print(lista[i])
```

# Persistencia de Datos

## Archivos JSON

Archivos json en Python

Leer un archivo  
json para  
recuperar la  
estructura de  
datos  
diccionario

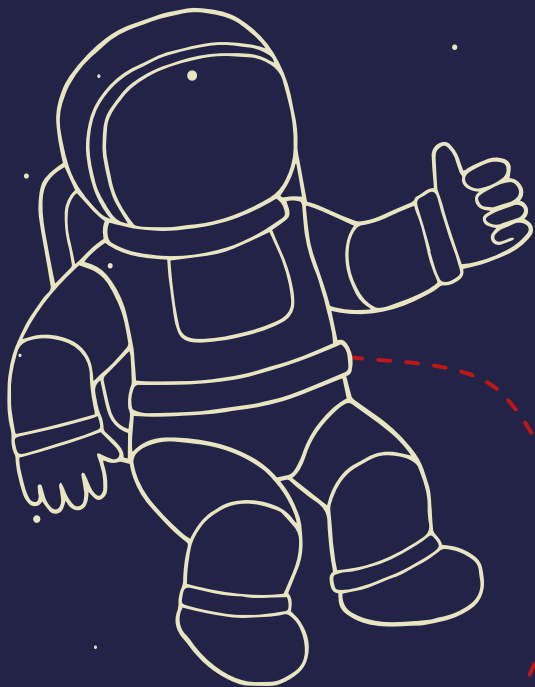
```
import json

with open("diccionario.json", "r") as archivo:
    diccionario=json.load(archivo)

archivo.close()

print("Diccionario: ",diccionario)

print(diccionario["3"])
```



# Programa académico CAMPUS



Ciclo 1:  
Fundamentos de  
Programación

