Ciclo 1
Fundamentos de programación con Python
Sesión 23: Persistencia de datos con Python 1

Programa Ciencias de la Computación e Inteligencia Artificial
Escuela de Ciencias Exactas e Ingeniería
Universidad Sergio Arboleda
Bogotá

Agenda

1. Persistencia de datos
2. Ventajas
3. Archivos de texto
4. Lectura de un archivo
5. Escritura de un archivo
6. Creación de un archivo
7. Lectura línea a línea de un archivo
8. Control de errores
9. Ejercicios propuestos

1. Persistencia de datos

- La persistencia de datos es la capacidad de un equipo para mantener la información incluso después de apagar el computador o cerrar un programa que utiliza la información.
- Persistencia de los datos en esencia conserva el estado de la información en un momento en que usted elija, lo guarda, por ejemplo, cuando hace clic en un botón "Guardar" o copiar archivos en un dispositivo de almacenamiento no volátil.
- Una de las maneras más simples de mantener sus datos es a través de archivos de texto. Otra alternativa es usar Bases de Datos.

2. Ventajas

- La mayoría de los programas de computador usan la memoria RAM para almacenar temporalmente la información que utiliza el programa, tales como: datos de la función o referencias a variables. Al cerrar el programa, el equipo se borran todos los datos en la RAM. Al persistir los datos sobre archivos de texto, permite que los datos con los que se trabajan no se pierdan.
- Los datos persistentes pueden ser recuperados, modificados y almacenados nuevamente, conservando los cambios realizados de manera permanente.

3. Archivos de Textos

- Un archivo de texto contiene una sucesión de caracteres que se puede considerar organizada en una secuencia de líneas.
- Los programas Python, por ejemplo, suelen residir en archivos de texto. Es posible generar, leer y modificar archivos de texto con editores de texto o con los propios programas.
- Desde el punto de vista de la programación un archivo de texto es un objeto en el que se puede leer y/o escribir información. Esto obliga a que siempre debamos seguir un protocolo establecido de tres pasos:

1. Abrir el archivo de texto
2. Leer o escribir información sobre el archivo de texto
3. Cerrar el archivo de texto

➢ Pasos para trabajar con archivos de texto

1. Abrir el archivo indicado. Las posibles operaciones son:

    - Lectura
    - Escritura
    - Adición

2. Leer o escribir la información
3. Cerra el archivo

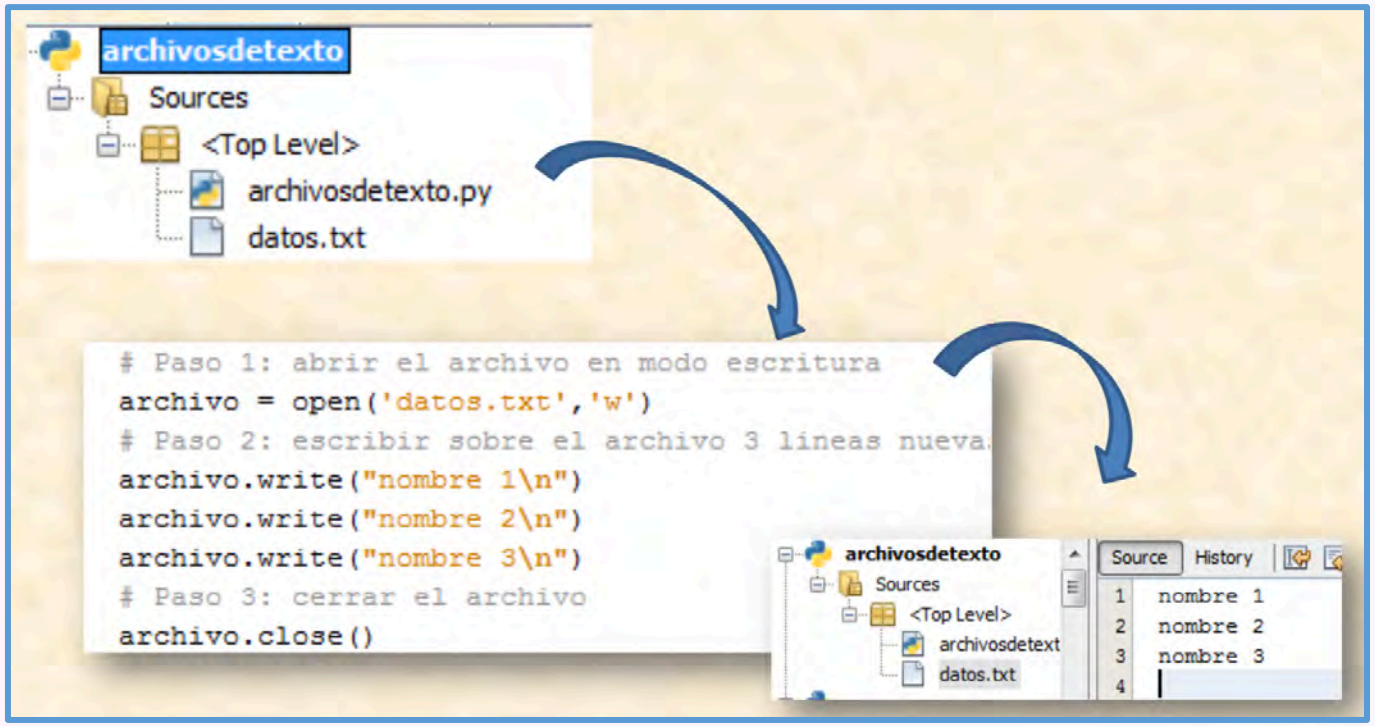
4. Lectura de un archivo

Los archivos se organizan en directorios y subdirectorios. Todo programa en ejecución tiene un "directorio actual". Existen 2 tipos de paths: relativos y absolutos.

- **Ejemplo;** lectura de líneas de un archivo de texto. (archivo= open('datos.txt', r))

5. Escritura de un archivo

- **Ejemplo:** escritura de líneas de un archivo de texto. (archivo= open('datos.txt', w))



6. Creación de un archivo

- Crear un archivo y agregar datos:

```
# Paso 1: abrir el archivo en modo escritura
archivo = open('datos.txt','w')
# Paso 3: cerrar el archivo
archivo.close()
```

```
# Paso 1: abrir el archivo en modo escritura
archivo = open('datos.txt','a')
# Paso 2: escribir sobre el archivo 3 lineas nuevas
archivo.write("nombre 4\n")
archivo.write("nombre 5\n")
archivo.write("nombre 6\n")
# Paso 3: cerrar el archivo
archivo.close()
```

7. Lectura línea a línea de un archivo de texto

La clase file tiene el método readline() que retorna toda una línea del archivo de texto y deja posicionado el puntero de archivo en la siguiente línea.
Cuando llega al final del archivo readline retorna un string vacío.

```
1  # Paso 1: abrir el archivo en modo escritura
2  archivo = open('datos.txt','r')
3  linea= archivo.readline()
4  while linea!="":
5      print linea
6      linea=archivo.readline()
7  archivo.close()
8
```

```
Output - archivosdetexto
nombre 1

nombre 2

nombre 3

nombre 4

nombre 5

nombre 6
```

7. Lectura línea a línea de un archivo de texto

- Lectura de las líneas de un texto

Se puede leer todo el contenido de un archivo de texto y almacenarlo en una lista (esto tiene sentido si el archivo de texto no es muy grande). Usamos el método readlines() en lugar de readline()

```
# Paso 1: abrir el archivo en modo escritura
archivo = open('datos.txt','r')
lineas= archivo.readlines()
print lineas
archivo.close()
```

```
['raul\n', 'pedro\n', 'juan\n']
```

8. Control de errores

Si trata de abrir en modo lectura un archivo inexistente, se obtiene un error y la ejecución del programa se aborta.

- Dos posibilidades:

1. Preguntar si el archivo existe
2. Capturar la excepción de la apertura

8. Control de errores

- Eliminar los saltos de línea:

```
archivo = open('datos.txt', 'r')
for linea in  archivo:
    if linea[-1] == '\n':
        linea = linea[:-1]
    print linea
archivo.close()
```

9. Ejercicios propuestos

- Desarrollar un programa que calcule el número de líneas de un archivo de texto.
- Diseña un programa que cuente el número de caracteres de un archivo de texto, incluyendo los saltos de línea.
- Diseñe una función que, dada una palabra y un nombre de archivo, diga si la palabra aparece o no en el archivo.
- Diseñe una función que, dado un nombre de archivo, muestre cada una de sus líneas precedida por su numero de línea.

El futuro digital es de todos

MinTIC

Preguntas

UNIVERSIDAD SERGIO ARBOLEDA

MisiónTIC2022