Archivos

...

Programación
2024

Almacenamiento de datos

- Las **variables** y **datos** utilizados en un programa se almacenan en la **memoria RAM**.
- El acceso a estos datos es muy rápido, lo cual favorece la eficiencia del programa.
- Almacenar en memoria tiene dos limitaciones:
    - Tamaño de la memoria RAM.
    - Temporalidad.

Flujo de datos

- Durante su ejecución, un programa puede **recibir** y **devolver** datos del usuario o del sistema.
- Este movimiento se conoce como **flujo de datos**.
- Hasta el momento, la recepción y devolución de datos fue, principalmente, por consola.
- Estos flujos se conocen como **entrada** y **salida estándar** (STDIN y STDOUT).

Archivos

- Un **archivo** o **fichero** es una colección de datos almacenada en la memoria secundaria (por ej: disco).
- Una vez escritos, los datos almacenados en un archivo **no se pierden** como sucede en la memoria RAM.
- Por lo general, se busca que los datos en un archivo deben estar relacionados entre sí y organizados (por ej, datos en filas, cada columna es un campo distinto).

Tipos de archivos

- En programación, se suele trabajar con dos tipos de archivos:
  - Archivos de texto:
    - Almacenan caracteres ASCII.
    - Son legibles por las personas
  - Archivos binarios:
    - Almacenan bytes.
    - Solo son interpretables por computadoras.
    - Se utilizan para almacenar programas, imágenes, audios, ...

http://www.linfo.org/binary_file.html

Tipos de archivos

Archivo de texto

Muestra1	NC_003317.1	99	1.94
Muestra1	NC_003317.1	199	1.98
Muestra1	NC_003317.1	299	1.98
Muestra1	NC_003317.1	399	1.98
Muestra1	NC_003317.1	499	1.98
Muestra1	NC_003317.1	599	1.98
Muestra1	NC_003317.1	699	1.98
Muestra1	NC_003317.1	799	1.98
Muestra1	NC_003317.1	899	1.98
Muestra1	NC_003317.1	999	1.98

Archivo binario

Operaciones con archivos

- Creación.
- Consulta.
- Actualización (altas, bajas, modificación)
- Destrucción (borrado).
- Fusión o combinación.

Abriendo archivos en Python

- La función ***open*** permite abrir archivos.
- El primer parámetro es el nombre de un archivo (como texto o almacenado en una variable).
- Después viene el modo en que se abrirá el archivo y el tipo de archivo, de ser necesario.

'r': read (lectura) *

'w': write (escritura)

'a': append (agregar)

'x': create (crear)

't': texto *

'b': binario

Abriendo archivos en Python

- Existen dos formas de abrir un archivo:

```
archivo = open("archivo.txt","r")
…
archivo.close()
```

```
with open("archivo.txt","r") as archivo:
    …
```

Acceso secuencial

- En un archivo de texto, los datos se escriben y acceden de forma **consecutiva**.
- Para acceder al registro n es necesario pasar por los n-1 registros anteriores.
- Al final, el sistema añade una marca que indica que no hay más datos (End Of File o EOF)

Figura 9.6. Organización secuencial.

Leyendo archivos en Python

- Hay 3 alternativas para leer un archivo:
  - La función **readline**, que lee solo una línea
  - La función **readlines**, que lee todas las líneas del archivo.
  - Recorrer la variable “archivo” con un bucle **for** (solo carga una línea en memoria)
- No importa el contenido de la línea, siempre se va a guardar como un **string**.

Abriendo archivos en Python

```
l = archivo.readline() # lee una linea

todas_las_lineas = archivo.readlines()



for l in archivo:
    # lee cada línea en archivo y la guarda en “l”
```

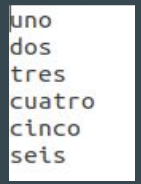
Leyendo archivos en Python

- Cuando se abre un archivo, el sistema crea una variable del tipo **puntero** que indica que línea se está leyendo.
- Cada vez que se lee una línea (con cualquiera de las alternativas), es imposible volver atrás.
- Si se usan la función **readlines** o cuando finaliza el bucle **for**, se alcanza el final del archivo (puntero apunta a EOF)

Leyendo archivos en Python I

- Que se obtiene al ejecutar estas líneas con el siguiente archivo?

```
with open("lista.txt", "r") as archivo:
    l = archivo.readline()
    print(l)
    l = archivo.readline()
    print(l)
```



Leyendo archivos en Python I

- Que se obtiene al ejecutar estas líneas con el siguiente archivo?

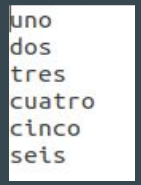
```
with open("lista.txt", "r") as archivo:
    l = archivo.readline()
    print(l)
    l = archivo.readline()
    print(l)
```

```
(base) matias@rfgenom002:~$ python test.py
uno

dos
```

Leyendo archivos en Python II

- Que se obtiene al ejecutar estas líneas con el mismo archivo?

```
with open("lista.txt", "r") as archivo:
    l = archivo.readline()
    print(l)
    archivo.readlines()
    l = archivo.readline()
    print(l)
```



Leyendo archivos en Python II

- Que se obtiene al ejecutar estas líneas con el mismo archivo?

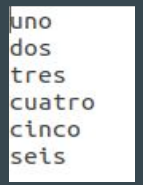
```
with open("lista.txt", "r") as archivo:
    l = archivo.readline()
    print(l)
    archivo.readlines()
    l = archivo.readline()
    print(l)
```

```
(base) matias@rfgenom002:~$ python test.py
uno
```

Leyendo archivos en Python III

- Que se obtiene al ejecutar estas líneas con el mismo archivo?

```
with open("lista.txt", "r") as archivo:
    for l in archivo:
        print(l)
```



Leyendo archivos en Python III

- Que se obtiene al ejecutar estas líneas con el mismo archivo?

```
with open("lista.txt", "r") as archivo:
    for l in archivo:
        print(l)
```

```
(base) matias@rfgenom002:~$ python test.py
uno

dos

tres

cuatro

cinco

seis
```

Lidiando con fines de línea

```
with open("lista.txt", "r") as archivo:
    for l in archivo:
        print(l)

with open("lista.txt", "r") as archivo:
   lineas = archivo.readlines()
   print(lineas)
```

```
(base) matias@rfgenom002:~$ python test.py
uno

dos

tres

cuatro

cinco

seis

['uno\n', 'dos\n', 'tres\n', 'cuatro\n', 'cinco\n', 'seis\n']
```

Lidiando con fines de línea

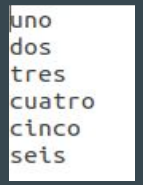
```
with open("lista.txt", "r") as archivo:
    for l in archivo:
        l = l.rstrip()
        print(l)
```

```
(base) matias@rfgenom002:~$ python test.py
uno
dos
tres
cuatro
cinco
seis
```

Combinando recorridos

- Que se obtiene al ejecutar estas líneas con el mismo archivo?

```
with open("lista.txt", "r") as archivo:
    for l in archivo:
        l1 = archivo.readline()
        l2 = archivo.readline()
        l1 = l1.rstrip()
        l2 = l2.rstrip()
        print(l1, l2)
```



Combinando recorridos

- Que se obtiene al ejecutar estas líneas con el mismo archivo?

```
with open("lista.txt", "r") as archivo:
    for l in archivo:
        l1 = archivo.readline()
        l2 = archivo.readline()
        l1 = l1.rstrip()
        l2 = l2.rstrip()
        print(l1, l2)
```

```
(base) matias@rfgenom002:~$ python test.py
dos tres
cinco seis
```

Escribiendo en un archivo

- Para escribir un archivo, primero es necesario abrirlo en modo **escritura** o **anexar**.
- La función **write** permite guardar en el archivo un **texto** o un **string** (siempre strings!!).
- La función write solo admite **un parámetro**: si se quieren escribir muchas cosas es necesario armar un solo string o llamar muchas veces a la función.

```
with open("archivo.txt","w") as archivo:
    archivo.write("Programacion.\n")

    var = "Biologia."
    archivo.write(var)
```

Valores separados por comas (CSV)

- Un formato de archivo de texto muy utilizado para almacenar datos el CSV.
- Estos archivos representan **tablas**, donde cada **columna** está **delimitada por un carácter** (por ej.: comas).
- Para leer un archivo en este formato es necesario leer cada línea y **separarla según el carácter delimitador**.

Valores separados por comas (CSV)

```
with open("archivo.csv","r") as archivo:
   for l in archivo:
       l = l.rstrip()
       valores = l.split(",")
    …
```

Ejemplo 1

- El recuento de células somáticas en leche permite determinar si la salud de una vaca: si el recuento es alto, es probable que el animal está sufriendo una enfermedad. A partir de la información del siguiente archivo, encuentre el animal que mayor incremento tuvo de células somáticas entre análisis y cual es su raza.

```
Caravana,Raza,CS_Anterior,CS
6441,HO,160,43
5995,XB,82,236
6084,HO,136,19
5646,HO,27,227
6270,HO,112,36
6409,XB,44,174
6230,XB,63,33
6239,HO,27,16
```

Ejemplo 1: pseudocódigo

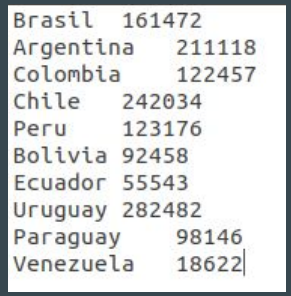
1. Inicializar mayor en -inf, animal y raza como “”
2. Saltear encabezado
3. Leer el archivo, línea por línea.
4. Separar la línea por comas.
5. Restar la columna 4 menos la columna 3.
6. Si la resta es mayor al valor máximo guardado:
    a. animal = columna 1
    b. raza = columna 2
    c. mayor = resta.
7. Devolver animal, raza.

Ejemplo 1: código

```
mayor = float("-inf")
animal = ""
raza = ""
with open("celulas.txt", "r") as archivo:
    archivo.readline()
    for linea in archivo:
        linea = linea.rstrip()
        valores = linea.split(",")
        if (int(valores[3]) - int(valores[2])) > mayor:
            mayor = int(valores[3]) - int(valores[2])
            animal = valores[0]
            raza = valores[1]
print ("El animal con mayor aumento de CS es: ", animal)
print ("La raza del animal es: ", raza)
```

Ejemplo 2

- “casos.tsv” es un archivo separado por tabulaciones que presenta una lista de casos de COVID en distintos países, cada millón de habitantes. Construya un gráfico de barras a partir de los datos brindados.



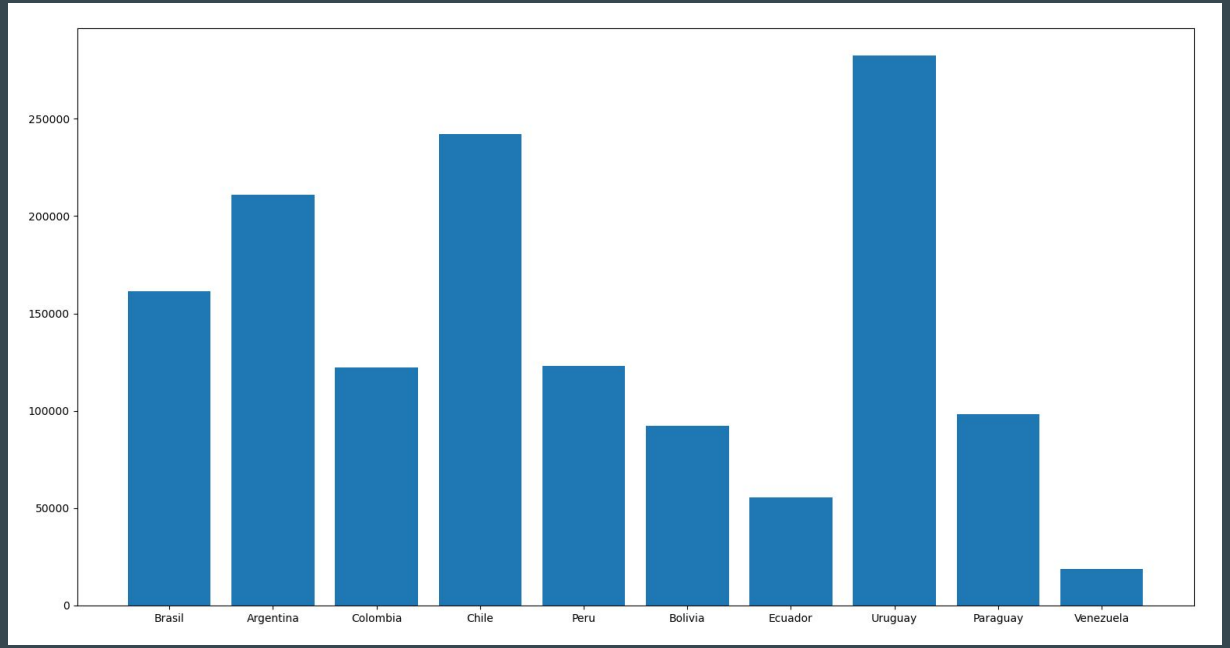
Ejemplo 2: pseudocódigo

1. Crear listas vacías para países y cantidad de casos
2. Cargar archivo y leer por líneas.
3. Separar linea por tabulación.
4. Agregar primer campo a la lista de países.
5. Agregar segundo campo a la lista de cantidades.
6. Graficar.

Ejemplo 2: código

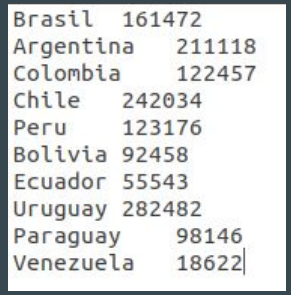
```
import matplotlib.pyplot as plt

paises = []
casos = []
with open("casos.tsv", "r") as archivo:
  for linea in archivo:
    linea = linea.rstrip()
    campos = linea.split("\t")
    paises.append(campos[0])
    casos.append(int(campos[1]))

plt.bar(paises, casos)
plt.show()
```

Ejemplo 2: resultados



Ejemplo 3

- A partir del archivo del ejemplo anterior, guardar en un archivo nuevo el promedio de casos en todos los países listados.



Ejemplo 3: pseudocódigo

1. Inicializar las variables suma y cantidad en 0.
2. Cargar archivo y leer por líneas.
3. Separar línea por tabulación.
4. Sumar la segunda columna a suma y sumar 1 a cantidad.
5. Calcular suma/cantidad.
6. Abrir archivo de salida.
7. Escribir promedio en archivo de salida.

Ejemplo 3: código

```
suma = 0
cantidad = 0
with open("casos.tsv", "r") as archivo:
  for linea in archivo:
    linea = linea.rstrip()
    campos = linea.split("\t")
    suma += int(campos[1])
    cantidad += 1

prom = suma/cantidad
with open("promedio.txt", "w") as salida:
    salida.write(str(prom)) # write solo acepta strings!!
```