



Clase 17 – Archivos

IIC1103 Sección 9 – 2019-2

Profesor: Felipe López

26-11-2019

Contenidos

1. Leer archivos
2. Abrir archivos
3. Escribir archivos
4. Ejercicios

- Ahora ya manejamos todas las herramientas necesarias para poder almacenar y trabajar con datos dentro de Python.

- Ahora ya manejamos todas las herramientas necesarias para poder almacenar y trabajar con datos dentro de Python.
- Sin embargo, surge la siguiente duda ¿Cómo podríamos cargar estos datos?

- Ahora ya manejamos todas las herramientas necesarias para poder almacenar y trabajar con datos dentro de Python.
- Sin embargo, surge la siguiente duda ¿Cómo podríamos cargar estos datos?
- Y después de modificarlos o procesarlos ¿Cómo podríamos guardar esta información?

Interacción con archivos

- Python puede interactuar con varios tipos de archivos.
- En este curso aprenderemos cómo interactuar con archivos de texto plano.
- Estos archivos tienen un contenido simple y fácil de interpretar. La ventaja es que su contenido se puede ver y editar con cualquier programa editor de texto, lo que permite una rápida creación y edición.
- Python posee herramientas para poder leer estos archivos, así como también escribir sobre ellos (o crear nuevos).

¿Para qué sirve trabajar con Archivos?

- Trabajar con archivos tiene muchísimas ventajas.
- La más clara es poder incorporar grandes volúmenes de información a Python.
 - Hemos visto como añadir información de forma manual a Python:
 - Creando una o más variables de forma explícita (por ejemplo, `a="texto"`)
 - Creando una o más listas de forma explícita (por ejemplo, `lista1=["a","b","c"]`)
 - Creando una o más listas de listas de forma explícita (por ejemplo, `matriz = [["a","b","c"],["d","e","f"],["g","h","i"]]`)
 - También aprendimos como permitir que el usuario ingresara información mediante el comando `input()`.
 - **Pero ninguno de los métodos anteriores permite cargar mucha información de forma rápida.**

¿Para qué sirve trabajar con Archivos?

Al leer archivos, podemos ingresar mucha información de forma muy rápida y eficiente.

Leer Archivos

- Para poder leer archivos en Python, ocupamos el comando `open()`.
- Esta es una función que posee Python que permite abrir un archivo (recordemos que trabajaremos solo con archivos de texto plano).

Leer Archivos

- Veamos un ejemplo de lectura de archivos:

```
informacion_archivo = open("ejemplo1.txt")  
print(informacion_archivo)
```

```
<_io.TextIOWrapper name='ejemplo1.txt' mode='r'  
encoding='UTF-8'>
```

- En este caso, el archivo está en el mismo directorio que el archivo Python.

Leer Archivos

- ¿Y si no estuviera en el mismo directorio?

```
informacion_archivo = open("Ejemplo/Ejemplo1/Exageracion_ejem-  
plo1/ejemplo2.txt")  
print(informacion_archivo)
```

```
<_io.TextIOWrapper name='ejemplo2.txt' mode='r'  
encoding='UTF-8'>
```

- En este caso, el archivo está en otro directorio.

read()

- Para poder acceder al contenido de un archivo, ocupamos la función `read()`:

```
informacion_archivo = open("ejemplo1.txt")  
contenido = informacion_archivo.read()  
print(contenido)
```

```
¡Hola!  
Bienvenidos  
a  
IIC1103  
2018-2
```

- Esta función lee todo el contenido de un archivo y lo agrega a una variable.

Leer Archivos

- Para poder acceder al contenido de un archivo, debemos hacer lo siguiente:

```
informacion_archivo = open("ejemplo1.txt")  
print(informacion_archivo.read())
```

CanciÃ³n
BorrÃ³n
LÃ³pez

¿Qué pasó?

Encoding

- Por defecto, el encoding de lectura es ASCII
- Muchísimos archivos están codificados en UTF-8 (Unicode)
 - Recomendable para evitar problemas con tildes, caracteres especiales, etc.
- Para solucionarlo, debemos especificar el encoding al abrir.

```
informacion_archivo = open("ejemplo1.txt",encoding="UTF-8")  
print(informacion_archivo.read())
```

Canción
Borró
López

readlines()

- En realidad, la mejor forma de leer un archivo es de la siguiente manera:

```
informacion_archivo = open("ejemplo1.txt")  
  
lineas = informacion_archivo.readlines()  
print(lineas)  
  
for linea in lineas:  
    print(linea)
```

```
['¡Hola!\n', 'Bienvenidos\n', 'a\n', 'IIC1103\n', '2018-2']  
¡Hola!  
  
Bienvenidos  
  
a  
  
IIC1103  
  
2018-2
```

strip()

- Del ejemplo anterior, pueden notar que los elementos de la lista “lineas” tienen todos un salto de línea (“\n”).
- Esto puede no ser conveniente. Por ejemplo, cuando imprimimos los elementos de “lineas”, se imprime una línea extra entre cada elemento.
- Por ello, es recomendado ocupar strip() sobre los elementos de la lista entregada por readlines(), para evitar problemas como estos.

readlines()

- Veamos el ejemplo anterior con strip()

```
informacion_archivo = open("ejemplo1.txt")

lineas = informacion_archivo.readlines()
lineas2 = []
print(lineas)

for linea in lineas:
    linea = linea.strip()
    lineas2.append(linea)
    print(linea)
print(lineas2)
```

```
['¡Hola!\n', 'Bienvenidos\n', 'a\n', 'IIC1103\n', '2018-2']
¡Hola!
Bienvenidos
a
IIC1103
2018-2
['¡Hola!', 'Bienvenidos', 'a', 'IIC1103', '2018-2']
```

Leer Archivos

- Trabajar con archivos de la forma que mostramos anteriormente es muy útil. Esto porque:
 - Como el archivo leído se transforma en una lista cuyos elementos son las líneas del archivo, esto nos permite saber cuántas líneas tiene el archivo.
 - Además, nos permite hacer un `for` sobre la lista de líneas del archivo, para poder acceder a cada una de ellas y trabajar de forma independiente con cada línea del archivo.

¿Para qué sirve trabajar con Archivos?

- Además de leer archivos, podemos crear o escribir archivos ya existentes.
- Sirve para poder guardar el procesamiento que hacemos sobre un archivo que ya leímos.
- También sirve para poder guardar lo que pasó en la ejecución de un código. Esto dado que cuando la ejecución de un código se termina, todo lo que ocurrió se pierde. Si uno quisiera guardar por ejemplo el valor de una variable para poder tener acceso a este la próxima vez que un código se ejecuta, se podría hacer escribiendo archivos.

open()

- Lo primero que debemos saber antes de poder escribir archivos es entender a cabalidad el comando `open()`.
- Lo que hace este comando es abrir un archivo para poder trabajar con él. Al hacerlo, tiene tres modos para hacerlo:
 - `r`: Viene de “read”. Este modo sirve para poder leer un archivo.
 - `w`: Viene de “write”. Este modo sirve para poder escribir en un archivo.

IMPORTANTE: Al elegir el modo “w” al abrir un archivo, se borra el contenido que este tenía.

- `a`: Viene de “append”. Este modos sirve para poder agregar contenido a un archivo ya existente.

open()

- Por lo tanto, cuando ocupamos el comando open() en los ejemplos anteriores, en realidad estábamos ocupando el modo “r”. De hecho, este modo viene seleccionado por defecto. Veamos el ejemplo anterior ocupando explícitamente el modo “r”.

```
informacion_archivo = open("ejemplo1.txt", "r")  
  
lineas = informacion_archivo.readlines()  
print(lineas)  
  
for linea in lineas:  
    print(linea)
```

```
['¡Hola!\n', 'Bienvenidos\n', 'a\n', 'IIC1103\n', '2018-2']  
¡Hola!  
  
Bienvenidos  
  
a  
  
IIC1103  
  
2018-2
```

write()

- Para poder escribir en archivos que ya tienen un contenido, ocupamos la función `write()`.

```
informacion_archivo = open("ejemplo1.txt", "a")  
informacion_archivo.write("En este curso \n aprenderán a escribir en Python")
```

ejemplo1.txt saved

```
1 ¡Hola!  
2 Bienvenidos  
3 a  
4 IIC1103  
5 2018-2|
```

Archivo de texto antes de la ejecución del código

¿Qué pasó?

ejemplo1.txt saved

```
1 ¡Hola!  
2 Bienvenidos  
3 a  
4 IIC1103  
5 2018-2|
```

Archivo de texto después de la ejecución del código

close()

- Es importante incorporar el uso de una nueva función al ocupar el comando `open()`. Esta función es `close()`, y lo que hace es terminar el uso de un archivo en la ejecución de nuestro código. Veamos un ejemplo:

```
informacion_archivo = open("ejemplo1.txt", "a")
informacion_archivo.write("En este curso \n aprenderán a escribir en Python")
informacion_archivo.close()
```

ejemplo1.txt saved

```
1 ¡Hola!
2 Bienvenidos
3 a
4 IIC1103
5 2018-2|
```

Archivo de texto antes de la ejecución del código

ejemplo1.txt saved

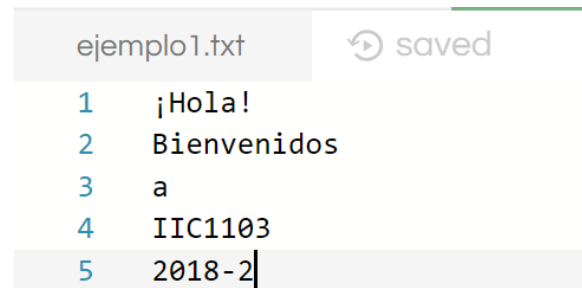
```
1 ¡Hola!
2 Bienvenidos
3 a
4 IIC1103
5 2018-2En este curso
6 |aprenderán a escribir en Python|
```

Archivo de texto después de la ejecución del código

print()

- También podemos escribir mediante la función `print()`

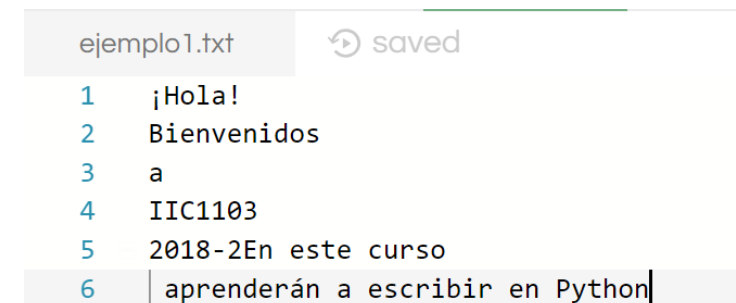
```
informacion_archivo = open("ejemplo1.txt","a")
print("En este curso \n aprenderán a escribir en Python",file=informacion_archivo)
informacion_archivo.close()
```



ejemplo1.txt saved

```
1 ¡Hola!
2 Bienvenidos
3 a
4 IIC1103
5 2018-2
```

Archivo de texto antes de la ejecución del código



ejemplo1.txt saved

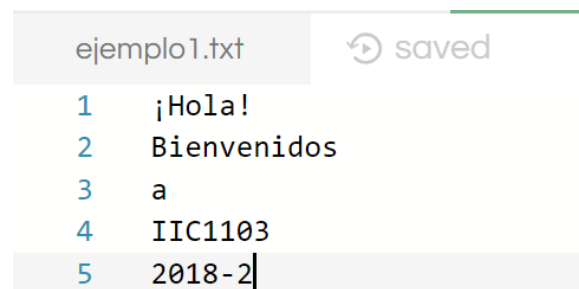
```
1 ¡Hola!
2 Bienvenidos
3 a
4 IIC1103
5 2018-2En este curso
6 aprenderán a escribir en Python
```

Archivo de texto después de la ejecución del código

write() en modo “w”

- Como vimos anteriormente, se puede ocupar el modo “w”. Recordemos que este modo borra el contenido original del archivo. Veamos un ejemplo:

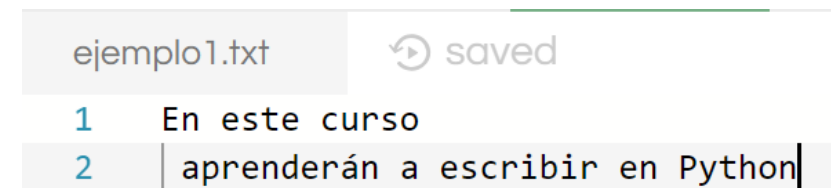
```
informacion_archivo = open("ejemplo1.txt", "w")
informacion_archivo.write("En este curso \n aprenderán a escribir en Python")
informacion_archivo.close()
```



ejemplo1.txt saved

```
1 ¡Hola!
2 Bienvenidos
3 a
4 IIC1103
5 2018-2|
```

Archivo de texto antes de la ejecución del código



ejemplo1.txt saved

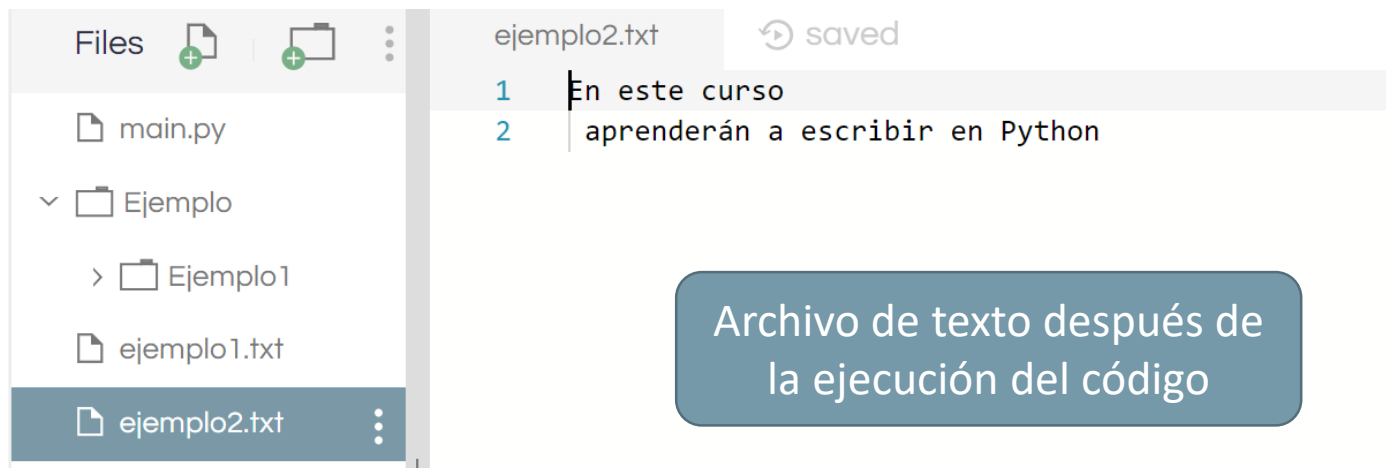
```
1 En este curso
2 aprenderán a escribir en Python|
```

Archivo de texto después de la ejecución del código

write() en modo “w”

- ¿Qué pasa si ocupamos el comando open en un archivo que no está creado? Veamos un ejemplo:

```
informacion_archivo = open("ejemplo2.txt","w")  
informacion_archivo.write("En este curso \n aprenderán a escribir en Python")  
informacion_archivo.close()
```



Archivo de texto después de
la ejecución del código

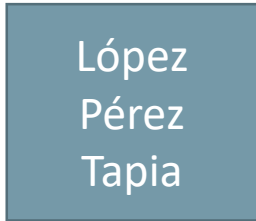
Ejercicio 1

- Hay dos archivos, de nombres “nombres.txt” y “apellidos.txt”. Cada uno contiene el primer nombre y primer apellido de ciertas personas respectivamente.
- Por ejemplo, si hay tres personas de nombres: Felipe López, Juan Pérez y María Tapia



Felipe
Juan
María

nombres.txt



López
Pérez
Tapia

apellidos.txt

Ejercicio 1

- Crea un código que junte el primer nombre y apellido de cada persona, y los guarde en un archivo de nombre “nombres_completos.txt” en el mismo orden que están almacenados en el archivo original.
- Importante: El código que crees debe servir para cualquier cantidad de nombres, y no solo para los tres del ejemplo.

Ejercicio 1

```
nombres = open("nombres.txt").readlines()
nombres2 = []

for i in nombres:
    nombres2.append(i.strip())

apellidos = open("apellidos.txt").readlines()
apellidos2 = []

for i in apellidos:
    apellidos2.append(i.strip())

print(nombres2)
print(apellidos2)

escribir = open("nombres_completos.txt", "w")
for i in range(len(nombres2)):
    escribir.write(nombres2[i]+" "+apellidos2[i)+"\n")
escribir.close()
```

Ejercicio 2

- Tenemos 5 archivos con las notas de la I1,I2,I3 y Ex de N personas.
- Estos archivos se denominan “ramo1.txt”, “ramo2.txt”, ... Puede asumir que están en el mismo directorio que el archivo Python.
- Puede asumir que estas personas solo han tenido estos 5 ramos.
- Puede asumir que cada persona aparece una vez en cada lista (no menos ni más)
- El formato de cada archivo es el siguiente:
Número alumno,Nota I1,Nota I2,Nota I3,Nota Ex
- Nos piden lo siguiente:
 1. Para cada archivo, calcular el promedio final del ramo (puede asumir que es el promedio simple entre las Ies y el Examen) por alumno e y guardarlo en un archivo de nombre “promedios.txt”.
 2. Calcular el PPA (promedio de los 5 ramos) por alumno y crear un nuevo archivo de nombre “PPA.txt” cuyas filas tengan el siguiente formato
Número alumno,PPA

Ejercicio 2

1. Para cada archivo, calcular el promedio final del ramo (puede asumir que es el promedio simple entre las les y el Examen) por alumno e y guardarlo en un archivo de nombre “promedios.txt”.

```

#Parte I
def calcular_prom(lista_de_notas, ramo):
    num_alumno = lista_de_notas.split(",")[0]
    notas = lista_de_notas.split(",")[1:5]
    #print(notas)

    suma = 0
    for i in notas:
        #aunque i tenga un salto de línea ("\n"), igualmente se puede transformar a float
        suma+=float(i)
    pf = suma/4

    return num_alumno+" "+ramo+" "+str(pf)

escribir2 = open("promedios.txt", "w")
ramos = ["ramo1.txt", "ramo2.txt", "ramo3.txt", "ramo4.txt", "ramo5.txt"]

for ramo in ramos:
    archivo = open(ramo)
    lineas = archivo.readlines()

    for linea in lineas:
        escribir2.write(calcular_prom(linea, ramo)+"\n")
escribir2.close()

```


Ejercicio 2

2. Calcular el PPA (promedio de los 5 ramos) por alumno y crear un nuevo archivo de nombre “PPA.txt” cuyas filas tengan el siguiente formato

Número alumno,PPA

```
promedios_read = open("promedios.txt").readlines()

promedios_por_ramo = []
for i in promedios_read:
    promedios_por_ramo.append(i.split(" "))

cant_alumnos = len(promedios_por_ramo)//5
promedios_por_alumno = []

for i_alumno in range(cant_alumnos):
    suma = 0
    lista_aux = []
    lista_aux.append(promedios_por_ramo[i_alumno][0])
    for i in range(5):
        lista_aux.append(float(promedios_por_ramo[i_alumno+cant_alumnos*i][2]))
    promedios_por_alumno.append(lista_aux)

archivo_final = open("PPA.txt", "w")

for alumno in promedios_por_alumno:
    ppa = (alumno[1]+alumno[2]+alumno[3]+alumno[4]+alumno[5])/5
    archivo_final.write(alumno[0]+", "+str(ppa)+"\n")

archivo_final.close()
```

Resumen de la clase

Para leer archivos:

- Abrir un archivo y asignarlo a una variable:
`variable_archivo = open(nombre_archivo).`
- Para poder leer un archivo ocupar la función `readlines()` en `variable_archivo`. Esta función devuelve una lista con todas las líneas del archivo leído.
`lineas_archivo = variable_archivo.readlines()`

Para escribir en archivos:

- Abrir un archivo y asignarlo a una variable. Puede ser en modo “a” para agregar información a un archivo que ya contiene información. O “w” para crear uno nuevo con contenido nuevo (este modo sobrescribe el contenido original del archivo):
`variable_archivo = open(nombre_archivo, (modo)).`
- Escribir usando la función `write()` en la variable que contiene al archivo.
`variable_archivo.write(texto).`
- Después de escribir el texto en el archivo, usar la función `close()` en la variable que almacena el archivo.
`variable_archivo.close()`

Bibliografía

- <http://runest.ing.puc.cl/files.html>
- A. B. Downey. Think Python: How to think like a computer scientist. Green Tea Press, 2013 -> Capítulo 14

Links

- <https://repl.it/@FelipeLopez/IIC1103Archivos> que contiene todos los ejemplos de la clase.



Clase 14 – Archivos

IIC1103 Sección 3 – 2018-2

Profesor: Felipe López

Fecha: 9 de octubre del 2018

26-11-2019