TEL101 - Iniciación a la Programación

Clase 7: Archivos

Prof. Nicolás Torres
nicolas.torresr@usm.cl
Ingeniería Civil Telemática
Departamento de Electrónica
Universidad Técnica Federico Santa María

Motivación

Todos los datos almacenados en variables se encuentran en memoria RAM, un sistema de almacenamiento a corto plazo. Los programas utilizan este tipo de memoria para guardar datos de forma temporal durante la ejecución. Sin embargo, toda la información se pierde cuando la ejecución termina y no es posible "recordar" los valores.

Cuando sea necesario almacenar datos de forma permanente en el tiempo, se deben colocar en dispositivos de almacenamiento persistentes como un Disco Duro (HDD), una Unidad de Estado Sólido (SSD), o una memoria USB (pendrive), entre otros.

Archivo

Un archivo es una secuencia de datos almacenados en un dispositivo persistente. Están identificados por un nombre y la ruta al directorio (carpeta) donde se ubica.

Los programas pueden leer datos desde archivos durante su ejecución, guardar datos en archivos para preservar información, e incluso editar el contenido de archivos para actualizar información.

Archivos de texto plano

Los **archivos de texto plano**, contienen únicamente caracteres y carecen de cualquier tipo de formato tipográfico (Arial, Times, Courier, negritas, subrayado, cursivas, tamaño, etc.).

Python permite manipular este tipo de archivos a través de operaciones de **lectura** y **escritura**.

Métodos avanzados de procesamiento de texto

Antes de poder trabajar con archivos de texto, se necesitan conocer métodos para manipular texto.

Método `replace`

El método `str.replace(a,b)` retorna una copia del string `str` con todas las ocurrencias del substring `a` reemplazadas por `b`.

```
In [1]: "cara".replace("a","o")
```

```
Out[1]: 'coro'
```

Método `split`

El método `str.split(s)` retorna una lista de los caracteres del string `str`, separados según el substring `s`. Si `s` no es proporcionado, el separador por defecto es cualquier espacio en blanco y caracteres especiales.

```
In [2]: " hola mundo\n ".split()
```

```
Out[2]: ['hola', 'mundo']
```

```
In [3]: "La ruta natural".split()
```

```
Out[3]: ['La', 'ruta', 'natural']
```

```
In [4]: "La ruta natural".split("a")
```

```
Out[4]: ['L', ' rut', ' n', 'tur', 'l']
```

```
In [5]: "1,2,3".split(",")
```

```
Out[5]: ['1', '2', '3']
```

Método `strip`

El método `str.strip(s)` retorna una copia del string `str` con los caracteres iniciales y finales que sean iguales al substring `s` eliminados. Si `s` no es proporcionado, por defecto se eliminan cualquier espacio en blanco y caracteres especiales del inicio y final.

```
In [6]: " hola mundo\n ".strip()
```

```
Out[6]: 'hola mundo'
```

```
In [7]: "abracadabra".strip("a")
```

```
Out[7]: 'bracadabr'
```

Método `join`

El método `str.join(iterable)` retorna un string compuesto por la concatenación de todos los strings en el `iterable`, separados por el string `str`. Todos los elementos del `iterable` deben ser strings; de lo contrario, ocurre una excepción.

```
In [8]: ".".join(['1', '2', '3', '4', '5'])
```

Out[8]: '1.2.3.4.5'

In [9]:
```
" ".join(['1', '2', '3', '4', '5'])
```

Out[9]: '1 2 3 4 5'

In [10]:
```
"->".join(['1', '2', '3', '4', '5'])
```

Out[10]: '1->2->3->4->5'

In [11]:
```
"".join(['1', '2', '3', '4', '5'])
```

Out[11]: '12345'

Método `format`

El método `str.format()` retorna una copia del string `str`, reemplazando cada campo con el valor del argumento correspondiente en formato texto. El string `str` puede contener caracteres literales o campos de reemplazo determinados por paréntesis curvos `{}`. Cada campo de reemplazo puede contener el índice numérico de un argumento posicional o el nombre. Si está vacío, se reemplaza en orden.

Este método permite trabajar con una plantilla que puede ser rellenada con información dinámica.

In [12]:
```
plantilla = "Soy {}, vivo en {} y tengo {} años."
```

In [13]:
```
plantilla.format('Wang', 'China', 5.5)
```

Out[13]: 'Soy Wang, vivo en China y tengo 5.5 años.'

In [14]:
```
plantilla.format('Vlad', 'Rusia', 600)
```

Out[14]: 'Soy Vlad, vivo en Rusia y tengo 600 años.'

In [15]:
```
"{1}{0}{2}{0}".format('a', 'v', 'c')
```

Out[15]: 'vaca'

`{0}`, `{1}`, ... se llaman **campos**, y el método `format` va rellenando los campos en orden.

In [16]:
```
x = "{nombre} vive en {pais}"
```

Los campos pueden definirse por nombres o identificadores, en lugar de números.

In [17]:
```
x.format(pais = 'Alemania', nombre = 'Johannes')
```

Out[17]: 'Johannes vive en Alemania'

F-strings

Las cadenas de texto formateadas (también llamadas *f-strings* para abreviar), permiten incluir el valor de expresiones dentro de una cadena al anteponer el prefijo `f` o `F`. Estas cadenas pueden contener campos de reemplazo determinados por paréntesis curvos `{}`.

In [18]:
```
a = 5
b = 10
print(f"{a} + {b} es {a + b}.")
```

```
5 + 10 es 15.
```

Apertura de archivos

La función `open()`, se usa más comúnmente con dos argumentos: `file` y `mode`.

- `file`: es un texto que contiene la ruta al directorio de trabajo actual donde se ubica el archivo que desea abrir. Si el código Python (`.py`) se encuentra en el mismo directorio que el archivo, solo es necesario el nombre.
- `mode`: es un carácter opcional que especifica el modo de acceso. El valor predeterminado es `'r'`, lo que significa que está abierto para lectura. Otros valores comunes son `'w'` para sobreescritura (vacía el contenido del archivo si ya existe, o en caso contrario, lo crea), y `'a'` para escritura al final del archivo (si no existe, lo crea).

Un tercer argumento opcional, `encoding`, establece la codificación que se utiliza para interpretar caracteres. Por ejemplo, `encoding='UTF-8'` permite reconocer cualquier carácter Unicode y gran cantidad de alfabetos.

La función retorna un tipo de dato `file`, que es una representación abstracta del archivo y no el contenido mismo.

Cierre de archivos

El método `f.close()`, propio del tipo de dato `file`, cierra el archivo `f` y libera cualquier recurso del sistema utilizado por él. Un archivo cerrado ya no se puede leer ni escribir hasta volver a abrirlo.

Lectura de archivos

Para leer el contenido de un archivo existente, se puede recorrer por medio de un ciclo `for` el valor que retorna la función `open()` al usar el argumento `mode` en `'r'`. Esto es eficiente en memoria, rápido y lleva a un código simple. La variable del ciclo siempre será una cadena de caracteres e irá cambiando de línea en cada iteración.

In [19]:
```
f = open('quijote.txt', 'r')
for linea in f:
    print(linea)
f.close()
```

```
En un lugar

de la Mancha

de cuyo nombre

no quiero acordarme

no ha mucho
```

In [20]:
```
f = open('quijote.txt', 'r')
for linea in f:
    print(linea.strip().upper())
f.close()
```

```
EN UN LUGAR
DE LA MANCHA
DE CUYO NOMBRE
NO QUIERO ACORDARME
NO HA MUCHO
```

Separadores

Los archivos CSV (del inglés *comma-separated values*), son archivos de texto plano utilizados para representar datos en forma de tabla, en los que las columnas se separan por comas (o punto y coma) y las filas por saltos de línea.

Para manipular archivos CSV, o cualquier otro documento donde sus líneas tengan campos de información separados por algún carácter, se utiliza el método `split` para separar los elementos en una lista.

In [21]:
```
suma = 0
f = open('productos.csv', 'r')
for linea in f:
    codigo,producto,precio = linea.strip().split(',')
    print(f"{producto}\t$ {precio}")
    suma += int(precio)
f.close()
print("-"*24)
print(f"TOTAL:\t\t$ {suma}")
```

```
Calcetines      $ 5000
Bicicleta       $ 100000
Pan de pascua   $ 900
Chocolate       $ 1000
Perfume Lancome $ 2000
------------------------
TOTAL:          $ 108900
```

Escritura de archivos

Para crear un archivo o sobreescribir información en uno ya existente, se debe abrir con la función `open()` en modo `'w'`.

Método `write`

El método `f.write()` recibe una cadena de texto y escribe los caracteres en el archivo `f`.

In [22]:
```
f = open('nuevo.txt', 'w')
f.write('En un lugar')
f.write('de la Mancha\n')
f.write('de cuyo nombre\nnoquiero acordarme')
f.close()
```

In [23]:
```
f = open('nuevo.txt', 'r')
```

```
for linea in f:
    print(linea)
f.close()
```

```
En un lugarde la Mancha

de cuyo nombre

noquiero acordarme
```