Introducción a la Programación

Prof. Agustín Gravano

Primer semestre de 2022

Clase teórica 13: Entrada/salida

Tengan abierto Spyder (o similar) para hacer algunas pruebas en Python.

Atención: Los temas nuevos presentados en esta clase **no** deben usarse en la resolución del TP1. Tampoco serán evaluados en el primer examen parcial, pero sí en el TP2.

Entrada/salida

Hasta ahora, las entradas (o valores iniciales) de nuestros programas estaban determinadas directamente en el código (*hardcodeadas*) y sus salidas (o resultados) se imprimían por pantalla con `print`.

Hoy veremos dos formas básicas en que un programa puede interactuar con el exterior:

- Mediante lectura y escritura de archivos de texto.
- Mediante entrada por teclado durante la ejecución.

Por supuesto, hay muchas otras formas: los programas pueden interactuar con otros programas en la misma computadora (a través del sistema operativo), o con programas en otras computadoras (a través de una red), o con el mundo físico mediante todo tipo de sensores, por mencionar algunas. Algunas de ellas serán vistas en *TD2: Sistemas de Computación* y en *TD4: Redes de Computadoras.*

Entrada/salida · Archivos

- Un archivo (en inglés, *file*) es una colección numerada y ordenada de bytes, que almacena datos procesables por un programa.
- Se almacena en un medio físico (p.ej., un disco rígido o un pendrive).
- Puede ser leído o escrito, en todo o en parte, por un programa en ejecución, el cual a su vez almacena sus datos en variables (p.ej., listas) en la memoria de la computadora.
- En el nombre del archivo suele indicarse, mediante la *extensión*, la forma en que se organiza el contenido del archivo.
- Según su contenido, existen dos tipos genéricos de archivos:
  - Archivos de texto, caracterizados por contener únicamente caracteres imprimibles (`.txt`, `.py`, `.html`, `.xml`, `.csv`, `.tex`, por ejemplo).
  - Archivos binarios, que organizan su contenido de distintas maneras (`.png`, `.wav`, `.mp4`, `.pdf`, `.xlsx`, `.docx`, por ejemplo).
- Por simplicidad, vamos a trabajar con archivos de texto, pero no hay mayores diferencias conceptuales con los archivos binarios.

Entrada/salida · Archivos de texto · Lectura

Con open abrimos un archivo, en modo lectura por defecto.

```
from typing import TextIO
f:TextIO = open('provincias.txt')
```

`f` es un file handler, que nos permite leer el contenido del archivo.

- **Nota:** En Python, para usar *type hints* de archivos, debemos importar `TextIO` de la biblioteca `typing`: `from typing import TextIO`.

Podemos leer todo el contenido del archivo en una variable de tipo `str`:

```
texto:str = f.read()
print(texto)
```

Ejemplo: ¿Cómo podríamos averiguar la cantidad de caracteres del archivo `provincias.txt`?

Entrada/salida · Archivos de texto · Lectura

También podemos iterar el archivo, leyendo línea a línea:

```
for linea in f:
    linea = linea.rstrip()          # Borra \n del final de la línea.
    print(linea)
```

Ejercicio: Escribir un programa que lea `provincias.txt` e imprima por pantalla únicamente el nombre de las provincias, uno por línea:

```
Buenos Aires
CABA
Catamarca
Chaco
...
```

Sugerencia: para separar los valores de cada línea, usar:

```
valores:List[str] = linea.split('___')
```

Entrada/salida · Archivos de texto · Escritura

Con `open(filename, 'w')` abrimos un archivo en modo escritura (`w` es por *write*). Si `filename` ya existía, se lo pisa (¡cuidado!); si no existía, se lo crea.

```
from typing import TextIO
f:TextIO = open('mundiales.txt', 'w')
```

Escribimos en el archivo con la instrucción `f.write('...')`.

```
mundiales:List[str] = ['Corea-Japón', 'Alemania',
                       'Sudáfrica', 'Brasil', 'Rusia']
for m in mundiales:
    f.write(m + "\n")       # Agregamos \n al final de la línea.
```

Al terminar de escribir, conviene cerrar el archivo para evitar problemas:

```
f.close()
```

Ejercicio: Escribir un programa que escriba los primeros 20 números naturales, uno por línea, en un archivo llamado `'20primeros.txt'`.

Entrada/salida · Entrada por teclado

Durante la ejecución del programa, Python permite tomar una entrada del usuario en forma interactiva, mediante el comando `input`.

```
nombre:str = input('Ingrese su nombre: ')
```

Esta instrucción hace lo siguiente:

1. Muestra por pantalla el texto pasado como argumento (en este ejemplo: `Ingrese su nombre:`).
2. Permite al usuario escribir un string por teclado, quien debe presionar la tecla Enter al terminar.
3. El string ingresado es el valor retornado por la función `input` (en este ejemplo, ese valor se almacena en el string `nombre`).

Ejemplo:

```
nombre:str = input('Ingrese su nombre: ')
print('¡Buenos días, ' + nombre + '!')
```

Ejercicio: Escribir un programa que elija al azar un número del 1 al 5 y permita al usuario adivinar hasta acertarle. Por ejemplo:

```
Adiviná en qué número del 1 al 5 estoy pensando: 3
Nop
Adiviná en qué número del 1 al 5 estoy pensando: 2
Nop
Adiviná en qué número del 1 al 5 estoy pensando: 5
Bien! Adivinaste en 3 intento/s.
```

(El usuario ingresó las respuestas 3, 2, 5. Los textos restantes fueron generados por el programa.)

Para elegir un número entero del 1 al 5 podemos usar la biblioteca `random` de esta manera:

```
import random
numero_secreto:int = random.randint(1,5)
```

Sintaxis del comando `input`:

```
respuesta:str = input('mensaje')
```

Repaso de la clase de hoy

- Manejo de archivos (`open`, `close`).
- Lectura de archivos de texto (`read`).
- Escritura de archivos de texto (`write`).
- Entrada de valores por teclado durante la ejecución (`input`).

Bibliografía complementaria:

- APPP2, capítulo 11.
- HTCSP3, capítulo 13.

Con lo visto, ya pueden resolver hasta el Ejercicio 6 (inclusive) de la Guía de Ejercicios 6.