Guía de Ejercicios 5: Entrada/salida

Objetivos:

- Ejercitar dos formas básicas en las que un programa puede interactuar con el mundo exterior: lectura/escritura de archivos de texto y entrada/salida interactiva con el usuario.

Lectura y escritura de archivos de texto

Cuidado: ¡Asegúrense de no sobreescribir archivos importantes!

Ejercicio 1.

(a) Escribir una función que tome como argumento el nombre de un archivo de texto y retorne un string con las longitudes de las líneas del archivo (sin contar el carácter especial de fin de línea), separadas por espacios. Por ejemplo, para un archivo con el siguiente contenido, debería devolver el string `'8 7 4 12 '`.

```
Figueroa\n
Alcorta\n
7350\n
Buenos Aires\n
```

(b) Escribir una función similar a la anterior, que reciba como segundo argumento el nombre de un archivo nuevo. Esta función, en lugar de devolver un string, debe escribir en el archivo nuevo las longitudes de las líneas del archivo original, una por línea.

Ejercicio 2. Escribir una función que, dado un entero `n`>0 y un string `filename`, escriba un archivo nuevo con nombre `filename`, que tenga los primeros `n` números primos, uno por línea.

Ejercicio 3. Escribir una función que reciba como argumentos dos nombres de archivos, `fuente` (el nombre de un archivo de Python existente), y `destino` (el nombre de un nuevo archivo de Python a crear). Esta función debe leer el archivo `fuente`, borrarle todos los comentarios que comienzan en `#`, y escribir el resultado en el archivo `destino`.

Por ejemplo, para el archivo de la izquierda, debe generarse el de la derecha:

```
# defino mis variables
i:int = 0       # índice
cant:int = 0    # contador
while i <= 100:
    # Si es primo, lo cuento
    if es_primo(i):
        cant = cant + 1
    i = i + 1
```

```

i:int = 0
cant:int = 0
while i <= 100:

    if es_primo(i):
        cant = cant + 1z
    i = i + 1
```

Sugerencia: Definir primero una función auxiliar que, dado un string `s`, devuelva el prefijo de `s` hasta el primer carácter `#`, no inclusive (si lo hay).

Testing: Antes de programar la función, diseñar pares de archivos `fuente`/`destino` de ejemplo para luego probar la función y poder verificar su correcto funcionamiento.

Interacción con el usuario

Ejercicio 4. Escribir programas que pregunten los argumentos necesarios al usuario en forma interactiva (con la función `input`) y realicen las siguientes operaciones. Reusar funciones definidas en guías anteriores cuando resulte conveniente. Suponer que el usuario siempre ingresará de manera correcta todos los argumentos.

(a) Dado un número entero `n` ≥ 0, imprimir por pantalla un cuadrado de asteriscos de lado `n`. Ejemplo de ejecución del programa (en rojo se indica el input del usuario):

```
Ingrese n: 5
*****
*****
*****
*****
*****
```

(b) Dado un string `s`, imprimir por pantalla la inversa de `s`. Ejemplo:

```
Ingrese un texto: universidad
dadisrevinu
```

Ejercicio 5. Escribir un único programa que realice las dos operaciones del Ejercicio 4. Primero debe preguntar el nombre de la operación a realizar (`'cuadrado'` o `'inversa'`) y después, los argumentos que correspondan para la operación correspondiente.

Ejercicio 6. Escribir un programa que elija al azar dos números enteros entre 10 y 20, permita al usuario ingresar el resultado del producto entre ambos números, y muestre un mensaje indicando si el resultado es correcto o incorrecto. Ejemplos:

```
Ingresar el resultado de 15*10: 150
Bien!
```

```
Ingresar el resultado de 18*16: 298
Mal! Resultado correcto: 288
```

Para elegir un número entero al azar entre `a` y `b` podemos usar la biblioteca `random`:

```
1 import random
2 numero:int = random.randint(a, b)
```