Guía de Ejercicios 6:
Entrada/salida. Representación de la información

Objetivos:

- Introducir y ejercitar algunas 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.
- Conocer y manejar las distintas formas de representar números enteros y reales en la computadora, para tener conciencia de sus limitaciones y sus implicancias.

Lectura y escritura de archivos de texto

Ejercicio 1.

(a) Escribir una función que tome como argumento el nombre de un archivo de texto y retorne una lista con la cantidad de caracteres contenidos en cada línea del archivo (sin contar el carácter especial de fin de línea).

(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 una lista, debe escribir en el archivo nuevo la cantidad de caracteres de cada línea del archivo original. (Cuidado: ¡No sobreescribir archivos importantes!)

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. (Cuidado: ¡No sobreescribir archivos importantes!)

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 imprimo!
    if es_primo(i):
        print(i)
    i = i + 1
```

```

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

    if es_primo(i):
        print(i)
    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).

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 $n \geq 0$, 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
```

(c) Dada una lista de enteros, determinar si está ordenada en forma estrictamente creciente. La lista debe ingresarse separando los enteros por comas. Ejemplos:

```
Ingrese una lista de enteros separados por comas: 1,3,7,10,1000
La lista ingresada está ordenada en forma estrictamente creciente.
```

```
Ingrese una lista de enteros separados por comas: 1,1,2
La lista ingresada no está ordenada en forma estrictamente creciente.
```

Ejercicio 5. Escribir un único programa que realice las tres operaciones del Ejercicio 4. Primero debe preguntar el nombre de la operación a realizar (`'cuadrado'`, `'inversa'` o `'creciente'`) 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
```

La elección de los números al azar puede hacerse con la función `randint` de la biblioteca `random`.

Representación de la información

Ejercicio 7. Conversión de decimal a binario y viceversa.

El siguiente algoritmo permite obtener la representación binaria de un número entero n:

> **entero_a_binario**($n \in \mathbb{Z}$) $\rightarrow$ string:
> *resultado* $\leftarrow$ string vacío
> Repetir mientras $n > 0$:
> Dividir a n por 2.
> Esto arroja un cociente c y un resto r, tales que $n = 2 \times c + r$.
> Agregar r (0 o 1) al principio del string *resultado*.
> $n \leftarrow c$
> Retornar el string almacenado en *resultado*.

El siguiente algoritmo permite obtener la representación binaria de un número fraccionario f, tal que $0 \leq f < 1$:

> **parte_fraccionaria_a_binario**($n \in \mathbb{Z}$) $\rightarrow$ string:
> *resultado* $\leftarrow$ string vacío
> Repetir mientras $f > 0$ (o hasta que `resultado` tenga la precisión deseada):
> Multiplicar a f por 2.
> Sean d la parte entera del resultado (0 o 1), y q la parte fraccionaria.
> Agregar d al final del string *resultado*.
> $f \leftarrow q$
> Retornar el string almacenado en *resultado*.

(a) Obtener a mano la representación binaria de los siguientes números decimales:

(I) 16

(II) 1234

(III) 0.5

(IV) 0.1

(V) 21.625

(b) Obtener a mano la representación decimal de los siguientes números binarios:

(I) `11111111`

(II) `0.11111111`

(III) `10101.101`

Ejercicio 8. Indicar el rango de representación de las siguientes codificaciones de enteros.
Por ejemplo, para "Signo más magnitud de 4 bits", el máximo valor representable es 7 (`0111`) y el mínimo es -7 (`1111`). Entonces, su rango de representación es: $-7 \ldots 7$.

(a) Sin signo de 8 bits.

(b) Signo más magnitud de 8 bits.

(c) Notación exceso 128 de 16 bits.

(d) Notación exceso 128 de 8 bits.

(e) Complemento a dos de 10 bits.

(f) Complemento a dos de 5 bits.

Ejercicio 9. Dada la secuencia de 10 bits $\boxed{1}\boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{0}\boxed{1}$, calcular el valor entero representado si se interpreta como:

(a) Notación sin signo

(b) Notación signo y magnitud

(c) Notación complemento a dos

(d) Notación exceso 128

Ejercicio 10. Obtener la secuencia de 10 bits que representa al número -300 según las siguientes codificaciones, de ser posible:

(a) Notación sin signo

(b) Notación signo y magnitud

(c) Notación complemento a dos

(d) Notación exceso 128

Ejercicio 11. Suponer un formato de **punto fijo** de 16 bits, con 1 bit para el signo, 8 bits para la parte entera y 7 bits para la parte fraccionaria.

(a) Calcular los máximos y mínimos valores representables.

(b) ¿Cuántas formas hay de representar el cero?

(c) Calcular el número representable más cercano a cero y distinto de cero.

(d) Determinar el rango en el cual ocurre error de *underflow*.

(e) Codificar el número 5.25.

(f) Codificar el número más cercano a $10/3$.

(g) Indicar qué número representa $\boxed{1}\boxed{0}\boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{1}\boxed{0}\boxed{1}\boxed{1}\boxed{0}\boxed{1}\boxed{0}\boxed{0}\boxed{0}\boxed{0}$.

Ejercicio 12. Suponer un formato de **punto flotante** de 16 bits, con 1 bit para el signo, 6 bits para el exponente (codificado en complemento a dos) y 9 bits para la fracción (interpretada como `1.xxxxxxxxx`).

(a) Calcular los máximos y mínimos valores representables.

(b) ¿Cuántas formas hay de representar el cero?

(c) Calcular el número representable más cercano a cero y distinto de cero.

(d) Determinar el rango en el cual ocurre error de *underflow*.

(e) Codificar el número -0.015625.

(f) Codificar el número más cercano a $10/3$.

(g) Indicar qué número representa $\boxed{1}\boxed{0}\boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{1}\boxed{0}\boxed{1}\boxed{1}\boxed{0}\boxed{1}\boxed{0}\boxed{0}\boxed{0}\boxed{0}$.