**Concepto de programación.**

Programación es la acción y efecto de programar. Este verbo tiene varios usos: se refiere a idear y ordenar las [acciones](https://definicion.de/accion) que se realizarán en el marco de un proyecto; al anuncio de las partes que componen un acto o espectáculo; a la preparación de máquinas para que cumplan con una cierta tarea en un momento determinado; a la elaboración de programas para la resolución de problemas mediante ordenadores; y a la preparación de los datos necesarios para obtener una solución de un problema a través de una calculadora electrónica.

**Tipos de programas.**

Programas de aplicación.

Programas utilitarios.

Programas de entretenimiento.

Programas de sistema.

**Procesamiento de datos.**

El procesamiento de datos es, en general, "la acumulación y manipulación de elementos de datos para producir información significativa."

El procesamiento de datos trata de un subconjunto del *procesamiento de la información*, el cambio (procesamiento) de la información de cualquier manera detectable por un observador. El procesamiento de datos es distinto del [*procesamiento de textos*](https://es.wikipedia.org/wiki/Procesamiento_de_textos), pues este último manipula textos nada más en lugar de los datos.

**Ciclo de procesamiento de datos.**

**.**

**Bit.**

Mientras que en el sistema de numeración decimal se usan diez dígitos (diez símbolos), en el binario se usan solo dos dígitos, el 0 y el 1. Un **bit** o dígito binario puede representar uno de esos dos valores: 0 o 1.

**Bytes.**

1 **byte** = 8 bits. 1 kilobyte (kB o kbyte) = 1024 **bytes**. 1 megabyte (MB o Mbytes) = 1024 kilobytes. 1 gigabyte (GB o Gbytes) = 1024 megabytes.

**Carácter.**

**Carácter** (tipo de dato) En terminología informática y de telecomunicaciones, un **carácter** es una unidad de información que corresponde aproximadamente con un grafema o con una unidad o símbolo parecido, como los de un alfabeto o silabario de la forma escrita de un lenguaje natural.

**Registro.**

Un registro, en [programación](https://es.wikipedia.org/wiki/Programaci%C3%B3n), es un [tipo de dato estructurado](https://es.wikipedia.org/wiki/Estructura_de_datos) formado por la unión de varios elementos bajo una misma estructura. Estos elementos pueden ser, o bien datos elementales (entero, real, carácter,...), o bien otras estructuras de datos. A cada uno de esos elementos se le llama [campo](https://es.wikipedia.org/wiki/Campo_(inform%C3%A1tica)).

Un registro se diferencia de un [vector](https://es.wikipedia.org/wiki/Vector_(programaci%C3%B3n)) en que éste es una colección de datos iguales, es decir, todos del mismo tipo, mientras que en una estructura los elementos que la componen, aunque podrían serlo, no tiene por qué ser del mismo tipo.

**Archivos.**

Todos los datos que un programa utiliza durante su ejecución se encuentran en sus variables, que están almacenadas en la [memoria RAM](http://es.wikipedia.org/wiki/Memoria_RAM) del computador.

La memoria RAM es un medio de almacenamiento **volátil**: cuando el programa termina, o cuando el computador se apaga, todos los datos se pierden para siempre.

Para que un programa pueda guardar datos de manera permanente, es necesario utilizar un medio de almacenamiento **persistente**, de los cuales el más importante es el [disco duro](http://es.wikipedia.org/wiki/Disco_duro).

Los datos en el disco duro están organizados en [archivos](http://es.wikipedia.org/wiki/Archivo_(inform%C3%A1tica)). Un **archivo** es una secuencia de datos almacenados en un medio persistente que están disponibles para ser utilizados por un programa. Todos los archivos tienen un nombre y una ubicación dentro del sistema de archivos del sistema operativo.

Los datos en un archivo siguen estando presentes después de que termina el programa que lo ha creado. Un programa puede guardar sus datos en archivos para usarlos en una ejecución futura, e incluso puede leer datos desde archivos creados por otros programas.

Un programa no puede manipular los datos de un archivo directamente. Para usar un archivo, un programa siempre abrir el archivo y asignarlo a una variable, que llamaremos el **archivo lógico**. Todas las operaciones sobre un archivo se realizan a través del archivo lógico.

Dependiendo del contenido, hay muchos tipos de archivos. Nosotros nos preocuparemos sólo de los **archivos de texto**, que son los que contienen texto, y pueden ser abiertos y modificados usando un editor de texto como el Bloc de Notas. Los archivos de texto generalmente tienen un nombre terminado en .txt.

**Lectura de archivos**

Para leer datos de un archivo, hay que abrirlo de la siguiente manera:

archivo = open(nombre)

nombre es un string que tiene el nombre del archivo. archivo es el archivo lógico a través del que se manipulará el archivo.

Si el archivo no existe, ocurrirá un **error de entrada y salida** (IOError).

Es importante recordar que la variable archivo es una representación abstracta del archivo, y no los contenidos del mismo.

La manera más simple de leer el contenido es hacerlo línea por línea. Para esto, basta con poner el archivo lógico en un ciclo for:

for linea in archivo:

# hacer algo

Una vez que los datos han sido leídos del archivo, hay que cerrarlo:

archivo.close()

Por ejemplo, supongamos que tenemos el archivo himno.txt que tiene el siguiente contenido:

Puro Chile

es tu cielo azulado

puras brisas

te cruzan también.

El archivo tiene cuatro líneas. Cada línea termina con un salto de línea (\n), que indica que a continuación comienza una línea nueva.

El siguiente programa imprime la primera letra de cada línea del himno:

archivo = open('himno.txt')

**for** linea **in** archivo:

**print** linea[0]

archivo.close()

El ciclo for es ejecutado cuatro veces, una por cada línea del archivo. La salida del programa es:

P

e

p

t

Otro ejemplo: el siguiente programa imprime cuántos símbolos hay en cada línea:

archivo = open('himno.txt')

**for** linea **in** archivo:

**print** len(linea)

archivo.close()

La salida es:

11

20

13

19

Note que el salto de línea (el “enter”) es considerado en la cuenta:

+---+---+---+---+---+---+---+---+---+---+---+

| P | u | r | o | | C | h | i | l | e | \n| = 11 símbolos

+---+---+---+---+---+---+---+---+---+---+---+

Para obtener el string sin el salto de línea, se puede usar el método strip, que elimina todos los símbolos de espaciado al principio y al final del string:

**>>>** s = ' Hola**\n**'

**>>>** s.strip()

'Hola'

Si modificamos el programa para eliminar el salto de línea:

archivo = open('himno.txt')

**for** linea **in** archivo:

**print** len(linea.strip())

archivo.close()

entonces la salida es:

10

19

12

18

Lo importante es comprender que los archivos son leídos línea por línea usando el ciclo for.

**Escritura en archivos**

Los ejemplos anteriores suponen que el archivo por leer existe, y está listo para ser abierto y leído. Ahora veremos cómo crear los archivos y cómo escribir datos en ellos, para que otro programa después pueda abrirlos y leerlos.

Uno puede crear un archivo vacío abriéndolo de la siguiente manera:

archivo = open(nombre, 'w')

El segundo parámetro de la función open indica el uso que se le dará al archivo. 'w' significa «escribir» (*write* en inglés).

Si el archivo señalado no existe, entonces será creado. Si ya existe, entonces será sobreescrito. Hay que tener cuidado entonces, pues esta operación elimina los datos del archivo que existía previamente.

Una vez abierto el archivo, uno puede escribir datos en él usando el método write:

a = open('prueba.txt', 'w')

a.write('Hola ')

a.write('mundo.')

a.close()

Una vez ejecutado este programa, el archivo prueba.txt será creado (o sobreescrito, si ya existía). Al abrirlo en el Bloc de Notas, veremos este contenido:

Hola mundo.

Para escribir varias líneas en el archivo, es necesario agregar explícitamente los saltos de línea en cada string que sea escrito. Por ejemplo, para crear el archivo himno.txt que usamos más arriba, podemos hacerlo así:

a = open('himno.txt', 'w')

a.write('Puro Chile**\n**')

a.write('es tu cielo azulado**\n**')

a.write('puras brisas**\n**')

a.write('te cruzan también.**\n**')

a.close()

Además del modo 'w' (*write*), también existe el modo 'a' (*append*), que permite escribir datos al final de un archivo existente. Por ejemplo, el siguiente programa abre el archivo prueba.txtque creamos más arriba, y agrega más texto al final de él:

a = open('prueba.txt', 'a')

a.write('**\n**')

a.write('Chao ')

a.write('pescao.')

a.close()

Si abrimos el archivo prueba.txt en el Bloc de Notas, veremos esto:

Hola mundo.

Chao pescao.

De haber abierto el archivo en modo 'w' en vez de 'a', el contenido anterior (la frase Holamundo) se habría borrado.

**Archivos de valores con separadores**

Una manera usual de almacenar datos con estructura de tabla en un archivo es la siguiente: cada línea del archivo representa una fila de la tabla, y los datos de una fila se ponen separados por algún símbolo especial.

Por ejemplo, supongamos que queremos guardar en un archivo los datos de esta tabla:

| **Nombre** | **Apellido** | **Nota 1** | **Nota 2** | **Nota 3** | **Nota 4** |
| --- | --- | --- | --- | --- | --- |
| Perico | Los Palotes | 90 | 75 | 38 | 65 |
| Yayita | Vinagre | 39 | 49 | 58 | 55 |
| Fulana | De Tal | 96 | 100 | 36 | 71 |

Si usamos el símbolo : como separador, el archivo, que llamaremos alumnos.txt, debería quedar así:

Perico:Los Palotes:90:75:38:65

Yayita:Vinagre:39:49:58:55

Fulanita:De Tal:96:100:36:71

El formato de estos archivos se suele llamar [CSV](http://en.wikipedia.org/wiki/CSV_(file_format)), que en inglés son las siglas de *comma-separated values* (significa «valores separados por comas», aunque técnicamente el separador puede ser cualquier símbolo). A pesar del nombre especial que reciben, los archivos CSV son archivos de texto como cualquier otro, y se pueden tratar como tales.

Los archivos de valores con separadores son muy fáciles de leer y escribir, y por esto son muy usados. Como ejemplo práctico, si usted desea hacer un programa que analice los datos de una hoja de cálculo Excel, puede guardar el archivo con el formato CSV directamente en el Excel, y luego abrirlo desde su programa escrito en Python.

Para leer los datos de un archivo de valores con separadores, debe hacerlo línea por línea, eliminar el salto de línea usando el método stripy luego extraer los valores de la línea usando el método split. Por ejemplo, al leer la primera línea del archivo de más arriba obtendremos el siguiente string:

'Perico:Los Palotes:90:75:38:65**\n**'

Para separar los seis valores, lo podemos hacer así:

**>>>** linea.strip().split(':')

['Perico', 'Los Palotes', '90', '75', '38', '65']

Como se trata de un archivo de texto, todos los valores son strings. Una manera de convertir los valores a sus tipos apropiados es hacerlo uno por uno:

valores = linea.strip().split(':')

nombre = valores[0]

apellido = valores[1]

nota1 = int(valores[2])

nota2 = int(valores[3])

nota3 = int(valores[4])

nota4 = int(valores[5])

Una manera más breve es usar las rebanadas y la función map:

valores = linea.strip().split(':')

nombre, apellido = valores[0:2]

nota1, nota2, nota3, nota4 = map(int, valores[2:6])

O podríamos dejar las notas en una lista, en vez de usar cuatro variables diferentes:

notas = map(int, valores[2:6])

Por ejemplo, un programa para imprimir el promedio de todos los alumnos se puede escribir así:

archivo\_alumnos = open('alumnos.txt')

**for** linea **in** archivo\_alumnos:

valores = linea.strip().split(':')

nombre, apellido = valores[0:2]

notas = map(int, valores[2:6])

promedio = sum(notas) / 4.0

**print** '{0} obtuvo promedio {1}'.format(nombre, promedio)

archivo\_alumnos.close()

Para escribir los datos en un archivo, hay que hacer el proceso inverso: convertir todos los datos al tipo string, pegarlos en un único string, agregar el salto de línea al final y escribir la línea en el archivo.

Si los datos de la línea ya están en una lista o una tupla, podemos convertirlos a string usando la función map y pegarlos usando el método join:

alumno = ('Perico', 'Los Palotes', 90, 75, 38, 65)

linea = ':'.join(map(str, alumno)) + '**\n**'

archivo.write(linea)

Otra manera es armar el string parte por parte:

linea = '{0}:{1}:{2}:{3}:{4}:{5}**\n**'.format(nombre, apellido,

nota1, nota2, nota3, nota4)

archivo.write(linea)

Como siempre, usted debe preferir la manera que le parezca más simple de entender.