

CLASE N°14

STRINGs E INPUT/OUTPUT

STRING

- El tipo string (`str`)
 - Es un tipo de dato que ya hemos utilizado durante el curso
 - No es más que una **lista de caracteres inmutable**
 - Podemos recorrerlo de forma parecida al tipo de dato **lista** (`list`), revisado en clases anteriores
 - Es un **objeto** y podemos solicitarle tomar alguna acción por medio de un conjunto de **métodos**
 - por ejemplo: `.lower()`, `.upper()`, `.capitalize()`

STRING



- La función nativa `raw_input()`
 - se usa para que el usuario **ingrese datos directamente**
 - Al igual que `input()`, devuelve lo ingresado por el usuario
 - A diferencia de `input()`, no evalúa lo ingresado como una expresión, sino que simplemente lo devuelve como **texto**
 - Al igual que `input()`, podemos **guardar el string** devuelto en una variable o **usarlo** en una expresión

3

ENTRADA DE DATOS



- En Python tenemos distintos métodos para alimentar a un programa con valores:
 - A través del teclado, con una **expresión válida en Python** usando `input()`
 - A través del teclado, con un **string** usando `raw_input()`
 - A través de un **archivo** en modo de lectura

4

MANEJO DE ARCHIVOS



■ Archivos

- Se identifica con un **nombre** y una **ruta** (ubicación del archivo en el medio de almacenamiento)
- En general trabajaremos con archivos en la **misma carpeta** del programa que lo accede para facilitar su uso
- Para utilizar un archivo primero se debe **abrir**:

```
<identificador> = open('<nombre archivo>', '<modo>')
```

5

MANEJO DE ARCHIVOS



- El modo en el que se abrirá un archivo depende del uso que se requiera:
 - **r**: el archivo **sólo se utilizará para lectura**, el programa falla si el archivo no se encuentra o se intenta modificarlo
 - **w**: el archivo **sólo se utilizará para escritura**, si el archivo no existe, éste se crea, y cualquier contenido previo en el archivo **se pierde**
 - **a**: el archivo **sólo se utilizará para escritura a partir del final del archivo**, también se crea si no existe, pero el contenido previo **no se pierde**

6

MANEJO DE ARCHIVOS



- La función primitiva `open()` devuelve un **objeto** de tipo **file**
 - Que **guardamos** en una variable para acceder al archivo
 - Acceso al contenido es por medio de sus **métodos**:
 - `<file>.close()`: **cierra el archivo**
 - `<file>.readline()`: **lee la siguiente línea del archivo** y se **devuelve como un string**. Si se usa `<file>.readlines()`, se genera una lista de strings donde cada string es una línea del archivo
 - `<file>.write(<str>)`: **escribe un string en el archivo**. Este método no escribe líneas, por lo que se debe insertar manualmente el salto de línea (`'\n'`)

7


SALIDA CON FORMATO



- En Python usamos un **string formato**
 - Incluyen **marcadores** para indicar la **ubicación** de un dato dentro del string
 - el marcador **%s** se usa para caracteres o un string
 - los marcadores **%d** o **%i** se usan para valores enteros
 - el marcador **%f** se usa para números flotantes
 - Indicamos los valores para cada marcador por medio del **operador %** entre un string formato y una lista inmutable de valores
 - Podemos **modificar los marcadores** para indicar largos mínimos, número de decimales, etc.


8

PARA LA PRÓXIMA CLASE



- Revisar ejemplo e intentar realizar la **tarea** propuesta
- Se estudiarán **maneras de afrontar problemas** de programación para generar soluciones antes de escribir el código
- A partir de ahora veremos la utilización y creación de **tipos de datos complejos** en Python

9



¿CONSULTAS?

10