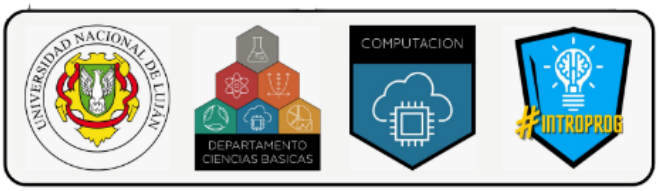

11071: Introducción a la Programación
Departamento de Ciencias Básicas
Universidad Nacional de Luján

Trabajo Práctico X

Manejo de Strings

1. Todas las variables y valores en Python tienen definido implícitamente un tipo. Mediante el uso de la función **type(...)**, podemos averiguar el tipo de un dato en determinado momento de programa. Utilice la función **type** para mostrar en pantalla los tipos de los siguientes valores:

```
'Hola mundo'
"Hola mundo"
100
'100'
```

Luego de haber determinado los tipos de los valores listados, responda:

- ¿De qué tipo son las cadenas de caracteres en Python?
- ¿Cuáles son las dos formas de escribir strings en Python? Investigue cuál es la diferencia entre ambas.
- ¿Es lo mismo que una variable tenga asignado el valor **100** a que tenga el valor **'100'**? ¿Cuál es la diferencia?

2. Las operaciones aritméticas tradicionales tienen un comportamiento especial cuando las aplicamos a strings. Utilice un script de Python para responder:

- ¿Qué función cumple el operador **+** entre strings?
- ¿Qué función cumple el operador ***** entre strings?
- Ejecute el siguiente código y describa con sus palabras cuál es el problema:

```
mi_valor = 'Hola' + 7
print(mi_valor)
```

3. Además de las operaciones que ya conocemos, los strings en Python tienen funciones predefinidas que nos facilitan su manipulación. Investigue el uso de cada una de las siguientes funciones, y escriba un ejemplo de su uso en un script de Python:

`len(string)`	`string.capitalize()`	`string.isnumeric()`
`string.lower()`	`string.replace(s1,s2)`	`string.ispace()`
`string.upper()`	`string.count(s1)`	`string.endswith(s1)`

4. Como su nombre lo indica, los strings son ***cadenas*** de caracteres, es decir, una sucesión de símbolos. Si lo entendemos de esta manera, podemos utilizar una estructura iterativa, por ejemplo el ***for***, para recorrer uno a uno los caracteres de un string, de la siguiente manera:

```
for letra in un_string:
  # código por cada caracter
```

Haga uso de esto para implementar un script que le solicite al usuario ingresar su nombre, y luego imprima cada una de las letras en un renglón diferente de la terminal.

5. Python nos permite averiguar si determinado caracter o palabra está dentro de un string, mediante el uso de la palabra reservada **in**. La misma se utiliza de la siguiente manera:

```
if contenido in palabra:
    # código a ejecutar si el contenido está dentro de la palabra
```

Haga uso de esta funcionalidad para pedirle al usuario que ingrese una frase por teclado, y luego verifique si la misma contiene alguna letra 'ñ', y si además contiene la palabra 'hola'. Informe en pantalla tanto en caso afirmativo como negativo.

6. Haciendo uso de las funciones para strings que ya conoce, implemente un script que haga lo siguiente:

I. Le solicite al usuario ingresar una palabra por teclado. Se debe validar que la palabra tenga al menos una 'ñ', que no sea sólo caracteres numéricos, y que no sean sólo espacios en blanco. En caso de no ser válida, se le debe pedir al usuario que la reingrese.
II. Informe en pantalla la cantidad de letras de la palabra ingresada.
III. Transforme la palabra a mayúsculas, reemplace todas las 'Ñ' por 'N', y luego muestre el resultado en pantalla.

7. Escribir funciones que dada una cadena de caracteres:
I. Imprima los dos primeros caracteres.
II. Imprima los tres últimos caracteres.
III. Imprimir dicha cadena en sentido inverso.

8. Escribir una función que reciba una cadena que contiene un largo número entero y devuelva una cadena con el número y las separaciones de miles. Por

ejemplo, si recibe '1234567890', debe devolver '1.234.567.890'.

9. Escribir funciones que dada una cadena de caracteres devuelva solamente las letras consonantes. Por ejemplo, si recibe 'algoritmos' debe devolver 'lgrtms'.

10. Escribir una función que reciba una cadena de unos y ceros (es decir, un número en representación binaria) y devuelva el valor decimal correspondiente.

¡A testear!

En los siguientes ejercicios aprenderemos a crear nuestros propios tests unitarios, y los vamos a definir con la siguiente estructura:

```
def test<NombreDeLaFuncion> ():

    assert funcion1(parametros) == <resultadoEsperado>
    assert funcion1(parametros) == <resultadoEsperado>
    assert funcion1(parametros) == <resultadoEsperado>

    print("Paso")
```

1. Cree un test unitario para el **ejercicio 9.**.
2. Cree un test unitario para el **ejercicio 10.**.