CADENA
(STRING)

Sucesión de caracteres encerrada entre “ ”

Ejemplos:

“11-4469-7500”, “Hola LUISA!”,
“Jeje! ☺”
“abc@gmail.com”

¿QUÉ SE PUEDE HACER CON ELLAS?

- Operador + (concatenación de cadenas): acepta dos cadenas como operandos y devuelve la cadena que resulta de unir la segunda a la primera.

- Operador * (repetición de cadena): acepta una cadena y un entero y devuelve la concatenación de la cadena consigo misma tantas veces como indica el entero.

PROBEMOS CONCATENAR…

```
>>> saludo = “Hola”
>>> nombre = “Luisa”
>>> saludo+nombre
```

¿Problemas?

claro… la salida será:

‘HolaLuisa’

y la idea no es generar cadenas que dificulten la lectura.

POSIBLES SOLUCIONES:

>>> saludo+“ ”+nombre

o…

>>>nombre= “ Luisa”

y…

>>> saludo+nombre

‘Hola Luisa’

AHORA REPETIR…

>>>saludo * 3 ‘HolaHolaHola’

¿otra vez problemas…?

…proponer una solución sin agregar espacio en la variable saludo !!!

¿PROBARON….

```
>>> (saludo+“ ”)*3
```

También se puede combinar operaciones de cadenas !!!

ALGUNOS MÉTODOS QUE PODEMOS USAR CON CADENAS

Podemos manipular cadenas, además, mediante **métodos** (funciones especiales) que les son propios:

- cadena.lower() (paso a minúsculas): devuelve una cadena con los caracteres de la cadena en minúsculas.
- cadena.upper() (paso a mayúsculas): devuelve una cadena con los caracteres de la cadena en mayúsculas.
- cadena.capitalize() (paso a palabras con inicial mayúscula): devuelve la cadena que empieza con mayúscula.
- cadena.swapcase() (invierte mayúsculas y minúsculas): devuelve la cadena convirtiendo minúsculas en mayúsculas y viceversa.

PROBANDO MÉTODOS…

```
>>> cadena = “Buen inicio de IP”

>>> print(cadena.lower())
>>> print(cadena.upper())

>>> print(cadena.capitalize())

>>> print(cadena.swapcase())
```

Probar en python!!

FUNCIONES PARA CADENAS

a = “Hola Mundo”

>>> len(a)

10

>>> len(“abcd” * 4)

16

>>> len(“a b”)

3

Devuelve la longitud del string

¿Qué valor devuelve len(“”)? 0

¿Qué valor devuelve len(“ ”)? 1

CONVIRTIENDO DATOS...

int: recibe una cadena cuyo contenido es una secuencia de dígitos y devuelve el número entero que describe.

float: acepta una cadena cuyo contenido describe un flotante y devuelve el flotante en cuestión.

str: se le pasa un entero o flotante y devuelve una cadena con una representación del valor como secuencia de caracteres.

```
>>> modelo = 2018
>>> modelo
2018
>>> type(modelo)
<class 'int'>
>>> str(modelo)
'2018'
```

```
>>> peso = "64.5"
>>> peso
'64.5'
>>> type(peso)
<class 'str'>
>>> float(peso)
64.5
```

```
>>> edad = "25"
>>> edad
'25'
>>> type(edad)
<class 'str'>
>>> int(edad)
25
```

Y PARA QUÉ CASTEAR DATOS...?

- Para conocer la longitud de un dato numérico... no podría aplicar la función **len( )**, primero debería castear el número a cadena.

- Para armar una dirección de e-mail combinando texto y números, ya que no se pueden concatenar distintos tipos de datos.

```
>>> nombre = “ana”
>>> anio = 2018
>>> mail = nombre+anio+”@gmail.com”
>>> mail = nombre+str(anio)+”@gmail.com”
>>> mail
‘ana2018@gmail.com’
>>> len(mail)
17
```

TypeError: object of type 'int' has no len()

```
>>> codigo = 13698547
>>> len(codigo)
>>> len(str(codigo))
8
```

TypeError: Can't convert 'int' object to str implicitly