El futuro digital es de todos

MinTIC

Funciones Básicas

Rogerio Orlando Beltrán Castro

Universidad Industrial de Santander

MisiónTIC2022

Para trabajar con texto:

- **print()** Imprime por pantalla el argumento que le pasemos
- **len()** Nos devuelve la longitud en caracteres de la cadena que le pasemos como parámetro
- **replace()** Remplaza una cadena por la que incluya como argumento.
- **upper()** Convierte la cadena pasada como argumento en mayúsculas
- **lower()** Convierte la cadena pasada como argumento en minúsculas

Ejercicio Print()

En los programas, para que python nos muestre texto o variables hay que utilizar la función print().

La función print() permite mostrar texto en pantalla. El texto a mostrar se escribe como argumento de la función:

```
print("Hola")
```

Las cadenas se pueden delimitar tanto por comillas dobles (") como por comillas simples (').

Ejercicio Print()

La función print() admite varios argumentos seguidos. En el programa, los argumentos deben separarse por comas. Los argumentos se muestran en el mismo orden y en la misma línea, separados por espacios:

```
print("Hola", "Adiós")
```

```
Hola Adiós
```

Cuando se trata de dos cadenas seguidas, se puede no escribir comas entre ellas, pero las cadenas se escribirán seguidas, sin espacio en blanco entre ellas:

```
print("Hola" "Adiós")
```

```
HolaAdios
```

Al final de cada print(), Python añade automáticamente un salto de línea:

```
print("Hola")
print("Adiós")
```

```
Hola
Adiós
```

Para generar una línea en blanco, se puede escribir una orden print() sin argumentos.

```
print("Hola")
print()
print("Adiós")
```

```
Hola

Adiós
```

Si se quiere que Python **no** añada un salto de línea al final de un print(), se debe añadir al final el argumento end="":

```
print("Hola", end="")
print("Adiós")
```

```
HolaAdios
```

Ejercicio Print()

Si se quiere que Python **no** añada un salto de línea al final de un print(), se debe añadir al final el argumento end="":

```
print("Hola", end="")
print("Adiós")
```

```
HolaAdios
```

En el ejemplo anterior, las dos cadenas se muestran pegadas. Si se quieren separar los argumentos en la salida, hay que incluir los espacios deseados (bien en la cadena, bien en el argumento end):

```
print("Hola. ", end="")
print("Adiós")
```

```
Hola. Adiós
```

```
print("Hola.", end=" ")
print("Adiós")
```

```
Hola. Adiós
```

El valor del parámetro end puede ser una cadena f:

```
texto = " y "
print("Hola", end=f"{texto}")
print("Adiós")
```

```
Hola y Adiós
```

Ejercicio Print()

Para incluir comillas dentro de comillas, se puede escribir una contrabarra (\) antes de la comilla para que Python reconozca la comilla como carácter, no como delimitador de la cadena:

```
print("Un tipo le dice a otro: \"¿Cómo estás?\"")
print('Y el otro le contesta: \'¡Pues anda que tú!\'')
```

```
Un tipo le dice a otro: "¿Cómo estás?"
Y el otro le contesta: '¡Pues anda que tú!'
```

O escribir comillas distintas a las utilizadas como delimitador de la cadena:

```
print("Un tipo le dice a otro: '¿Cómo estás?'")
print('Y el otro le contesta: "¡Pues anda que tú!"')
```

```
Un tipo le dice a otro: '¿Cómo estás?'
Y el otro le contesta: "¡Pues anda que tú!"
```

La función print() permite incluir variables o expresiones como argumento, lo que nos permite combinar texto y variables:

```
nombre = "Pepe"
edad = 25
print("Me llamo", nombre, "y tengo", edad, "años.")
```

```
Me llamo Pepe y tengo 25 años.
```

```
semanas = 4
print("En", semanas , "semanas hay", 7 * semanas, "días.")
```

```
En 4 semanas hay 28 días.
```

Ejercicio Print()

La función print() muestra los argumentos separados por espacios, lo que a veces no es conveniente. En el ejemplo siguiente el signo de exclamación se muestra separado de la palabra.

```
nombre = "Pepe"
print("¡Hola,", nombre, "!")
```

```
¡Hola, Pepe !
```

Antes de Python 3.6 estas situaciones se podían resolver de una forma algo engorrosa (que se comenta en la lección de temas obsoletos), pero a partir de Python 3.6 se pueden utilizar las cadenas "f".

```
nombre = "Pepe"
print(f"¡Hola, {nombre}!")
```

```
¡Hola, Pepe!
```

Ejercicio len()

- La función len devuelve la longitud (el número de elementos) de un objeto. El argumento puede ser una secuencia (como un string, bytes, tupla, listado o rango) o una colección (como un diccionario, conjunto o conjunto helado).

En este primer ejemplo se cuenta el número de caracteres de una cadena de texto:

```
print(len("Python"))
```

```
In [1]: print(len("Python"))
        6
```

Ejercicio len()

También puede usarse para contar el número de elementos de un diccionario:

```
d = dict([("uno", 1), ("dos", 2)])
print(len(d))
```

```
In [2]: d = dict([("uno", 1), ("dos", 2)])
        print(len(d))

        2
```

Ejercicio len()

En este ejemplo la función devuelve el número de elementos de una lista:

```
m = ["a", "b", 1, 2, "c"]
print(len(m))
```

```
In [3]: m = ["a", "b", 1, 2, "c"]
        print(len(m))

        5
```

Ejercicio replace()

```
1. >>> dir_name = 'S2A_MSIL1C_20190106T105431_N0207_R051_T30SXH_20190106T112304'
2. >>> new_dir_name = dir_name.replace('MSIL1C', 'MSIL2A')
3. >>> new_dir_name
4. 'S2A_MSIL2A_20190106T105431_N0207_R051_T30SXH_20190106T112304'
```

Ejercicio replace()

Limitando el número de ocurrencias a reemplazar

Al método replace() se le puede pasar un tercer parámetro opcional de tipo integer. Este parámetro indica el número máximo de ocurrencias a ser reemplazadas:

```
1.  >>> aes = 'aaaaaa'
2.  >>> nueva = aes.replace('a', 'b', 3)
3.  >>> nueva
4.  'bbbaaa'
```

Ejercicio upper()

- Para convertir una cadena a mayúsculas, utiliza el método upper(). upper() devuelve una copia de la cadena en mayúsculas.

```
>>> hola = 'Hola pythonista'
>>> hola_upper = hola.upper()
>>> print(hola)
Hola pythonista
>>> print(hola_upper)
HOLA PYTHONISTA
```

Ejercicio **lower()**

- Para convertir una cadena a minúsculas, utiliza el método lower(). lower() devuelve una copia de la cadena en minúsculas.

```
>>> hola = 'HOLA Pythonista'
>>> hola_lower = hola.lower()
>>> print(hola)
HOLA Pythonista
>>> print(hola_lower)
hola pythonista
```

Ejercicio Convertir solo la primera letra de un texto a mayúsculas

- Para convertir la primera letra de una cadena a mayúsculas, utiliza el método capitalize(). capitalize() devuelve una copia de la cadena con la primera letra en mayúsculas..

```
frase=input('Ingrese una frase:')
print(frase)
frase=frase.capitalize()
print(frase)
```

Ejercicio Cambiar mayúsculas por minúsculas y viceversa

- Para cambiar los caracteres en mayúsculas de un texto por minúsculas y viceversa, utiliza el método swapcase(). swapcase() devuelve una copia de la cadena con los caracteres en mayúsculas cambiados por minúsculas y viceversa.

```
hola = 'Hola Pythonista. ¿Te gusta Python?'
hola_swaped = hola.swapcase()
print(hola_swaped)
```

Convertir un texto a formato título

Para convertir un texto a formato título, utiliza el método title(). title() convierte la primera letra de cada palabra de una cadena a mayúsculas.

```
hola = 'hola pythonista. ¿te gusta python?'
hola_title = hola.title()
print(hola_title)
```

Comprobar si un texto está todo en mayúsculas

Para comprobar si un texto está todo en mayúsculas, utiliza el método isupper(). isupper() devuelve True si todos los caracteres de una cadena están en mayúsculas, False en caso contrario.

```
frase=input('Ingrese una frase:')
if frase.isupper():
    print('Esta en mayúscula')
else:
    print('No esta en mayúscula')
```

Comprobar si un texto está todo en minúsculas

Para comprobar si un texto está todo en minúsculas, utiliza el método islower(). islower() devuelve True si todos los caracteres de una cadena están en minúsculas, False en caso contrario.

```
frase=input('Ingrese una frase:')
if frase.islower():
    print('Esta en minúscula')
else:
    print('No esta en minúscula')
```

Funciones numéricas

- **sum()** Suma la lista de números pasados como argumento.
- **min()** Determina el valor mínimo de una lista de números pasados como argumento.
- **max()** Determina el valor máximo de una lista de números pasados como argumento.
- **range()** Crea una lista aritmética de números según el valor pasado como argumento.
- **str()** Convierte un valor numérico en texto.

Ejercicio **sum()**

- **sum()** Suma la lista de números pasados como argumento.

Ejercicio **min()**

- **min()** Determina el valor mínimo de una lista de números pasados como argumento.

Ejercicio **max()**

- **max()** Determina el valor máximo de una lista de números pasados como argumento.
- **range()** Crea una lista aritmética de números según el valor pasado como argumento.
- **str()**

Ejercicio **range()**

- **range()** Crea una lista aritmética de números según el valor pasado como argumento.
- **str()**

Ejercicio **str()**

- Diseñar el algoritmo correspondiente a un programa que al introducir una cantidad de dinero expresado en pesos nos indique cuántos billetes y monedas se puede tener como mínimo (Billetes de 1000, 2000, 5000, 10000, 20000 y 50000, monedas de 20, 50, 100, 200 y 500).