1000 PROGRAMADORES

Introducción a la
Programación en Python

MÓDULO 3

Entradas, uso de la terminal. Scripts
Salidas y formatos.

Ministerio de Educación
Cultura, Ciencia y Tecnología
Gobierno de Salta

Ministerio de Economía
y Servicios Públicos
Gobierno de Salta

Universidad
Nacional de Salta

INTRODUCCIÓN

Los programas serían de muy poca utilidad si no fueran capaces de interactuar con el usuario, incluso podrían no cumplir con el objetivo de encontrar la solución a un problema.

La entrada de datos en Python es bastante simple por medio de la función `input()`. En este módulo conoceremos una de las funciones para leer leer datos por teclado y asignar ese valor recibido a una variable. Además algunas consideraciones a tener en cuenta al momento de usarla.

La forma general de mostrar información por pantalla es mediante una consola de comando, generalmente podemos mostrar texto y variables separándolos con comas, para esto se usa la sentencia `print()`.

ENTRADAS

Entrada de datos por consola

Para pedir al usuario que introduzca algún dato a través del teclado, usaremos el método input(). Este método, recibe como parámetro un **`str`** con el mensaje a mostrar al usuario:

```
edad = input('¿Qué edad tienes? ')
```

```
¿Qué edad tienes?
```

salida

En este momento, la consola queda a la espera de la entrada de datos por parte del usuario. Al introducir cualquier texto, será asignado a la variable edad.

La entrada es convertida siempre a un objeto de tipo str UTF-8. Por tanto, si lo que necesitamos es cualquier otro tipo, por ejemplo, un int, habrá que hacer la conversión correspondiente.

En el siguiente ejemplo, se pide un número al usuario y se desea mostrar su consecutivo.

```
x=input("ingrese nro: ")
x=x+1
print(x)
```

```
ingrese nro: 3          #Solicita el ingreso del dato
---------------------------------------------------------------------
TypeError                          Traceback (most recent call last)
<ipython-input-45-7067bfee6ec2> in <module>()
      1 x=input("ingrese nro: ")
----> 2 x=x+1            #Indica error en esta línea
      3 print(x)
TypeError: can only concatenate str (not "int") to str
```

salida

Como el dato ingresado se almacena como cadena de texto, no es posible encontrar

su consecutivo. Veamos cómo se consigue una solución al problema:

```
x=int(input("ingrese nro: ")) #Usamos función int()
x=x+1
print(x)
```

```
ingrese nro: 3
4
```

---

salida

En el código anterior se usó la función predefinida **`int()`** para convertir el dato ingresado de cadena de texto a valor numérico, por lo tanto la operación para obtener su consecutivo está permitida.

Entrada de datos por Script

Hasta aquí lo que hemos escrito código en el intérprete, y/o escribir pequeños programas Python, pero los programas informáticos más complejos no funcionan así. Se basan en escribir todas las instrucciones en archivos llamados *scripts*, que no son más que guiones de instrucciones. Luego se envía este archivo al intérprete como parámetro desde la terminal y éste ejecutará todas las instrucciones en bloque.

Aparte de ser la base del funcionamiento de los programas, la característica de los *scripts* es que pueden recibir datos desde la propia terminal de comando en el momento de la ejecución, algo muy útil para agregar dinamismo a los *scripts* a través de parámetros personalizables.

A continuación, un ejemplo el cual simula una sala de chat, validando datos de entradas numéricas y tipo cadena de caracteres e interactúa con el usuario.

```
print ("\nSimulando a SalaChat")
print ("====================")
print ("\nSala de Chat > De 19 a 29 años")
print ("-----------------------------\n")
print ('Ciro: ¿Cómo se llama usted?: ' )
nombre = input('Yo: ')
print ('Ciro: Hola', nombre, ', encantada de conocerte :3')
print ('Ciro: ¿Que edad tiene usted?: ')
edad = input('Yo: ')
print ('Usted tiene', edad, ', y yo ya no digo mi edad xD')
```

```
Simulando a SalaChat
====================

Sala de Chat > De 19 a 29 años
------------------------------

Ciro: ¿Cómo se llama usted?:
Yo: cecilia
Ciro: Hola cecilia , encantada de conocerte :3
Ciro: ¿Que edad tiene usted?:
Yo: 29
Usted tiene 29 , y yo ya no digo mi edad xD
```

salida

Además de la entrada por consola, es posible obtener datos desde un archivo de texto de diversas extensiones o desde una página web. El primero de los ítems lo desarrollaremos más adelante estudiando el uso de ficheros JSON y CSV, el segundo tema conocido como Web Scraping lo dejaremos para un curso más avanzado y como objeto de investigación de los estudiantes.

SALIDAS Y FORMATOS

Mostrar datos por pantalla no solo es sencillo sino además flexible. En general, es cuestión de una sola línea de código con una sola función llamada print().

La función print() de Python, recibe entre los paréntesis lo que sea que quieras mostrar y además permite indicar algunos detalles adicionales.

Mostrar mensajes de texto

Se realiza colocando como argumento un texto encerrado entre comillas (ya sean dobles "" o simples '').

```
print("mostremos un mensaje corto") #comillas dobles
print('mostremos un mensaje corto') #comillas simples
```

```
mostremos un mensaje corto
mostremos un mensaje corto
```

---

salida

No hay un límite en la cantidad de caracteres a imprimir, pero si el texto es muy largo puede ensuciar el código.

```
print("mostremos un mensaje largo para ver algunas características de la función print")
```

```
mostremos un mensaje largo para ver algunas características de la función print
```

---

salida

Tratemos de dividirlo en varias llamadas a print para que el código sea más legible:

```
print("mostremos un mensaje largo ")
print("para ver algunas características ")
print("de la función print")
```

```
mostremos un mensaje largo
para ver algunas características
de la función print
```

---

salida

Sin embargo el resultado obtenido es completamente diferente al esperado, pues, Python puso cada mensaje en una línea diferente en la pantalla. Esto sucede, debido a que, por defecto, Python crea saltos de línea al final de cada llamado a *print()*.

Esto puede resolverse usando parámetros especiales de *print()*, indicando que queremos poner un espacio en blanco, en lugar de un salto de línea, de modo que todo quede en una sola línea en pantalla y con sus debidos espacios. Lo que haremos es cambiar el final de línea de *print()* en Python indicándolo en un parámetro llamado *end*:

```
print("mostremos un mensaje largo", end=" ")
print("para ver algunas características", end=" ")
print("de la función print")
```

```
mostremos un mensaje largo para ver algunas características de la
función print
```

---

salida

Mostrar variables o expresiones

Además de texto puro es posible mostrar valores asignados a una variable o directamente expresiones algebraicas sin ser asignadas previamente.

```
a=5
print(a)
print(a+1)
print(5*8)
```

```
5
6
40
```

salida

Por supuesto que una salida es más clara o explícita cuando se combinan los valores que se desean mostrar con mensajes de texto que describen esos valores.

```
c=5*8
print("El resultado de 5*8 es",5*8)
print("El resultado de 5*8 es",c)
```

```
El resultado de 5*8 es 40
El resultado de 5*8 es 40
```

salida

Otro ejemplo:

```
a=5
b=8
print("El resultado de",a,"x",b,"es",a*b)
```

```
El resultado de 5 x 8 es 40
```

salida

Por defecto el separador de *print* es un espacio en blanco, es decir, separa fin de cadena de texto o valor de variable. Es posible modificarlo usando el parámetro *sep*.

```
a=5
print("La sucesión es ",a,a+1,a+2,sep="-")
```

```
La sucesión es -5-6-7
```

salida

Mejoramos el ejemplo anterior combinando valores de los parámetros ya vistos:

```
a=5
print("La sucesión es ",a,end="")
```

```
print("",a+1,a+2,sep="-")
```

```
La sucesión es  5-6-7
```

salida

Muestra Avanzada

Hasta aquí hemos visto salidas en pantalla usando sentencias básicas, esta última forma que te explicaremos abarca las ya vistas y ayuda a una mejor legibilidad de una salida dentro de un programa Python.

Se propone una nueva forma de interpolar o intercalar expresiones en cadenas y aplicar formatos con conversiones implícitas, llamada *cadenas f* conocida como ***f-strings***. Un nuevo instrumento disponible desde Python 3.6 que simplifica y unifica las posibilidades anteriores.

```
curso = 'Python'
plataforma = 'Moodle'
print(f"El nuevo curso de {curso} en {plataforma} está increíble")
```

```
El nuevo curso de Python en Moodle está increíble
```

salida

Tenemos una línea más corta, más legible ofreciendo la oportunidad de invocar funciones, por ejemplo si quisiéramos colocar el valor de *curso* en mayúsculas.

```
curso = 'Python'
plataforma = 'Moodle'
print(f"El nuevo curso de {curso.upper()} en {plataforma} está increíble")
```

```
El nuevo curso de PYTHON en Moodle está increíble
```

salida

Ya que las f-strings son evaluadas al momento de la ejecución tenemos la posibilidad de colocar cualquier expresión válida.

```
print(f"{100 + 28}")
print(f"{25 * 7}")
```

```
128
175
```

---

salida

Si necesitamos dividir nuestro mensaje en múltiple líneas podemos hacerlo de esta forma.

```
curso = 'Python'
plataforma = 'Moodle'
message = (
    f"Nuevo curso de {curso} "
    f"disponible en {plataforma} "
)
print(message)
```

```
Nuevo curso de Python disponible en Moodle
```

---

salida

Es muy importante colocar la ***f*** al inicio de cada línea de lo contrario Python imprimirá la línea como texto, con llaves y variables inclusive.

BIBLIOGRAFÍA

[1] Python para todos. R. Gonzalez Duque. Licencia Creative Commons Reconocimiento 2.5 España.
Descarga gratuita de versión más reciente en http://mundogeek.net/tutorial-python/
[2] Python 3 al descubierto 2da Edición. A. Fernández Montoro. Alfaomega. Año 2013
[3] Tutoriales Web varios.