ENTRADA Y SALIDA DE DATOS

Guardar datos de entradas

Python también nos permite guardar valores que le pasamos por entrada de consola. Podemos utilizar estos para darle una mejor dinámica a nuestro programa dependiendo del dato ingresado.

```
entrada    = input ("Ingrese el valor que quiera guardar en la variable")
```

Una vez que esta línea aparezca en pantalla, el programa se pausara esperando que el usuario ingrese un valor. Una vez ingresado, presionando "**enter/intro**", el flujo del programa seguirá de manera normal, pero utilizando esta nueva entrada de datos como parte del programa.

Recordemos un detalle, la máquina no sabe lo que nosotros pensamos, así que, si nosotros le pasamos un valor, ella siempre lo interpretará como válido, al menos que le indiquemos lo contrario

Input

El objetivo de la instrucción input es parar un programa en Python, esperar que el usuario introduzca un dato mediante el teclado y tras apretar la tecla enter (o intro o return), almacenar este dato en una variable.

Ejemplo:

```
entrada  = input ("Ingrese el valor que quiera guardar en la variable") # 9
print(entrada) # 9
```

Diferencia entre PRINT e INPUT:

- Print, solamente muestra en pantalla un mensaje o un valor que quiera mostrar en la terminal.
- Input, puede mostrar un mensaje en pantalla y además espera que el usuario ingrese un valor y/o presione la tecla Enter para continuar.

Type y Casting

Con todo lo estudiado hasta ahora, por fin podemos hablar de Type y Casting.
Estas son herramientas de los tipos de datos, una nos indica de qué tipo de dato estamos tratando una variable (type) y la otra es para poder cambiar a una variable su tipo de dato (cast).
Con la función type() podremos saber qué tipo de dato es. Y con int(), float(), str(), list(), dict(), tuple() cambiaremos el tipo de datos.

```
variable_1 = 5
print(type(variable_1))          # <class 'int'>
variable_2 = str(variable_1)     # aquí es donde hacemos el casting de int a str.
print(type(variable_2))          # <class 'str'>
```

La asignación múltiple de variables, también puede darse utilizando como valores, el contenido de una tupla o una lista.

Tipo de dato del input

La instrucción input devuelve siempre una cadena. ¡**Siempre!**, aunque tecleemos un número.

Por ejemplo:

```
dni = input ("Ingrese un DNI: ")  # 35236211
print(type(dni))       # <class 'str'>
```

Concatenación

Vimos previamente que el operador suma (+) nos permite concatenar strings en Python.

Por ejemplo:

```
nombre = 'Juan Pablo'
apellidos = ‘Sosa Gómez'
nombre_completo = nombre + apellidos
print(nombre_completo)      #      'Juan PabloSosa Gómez'
```

Para agregar un espacio entre ambos valores, podría concatenar un string con un espacio:

```
nombre_completo = nombre + “ ” + apellidos
```

Concatenar otros elementos

Como no podemos concatenar con el operador + un **string** con una variable numérica, podemos utilizar la función **STR** para convertir esa variable:

Por ejemplo:

```
suma = 1 + 2
res = 'El resultado de 1 + 2 es: ' + str(suma)
print(res)      #      'El resultado de 1 + 2 es: 3'
```

Cadenas f-Strings

A partir de la versión 3.6 se introdujo f-Strings. También llamado formatted string literals ó Formateo literal de cadenas, f-Strings tiene una sintaxis simple y fluida que hará más sencillo el darle formato a cadenas de texto.

Una cadena de texto normalmente se escribe entre comillas dobles ("") o comillas simples (''). Para crear f-strings, solo tienes que agregar la letra f o F mayúscula antes de las comillas.

Por ejemplo:

```
lenguaje = "Python"
lugar = "dydInformática"
print(f"Curso de {lenguaje} en {lugar}.")      #      Curso de Python en dydInformática
```

También es posible realizar una operación dentro de las llaves: {num1 * num2}