Entrada y salida de datos

Ya se ha observado que la funcion print() imprime una linea completa, ademas de combinar cadenas con la operacion concatear "+". También se puede utilizar el caracter "," como separador de variables (ademas de incluir espacios alrededor) y estas no necesariemente tienen que ser todas cadenas.

In [ ]:

```
dias_febrero = 28
print("El mes de febrero tiene", dias_febrero, "dias")
```

Y esto sirve tambien para operaciones:

In [ ]:

```
nombre = None      # Esriba su nombre en la variable
print("Bienvenido al lado oscuro", nombre)
```

In [ ]:

```
print("Esta frase" + "..." + " continua aqui")
```

In [ ]:

```
print("El modulo de", 3+4j, "es", abs(3+4j))
```

In [ ]:

```
i = 1
print("El valor de i es",i)
print("El valor de i+1 es",i+1)
```

Se puede apreciar que *print* imprime cadenas, variables, operaciones con variables. Ademas, *print* siempre imprime una línea entera (es decir, ingresa una "nueva linea" al final). Se puede evitar esto colocando el espcificador *end=" "* como argumento de la función *print*

In [ ]:

```
i = 1
print("El valor de i es",i, end="")
print("El valor de i+1 es",i+1)
```

O especificar dentro de los " " algun caracter que se utilizara para separar ambos string.

In [ ]:

```
i = 1
print("El valor de i es",i, end=".")
print(" El valor de i+1 es",i+1)
```

Salida con formato

Como se puede apreciar, la función print permite imprimir textos y numeros con mucha flexibilidad. Hay muchas maneras de utilizar los diferentes parametros de la función print y hay basicamente tres maneras de tener el control sobre el formato de una salida.

In [ ]:

```
q = 459
p = 0.098
print(q, end=" ")
print(p, end=" ")
print()
print(q, p, p*q)
print(q, p, p*q, sep=",")
print(q, p, p*q, sep="->")
```

En este primer método, se utilizan las especificaciones de *print* para controlar la seperación de caracteres(,), el final de linea (end=" ") y el tipo de caracter de separación (sep = ","). La instrucción *print()* genera una linea en blanco

In [ ]:

```
dni = 18765367
peso = 60.45
altura = 1.60
print("El paciente con DNI %d mide %1.1f metros y pesa %3.2f kilogramos" % (dni, altura
, peso))
```

Este segundo método es conocido como string modulo y hace referencia al operador % que separa el texto a mostrar de los datos a insertar. Este método es parecido al uso de las instruccones printf en C, en donde se colocan unos especificadores de formato (%d para enteros, %f para flotantes o %a.bf donde a indica el numero de digitos de la parte entera y b el numero de digitos de la parte decimal) y luego el operador % para agregar los datos a incluír. Este método esta en desuso actualmente el Python.

In [ ]:

```
id_alumno = 2635363
nota1, nota2, nota3 = 12, 14, 16
print("El alumno {} tiene en sus practicas {}, {} y {}".format(id_alumno, nota1, nota2,
nota3))
```

Esta es la forma preferida pues permite la mayor flexibilidad. Se requiere del uso de un método de la función *print* llamado *format*. Tiene la forma:

```
print(" { }  ".format( ))
```

Donde cada una de las *{ }* contendra cada uno de los valores dentro del método *format*.

In [ ]:

```
print("Los tres primeros numeros: {}, {} y {}".format(1,2,3))
```

Su puede utilizar un parametro posicional {i}, donde i = 0, 1, 2, ... para no redundar en los datos dentro de format:

In [ ]:

```
print("{0}, {1} y {2} en inverso con {2}, {1} y {0}".format(1,2,3))
```

Asi tambien, se puede especificar el formato de los numeros a imprimir:

In [ ]:

```
print("Art: {0:5d} Precio: {1:8.2f}".format(345,120.50))
```

{0:5d} imprime el primer valor como entero en un espacio de 5 caracteres {1:8,2f} imprime el segundo valor como flotante en un espacio de 8 caracteres, donde se reservan los 2 ultimos para decimales

Ejercicios

Complete el bloque de código para que muestre el texto

"*nombre*, la ID = 087252 ingresada no es valida. Error = -1"

In [ ]:

```
nombre =
identidad =
print()
```

Modifique el bloque de código para que se muestren las operaciones de suma, resta, multiplicacion y division de dos variables p y q, utilizando el metodo *format*

In [ ]:

```
p =
q =
print()   #"valor p + valor q = p + q"
print()   #"valor p - valor q = p - q"
print()   #"valor p * valor q = p * q"
print()   #"valor p / valor q = p / q"
```

Entrada de datos

Para ingresar datos y cargarlos a una variable, se utiliza la función *input*

In [ ]:

```
nombre = input("Ingrese su nombre: ")
print("Hola", nombre)
```

Es importante notar que la función input retorna un valor str.

In [ ]:

```
num1 = input("Ingrese un numero: ")
num2 = input("Ingrese otro numero: ")
print("{} + {} = {}".format(num1, num2, num1 + num2)
```

Si lo que se espera es ingresar datos numéricos se deben de utilizar las funciones de conversión de datos:

In [ ]:

```
num1 = int(input("Ingrese un numero: "))
num2 = int(input("Ingrese otro numero: "))
print("{} + {} = {}".format(num1, num2, num1 + num2))
```

Se puede utilizar el función split() para pedir varios valores en un mismo input y almacenarlos en diferentes variables.

In [ ]:

```
num1, num2 = input("Ingrese dos numeros: ").split()
print(num1)
print(num2)
```

Ejercicios

Complete el bloque de codigo para que el programa pida al usuario que ingrese su nombre en el formato Nombre Apellido e imprima el texto "Bienvenido Apellido, Nombre"

```
Ingrese su nombre: Elmer Curio
Bienvenido Curio, Elmer
```

In [ ]:

```
```

Escriba un programa que pida al usuario que ingrese dos numeros en una sola línea y luego muestre los resultados de todas las operaciones aritméticas:

```
Ingrese dos numeros : 5 2

5 + 2 = 7
5 - 2 = 3
5 * 2 = 10
5 / 2 = 2.5
5 % 2 = 1
```

In [ ]:

```
```