



Instructivo 1 Algoritmos

Competencia: Desarrollo de la solución de software

Introducción

El siguiente instructivo esté enfocado en guiarlo en el diseño y construcción de algoritmos, los cuales son la forma escrita de representación de la algoritmia. Para esta primera parte de la competencia se realizarán todas las prácticas, ejercicios y talleres en lenguaje Python.

Contenido

Instructivo 1 Algoritmos.....	1
Introducción.....	1
Estructura General de un Programa.....	2
Variables.....	3
Imprimir en pantalla	9
Constantes.....	13
Lectura de datos	14
Comentarios	16
Comentarios de una línea	16
Comentarios de bloque o multilínea.....	16



Estructura General de un Programa

Un programa puede considerarse como una secuencia de acciones (instrucciones) que manipulan un conjunto de datos para que realice una tarea específica. En general un programa está formado por dos bloques:

- Bloque de declaraciones: en este bloque se especifican todos los objetos que utilizaría el programa (constantes, variables, tablas, registros, archivos, etc.). Las declaraciones se utilizan en aquellos lenguajes de programación que no tienen declaración explícita de los objetos. Su misión consiste en indicar al procesador que reserve espacio en la memoria para un objeto del programa, indicando asimismo su nombre, tipo y características.
- Bloque de instrucciones: lo constituye el conjunto de operaciones y la secuencia de instrucciones que se han de realizar para la obtención de los resultados deseados. Dentro de este bloque se diferencia tres partes fundamentales:
 - Entrada de datos: conformada por todas las instrucciones que toman datos de un dispositivo externo, almacenándolos en la memoria central para que puedan ser procesados.
 - Proceso: formado por las instrucciones que modifican/procesan los datos, dejando estos disponibles en la memoria central.
 - Salida de resultados: conjunto de instrucciones que toman los datos finales de la memoria central y los envían a los dispositivos externos.

Existen varios tipos de instrucciones en un programa:

- Básicas (Primitivas): son aquellas que ejecuta el procesador de modo inmediato. Las principales son asignación, entrada y salida:
 - Instrucción de asignación: consiste en calcular/indicar el valor de una expresión y almacenarlo en una variable.
 - Instrucción de entrada: toma un dato de un dispositivo de entrada y lo almacena en una variable.
 - Instrucción de salida: toma el valor de una expresión o variable y lo lleva a un dispositivo externo.



- De Control: este tipo de instrucciones controlan la ejecución de otras instrucciones. Existen varios tipos:
 - Selectivas(alternativas): controlan la ejecución de unas u otras instrucciones según una condición.
 - Salto: alternan la secuencia normal de ejecución de un programa únicamente en el caso de incumplimiento de una condición asociada a la propia instrucción.
 - Iterativas: repiten, un número finito de veces, una o varias instrucciones.
- Compuestas: son aquellas que el procesador no puede ejecutar directamente, sino que realiza una llamada a un subprograma, subrutina o párrafo.

Variables

Las variables se suelen utilizar en muchos ámbitos diferentes, como, por ejemplo, en matemáticas.

$$x^2 + 2x + 1 = 0$$

Variables

Pero, en programación Una **variable** es donde se guarda en memoria (y se recupera) datos que se utilizan en un programa. Cuando programamos utilizamos variables para:

- **Guardar** *datos y estados*.
- **Asignar** valores de una variable a otra.
- **Representar** valores dentro de una expresión matemática.



- **Mostrar** valores por pantalla.

Todas las variables deben ser de un tipo de datos, ya sea un *dato de tipo primitivo*, como un **número** o **texto**, o un *dato abstracto*, como un **objeto** que se ha creado. En **Python**, no es necesario declarar el tipo de variable. Simplemente asigna un valor y Python inferirá el tipo automáticamente. Se dice que es una variable, porque el valor de esta puede cambiar en cualquier parte de nuestro programa.

Sintaxis:

```
nombreVariable = valor
```

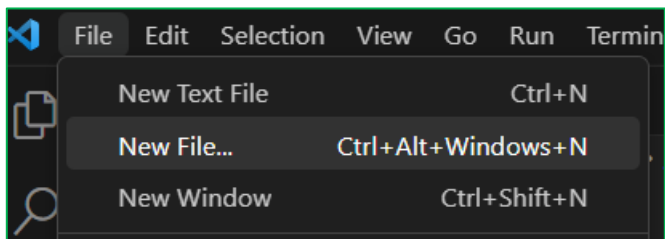
Ejemplos:

```
name = "Arnulfo"
age = 35
average = 4.5
```

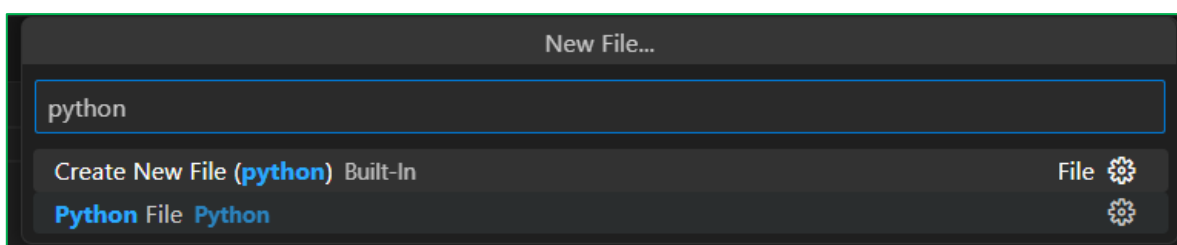
Cada declaración o asignación de una variable debe ir en una línea de código. En el caso de números decimales se usa el punto como separador decimal. Y para cadenas de texto el valor debe estar entre comillas dobles o comillas simples.

Tipo Variable	Descripción	Ejemplo
Entero	almacena números enteros positivos y negativos	x=12, z=12+6, suma=0, resta = -100
Decimal	almacena números decimales positivos y negativos	x=12.45, z=12+6.1, suma=0.0, resta= -80.345
Cadena	almacena una secuencia de letras, espacios, números y otros caracteres	Nombre="Carlos", msg="Hola Mundo", resultado="El nombre es: ". nombre
Booleano	almacena sólo falso o verdadero	Encendido=False, soltero=True

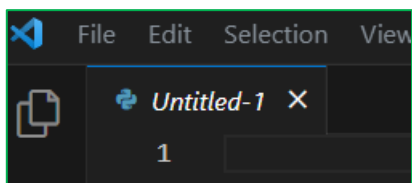
Para iniciar con nuestro ejemplo en Python, abriremos Visual Studio Code. Luego creamos un nuevo archivo, vamos al menú *File / New File*:



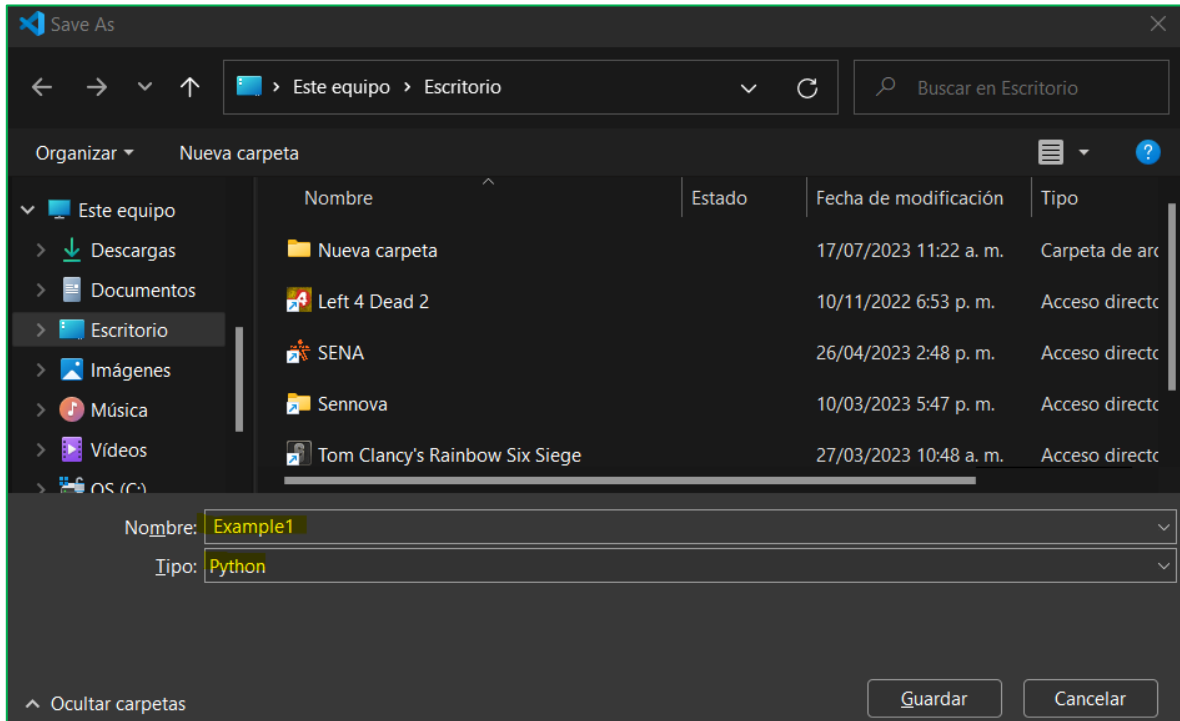
En la ventana emergente, escribimos **Python**. Y luego seleccionamos la opción **Python File**



Vemos que se crea una nueva pestaña:

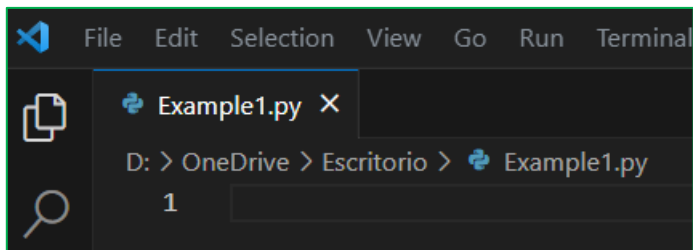


Ahora presionamos las teclas *Control + S* o también *Menú File / Save*. Esto nos muestra la ventana de guardar como:

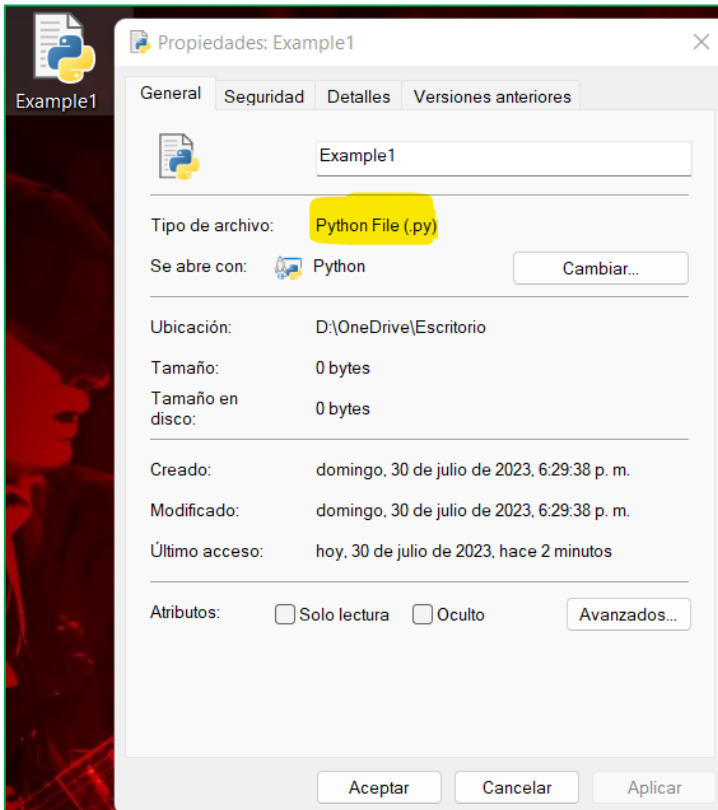


Allí colocaremos como nombre de archivo **Example1** y asegúrate que en el tipo diga **Python**. Damos clic en *Guardar*.

Note que ahora el nombre de la pestaña a cambiado y además se muestra la ruta donde se encuentra el archivo guardado:



Todos los archivos con código fuente en Python tienen la extensión **.py** y se ven así en Windows:



Nota: no necesariamente si das doble clic al archivo este te abrirá en VS Code, por defecto lo que hace es ejecutarlo o abrirlo en el editor de texto que trae Python.

Ahora declaramos 4 variables de diferente tipo de la siguiente forma:

```
Example1.py X
D: > OneDrive > Escritorio > Example1.py > ...
1  name = "Arnulfo Archundia"
2  age = 35
3  average = 4.5
4  married = True
5
```

Nota: guarde cambios constantemente en sus archivos con *Control + S* ya que de no hacerlo no se reflejarán los cambios al ejecutar el programa, o cuando se



trabaje con varios archivos a la vez no se tendrán en cuenta cambios no guardados...o si se va la energía... 

Ahora veamos ejemplos de uso de variables más complejos, pero antes debemos conocer cuáles son los operadores matemáticos básicos en Python.

Operador Aritmético	Símbolo
Suma	+
Resta	-
Multiplicación	*
División	/
División entera	//
Modulo	%
Potencia	**
Asignación	=

Nota: el operador potencia eleva un número (o variable), a un número (o variable). Por ej.; $2**3$, $x**2$, $n**m$. El operador asignación permite darle o asignarle un valor a una variable. Por ej.: $x=2$, $y=2+3$, $z=x+2$, $w=x*y$.


Vamos a nuestro archivo **Example1.py** y digitaremos las siguientes sentencias:

```
6 add = 5 + 10
7 sub = 10 - 8
8 mult = 6 * 9
9 div = 5 / 2
10 div2 = 5 // 2
11 pow = 3 ** 2
```

Note que en cada variable estamos realizando operaciones matemáticas con números constantes, pero ¿qué pasa si ya tengo un número almacenado en una variable que ya existe? Esto nos indica que podemos realizar operaciones matemáticas entre variables ya existentes. Incluyamos las siguientes sentencias:



```
13 add2 = age + 1
14 div3 = div2 + average
15 pow = pow * 2
```

¿Nota algo en la línea 15? ¿Una variable puede asignarse a si misma? 

¿Ahora, como sabemos los resultados de lo que tenemos hasta ahora? Lo que haremos es imprimir en pantalla las variables para ver los resultados de nuestras operaciones.

Imprimir en pantalla

En Python podemos imprimir en la consola o terminal de Windows para el resultado de nuestros programas. Podemos imprimir texto, números, variables o todas las combinaciones posibles. Para ello hacemos uso de la función **print** de Python.

Sintaxis básica:

```
print("texto a imprimir")
```

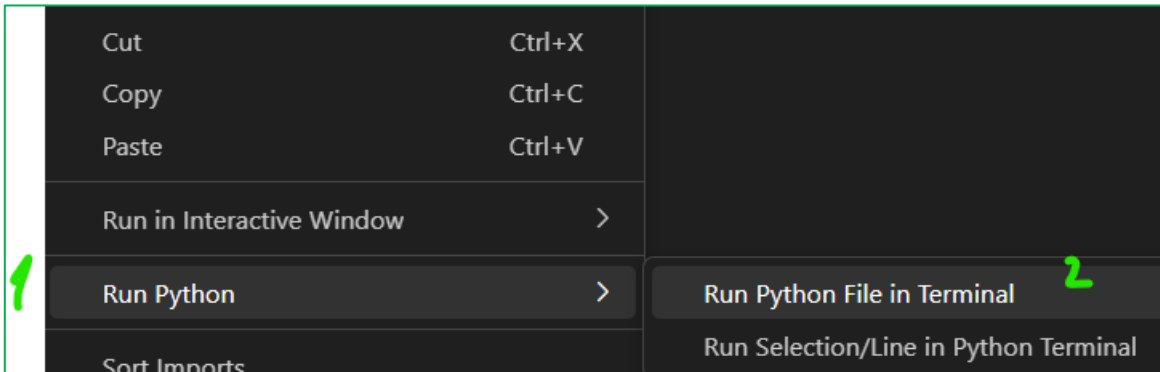
```
print(variable)
```

```
print("texto" . variable)
```

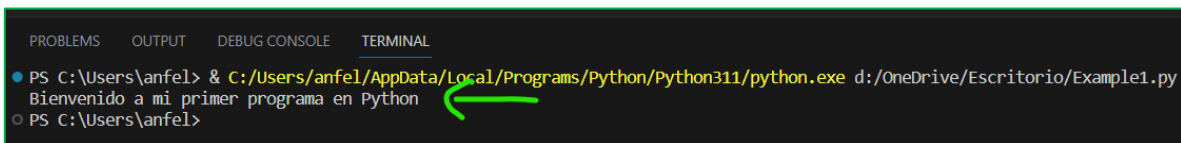
Vamos a **Example1.py** y en la línea 1 de nuestro código vamos a colocar la siguiente línea, colocando debajo lo que ya teníamos:

```
Example1.py X
D: > OneDrive > Escritorio > Example1.py > ...
1 print("Bienvenido a mi primer programa en Python")
2 name = "Arnulfo Archundia"
```

Guardamos cambios, para probar si esto funciona debemos ejecutar el archivo. Para ello damos clic derecho en cualquier parte de nuestro código, seleccionamos *Run Python / Run Python File in Terminal*

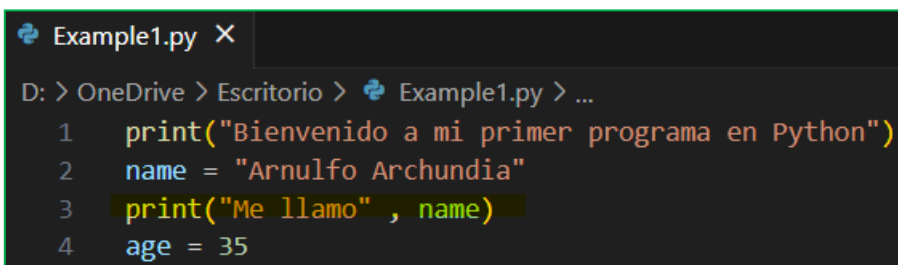


Si todo está bien escrito y sin errores veremos algo como lo siguiente:

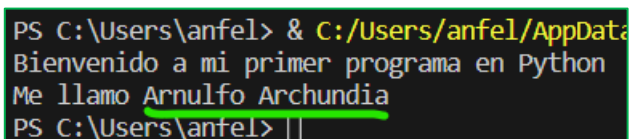


VS Code abre dentro del editor una Terminal de Windows (DOS) y allí nos muestra el resultado de nuestra impresión en pantalla.

Ahora, en la línea 3 introduzca la siguiente sentencia:



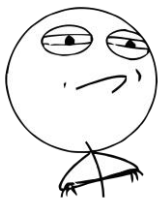
Aquí estamos imprimiendo un texto y le estamos “concatenando” una variable. Es decir, que Python debe imprimir el texto y luego lo que tenga almacenado esa variable en ese momento. Ejecutamos y el resultado debe ser:



Ejercicio:

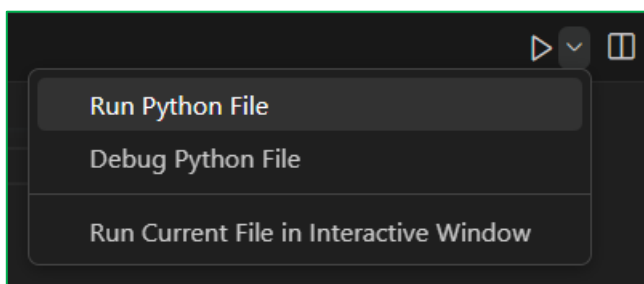


CHALLENGE ACCEPTED



Ej: Al igual que con la variable name, imprima cada valor de cada variable de Example1.py, incluyendo las variables de operaciones matemáticas.

Nota: también se puede ejecutar el archivo con el botón *Run* y luego elegir la opción *Run Python File*:



Más abajo está la solución, pero no la veas hasta que no termines el ejercicio





```
1  print("Bienvenido a mi primer programa en Python")
2  name = "Arnulfo Archundia"
3  print("Me llamo" , name)
4  age = 35
5  print("Edad" , age)
6  average = 4.5
7  print("Promedio" , average)
8  married = True
9  print("Casado" , average)
10
11  add = 5 + 10
12  print("Suma" , add)
13  sub = 10 - 8
14  print("Resta" , sub)
15  mult = 6 * 9
16  print("Multiplicación" , mult)
17  div = 5 / 2
18  print("División" , div)
19  div2 = 5 // 2
20  print("División entera" , div2)
21  pow = 3 ** 2
22  print("Potencia" , pow)
23
24  add2 = age + 1
25  print("Suma con variable" , add2)
26  div3 = div2 + average
27  print("División con variable" , div3)
28  pow = pow * 2
29  print("Potencia reasignada" , add2)
30
```

Resultado de la ejecución:



```
PS C:\Users\anfel> & C:/Users/anfel/AppDat
Bienvenido a mi primer programa en Python
Me llamo Arnulfo Archundia
Edad 35
Promedio 4.5
Casado 4.5
Suma 15
Resta 2
Multiplicación 54
División 2.5
División entera 2
Potencia 9
Suma con variable 36
División con variable 6.5
Potencia reasignada 36
```

Constantes

En el caso de Python, las constantes pueden ser simuladas utilizando variables con nombres en mayúsculas, aunque no son realmente constantes. Una **constante** es un número, un carácter o una cadena de caracteres que se puede utilizar como valor en un programa y estos no pueden modificarse. La sintaxis es la misma que la de las variables.

Por convención, si se deben usar constantes en Python, el nombre de ellas deberá ser escrito en mayúsculas sostenidas. Ejemplo:

```
PI = 3.1416
```

En **Example1.py**, debajo de la última línea escribimos las sentencias:

```
31 PI = 3.14159
32 MAX_NUMBER = 100
33 MIN_NUMBER = 0
```

Las constantes también podemos imprimirlas en pantalla:

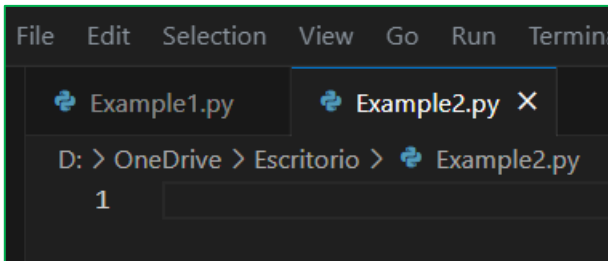
```
print("Constantes: ", PI, MAX_NUMBER, MIN_NUMBER)
```

Analice el resultado en la terminal de este último print.



Lectura de datos

Para esta parte crearemos un nuevo archivo Python llamado **Example2.py**.



Este proceso de lectura de datos permite que el usuario pueda ingresar datos por medio del teclado en un programa Python. Esto genera una interrupción en el programa, esperando por dicha entrada. Dicha entrada de datos se completa cuando el usuario presiona la tecla *Enter*. Para ello, se usa la función *input()* de Python la cual debe ir acompañada de un mensaje para clarificarle al usuario qué es lo que debe digitar.

Sintaxis:

```
Variable = input("mensaje")
```

Ejemplos:

```
edad = input("Digita tu edad")
```

En este caso lo que el usuario digite hasta presionar *Enter* se guardará en la variable *edad*. Note que si no damos contexto con un mensaje el usuario no sabrá qué es lo que debe digitar. Por tanto, realicemos un ejemplo más completo en el archivo **Example2.py**. Digitemos las siguientes sentencias:

```
D: > OneDrive > Escritorio > Example2.py > ...
1  print('Bienvenido a Example2.py')
2
3  name = input('Por favor, digite su nombre: ')
4
5  print('Hola', name, ', Gracias por su tiempo')
```



Note que la variable *name* al momento que guarda lo ingresado por el usuario s está creando en memoria, es decir, aquí en Python, no hay necesidad de primero crear la variable e inicializarla y luego asignar su valor.

Importante: todos los datos ingresados por teclado con la función `input` se almacenan como cadenas de texto o strings. Por tanto, si necesitamos capturar un número del usuario, entero o flotante, y luego realizar operaciones matemáticas con él, debemos convertir el dato capturado de cadena a número.

Para ello hacemos usos de las funciones `int()` y `float()`.

Sintaxis:

```
variable = int(variable)
```

```
variable = float(variable)
```

Vamos a **Example2.py** y digitemos las siguientes sentencias debajo de la última.

```
7  number1 = input('Digita un número entero: ')
8  number2 = input('Digita un número decimal: ')
9
10 number1 = int(number1)
11 number2 = float(number2)
12
13 add = number1 + number2
14 print('La suma es igual a', add)
```

En este ejemplo le pedimos al usuario que digite dos números, uno entero y otro flotante (En la terminal, el número flotante se digita usando el punto como separador decimal). Luego por cada variable que ha capturado estos datos le realizamos la conversión respectiva. Finalmente, sumamos ambas variables e imprimimos el resultado en pantalla.

```
Bienvenido a Example2.py
Por favor, digite su nombre: Arnulfo
Hola Arnulfo , Gracias por su tiempo
Digita un número entero: 5
Digita un número decimal: 2.3
La suma es igual a 7.3
```



Comentarios

Los comentarios son bloques de texto que los programadores escriben en sus programas para documentarlos. Estos comentarios no son leídos ni interpretados por el lenguaje de programación, es decir no los tiene en cuenta cuando se ejecuta el programa. Se deben tener comentarios como buena práctica de programación para documentar el objetivo del programa, el autor, fechas de edición y bloques del programa que necesiten ser aclarados.

Comentarios de una línea

Cuando necesitemos agregar un comentario que solo ocupará una sola línea usamos el símbolo numeral (#) y luego el texto que queramos documentar. Ejemplo:

```
10  # Convertimos los datos ingresados por el usuario
11  number1 = int(number1)
12  number2 = float(number2)
```

Comentarios de bloque o multilinea

Cuando necesitemos agregar comentarios que ocuparán varias líneas usamos tres comillas dobles como inicio del bloque de comentarios y cerramos con otras tres comillas dobles. Ejemplo:

```
1  """
2  Objetivo: programa que muestra el uso de la funcion input
3  y además conversiones de datos
4  Autor: afescobarv
5  Fecha: 01/08/2023
6  """
7
8  print('Bienvenido a Example2.py')
```