

Python

Python es un lenguaje de programación de alto nivel, interpretado, orientado a objetos y de uso generalizado con semántica dinámica, que se utiliza para la programación de propósito general.

Python fue creado por Guido van Rossum, nacido en Haarlem Países Bajos

Todos los usuarios que no sean Linux pueden descargar una copia en:

<https://www.python.org/downloads/>. Buscar version para 64 bits.

Deja las configuraciones predeterminadas que el instalador sugiere por ahora, con una excepción: mira la casilla de verificación denominada **Agregar Python 3.x a PATH y selecciónala**. Esto hará las cosas más fáciles.

Comenzando tu trabajo con Python

Ahora que tienes Python 3 instalado, es hora de verificar si funciona y de hacer el primer uso.

Este será un procedimiento muy simple, pero debería ser suficiente para convencerte de que el entorno de Python es completo y funcional.

Hay muchas formas de utilizar Python, especialmente si vas a ser un desarrollador de Python.

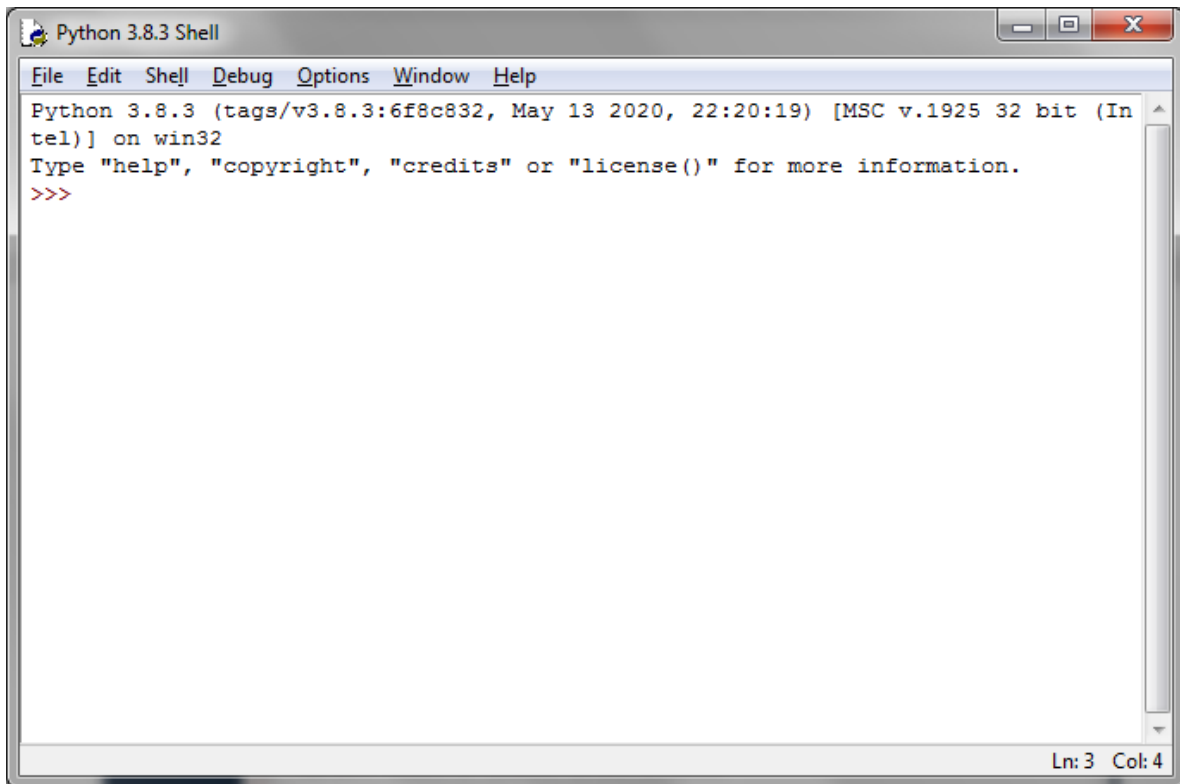
Para comenzar tu trabajo, necesitas las siguientes herramientas:

- Un editor que te ayudará a escribir el código (debes tener algunas características especiales, no disponibles en herramientas simples); este editor dedicado te dará más que el equipo estándar del sistema operativo.
- Una consola en la que puedes iniciar tu código recién escrito y detenerlo por la fuerza cuando se sale de control.
- Una herramienta llamada depurador, capaz de ejecutar tu código paso a paso y te permite inspeccionarlo en cada momento de su ejecución.

Además de sus muchos componentes útiles, la instalación estándar de Python 3 contiene una aplicación muy simple pero extremadamente útil llamada IDLE.

IDLE es un acrónimo de: Integrated Development and Learning Environment (Desarrollo Integrado y Entorno de Aprendizaje).

Navega por los menús de tu sistema operativo, encuentra IDLE en algún lugar debajo de Python 3.x y ejecútalo. Esto es lo que deberías ver:



Como puedes ver, IDLE abre una nueva ventana para ti. Puedes usarla para escribir y modificar tu código.

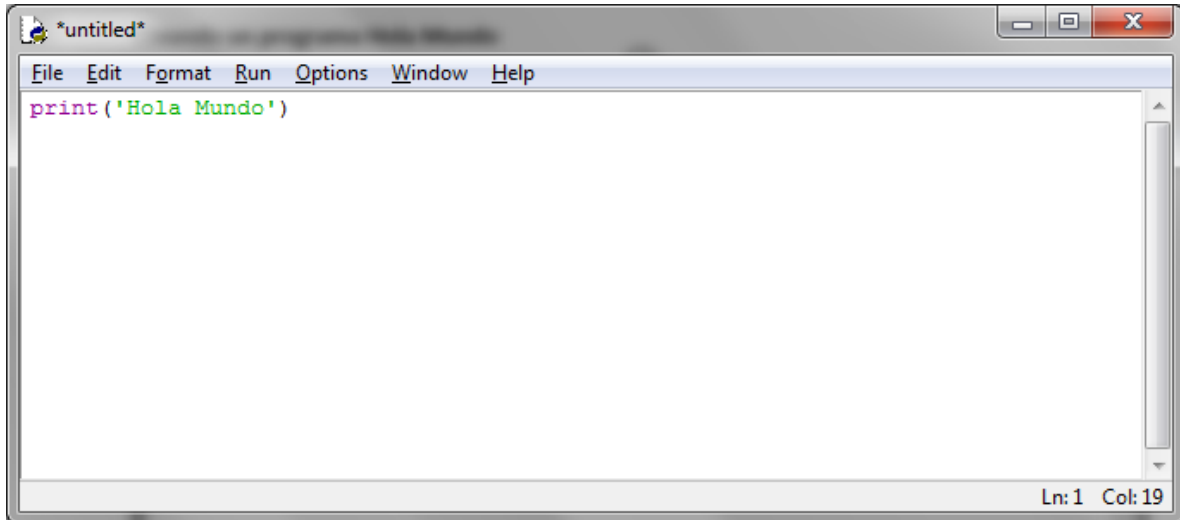
Esta es la ventana del editor. Su único propósito es ser un lugar de trabajo en el que se trate tu código fuente. No confundas la ventana del editor con la ventana de shell. Realizan diferentes funciones.

Creando un programa Hola Mundo

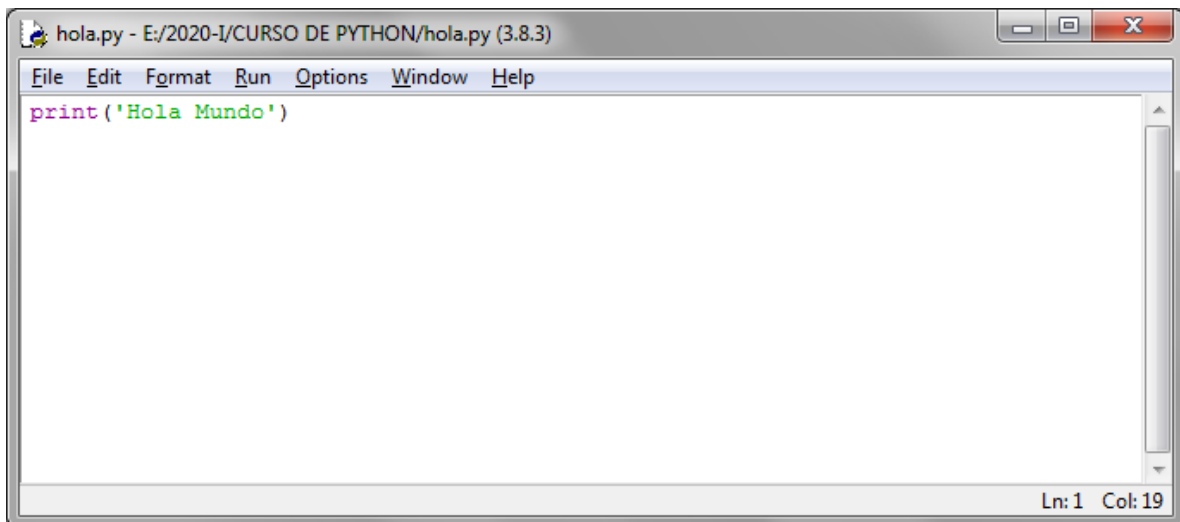
Haz clic en File, luego haz clic en New File

Sale el editor y escribimos

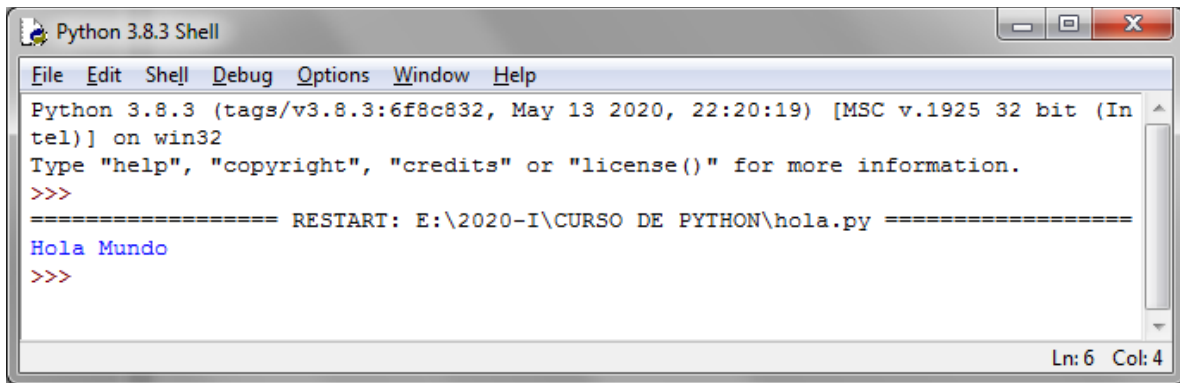
```
print('hola Mundo')
```



Luego grabamos el archivo, por ejemplo con el nombre hola, y esto se grabara con el nombre hola.py



Para ejecutarlo escogemos del Menu la opcion Run. Y vemos el resultado en la Consola



The screenshot shows a 'Python 3.8.3 Shell' window. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The text area contains the following content: 'Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32', 'Type "help", "copyright", "credits" or "license()" for more information.', and a prompt '>>>'. Below the prompt is a line of text: '===== RESTART: E:\2020-I\CURSO DE PYTHON\hola.py ====='. This is followed by the output 'Hola Mundo' in blue text, and another prompt '>>>'. The status bar at the bottom right indicates 'Ln: 6 Col: 4'.

La sintaxis de Python es bastante específica requiere que no haya más de una instrucción por una línea.

Una línea puede estar vacía (por ejemplo, puede no contener ninguna instrucción) pero no debe contener dos, tres o más instrucciones. Esto está estrictamente prohibido.

La funcion print

La función toma los argumentos (puede aceptar más de un argumento o ninguno) los convierte en un formato legible para el ser humano si es necesario y envía los datos resultantes al dispositivo de salida (generalmente la consola); en otras palabras, cualquier cosa que se ponga en la función de print() aparecerá en la pantalla.

Ejemplos

- Si deseamos mostrar un mensaje escribiremos el mensaje entre comillas dobles o simples:
`print("Pytnon es un programa muy facil de aprender")`
tambien podria ser:
`print('Pytnon es un programa muy facil de aprender')`

- Cada vez que se utiliza la funcion print muestra en la consola y se va a la siguiente linea.

```
print('El Lenguaje de Programacion Python')  
print('fue creado por:')  
print('Guido van Rossum')
```

Se vera en la Consola:

```
El Lenguaje de Programacion Python  
fue creado por:  
Guido van Rossum
```

- Si usamos la funcion print sin argumentos : `print()` genera una linea vacia.

- Se pueden utilizar caracteres de escape en la función print.

La barra invertida (\) tiene un significado muy especial cuando se usa dentro de las cadenas, es llamado el carácter de escape.

La palabra escape debe entenderse claramente- significa que la serie de caracteres en la cadena se escapa (detiene) por un momento (un momento muy corto) para introducir una inclusión especial.

En otras palabras, la barra invertida no significa nada, sino que es solo un tipo de anuncio, de que el siguiente carácter después de la barra invertida también tiene un significado diferente.

La letra n colocada después de la barra invertida proviene de la palabra newline (nueva línea).

Por ejemplo:

```
print('Trujillo\nCapital de la\nEterna Primavera')
```

se vera en la Consola:

```
Trujillo
Capital de la
Eterna Primavera
```

- Se pueden colocar varios argumentos a la función print, estos van separados por comas

```
print("Mi nombre es", "Juan Perez", "estudio", "Ingenieria de Sistemas")
```

se vera en la Consola:

```
Mi nombre es Juan Perez estudio Ingenieria de Sistemas
```

Al mostrar python le coloca un espacio entre cada argumento

- Se puede pasar argumentos en la función print para cambiar su comportamiento a la hora de mostrar los resultados. Estos son los argumentos con palabra clave end y sep
Cualquier argumento de palabra clave debe ponerse después del último argumento (esto es muy importante).

Si escribimos el siguiente código:

```
print("Mi nombre es", "Bond.", end=" ")
```

```
print("James Bond.")
```

se vera en la Consola

```
Mi nombre es Bond. James Bond.
```

Como puedes ver, el argumento de palabra clave end determina los caracteres que la función

print() envía a la salida una vez que llega al final de sus argumentos posicionales.

comportamiento predeterminado refleja la situación en la que el argumento de la palabra clave end se usa implícitamente de la siguiente manera: end="\n".

- Se estableció anteriormente que la función print() separa los argumentos generados con espacios. Este comportamiento también puede ser cambiado

El argumento de palabra clave que puede hacer esto se denomina **sep** (como separador).

```
print("Mi nombre es", "Juan Perez", "estudio", "Ingenieria de Sistemas", sep="-")
```

```
Mi nombre es-Juan Perez-estudio-Ingenieria de Sistemas
```

La función print() ahora utiliza un guión, en lugar de un espacio, para separar los argumentos generados.

- Los argumentos de palabras clave end y sep se pueden mezclar en una invocación a la función print. Por ejemplo:

```
print("Mi", "nombre", "es", sep="_", end="*")  
print("James", "Bond.", sep="*", end="*\n")
```

mostrara en la Consola:

Mi_nombre_es*James*Bond.

Literales

Un literal se refiere a datos cuyos valores están determinados por el literal mismo

Enteros

Son aquellos que no tienen parte fraccionaria.

Por ejemplo: 123, -12, 18, 378 son literales enteros.

La siguiente instrucción

```
print('La edad de Luis es: ',19)
```

El resultado en la consola sera

La edad de Luis es: 19

Enteros Octales

Si un número entero está precedido por un código 0O o 0o (cero-o), el número será tratado como un valor octal. Esto significa que el número debe contener dígitos en el rango del [0..7] únicamente.

0o123 es un número octal con un valor (decimal) igual a 83

Por ejemplo si colocamos

```
print(0o123)
```

El resultado es: 83. La función print() realiza la conversión automáticamente.

Enteros Hexadecimales

Los números en hexadecimal deben ser precedidos por el prefijo 0x o 0X (cero-x)

0x123 es un número hexadecimal con un valor (decimal) igual a 291

Por ejemplo si colocamos:

```
print(0x123)
```

el resultado es: 291

Reales(Flotantes)

Son números que tienen (o pueden tener) una parte fraccionaria después del punto decimal.

Ejemplo: 2.5, -0.4, .12, 6.

Enteros vs Reales (flotantes)

4 es entero

4.0 es un numero real

Se puede pensar que son idénticos, pero Python los ve de una manera completamente distinta.

4 es un número entero, mientras que 4.0 es un número punto-flotante .

El punto decimal es lo que determina si es flotante.

Reales – Notacion cientifica

Si queremos mostrar 3×10^8 se puede colocar 3E8

La letra E (también se puede utilizar la letra minúscula e - proviene de la palabra exponente) la cual significa por diez a la n potencia.

Por ejemplo la constante de planck 6.62607×10^{-34} se puede escribir : 6.62607E-34

El numero : 1×10^{-22} se puede escribir: 1E-22

Cadenas

Las cadenas se emplean cuando se requiere procesar texto (como nombres de cualquier tipo, direcciones, novelas, etc.), no números.

Las cadenas requieren comillas simples o dobles.

Ejemplos: 'Python', "Trujillo", "Emiratos Arabes Unidos",

Si quisieramos mostrar: Me gusta, "Jugar Futbol"

La primera solucion seria:

```
print("Me gusta \"Jugar Futbol\"")
```

Una comilla precedida por una diagonal invertida cambia su significado, no es un limitador, simplemente es una comilla.

La segunda solucion seria:

```
print('Me gusta "Monty Python"')
```

Delimitamos la cadena con comillas simples

¿Cómo se puede insertar un apóstrofe en una cadena la cual está limitada por dos apóstrofes?

Intenta imprimir una cadena que contenga el siguiente mensaje:

I'm Monty Python

```
print('I\'m Monty Python.')  
O  
print("I'm Monty Python.")
```

Cadenas Vacía

Se puede representar de la siguiente manera:

```
' '
```

```
" "
```

Valores Booleanos

Tienen 2 valores : True y False (Verdadero y falso)

Expresiones

Una expresión es una combinación de valores (o variables, operadores, llamadas a funciones) las cuales son evaluadas y dan como resultado un valor, por ejemplo, 1+2

Operadores

Los operadores son símbolos especiales o palabras clave que son capaces de operar en los valores y realizar operaciones matemáticas, por ejemplo, el * multiplica dos valores: x*y.

Operadores Aritméticos

	Operador
+	Suma
-	Resta
*	Multiplicacion
/	Division
**	Exponenciacion
//	Division Entera
%	Modulo (residuo de la division de 2 numeros)

El operador Suma: +

El símbolo del operador de suma es el + (signo de más), el cual esta completamente alineado a los estándares matemáticos.

De nuevo, observa el siguiente fragmento de código:

```
print(-4 + 4)
```

```
print(-4. + 8)
```

El resultado seria

```
0          entero
```

```
4.0       real
```

También hay un operador + unario. Se puede utilizar de la siguiente manera:

```
print(+2)
```

El operador conserva el signo de su único argumento, el de la derecha.

Aunque dicha construcción es sintácticamente correcta, utilizarla no tiene mucho sentido, y sería difícil encontrar una buena razón para hacerlo

El operador resta: -

El símbolo del operador de resta es el - (signo de menos), el cual esta completamente alineado a los estándares matemáticos. Sirve para restar 2 numeros o para cambiar el signo de un numero.

Esta es una gran oportunidad para mencionar una distinción muy importante entre operadores unarios y binarios.

En aplicaciones de resta, el operador de resta espera dos argumentos: el izquierdo (un minuendo en términos aritméticos) y el derecho (un sustraendo).

Por esta razón, el operador de resta es considerado uno de los operadores binarios, así como los demás operadores de suma, multiplicación y división.

Pero el operador negativo puede ser utilizado de una forma diferente:

```
print(-4 - 4)
```

```
print(4. - 8)
```

```
print(-1.1)
```

El resultado es:

```
-8
```

```
-4.0
```

```
-1.1
```

Operador Multiplicacion: *

Sirve para multiplicar 2 numeros

```
print(2 * 4)
print(5.0 * -8)
print(3.5 * 7)
8
-40.0
24.5
```

Operador Division : /

Un símbolo de / (diagonal) es un operador de división.

El valor después de la diagonal es el dividendo, el valor antes de la diagonal es el divisor.

Ejecuta el código y analiza los resultados.

```
print(6 / 3)
print(6 / 3.)
print(6. / 3)
print(6. / 3.)
2.0
2.0
2.0
2.0
```

El resultado producido por el operador de división siempre es **flotante**.

Operador Division Entera : //

El resultado carece de la parte fraccionaria, está ausente (para los enteros), o siempre es igual a cero (para los flotantes); esto significa que los resultados siempre son redondeados.

Se ajusta a la regla entero vs flotante.

```
print(6 // 3)
print(6 // 3.)
print(6. // 3)
print(6. // 3.)
2
2.0
2.0
2.0
```

Como se puede observar, una división de entero entre entero da un resultado entero. Todos los demás casos producen flotantes.

```
print(6 // 4)
```

```
print(6. // 4)
```

1

1.0

Lo que se obtiene son dos unos, uno entero y uno flotante.

El resultado de la división entera siempre se redondea al valor entero inferior mas cercano del resultado de la división no redondeada.

```
print(-6 // 4)
```

```
print(6. // -4)
```

-2

-2.0

Operador residuo (modulo) : %

El resultado de la operación es el **residuo que queda de la división entera**.

```
print(14 % 4)
```

2

Como puedes observar, el resultado es dos. Esta es la razón:

- $14 // 4$ da como resultado un 3 → esta es la parte entera, es decir el cociente.
- $3 * 4$ da como resultado 12 → como resultado de la multiplicación entre el cociente y el divisor.
- $14 - 12$ da como resultado 2 → este es el residuo.

```
print(12 % 4.5)
```

3.0

$12 // 4.5$ da 2.0

$2.0 * 4.5$ da 9.0

$12 - 9.0$ da 3.0

Operador exponenciación : **

Un signo de ** (doble asterisco) es un operador de exponenciación (potencia). El argumento a la izquierda es la base, el de la derecha, el exponente.

Las matemáticas clásicas prefieren una notación con superíndices, como el siguiente: 2^3 . Los editores de texto puros no aceptan esa notación, por lo tanto Python utiliza ** en lugar de la notación matemática, por ejemplo, `2 ** 3`.

Cuando ambos ** argumentos son enteros, el resultado es entero también.

Cuando al menos un ** argumento es flotante, el resultado también es flotante.

Esta es una distinción importante que se debe recordar.

```
print(2**3)
```

```
print(3.5**2)
```

```
print(16**0.5)
```

8

12.25

4.0

Prioridad de los Operadores Aritmeticos

Prioridad	Operador	Tipo
1	+, -	Unario
2	**	
3	*, /, %, //	
4	+, -	Binario

Operadores y paréntesis

Por supuesto, se permite hacer uso de paréntesis, lo cual cambiará el orden natural del cálculo de la operación.

De acuerdo con las reglas aritméticas, las sub-expresiones dentro de los paréntesis siempre se calculan primero.

Se pueden emplear tantos paréntesis como se necesiten, y seguido son utilizados para mejorar la legibilidad de una expresión,

La asociatividad de un operador binario define cómo se operan tres o más operandos. La asociatividad por la izquierda quiere decir que si aparecen tres o más operandos, primero se evalúan los dos operandos de más a la izquierda, y el resultado de esa operación se opera con el siguiente operando por la izquierda, y así sucesivamente. En la asociatividad por la derecha, los operandos se evalúan de derecha a izquierda.

Los operadores `+`, `-`, `*`, `/`, `//`, `%` son asociativos por la izquierda. El operador `**` es asociativo por la derecha.

Ejemplo:

```
- print(2 ** 2 ** 3)
  Primero se ejecuta 2**3 = 8 luego 2**8 = 256
- print(9 % 6 % 2)
  Primero se ejecuta 9%6 = 3 y luego 3%2 = 1
- print(2+3*5)
  Primero se ejecuta 3*5 = 15 y luego 2+15 =17
- print((-2/4), (2/4), (2//4), (-2//4))
  -0.5 0.5 0 -1
```

Variables

Posiciones de memoria donde se guardan un valor el cual puede ir cambiando durante la ejecución de un programa.

En python no se declaran las variables, se van utilizando cuando se guardan valores en ellas.

Como nombrar a una Variable

Si se desea nombrar una variable, se deben seguir las siguientes reglas:

El nombre de la variable debe de estar compuesto por letras MAYUSCULAS, minúsculas, dígitos, y el carácter `_` (guion bajo). El nombre de la variable debe comenzar con una letra. El carácter guion bajo es considerado una letra.

Se diferencian las mayusculas de las minusculas. Por ejemplo: Area es distinto de area y distinto de AREA.

El nombre de las variables no puede ser igual a algunas de las palabras reservadas de python

Python no impone restricciones en la longitud de los nombres de las variables.

Además, Python permite utilizar no solo las letras latinas, sino caracteres específicos de otros idiomas que utilizan otros alfabetos.

Ejemplo de nombres de variables correctos:

Longitud, x, año, alfa_123, _importe

Ejemplo de nombres de variables incorrectos

1er_lado → comienza con un número

Area del triángulo → contiene espacios en blanco

Palabras Reservadas

Observa las palabras que juegan un papel muy importante en cada programa de Python.

```
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

Operador de Asignación (=)

Una variable se crea cuando se le asigna un valor. A diferencia de otros lenguajes de programación, no es necesario declararla.

La creación (o su sintaxis) es muy simple: solo utiliza el nombre de la variable deseada, después el signo de igual (=) y el valor que se desea colocar dentro de la variable

Variable = valor

Asigna valor a la variable.

La asignación es de izquierda a derecha

Ejemplo:

A = 10

Asignamos el valor 10 a la variable A

nombre="Jorge"

Asignamos la cadena Jorge a la variable nombre

éxito = False

Asignamos False a la variable éxito

print(A,nombre,éxito)

10 Jorge False

No se pueden utilizar variables que no existan

Se puede utilizar print() para combinar texto con variables utilizando el operador + para mostrar cadenas con variables, por ejemplo

var = "3.7.1"

print("Versión de Python: " + var)

Versión de Python: 3.7.1

Trabajando con variables

```
x = 2
x = x + 5
x = x*10
print(x)
70
```

Si tenemos
A = 20
B = 25
No es lo mismo colocar
A=B que B=A
A= B Asigna el valor de B a la variable A → A= 25 y B=25
B= A Asigna el valor de A a la variable B → B=20, A=20

Se pueden asignar varios valores en una línea

```
x, y, z = 5, 10, 8
print(x,y,z)
```

Operadores Abreviados

Si op es un operador de dos argumentos (esta es una condición muy importante) y el operador es utilizado en el siguiente contexto:

variable = variable op expresión

Puede ser simplificado de la siguiente manera:

variable op= expresión

Ejemplos:

$i = i + 2 * j \Rightarrow i += 2 * j$

$var = var / 2 \Rightarrow var /= 2$

$rem = rem \% 10 \Rightarrow rem \% = 10$

$j = j - (i + var + rem) \Rightarrow j -= (i + var + rem)$

$x = x ** 2 \Rightarrow x ** = 2$

Comentarios en un programa

Un texto insertado en el programa el cual es, omitido en la ejecución, es denominado un comentario.

En Python, un comentario es un texto que comienza con el símbolo # y se extiende hasta el final de la línea.

Ejemplo:

```
# Esta programa calcula la hipotenusa (c)
# a y b son las longitudes de los catetos
a = 3.0
b = 4.0
c = (a ** 2 + b ** 2) ** 0.5 # se utiliza ** en lugar de la raíz cuadrada
print("c =", c)
```

La función input() sin argumentos

Sirve para leer datos desde el teclado.

La función input() es capaz de leer datos que fueron introducidos por el usuario y pasar esos datos al programa en ejecución.

El resultado de la función input es una cadena.

Se puede usar sin argumentos.

Ejemplo :

```
print('Ingrese nombre:')
nombre=input()
print('Ingrese numero:')
num = input()
print('nombre:', nombre,'numero :',num)
nombre: juan numero : 12
```

En el programa nombre y num serán variables tipo cadena.

La funcion input() con argumento

La función input() puede hacer algo más: puede mostrar un mensaje al usuario sin la ayuda de la función print().

El programa anterior lo podemos escribir así:

```
nombre=input('Ingrese nombre:')  
num = input('Ingrese numero:')  
print('nombre:', nombre,'numero :',num)
```

Ingrese nombre:luis

Ingrese numero:12

nombre: luis numero : 12

Conversión de datos o casting

Al ingresar si queremos convertir la cadena a un entero o la cadena a un real. Usaremos las funciones int() y float()

- La función int() toma un argumento (por ejemplo, una cadena: int(string)) e intenta convertirlo a un valor entero; si llegase a fallar, el programa entero fallará también (existe una manera de solucionar esto, se explicará mas adelante).
- La función float() toma un argumento (por ejemplo, una cadena: float(string)) e intenta convertirlo a flotante (el resto es lo mismo).

Si queremos ingresar el nombre, edad y peso de una Persona.

```
nombre=input('Ingrese nombre:')  
edad = int(input('Ingrese edad:'))  
peso = float(input('Ingrese peso:'))  
print('nombre:', nombre,'edad:',edad,'peso:',peso)
```

Al ejecutar el programa veremos:

Ingrese nombre:Luis

Ingrese edad:24

Ingrese peso:75

nombre: Luis edad: 24 peso: 75.0

Operadores de Cadena

Concatenación

El signo de + (más), al ser aplicado a dos cadenas, se convierte en un operador de concatenación:

`string + string`

Simplemente concatena (junta) dos cadenas en una. Además, puede ser utilizado más de una vez en una misma expresión.

No olvides, si se desea que el signo + sea un concatenador, no un sumador, solo se debe asegurar que ambos argumentos sean cadenas.

No se pueden mezclar los tipos de datos aquí

Replicación

El signo de * (asterisco), cuando es aplicado a una cadena y a un número (o a un número y cadena) se convierte en un operador de replicación.

`cadena * número`

`número * cadena`

Replica la cadena el numero de veces indicado por el número.

Por ejemplo:

- `"James" * 3` nos da `"JamesJamesJames"`.
- `3 * "an"` nos da `"ananan"`.
- `5 * "2"` (o `"2" * 5`) da como resultado `"22222"` (no 10).

Conversión de tipos de datos: `str()`

También se puede convertir un número a una cadena, lo cual es más fácil y rápido, esta operación es posible hacerla siempre.

Una función capaz de hacer esto se llama `str()`:

Ejemplo

```
a = float(input("Ingresa la longitud del primer cateto: "))
```

```
b = float(input("Ingresa la longitud del segundo cateto: "))
```

```
print("La longitud de la hipotenusa es: " + str((a**2 + b**2) **.5))
```

Como ven gracias a la función `str` podemos pasar el resultado entero a la función `print()` como una sola cadena, sin utilizar las comas.

Ejercicios

1. Calcular el perímetro, el área y la diagonal de un rectángulo si se ingresan el valor de sus lados.

Perímetro = $2 * (\text{lado1} + \text{lado2})$

Área = $\text{lado1} * \text{lado2}$

Diagonal = $(\text{lado1}^2 + \text{lado2}^2)^{1/2}$

Programa para ingresar los valores de los lados de un rectangulo

y reportar el area, el perimetro y la diagonal

```
lado1= float(input('Valor del primer lado:'))
```

```
lado2=float(input('Valor del segundo lado:'))
```

```
area=lado1*lado2
```

```
perimetro=2*(lado1+lado2)
```

```
diagonal = (lado1**2+lado2**2)**(1/2)
```

```
print('El area es : ',area)
```

```
print('El perimetro es: ',perimetro)
```

```
print('La diagonal es: ',diagonal)
```

Resultado:

Valor del primer lado:3

Valor del segundo lado:4

El area es : 12.0

El perimetro es: 14.0

La diagonal es: 5.0

2. Calcular el salario neto de un trabajador. Se debe leer el nombre, horas trabajadas, precio de la hora y sabiendo que los impuestos aplicados son el 10 por ciento sobre el salario bruto.

$sb = ph * ht$

$sn = sb - 0.10 * sb$

Programa para ingresar los valores de los lados de un rectangulo

y reportar el area, el perimetro y la diagonal

```
lado1= float(input('Valor del primer lado:'))
```

```
lado2=float(input('Valor del segundo lado:'))
```

```
area=lado1*lado2
```

```
perimetro=2*(lado1+lado2)
```

```
diagonal = (lado1**2+lado2**2)**(1/2)
```

```
print('El area es : ',area)
```

```
print('El perimetro es: ',perimetro)
```

```
print('La diagonal es: ',diagonal)
```

Salida

Ingrese nombre :Carlos

Numero de horas trabajadas :20

Precio de la hora :50

El salario neto del trabajador Carlos es : 900.0

3. Leer un número entero que represente días y determinar a cuanto equivale en años, meses y días (Asumir que todos los años tienen 360 días y que todos los meses tienen 30 días).

Ejemplo: Si se ingresa 4600 días el programa debería reportar: 12 años, 9 meses, 10 días.

```
# 3. Leer un número entero que represente días y determinar a cuanto
# equivale en años, meses y días (Asumir que todos los años tienen
# 360 días y que todos los meses tienen 30 días).
# Ejemplo: Si se ingresa 4600 días el programa debería reportar:
# 12 años, 9 meses, 10 días
```

```
dias = int(input('Cantidad de dias : '))
años = dias // 360
dias = dias % 360
meses = dias // 30
dias = dias % 30
print('Equivalen a :',años,'años',meses,'meses',dias,'dias')
```

Cantidad de dias : 4600

Equivalen a : 12 años 9 meses 10 dias

4. Dado un número natural de 4 cifras, hacer un programa que determine la suma y el producto de las cifras del número.

```
numero = int(input('Ingrese numero entero de 4 cifras: '))
u = numero % 10
numero = numero // 10
d = numero % 10
numero = numero // 10
c = numero % 10
um = numero // 10
suma = u + d + c + um
producto = u*d*c*um
print('La suma de digitos es: ',suma)
print('El producto de digitos es: ',producto)
```

Salida:

Ingrese numero entero de 4 cifras: 1234

La suma de digitos es: 10

El producto de digitos es: 24

5. Programa para intercambiar el valor de 2 variables numericas.

```
num1 = float(input('Primer valor : '))  
num2 =float(input('Segundo valor : '))  
num1,num2 = num2,num1  
print('num1 = ',num1,'num2=',num2)
```

resultado

Primer valor : 6

Segundo valor : 10

num1 = 10.0 num2= 6.0

Ejercicios Propuestos

- 1) Hacer un programa para que se ingresen 2 números y reporte su suma, resta y multiplicación.
- 2) Calcular la altura que cae un objeto. Se debe ingresar el tiempo recorrido en segundos.
$$H = 0.5 * G * T^2$$
$$G = 9.8 \text{ m/seg}^2$$
- 3) La gaseosa en la planta embotelladora se almacena en tanques cilíndricos de un radio de 2 metros. Se necesita un programa que ingresando la altura hasta la que llega la gaseosa, calcule el volumen que se tiene.
$$(\text{Volumen del cilindro} = \text{Pi} * \text{radio}^2 * \text{altura})$$
- 4) Una empresa paga a sus vendedores un sueldo básico mensual de S/.300. El sueldo bruto es igual al sueldo básico más una comisión, que es igual al 9% del monto total vendido. Por ley, todo vendedor se somete a un descuento del 11 %. Diseñe un programa que calcule la comisión, el sueldo bruto, el descuento y el sueldo neto de un vendedor de la empresa. Se debe ingresar el monto total vendido.
- 5) La distancia entre dos puntos (x1, y1) y (x2, y2) de un plano se puede obtener sacando la raíz cuadrada de la expresión $(x2 - x1)^2 + (y2 - y1)^2$. Escribir un programa que, dados dos puntos por el usuario, calcule la distancia entre estos dos puntos.
- 6) Dada una cantidad de dinero en soles, diseñe un programa que descomponga dicha cantidad en billetes de S/. 100, S/. 50, S/.10 y monedas de S/. 5, S/. 2 y S/.1. Así, por ejemplo, S/. 3778 puede descomponerse en 37 billetes de S/. 100, más 1 billete de S/. 50, más 2 billetes de S/. 10, más 1 moneda de S/. 5, más 1 moneda de S/.2 y más 1 moneda de S/. 1.
- 7) Dado un número natural de cuatro cifras, diseñe un algoritmo que forme un número con la cifra de los millares y la cifra de las unidades, en ese orden. Así, por ejemplo, si se ingresara el número 8235, el número formado sería 85.
- 8) Dado un número natural de tres cifras, diseñe un algoritmo que permita obtener el revés del número. Así, si se ingresa el número 238 el revés del número es 832.
- 9) Construya una aplicación que determine cuanto se le debe cobrar a los clientes de un grifo, dado que los surtidores registran lo que venden en galones, pero el precio de la gasolina está fijado en litros. Cada galón tiene 3.785 litros. El precio del litro es S/ 3.86.
- 10) Calcular el interés generado por un capital depositado durante cierta cantidad de períodos a una tasa de interés determinada y expresada en porcentaje.
Aplicar las siguientes fórmulas:
 - i. $\text{Monto} = \text{Capital} * (1 + \text{tasa}/100)^{\text{Número Períodos}}$
 - ii. $\text{Interés} = \text{Monto} - \text{Capital}$