

Variables

Variables en Python

En Python, una **variable** es un nombre que se asocia a un valor, permitiendo almacenar datos para su uso posterior. Cuando creamos una variable, en esencia estamos asignando un nombre simbólico a un valor específico, de modo que podemos reutilizarlo a lo largo de nuestro código.

Asignación de variables

La **asignación** de variables es el proceso mediante el cual vinculamos un valor a un nombre. Esto nos permite evitar escribir el mismo valor repetidamente, haciendo que el código sea más legible y fácil de mantener.

Por ejemplo, si queremos almacenar el número 5 en una variable llamada `num`, lo hacemos así:

```
num = 5
```

En Python, las **variables** no tienen un tipo explícito cuando se declaran, pero los valores que almacenan sí tienen un tipo de dato.

A continuación se presenta una lista de los principales tipos de variables (o tipos de datos) en Python:

Flotantes

Un **float** es un tipo de variable numérica que contiene números decimales. Fíjate que los decimales se escriben con un punto decimal.

```
altura = 1.75
```

Cadenas

Las cadenas de caracteres son conjuntos de letras, que normalmente suelen asociarse a una palabra o una frase:

```
nombre = "Juan"
```

Las **cadenas**, también llamadas **strings**, se pueden escribir entre `'` o entre `"""`. Si no pones comillas antes y después, Python creará que se trata de una variable, y no un texto.

Booleanos

Un booleano es un tipo de variable que solo puede tener dos valores: verdadero (`True`) o falso (`False`).

Un ejemplo de variable booleano es:

```
es_mayor_de_edad = True
tienecarnet = False
```

¿Cómo podemos llamar a nuestras variables?

Existen ciertas reglas a la hora de poner un nombre a una variable.

La primera es que no podemos usar palabras del propio lenguaje de programación (print, input... no son nombres de variables permitidos). Se llaman palabras reservadas.

Por tanto, esto es incorrecto:

```
print = 4
```

El nombre debe consistir en una sola palabra. No puede haber espacios. Esto sería incorrecto:

```
edad usuario = 10
```

En su lugar, mejor usar guiones o camelcase:

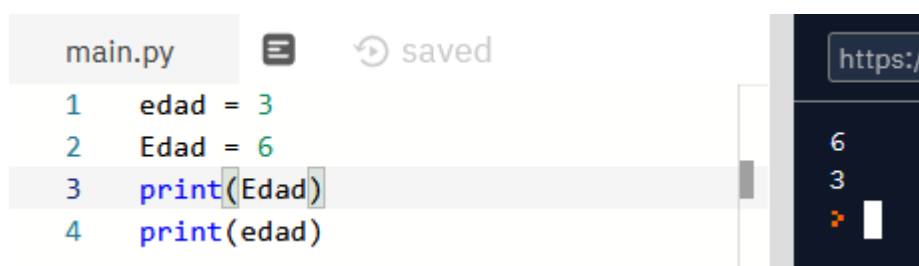
```
edad_usuario = 10
edadUsuario = 20
```

Además de estas reglas, sólo pueden contener letras, números y el guion bajo. Tampoco pueden empezar con un número


Mayúsculas y minúsculas

Aunque se pueden usar mayúsculas, y Python es muy cuidadoso con eso, entendiendo que Nombre, NOMBRE y nombre son variables DISTINTAS, lo mejor que podemos hacer es no liarnos siguiendo ciertas recomendaciones.

Fijaos en el ejemplo:



También es típico empezar con minúsculas, como hicimos en nombre. Y usar alguno de estos dos estilos para separar palabras:

```
main.py  saved  
1 numeroCarnet = '43156667X'  
2 nombre_Hijo = 'Calamardo'
```

Consejos

Los acentos suelen dar problemas por lo que mejor evitarlo.

Para la ñ hay quien para no escribir año usa anio , o bien, anyo .

En programas elaborados es muy conveniente elegir nombres de variables que nos den una pista de qué están representando

Ejemplo nombre y edad

Vamos a hacer un programa que pida al usuario su nombre y año de nacimiento para decirle al final:

Hola, Paco, tienes 45 años.

Para pedirle el nombre y el año de nacimiento usamos la misma estructura que antes. Una línea con un PRINT para que sepan lo que queremos y otra línea con un INPUT para recoger lo que nos han dicho.

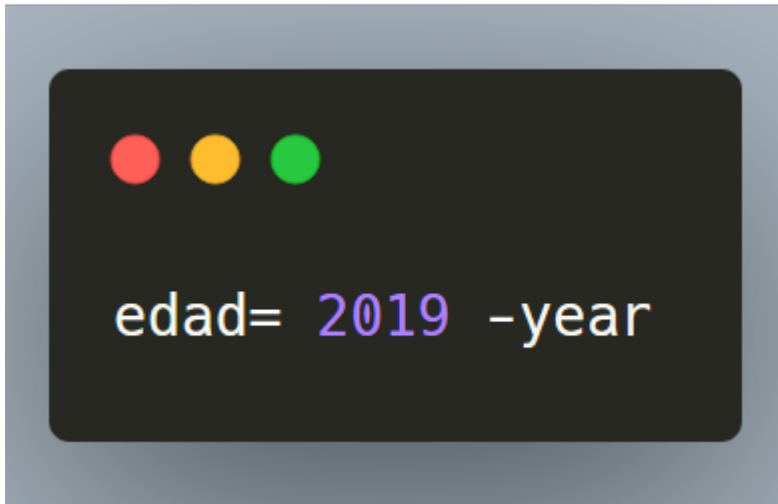
Todas las líneas que pongo aquí las tienes que escribir unas debajo de las otras en el mismo ejercicio. Va todo junto.

```
print('¿Cómo te llamas')  
nombre = input()  
print('¿En qué año naciste?')  
year = input()
```

Estupendo. Ahora cómo calculamos la edad. Bueno, pues sabemos que la edad será el año actual menos el año de nacimiento.

Pero fíjate, esa resta nos da un valor, que es justo lo que queremos, pero que...

```
edad = 2022 - anyo
```

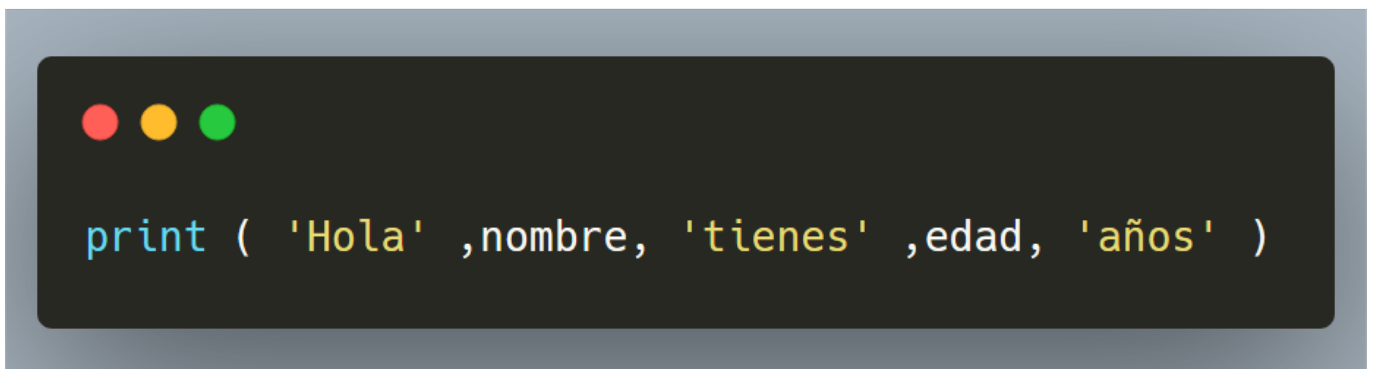


Cálculos con variables

El siguiente paso será calcular nuestra edad.

Con esto creamos la variable edad y le ASIGNAMOS el valor de la resta entre el año 2019 y el valor que tenga la variable year , que nos acaba de dar el usuario, y QUE SERÁ DISTINTO EN CADA EJECUCIÓN.

Pintar información por pantalla



Escribe y ejecuta el código anterior junto y adjunta una captura de pantalla. Te dará error, pero es normal. Luego lo arreglaremos

Arreglar errores

Al ejecutar el programa veremos que nos aparece en rojo un error. LOS ERRORES SON NORMALES. Leamos el mensaje de error.

```
Traceback (most recent call last):  
File "main.py", line 8, in <module>  
edad=2019-year  
TypeError: unsupported operand type(s) for -: 'int' and 'str'
```

La línea 8 es donde yo tengo escrita la operación `edad= 2019 -year`. En la última línea me dice que el operador "menos" no puede trabajar con un tipo entero y un tipo string a la vez.

El asunto es que TODAS LAS ENTRADAS POR TECLADO se toman como CADENAS DE CARACTERES.

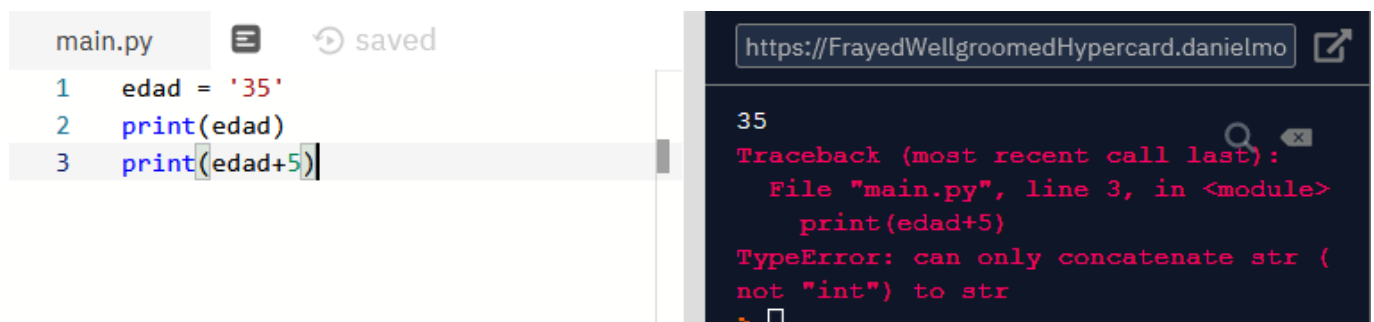
Así que el año que introdujo el usuario, para Python son LETRAS (caracteres), y no sabe cómo restar letras y números.

Convertir letras en números

Las variables pueden ser de diferentes tipos, y podemos transformar una variable de un tipo en otro. En este caso es necesario, puesto que para hacer operaciones matemáticas hay que convertir las letras en números.

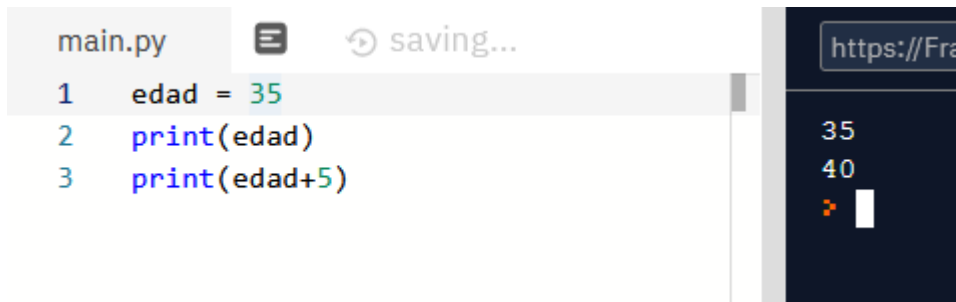
¿Hay una manera en Python de tomarse 45 como si fuera un número o como si fuera una cadena de caracteres? Sí.

Hay que usar la función `int()`. Convierte, si se puede, una cadena de caracteres en un número entero. Veamos el siguiente ejemplo. Edad contiene un string, es decir, las "letras" 3 y 5. Por eso va entre comillas. Cuando queremos sumar una palabra y un número no podemos (no es posible). Los dos objetos a sumar deben ser números o floats (decimales), por eso falla.



The screenshot shows a code editor with a file named `main.py`. The code contains three lines: `1 edad = '35'`, `2 print(edad)`, and `3 print(edad+5)`. The third line is highlighted. To the right, a terminal window shows the output of the script. It displays the number `35` on the first line, followed by a traceback for the second line. The traceback indicates a `TypeError: can only concatenate str (not "int") to str` at line 3, column 10, where the operation `print(edad+5)` was attempted.

En este otro caso, veréis que la variable `edad` no lleva comillas. Ahora está guardando 35 como número, no como palabra. Por ello, cuando quiero sumar la variable `edad` y el número 5, al ser ambos números, funciona.



The screenshot shows a code editor with a file named 'main.py' and a status bar indicating 'saving...'. The code in the editor is:

```
1 edad = 35
2 print(edad)
3 print(edad+5)
```

To the right of the code editor, there is a terminal window showing the output of the script:

```
https://Fra
35
40
>
```

Convertir tipos de variables

Cambiar de un tipo a otro se llama **casting**. Si queremos hacer una operación y necesitamos convertir una variable de un tipo a otro, podemos utilizar unas funciones que los transforman.

```
nombre = input("como te llamas: ")
edad = int(input("que edad tienes: "))
print('Hola', nombre, 'tienes', edad, 'años')
```