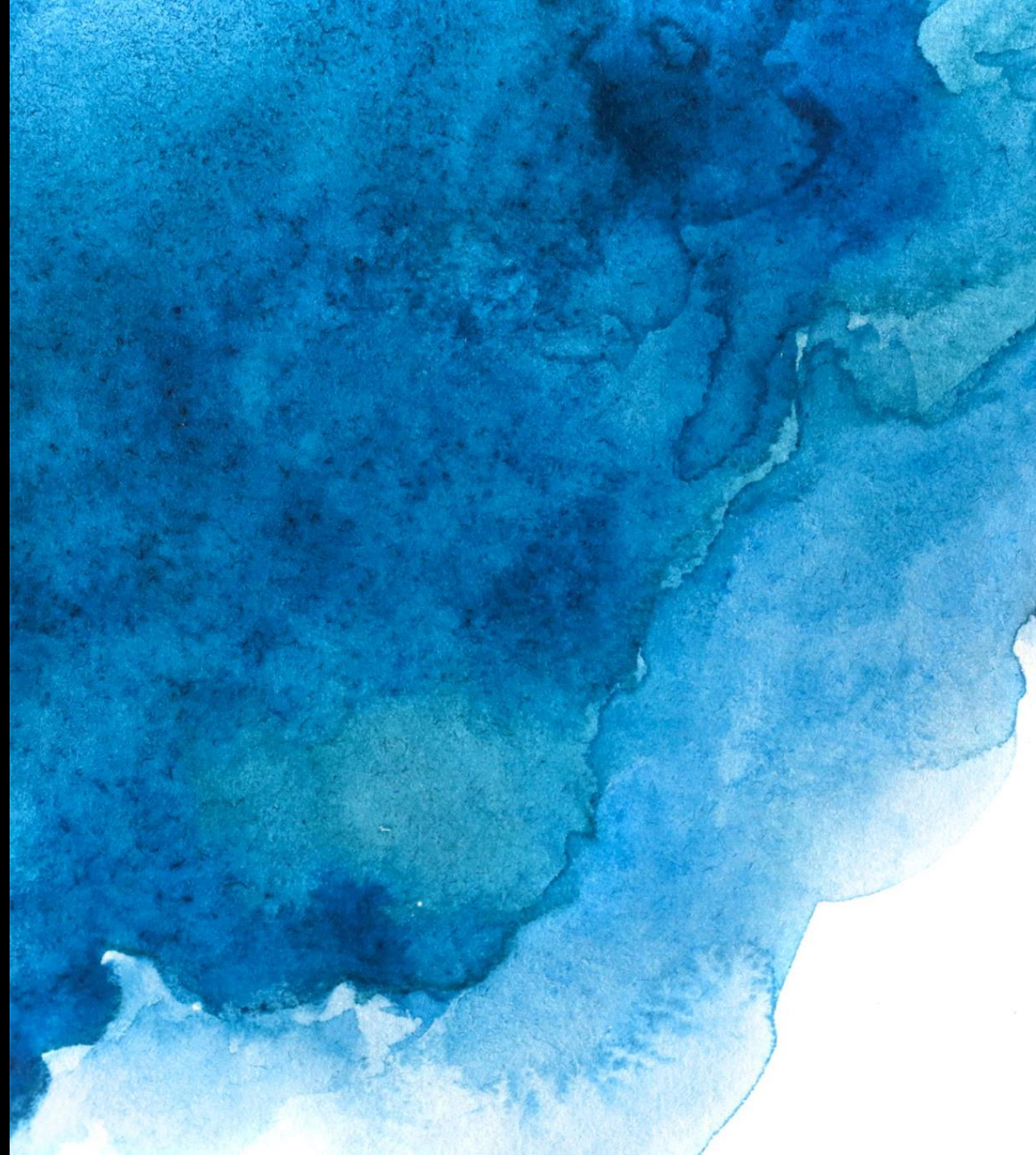


VARIABLES



¿QUÉ SON?

Es justo que Python nos permita codificar literales, las cuales contengan valores numéricos y cadenas.

Ya hemos visto que se pueden hacer operaciones aritméticas con estos números: sumar, restar, etc. Esto se hará una infinidad de veces en un programa.

Pero es normal preguntar como es que se pueden almacenar los resultados de estas operaciones, para poder emplearlos en otras operaciones, y así sucesivamente.

¿CÓMO ALMACENAR LOS RESULTADOS INTERMEDIOS, Y DESPUÉS UTILIZARLOS DE NUEVO PARA PRODUCIR RESULTADOS SUBSECUENTES?

Python ayudará con ello. Python ofrece "cajas" (contenedores) especiales para este propósito, estas cajas son llamadas variables - el nombre mismo sugiere que el contenido de estos contenedores puede variar en casi cualquier forma.



¿CUÁLES SON LOS COMPONENTES O ELEMENTOS DE
UNA VARIABLE EN PYTHON?

```
class Mascota:
```

```
    def __init__(self, edad):
```

Un nombre.

```
    pass
```

Un valor (el contenido del contenedor).

```
mascota = Mascota(2)
```

```
print(type(mascota) is Mascota) # True
```

Comencemos con lo relacionado al nombre de la variable.

Las variables no aparecen en un programa automáticamente. Como desarrollador, tu debes decidir cuantas variables deseas utilizar en tu programa.

También las debes de nombrar.

SI SE DESEA NOMBRAR UNA VARIABLE, SE DEBEN SEGUIR LAS SIGUIENTES REGLAS:

- El nombre de la variable debe de estar compuesto por MAYÚSCULAS, 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.
- Las mayúsculas y minúsculas se tratan de forma distinta (un poco diferente que en el mundo real - Alicia y ALICIA son el mismo nombre, pero en Python son dos nombres de variable distintos, subsecuentemente, son dos variables diferentes).
- El nombre de las variables no pueden ser igual a alguna de las palabras reservadas de Python (se explicará más de esto pronto).



NOMBRES CORRECTOS E INCORRECTOS DE VARIABLES

Nota que la misma restricción aplica a los nombres de funciones.

Python no impone restricciones en la longitud de los nombres de las variables, pero eso no significa que un nombre de variable largo sea mejor que uno corto.

AQUÍ SE MUESTRAN ALGUNOS NOMBRES DE VARIABLE QUE SON CORRECTOS, PERO QUE NO SIEMPRE SON CONVENIENTES:

```
MiVariable , i , t34 , Tasa_Cambio , contador , dias_para_navidad ,  
ElNombreEsTanLargoQueSeCometeranErroresConEl , _ .
```


ADEMÁS, PYTHON PERMITE UTILIZAR NO SOLO LAS LETRAS LATINAS, SINO CARACTERES ESPECÍFICOS DE OTROS IDIOMAS QUE UTILIZAN OTROS ALFABETOS. ESTOS NOMBRES DE VARIABLES TAMBIÉN SON CORRECTOS:

Adiós_Señora , sŭr_la_mer , Einbahnstraße , переменная .

AHORA VEAMOS ALGUNOS NOMBRES INCORRECTOS:

`10t` (no comienza con una letra), `Tasa Cambio` (contiene un espacio)

NOTA

PEP 8 -- Style Guide for Python Code recomienda la siguiente convención de nomenclatura para variables y funciones en Python:

Los nombres de las variables deben estar en minúsculas, con palabras separadas por guiones bajos para mejorar la legibilidad (por ejemplo: `var`, `mi_variable`).

Los nombres de las funciones siguen la misma convención que los nombres de las variables (por ejemplo: `fun`, `mi_función`).

También es posible usar letras mixtas (por ejemplo: `miVariable`), pero solo en contextos donde ese ya es el estilo predominante, para mantener la compatibilidad retroactiva con la convención adoptada.

PALABRAS CLAVE

- 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']`

- Son llamadas palabras clave o (mejor dicho) palabras reservadas. Son reservadas porque no se deben utilizar como nombres: ni para variables, ni para funciones, ni para cualquier otra cosa que se desee crear.
- El significado de la palabra reservada está predefinido, y no debe cambiar.

- Afortunadamente, debido al hecho de que Python es sensible a mayúsculas y minúsculas, cualquiera de estas palabras se pueden modificar cambiando una o varias letras de mayúsculas a minúsculas o viceversa, creando una nueva palabra, la cual no esta reservada.

- Por ejemplo - no se puede nombrar a la variable así:
- `Import`
- No se puede tener una variable con ese nombre, esta prohibido, pero se puede hacer lo siguiente:
- `Import`
- Estas palabras podrían parecer un misterio ahorita, pero pronto se aprenderá acerca de su significado.

CREANDO VARIABLES

¿Qué se puede poner dentro de una variable?

Cualquier cosa.

Se puede utilizar una variable para almacenar cualquier tipo de los valores que ya se han mencionado, y muchos mas de los cuales aun no se han explicado.

El valor de la variable es lo que se ha puesto dentro de ella. Puede variar tanto como se necesite o requiera. El valor puede ser entero, después flotante, y eventualmente ser una cadena.

Hablemos de dos cosas importantes - como son creadas las variables, y como poner valores dentro de ellas (o mejor dicho, como dar o pasarles valores).

RECORDAMOS

- Una variable se crea cuando se le asigna un valor. A diferencia de otros lenguajes de programación, no es necesario declararla.
- Si se le asigna cualquier valor a una variable no existente, la variable será automáticamente creada. No se necesita hacer algo más.
- 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.

Puedes colocar

CUALQUIER

float

words

value **COSA**²³ x

dentro de la variable

OBSERVA EL SIGUIENTE FRAGMENTO DE CÓDIGO:

```
var = 1
```

```
print(var)
```

- Consiste de dos simples instrucciones:
- La primera crea una variable llamada var, y le asigna un literal con un valor entero de 1.
- La segunda imprime el valor de la variable recientemente creada en la consola.
- Nota: print() tiene una función más â€” puede manejar variables tambi n. ¿Puedes predecir cu l ser  la salida (resultado) del c digo?

UTILIZANDO VARIABLES

Se tiene permitido utilizar cuantas declaraciones de variables sean necesarias para lograr el objetivo del programa, por ejemplo:

```
var = 1
account_balance = 1000.0
client_name = 'John Doe'
print(var, account_balance, client_name)
print(var)
```

Sin embargo, no se permite utilizar una variable que no exista, (en otras palabras, una variable a la cual no se le ha dado un valor).

Este ejemplo ocasionará un error:

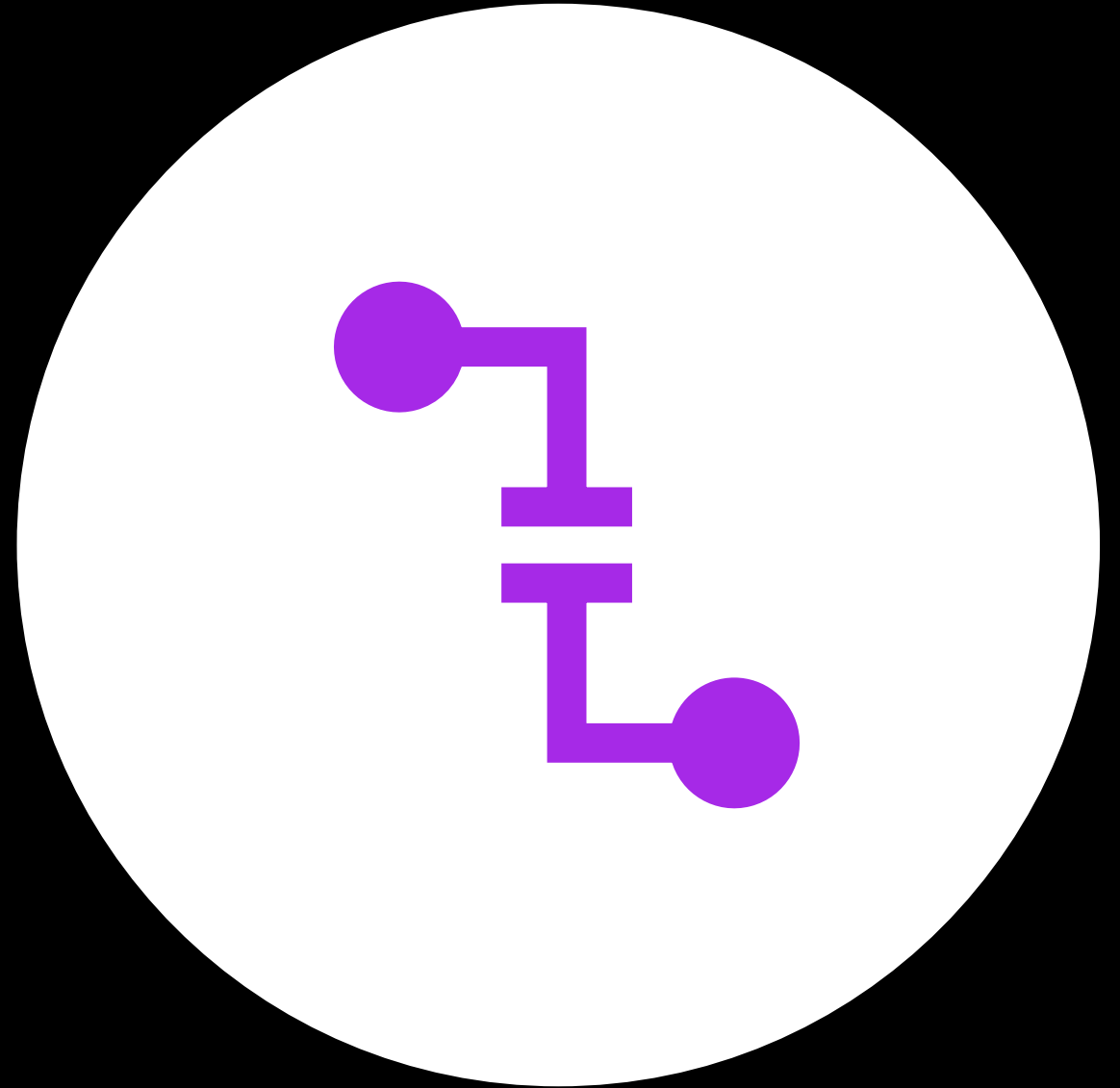
```
var = 1
```

```
print(Var)
```

RECUERDA


- Se puede utilizar `print()` para combinar texto con variables utilizando el operador `+` para mostrar cadenas con variables, por ejemplo:
- `var = "3.8.5"`
- `print("Versión de Python: " + var)`

¿PUEDES PREDECIR
LA SALIDA DEL
FRAGMENTO DE
CÓDIGO?




ASIGNAR UN VALOR NUEVO A UNA VARIABLE YA EXISTENTE

¿Cómo se le asigna un valor nuevo a una variable que ya ha sido creada? De la misma manera. Solo se necesita el signo de igual.



El signo de igual es de hecho un operador de asignación. Aunque esto suene un poco extraño, el operador tiene una sintaxis simple y una interpretación clara y precisa.



Asigna el valor del argumento de la derecha al de la izquierda, aún cuando el argumento de la derecha sea una expresión arbitraria compleja que involucre literales, operadores y variables definidas anteriormente.

OBSERVA EL SIGUIENTE CÓDIGO:

```
var = 1
```

```
print(var)
```

```
var = var + 1
```

```
print(var)
```

EL CÓDIGO ENVÍA DOS LÍNEAS A LA
CONSOLA:



La primer línea del código crea una nueva variable llamada var y le asigna el valor de 1.



La declaración se lee de la siguiente manera: asigna el valor de 1 a una variable llamada var.



De manera más corta: asigna 1 a var.



Algunos prefieren leer el código así: var se convierte en 1.



La tercera línea le asigna a la misma variable un nuevo valor tomado de la variable misma, sumándole 1. Al ver algo así, un matemático probablemente protestaría, ningún valor puede ser igualado a si mismo más uno. Esto es una contradicción. Pero Python trata el signo = no como igual a, sino como asigna un valor.



Entonces, ¿Cómo se lee esto en un programa?



Toma el valor actual de la variable `var`, sumale 1 y guárdalo en la variable `var`.



En efecto, el valor de la variable `var` ha sido **incrementado** por uno, lo cual no está relacionado con comparar la variable con otro valor.

¿PUEDES PREDECIR CUÁL SERÁ EL
RESULTADO DEL SIGUIENTE FRAGMENTO DE
CÓDIGO?

Variables >  example_variables.py > ...

```
1    var = 100  
2    var = 200 + 300  
3    print(var)
```

- 500 - ¿Porque? Bueno, primero, la variable var es creada y se le asigna el valor de 100. Después, a la misma variable se le asigna un nuevo valor: el resultado de sumarle 200 a 300, lo cual es 500.

OPERADORES ABREVIADOS

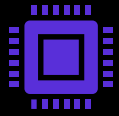


Es tiempo de explicar el siguiente conjunto de operadores que harán la vida del programador/desarrollador más fácil.



Muy seguido, se desea utilizar la misma variable al lado derecho y al lado izquierdo del operador =.

- Por ejemplo, si se necesita calcular una serie de valores sucesivos de la potencia de 2, se puede usar el siguiente código:
- $x = x * 2$
- También, puedes utilizar una expresión como la siguiente si no puedes dormir y estas tratando de resolverlo con alguno de los m
- `sheep = sheep + 1`



Python ofrece una manera más corta de escribir operaciones como estas, lo cual se puede codificar de la siguiente manera:



```
x *= 2  
sheep += 1
```

A continuación se intenta presentar una descripción general para este tipo de operaciones.

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

It can be simplified and shown as follows:

variable op= expresión

Observa los siguientes ejemplos. Asegúrate de entenderlos todos.

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

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

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

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

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

