Programación Web

Más sobre variables

Subsecretaría de Empleo
Chaco Gobierno de todos

Ministerio de Producción, Industria y Empleo
Chaco Gobierno de todos

CHACO
Gobierno de todos

INFORMATORIO

HABLEMOS SOBRE VARIABLES

Vamos a darte un ejemplo bien práctico para que comprendas mejor el concepto de variable.

Supongamos que nuestra **variable** es una caja y las cosas que vamos a poner adentro de esta caja, van a ser el **valor**.

Entonces, mi caja a la cual le di un nombre, en este caso como vés, se llama **variable**, y las cosas que voy a guardar y sacar de allí adentro van a ser valores, pero para simplificar vamos a decir, **valor**.

Con esto comprendido, vamos a trasladarlo a código.

```
variable = 'valor'
```

Y como verás, para **asignar** un valor a una variable, utilizamos el signo de igual (=).

Además vemos que la estructura consta de **3 partes**:
nombre de la variable, **signo de asignación** y **valor**

Como te explicamos en el apunte de variables, hay varios tipos de datos que puede almancenar una variable ¿te acordás de esos datos? Te refrescamos algunos de los más importantes:

```
int     #Números enteros
float   #Números decimales

str     #Strings o cadena de caracteres

bool    #Booleanos
```

Entonces siguiendo con nuestro ejemplo, ahora vamos a hacerlo más práctico y vamos a imaginarnos que estamos de mudanza, y, porque somos muy organizados, tenemos cajas etiquetadas, donde cada caja contiene solo las cosas que dice el nombre de la caja, por ejemplo 'ropa de invierno'

Y si esto lo trasladecemos a código:

```
caja1 = 'ropa de invierno'
```

¿Lo vas comprendiendo mejor? ¿Y si ahora cargamos más cajas? ¡Pero en en código!

```
caja1 = 'ropa de invierno'
caja2 = 'ropa de verano'
caja3 = 'elementos de la cocina'
```

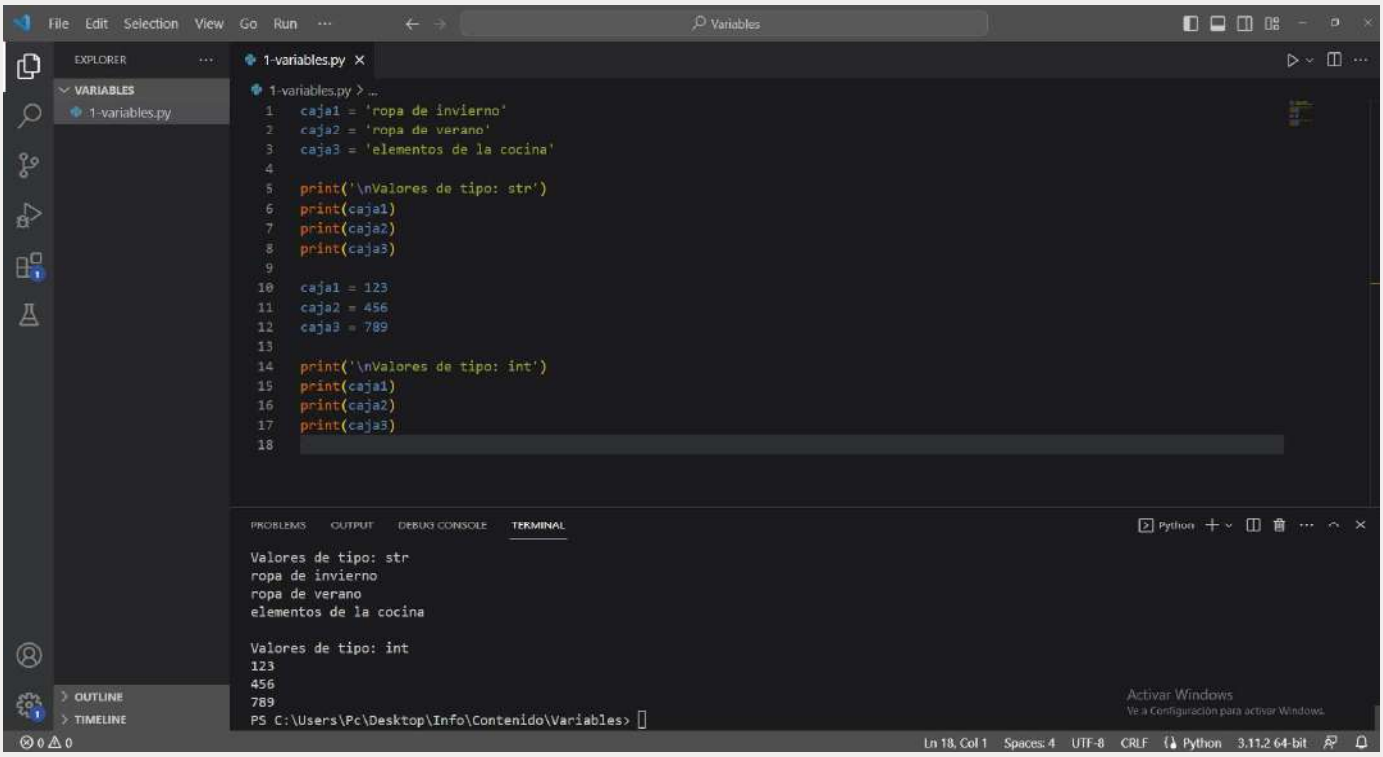
Como podés ver, tenemos 3 cajas o en este caso, 3 variables, y cada variable tiene un tipo de elemento o dato, que en este caso, es una cadena de caracteres. Si te alcanzaste a dar cuenta, cada valor tiene comillas simples ('') y por más que el valor sea distinto en cada variable, el tipo de dato sigue siendo el mismo: **STRING** o como se escribe en Python **str**.

¿Pero que pasa si ahora queremos guardar números enteros?

¡Fácil! Al ser Python de **tipado dinámico**, el valor de nuestra variable puede cambiar en tiempo de ejecución, y esto quiere decir, que solo debemos asignar un nuevo valor a la variable.

```
caja1 = 123
caja2 = 456
caja3 = 789
```

¿Y como sabemos que el valor cambió y fue aceptado por Python? Simplemente vamos a usar una función integrada de Python, la función: **print()**



Como podés ver, la función **print** (imprimir en inglés) sirve para mostrar en pantalla el valor que tienen las variables.

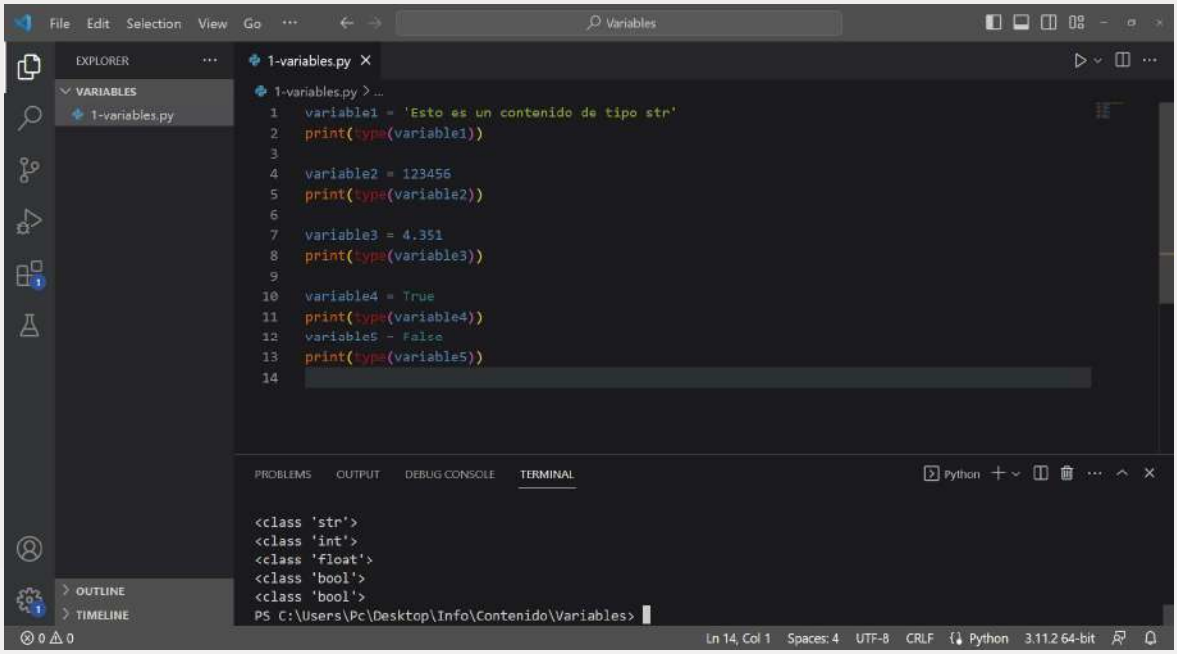
Lo podemos ir viendo en la parte de abajo, que es la consola, donde podemos ir ejecutando nuestro código (en los primeros apuntes que ya leíste, aclaramos que vamos a usar Visual Studio Code)

También, seguro que observaste que agregamos una pequeña oración para mostrar antes de mostrar los valores de las variables, esto solo fue para que quede más ordenado y puedas comprender mejor como podemos ir mostrando los valores de las variables.

Además, seguro que viste que luego de la función **print()** van paréntesis de apertura y cierre, y dentro de los paréntesis, el nombre de la variable que queremos mostrar su valor o, el valor que queremos mostrar directamente, como la oración que pusimos antes de mostrar el valor de la variable.
Este valor que mostramos directamente, es una cadena de caracteres o **string**, y como todo string, va entre comillas (acordate que pueden ser comillas simples o dobles, pero también recordá que por convención usamos comillas simples).

Y ahora volvemos con el concepto de **tipado dinámico**. En la primeras líneas de código de nuestro programa las variables tenían valores de tipo **str**, luego (en las siguientes líneas de código), cambiamos esos valores por números enteros o, mejor dicho, valores de tipo **integer,** o como se escribe en Python **int**

Para estar seguros de que tipo de dato contiene nuestra variable, podemos usar otra función con la cual podemos identificar esto. Esta función es: **type()**
Y para ver los tipos de datos más comunes que vamos a utilizar en Python, vamos a crear nuevas variables con nuevos valores.



Analicemos la imagen anterior:

* En variable1 (recordá que las variables pueden llevar cualquier nombre, pero siempre tenemos que intentar que éstos sean significativos con el valor que tenemos dentro de nuestra variable), tenemos un valor de tipo **str** y lo podemos ver en la consola cuando ejecutamos mediante la función **type()**

```
1   variable1 = 'Esto es un contenido de tipo str'
2   print(type(variable1))
```

```
<class 'str'>
```

* En variable2 tenemos un valor de tipo **int** y lo vemos en la consola.

```
4   variable2 = 123456
5   print(type(variable2))
```

```
<class 'int'>
```

* En variable3 tenemos un valor de tipo **float** y lo vemos en la consola.

```
7   variable3 = 4.351
8   print(type(variable3))
```

```
<class 'float'>
```

* En variable4 y en la variable5 tenemos un valor de tipo **bool.** Recordá que los valores que puede tomar un tipo bool van a ser verdadero o falso. En Python se debe escribir en inglés y con la primera en mayúscula: **True o False**

```
10   variable4 = True
11   print(type(variable4))
12   variable5 = False
13   print(type(variable5))
```

```
<class 'bool'>
<class 'bool'>
```

¡Y ahora es un buen momento para dar un pequeño repaso!

INFORMATORIO

Repasando un poco...

- Poner nombres a nuestras variables referidos al valor que va a tener:

* Si, por ejemplo, vamos a tener un nombre de usuario como valor de nuestra variable. El nombre que debería llevar nuestra variable sería: **nombre**

```
nombre = 'Informatorio'
```

* Si, por ejemplo, vamos a tener una edad como valor de nuestra variable. El nombre que debería llevar nuestra variable sería: **edad**

```
edad = 12
```

* Si, por ejemplo, vamos a tener una altura como valor de nuestra variable. El nombre que debería llevar nuestra variable sería: **altura**

```
altura = 1.85
```

* Si, por ejemplo, vamos a tener un estado como valor de nuestra variable (por ejemplo, en el caso de un empleado, si se encuentra activo o no). El nombre que debería llevar nuestra variable sería: **estado o activo**

```
estado = True
```

o

```
estado = False
```

```
activo = True
```

o

```
activo = False
```

Como podés ver, repasamos la estructura para asignar un valor a una variable, usamos los 4 tipos de datos que más vamos a utilizar y, un nombre de variable referido al valor que va a contener.
Estos 4 tipos de datos fueron: **str, int, float, bool**
Además aprendimos a utilizar la función **print()** y **type()**, que son algunas de las 69 funciones integradas de Python y, nos referimos a integradas, ya que no hay que importar ningún tipo de módulo para usarlas (más adelante veremos como importar módulos).

Vamos a interiorizarnos más y seguimos repasando...

1) Poner nombres a nuestras variables: Para definir nombres de variables hay que tener en cuenta las siguientes cuatro simples reglas:

a. El nombre de una variable puede empezar con una letra o un guión bajo, y aunque es permitido usar letras mayúsculas, por convención no lo hacemos.

b. Pueden contener letras, números y se puede usar el guión bajo (_).

c. En el nombre de una variable se distingue si contienen mayúsculas o minúsculas (significa que Python es un lenguaje case sensitive).

d. No se pueden utilizar palabras claves variables. Por ejemplo: no puedes definir una variable con el nombre "elif", pues esta palabra se usa en condicionales.

Otros ejemplos de Palabras reservadas:

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

2) Como vimos, para modificar el valor de una variable en Python, basta con asignarle un nuevo valor en cualquier momento y lugar, después de definirla.

3) A una variable se le puede asignar un valor literal (string, int, float, bool), una expresión, una llamada a una función o una combinación de todos ellos.

4) Se puede asignar un mismo valor a múltiples variables a la vez.

Por ejemplo: `a = b = c = 1`

5) En Python **todo es un objeto**. Entonces si asigno a la variable "a" el valor "1", realmente la variable "a" hace referencia al objeto que representa al número entero con valor "1".
Si ahora creamos una nueva variable "b" y le asignamos también el valor "1" , la variable "b" estará haciendo referencia al mismo objeto que la variable "a". En otros lenguajes de programación existen las

variables y constantes, pero en Python no. Aquí todos son **objetos**, pero de manera tradicional nos vamos a referir como variables a estos objetos (como lo venimos haciendo hasta ahora).

En definitiva, "a" y "b" hacen referencia al mismo objeto y, por tanto, están asociadas a la misma dirección de memoria.

En otros lenguajes "a" y "b" estarían asociadas a direcciones de memoria diferentes.

OPERADORES

Los **operadores** nos permiten manipular datos, sean variables, constantes, otras expresiones, objetos, atributos de objetos, entre otros, de manera que podamos:

a) transformarlos

b) usarlos en decisiones para controlar el flujo de ejecución de un programa

c) formar valores para asignarlos a otros datos

El tipo de datos involucrado en una expresión se relaciona con los operadores utilizados. Vamos con algunos de estos operadores.

OPERADORES MATEMÁTICOS

SUMA	RESTA
>> 2 + 2 Resultado: 4	>> 50 - 10 Resultado: 40
DIVISIÓN	**MULTIPLICACIÓN**
>> 25 / 3 Resultado: 8.333 Siempre retorna un punto flotante >> 25 // 5 Resultado: 5 Siempre retorna un número entero	>> 25 * 2 Resultado: 50
MÓDULO	**POTENCIA**
>> 25 % 3 Resultado: 1 Retorna el resto de la división	>> 8 ** 2 Resultado: 64

OPERADORES DE COMPARACIÓN

> MAYOR QUE	>= MAYOR O IGUAL QUE
>> a > b Resultado: **True** si el operando de la izquierda es estrictamente mayor que el de la derecha; **False** en caso contrario	>> a >= b Resultado: **True** si el operando de la izquierda es mayor o igual que el de la derecha; **False** en caso contrario.
< MENOR QUE	**<= MENOR O IGUAL QUE**
>> a < b Resultado: **True** si el operando de la izquierda es estrictamente menor que el de la derecha; **False** en caso contrario.	>> a <= b Resultado: **True** si el operando de la izquierda es menor o igual que el de la derecha; **False** en caso contrario.
== IGUAL	**!= DISTINTO**
>> a == b Resultado: **True** si el operando de la izquierda es igual que el de la derecha; **False** en caso contrario	>> a != b Resultado: **True** si los operandos son distintos; **False** en caso contrario

OPERADORES LÓGICOS

AND

>> a = true
>> b = true
>> x = a AND b

Resultado: True
Devuelve True solo si ambos valores son True, en cualquier otro caso devuelve False

OR

>> a = true
>> b = true
>> x = a OR b

Resultado: True

>> a = true
>> b = false
>> x = a OR b

Resultado: True

Devuelve False solo si ambos valores son False. Devuelve True si uno de los valores es True.

NOT

>> a = true
>> x = NOT a

Resultado: False
Cambia el valor de verdad de la variable a la que se aplica la operación

Además, podemos unir esto y usarlo de distintas formas por ejemplos con las operaciones con strings.

OPERACIONES CON CADENAS/STRINGS

CONCATENACIÓN	MULTIPLICACIÓN
>> 'Hola' + 'mundo' Resultado: Hola mundo Si tenemos dos cadenas o más entre comillas una al lado de la otra se concatenan automáticamente >> 'Hola' 'mundo' Resultado: Hola mundo	>> 3 * 'Hola' Resultado: HolaHolaHola
MEZCLA	
Se pueden mezclar las operaciones de concatenación y multiplicación >> 3 * 'Hola' + 'mundo' Resultado: HolaHolaHola mundo	

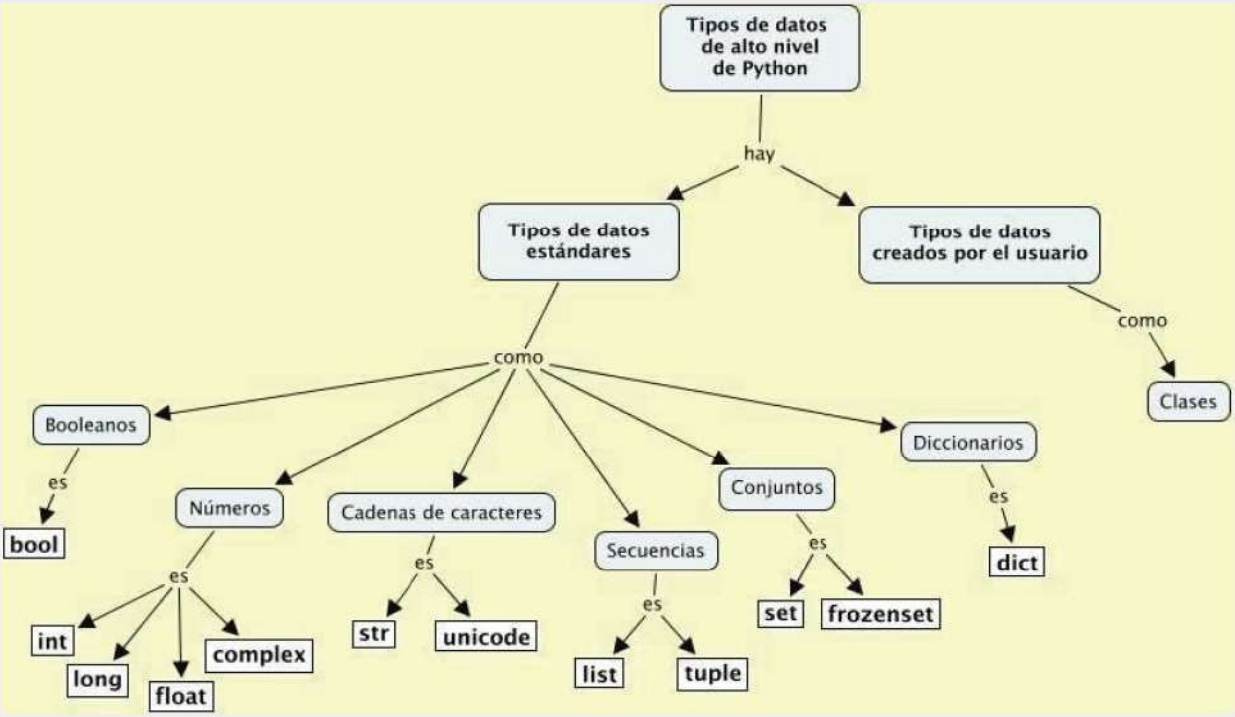
Dato. Clasificación de Datos. Tipos de Dato

Definición de Dato

Un dato es una representación simbólica (numéricas, alfabéticas, algorítmicas, etc.) de un atributo o cualidad de una entidad. Los datos aisladamente pueden no contener información humanamente relevante. Sólo cuando un conjunto de datos se examina conjuntamente a la luz de un enfoque, hipótesis o teoría se puede apreciar la información contenida en dichos datos y se puede utilizar en la realización de cálculos o toma de decisiones.
Un dato puede ser un caracter leído de un teclado, información almacenada en un disco, un número que se encuentra en la memoria central, etc.

Tipos de Dato

El tipo de un dato está definido por el conjunto de valores que puede tomar a lo largo de un programa.
En Python específicamente encontramos los siguientes tipos de Dato:



NÚMEROS

Enteros: Son números positivos o negativos que no tienen decimales. Estos números se conocen como de **tipo 'int'** (entero) o 'long' (entero largo para más precisión).

Por ejemplo: x = 2

REALES

Son números de tipo decimal y en Python se conocen como de tipo 'float'.

Por ejemplo: x = 2.5

COMPLEJOS

Son números que tienen una parte real y una parte imaginaria, en Python se conocen como de **tipo 'complex'**.

Por ejemplo: x = 2,1 + 6j

CADENAS o STRINGS

Se conocen como de **tipo 'str'** y el texto va encerrado entre comillas (simples o dobles).
Las cadenas admiten operadores como la suma o la multiplicación.

Por ejemplo: x = "Hola mundo" x = 'Hola Mundo'

BOOLEANOS

Puede tomar únicamente los valores de Verdadero o Falso. Se define usando el **tipo 'bool'**

Por ejemplo: x = true z = false

CONJUNTOS

Es una colección de datos desordenada que no contiene elementos que se repiten.
Se conoce como de **tipo 'set'**.

Por ejemplo: conjunto = {'naranja', 1, 'c', 2.5, True, 'ciruela'}

LISTAS

Contienen vectores(o como se los conoce en otros lenguajes: arrays), es decir, que contienen un conjunto de valores que pueden contener distintos tipos de dato. Se conocen como de **tipo 'list'**.

Por ejemplo: z = [0.5 , 'manzana', 1500 , 'azul', True, False]

TUPLAS

Es una lista que no se puede modificar después de su creación, es inmodificable o inmutable. Se puede anidar (o unir) una tupla dentro de otra. Se conocen como de **tipo 'tuple'**.

Por ejemplo: numero = 1, 25, 1500, 'Hola Mundo'
anidada = numero, 'Hola', 'Mundo', 15

DICCIONARIOS

Es un tipo de dato similar a los listas, pero trabajan con clave y valor en vez de índices.
Cada valor que está almacenado en un diccionario puede ser accedido usando la clave y obtener el valor en vez de usar un índice para referirse (vamos a practicar sobre esto para que lo comprendas mejor). Se conocen como de **tipo 'dict'**.

Por ejemplo:

```
agendatelefonica = {
'Juan' : 3624123456,
'Ana' : 3784546230,
'María' : 3794547895
}
```

```
turnos = {
1: 'Ariana',
2 : 'Belén',
3 : 'Carlos
}
```

NONE

Python incorpora un quinto tipo de dato que estrictamente hablando se llama **NoneType** y cuyo único valor posible es **None.**

A menudo, None es utilizado cuando se quiere crear una variable (recordá que es un objeto en realidad) pero aún no se le quiere asignar ningún valor en particular. Sin embargo, None es también un valor pero en este caso, es un valor nulo.

Por ejemplo: x = None

En clases vamos a poner todos estos conocimientos, que adquiriste, a prueba. Vamos a realizar ejercicios y ejemplos prácticos para comprender correctamente todo !No faltes!