INFORMATORIO

Taller de Programación Web

Manejo de Variables

Subsecretaría de Empleo
Chaco Gobierno de todos

Ministerio de Producción, Industria y Empleo
Chaco Gobierno de todos

CHACO
Gobierno de todos

INFORMATORIO

Variables. Manejo de Variables

Las **variables** nos permiten guardar valores y reutilizarlos en distintos lugares del código, nos permite acceder fácilmente a un dato para ser ***manipulado y transformado*** a lo largo de un programa.

Para definir una variable en Python, basta con nombrarla dentro del entorno y definir un valor ya que como dijimos anteriormente es un lenguaje completamente orientado a objetos, por lo que no es necesario declarar variables o el tipo antes de usarlas como en otros lenguajes.

```
<nombre_variable> = <valor>  |  Por ejemplo:    x = 10
```

En los ejercicios que desarrollemos a lo largo del curso las **variables serán la forma de identificar**, de forma sencilla, **un dato** que se encuentra almacenado en la memoria de la computadora por lo que debemos darle nombre que sean significativos.

Manejo de Variables

1. Para **crear** y darle **valores** a variables utilizamos la **operación de asignación (=)**.

   El operador de asignación enlaza un nombre, en el lado izquierdo del operador, con un valor en el lado derecho.

   ```
   Por ejemplo:    saludo = "¿Qué Onda?"

                   pi = 3.1414159265358979323846…
   ```

2. 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** o **reservadas** de Python como nombres de variables. Por ejemplo: no puedes definir una variable con el nombre “elif”, pues esta palabra se usa en condicionales.

Otros ejemplos de Palabras reservadas:

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

3. Para **modificar el valor** de una variable en Python, basta con **asignarle un nuevo valor** en cualquier momento y lugar después de la definición.
4. A una variable se le puede asignar un **valor literal** (número, string, booleano, none), una **expresión**, una llamada a una **función** o una **combinación** de todos ellos.
5. Se puede asignar un mismo valor a múltiples variables a la vez.

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

6. 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 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.

INFORMATORIO

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.

INFORMATORIO

== 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

<table>
<tr><th>AND</th><th colspan="2">OR</th></tr>
<tr><td>>> a = true
>> b = true
>> x = a AND b

Resultado: True
Devuelve True solo si ambos valores son True, en cualquier otro caso devuelve False</td><td>>> a = true
>> b = true
>> x = a OR b

Resultado: True</td><td>>> a = true
>> b = false
>> x = a OR b

Resultado: True</td></tr>
<tr><td rowspan="2"></td><td colspan="2">Devuelve False solo si ambos valores son False.
Devuelve True si uno de los valores es True.</td></tr>
<tr><th>NOT</th><td colspan="2" rowspan="2"></td></tr>
<tr><td>>> a = true
>> x = NOT a

Resultado: False
Cambia el valor de verdad de la variable a la que se aplica la operación</td></tr>
</table>

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	

Función type()

La función type() devuelve el tipo de objeto que recibe como argumento, podemos utilizarla para saber de qué tipo es una variable.

Por ejemplo:

```
>> nombre_variable =  'Hola mundo'
>> type( nombre_variable)
>> <class 'str'>
```

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.

Clasificación de Datos

Los datos se pueden clasificar:

1. De acuerdo a si varían o no durante la ejecución de un programa:
    a. **Constantes**: Su valor no varía a lo largo de la ejecución de un programa
    b. **Variables**: Su valor varía a lo largo de la ejecución de un programa

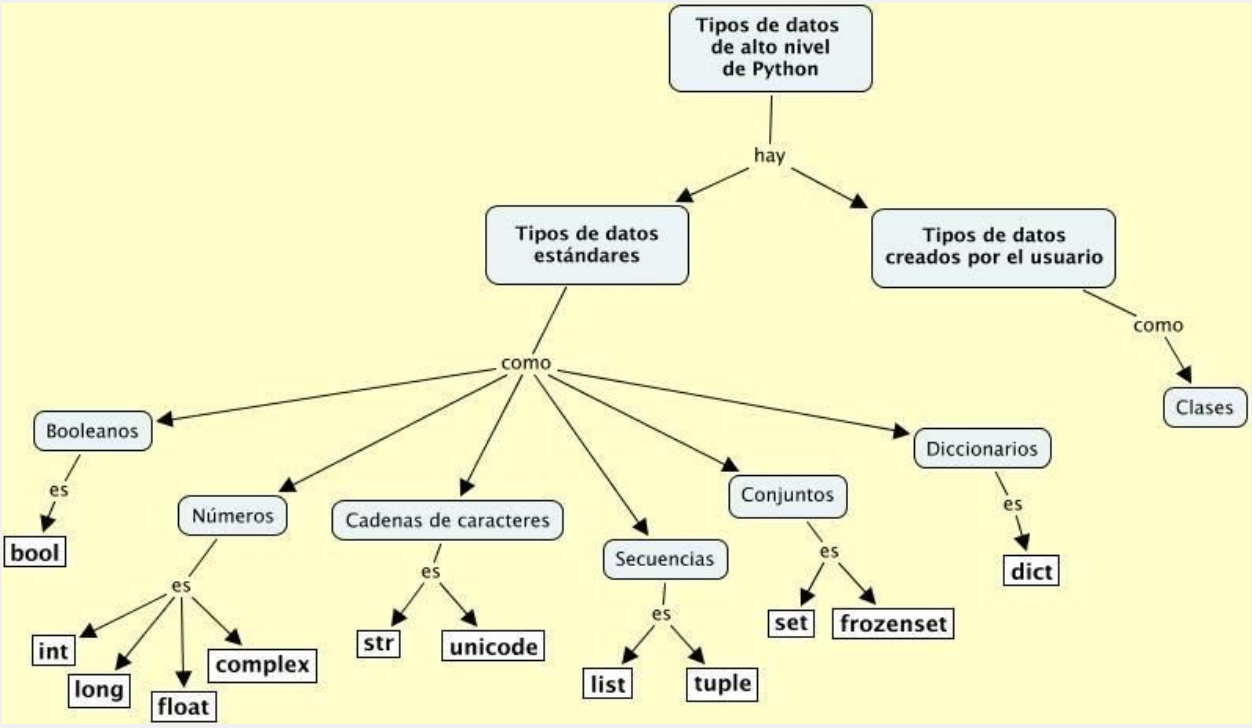
> *En Python no existen constantes ni variables, tan sólo objetos con los que hacer modificaciones.*
> *Lo que mal-llamamos variables (o constantes) no son otra cosa que* ***"referencias" a objetos****, como etiquetas para poder identificarlos.*
>
> ***Los objetos en Python pueden ser 'mutables' si puede modificarse solo algunas posiciones del dato o 'inmutables' si no pueden modificarse.***

2. De acuerdo a lo que representan pueden ser:
    a. **Números**
    b. **Cadenas o Strings**
    c. **Booleanos o Lógicos**

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` `z = -25`

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

Por ejemplo: `x = 2.5` `z = -0.5`

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`

```
2
3   numero1 = int(input("Favor ingresar el primer numero: "))
4   numero2 = int(input("Favor ingresar el segundo numero: "))
5
6   print "La suma de los dos numeros es de: " + str(numero1 + numero2)
```

CADENAS o STRINGS

Se conocen como de **tipo 'str'** texto 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(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']`

TUPLAS

Es una lista que no se puede modificar después de su creación, es inmodificable. Se puede anidar 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 arreglos, pero trabajan con llaves y valores en vez de índices. Cada valor está almacenado en un diccionario que puede ser accedido usando una clave en cualquier tipo de objeto (una cadena, número, lista, etc) en vez de usar un índice para referirse. Se conocen como de **tipo 'dict'**

Por ejemplo:

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

```
famosos= {
    1: "Susana Gimenez",
    5: "Marcelo Tinelli",
    7: "Mirtha Legrand"
}
```

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 (puesto que Python no distingue la creación de la asignación: crear una variable es simplemente darle un valor) pero aún no se le quiere asignar ningún valor en particular; aunque, en definitiva, como dijimos, None es también un valor.

Por ejemplo: `x = None`