DESCUBRIENDO EL PODER DE LA PROGRAMACION

CURSO INICIAL DE PYTHON

Jimmy Chung
Alexander Solis

Clase 2: Comentarios, Variables, Constantes y Tipos de datos

COMENTARIOS

- ✓ **Los comentarios son bloques de texto usados para comentar el código.**
- ✓ **Los comentarios no se ejecutan, solo son texto informativo.**
- ✓ **Ayudan a dar una descripción de una variable, una función, o cualquier otro elemento de nuestro código fuente.**
- ✓ **Los comentarios se inician con #**
- ✓ **Para comentar bloques (varias líneas) utilizamos comillas triples bien sean simples ''' o dobles """**

```
# Esto es un comentario
```

```
'''
Esto es un comentario
de varias líneas
de código
'''
```

VARIABLES

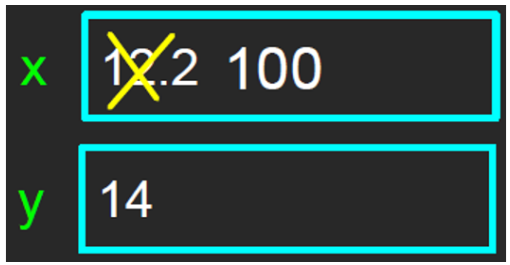
- **Una variable es un lugar designado en la memoria donde el programador puede guardar los datos y luego recuperar esos datos utilizando el “nombre” de la variable**
- **Los programadores elijen los nombres de las variables**
- **Usted puede cambiar el contenido de una variable en un enunciado posterior.**

```
x = 12.2
y = 14
```

VARIABLES

- **Una variable es un lugar designado en la memoria donde el programador puede guardar los datos y luego recuperar esos datos utilizando el "nombre" de la variable**
- **Los programadores elijen los nombres de las variables**
- **Usted puede cambiar el contenido de una variable en un enunciado posterior.**

```
x = 12.2
y = 14
x = 100
```



VARIABLES: Función id()

La función id() nos permite conocer la dirección de memoria de un objeto en Python.

```
>>> id(a)
4445989712

>>> id(b)
4445989712
```

Cada vez que asignamos un nuevo valor a una variable, ésta apunta a una nueva zona de memoria.

```
>>> a = 5
>>> id(a)
4310690224

>>> a = 7
>>> id(a)
4310690288
```

Reglas para nombrar Variables

- ✓ Solo pueden constar de letras, números y guión bajo.
- ✓ Debe empezar con una letra o un guión bajo _, nunca con un número.
- ✓ No pueden ser una palabra reservada del lenguaje («keywords»).

Válido	Inválido	Razón
`a`	`3`	Empieza por un dígito
`a3`	`3a`	Empieza por un dígito
`a_b_c___95`	`another-name`	Contiene un carácter no permitido
`_abc`	`with`	Es una palabra reservada del lenguaje
`_3a`	`3_a`	Empieza por un dígito

Las variables son sensibles a la mayúscula y minúscula «case-sensitive»
Ejemplo: user y User son variables diferentes.

Elegir Buenos Nombres de Variables

- Es nuestra responsabilidad escoger los nombres de las Variables.
- Nombramos a las variables de un modo que nos permita recordar qué nos proponemos guardar en ellas (“nemotécnica”= “ayuda memoria”)

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print(x1q3p9afd)
```

Qué está haciendo este código?

PEP 8: Style Guide for Python Code: https://peps.python.org/pep-0008/
PEP (Python Enhancement Proposals) Propuestas de mejora de Python

Elegir Buenos Nombres de Variables

- Es nuestra responsabilidad escoger los nombres de las Variables.
- Nombramos a las variables de un modo que nos permita recordar qué nos proponemos guardar en ellas (“nemotécnica”= “ayuda memoria”)

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print(x1q3p9afd)
```

Qué está haciendo este código?

```
a = 35.0
b = 12.50
c = a * b
print(c)
```

Qué está haciendo este código?

PEP 8: Style Guide for Python Code: https://peps.python.org/pep-0008/
PEP (Python Enhancement Proposals) Propuestas de mejora de Python

Elegir Buenos Nombres de Variables

- Es nuestra responsabilidad escoger los nombres de las Variables.
- Nombramos a las variables de un modo que nos permita recordar qué nos proponemos guardar en ellas (“nemotécnica”= “ayuda memoria”)

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print(x1q3p9afd)
```

```
a = 35.0
b = 12.50
c = a * b
print(c)
```

```
Horas = 35.0
Tarifa = 12.50
Salario = horas * tarifa
print(salario)
```

PEP 8: Style Guide for Python Code: https://peps.python.org/pep-0008/
PEP (Python Enhancement Proposals) Propuestas de mejora de Python

Palabras reservadas <<keywords>>

```
>>> help('keywords')

Here is a list of the Python keywords.  Enter any keyword to get more help.

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

Estas palabras no se pueden utilizar para nombrar variables

Variables: Otras consideraciones

Como regla general

- Usar **nombres** para las variables (ejemplo: producto).
- Usar **verbos** para funciones (ejemplo: obtener_descuento())
- Usar **adjetivos** para booleanos (ejemplo: disponible).

En Python se usa el símbolo = para **asignar** un valor a una variable:

Variables: Otras consideraciones

Python nos ofrece la posibilidad de hacer una asignación múltiple de la siguiente manera:

```
>>> tres = three = drei = 3
```

En este caso las tres variables utilizadas en el «lado izquierdo» tomarán el valor 3

Asignando una variable a otra variable

```
>>> people = 157503
>>> total_population = people
>>> total_population
157503
```

CONSTANTES

EN PYTHON LAS CONSTANTES NO EXISTEN.

Para guardar un valor constante se utiliza una variable pero se utilizan las letras mayúsculas para darle nombre a dicha variable.

Entonces, si en Python nos encontramos con un una variable en mayúsculas, significa NO MODIFICAR!

Ejemplos:

```
IVA=0.19
PI=3.14592
ALTO_PUERTA=210
DIAS_SEMANA=7
```

Valor de una Variable: print()

Para conocer el valor de una variable, utilizamos la función:

print()

print(): Imprime en pantalla el valor actual de una variable.

```
x = 10
y = "Nombre"

print(x)
print(y)
```

Tipo de dato de una Variable: type

Para conocer el tipo de dato que contiene una variable, utilizamos la función: type()

```
>>> type(9)
int

>>> type(1.2)
float

>>> height = 3718
>>> type(height)
int

>>> sound_speed = 343.2
>>> type(sound_speed)
float
```

TIPOS DE DATOS

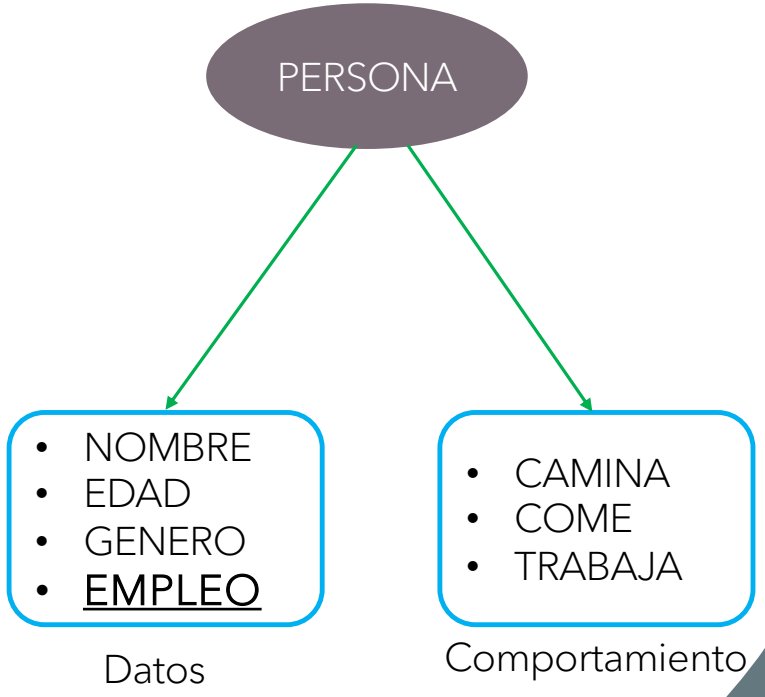
- Python es un lenguaje de programación orientado a objetos.
- En Python todo son objetos.
- Un objeto tiene, al menos, los siguientes campos (atributos):

  ✓ Un tipo del dato almacenado.

  ✓ Un identificador único para distinguirlo de otros objetos.

OBJETOS

Un objeto es una entidad que combina datos (**Variables o Constantes**) y comportamiento (**funciones**) relacionados en un solo elemento. Puede representar cualquier cosa del mundo real o abstracta.



TIPOS DE DATOS

Nombre	Tipo	Ejemplos
Booleano	bool	`True`, `False`
Entero	int	`21`, `34500`, `-34500`
Flotante	float	`3.14`, `1.5e3`
Complejo	complex	`2j`, `3 + 5j`
Cadena	str	`'tfn'`, `'''tenerife - islas canarias'''`
Tupla	tuple	`(1, 3, 5)`
Lista	list	`['Chrome', 'Firefox']`
Conjunto	set	`set([2, 4, 6])`
Diccionario	dict	`{'Chrome': 'v79' , 'Firefox': 'v71'}`

* String

NÚMEROS

Podemos considerar tres tipos de datos cuya naturaleza es numérica

Booleanos: Usado para representar valores **True (Verdadero)** o **False (Falso)**. Es un tipo de dato binario posibles valores True=1, False=0.

Enteros: Cualquier número que no tenga decimales.

Flotantes: Cualquier número con una parte decimal.

BOOLEANOS (bool)

George Boole es considerado como uno de los fundadores del campo de las ciencias de la computación y fue el creador del Álgebra de Boole que da lugar, entre otras estructuras algebraicas, a la Lógica binaria. En esta lógica las variables sólo pueden tomar dos valores: **verdadero** o **falso**. De aquí proviene el tipo de dato **bool** y admite dos posibles valores:

True: Que se corresponde a verdadero o 1 (por su representación numérica).

False: Que se corresponde a falso o 0 (por su representación numérica).

```
es_miercoles = True
print(es_miercoles)
```

```
True
```

```
es_martes = False
print(es_martes)
```

```
False
```

```
type(es_miercoles)
```

```
bool
```

```
type(es_martes)
```

```
bool
```

Importante: True y False con **la primera letra en mayúsculas.**

BOOLEANOS (bool)

True y False: Primera letra en mayúsculas.

```
es_una_falla = true
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[5], line 1
----> 1 es_una_falla = true

NameError: name 'true' is not defined
```

```
es_una_fallla = false
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[6], line 1
----> 1 es_una_fallla = false

NameError: name 'false' is not defined
```

Importante: True y False con **la primera letra en mayúsculas.**

ENTEROS (int)

El tipo de dato int permite almacenar un valor numérico no decimal ya sea positivo o negativo de cualquier valor.

Regla: No podemos comenzar un número entero por 0.

Tip: Python podemos dividir los números enteros con _ para facilitar su lectura/escritura (es como si esos guiones bajos no existieran).

```
>>> 8
8
>>> 0
0
>>> 08
  File "<stdin>", line 1
    08
     ^
SyntaxError: invalid token
```

```
>>> 99
99
>>> +99
99
>>> -99
-99
>>> 3000000
3000000
>>> 3_000_000
3000000
```

ENTEROS (int)

Es posible asignar valores en binario, hexadecimal y octal a un int así:

- El prefijo 0b indica que el valor será interpretado como un número binario.
- El prefijo 0x indica que el valor será interpretado como un número hexadecimal.
- [illegible] número octal.

```
a = 0b100
b = 0x17
c = 0o720
print(a, type(a)) #4 <class 'int'>
print(b, type(b)) #23 <class 'int'>
print(c, type(c)) #464 <class 'int'>
```

FLOTANTES (float)

float permite representar un número positivo o negativo con una parte decimal.

```
f = 0.10093
print(f)        #0.10093
print(type(f)) #<class 'float'>
```

```
>>> 4.000_000
4.0
```

Rangos: La mínima precisión es 2.2250738585072014e-308

La máxima precisión es 1.7976931348623157e+308

```
>>>import sys
>>>print(sys.float_info.min) #2.2250738585072014e-308
>>>print(sys.float_info.max) #1.7976931348623157e+308
```

CADENAS O STRING

Los Strings son una secuencia de caracteres como letras, números y símbolos, y deben escribirse entre comillas dobles ("texto") o sencillas ('texto').

```
s = "Esto es una cadena"
print(s)        #Esto es una cadena
print(type(s)) #<class 'str'>
```

```
s = 'Esto es otra cadena'
print(s)         #Esto es otra cadena
print(type(s))  #<class 'str'>
```

CADENAS O STRING

Para incluir comillas dobles dentro de
la cadena de texto la iniciamos con comillas sencillas

```
>>> 'Los llamados "strings" son secuencias de caracteres'
'Los llamados "strings" son secuencias de caracteres'
```

Para incluir comillas sencillas dentro de
la cadena de texto la iniciamos con comillas dobles

```
>>> "Los llamados 'strings' son secuencias de caracteres"
"Los llamados 'strings' son secuencias de caracteres"
```

CADENAS O STRING

Comillas triples:

Se utilizan para crear cadenas de texto multilínea:

```
>>> poem = '''To be, or not to be, that is the question:
... Whether 'tis nobler in the mind to suffer
... The slings and arrows of outrageous fortune,
... Or to take arms against a sea of troubles'''
```

Los tres puntos ... que aparecen a la izquierda de las líneas no están incluidos en la cadena de texto. Es el símbolo que ofrece el intérprete de Python cuando saltamos de línea.

ENTRADA DE DATOS (input)

Python nos provee la función input(), para capturar datos por pantalla.

Con la instrucción input(),
Python hace una pausa para
leer los datos que el usuario ingresa.

```
nombre = input('Quien es usted')
print('Bienvenido:', nombre)
```

```
Quién es usted

Jimmy

Bienvenido Jimmy
```

CONVERTIR ENTRADA DE DATOS

Si queremos leer un número del usuario, debemos convertirlo de una cadena a un número utilizando la función type conversion (conversión de tipo)

```
# Convertir pisos del elevador
inp = input('Piso europeo')
usf = int(inp) + 1
print('piso de EUA', usf)
```

```
Piso europeo 0
Piso de EUA 1
```

Ejercicios

- Muestra en pantalla el saludo Hola Pythonistas.
- Crea una variable llamada nombre, asígnale tu nombre, y muéstralo en pantalla.
- Asigna el valor 2001 a la variable my_dato y muéstralo en pantalla.
- Descubre el tipo del literal 'Hola curso, saludos'.
- Identifica el tipo del literal True.
- Asigna la expresión 10 * 3.0 a la variable result y muestra su tipo.
- Muestre en pantalla la longitud del siguiente texto: Curso de Python.

Ejercicios

- Escribe un programa que reciba el nombre, la edad, y el deporte, y luego imprima: Hola nombre, tienes edad años y te gusta el deporte
- Escribe un programa que imprima la suma de dos números.
- Escribe un programa que dados dos números devuelva el menor.
- Escriba un programa que reciba un numero, lo multiplique por 5, y muestre el resultado.
- Es criba un programa que reciba dos cadenas de texto (texto_1 y texto_2), luego arme una nueva cadena de texto separandolas con guion al medio "-" y muestre el resultado.

Ejercicios

- Escriba un programa que reciba los numeros 1 o 0, luego lo convierta a booleano, y muestre el valor y el tipo del booleano convertido.
- Escriba un programa que reciba False, lo convierta a booleano, y luego imprima el resultado.
- Escriba un programa que reciba un texto de entreda, pero no digite ningun valor (solo presione Enter), convierta el valor a booleano, y luego imprima el resultado.
- Escriba un progama que reciba un numero y conviertalo a flotante, imprima el valor, y el tipo de dato.

Tarea: Funciones «built-in»

Funciones «built-in»: Incorporadas por defecto en el propio lenguaje

abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	any()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

https://docs.python.org/es/3/library/functions.html?highlight=built

Hemos utilizado print(), type(), len().
Pero existen muchas mas.

Ayuda Tarea: «funcion help()»

En Python podemos pedir ayuda con la función help().

Supongamos que queremos obtener información sobre la función id()

Escribimos la siguiente instruccion: help(id) o id?

```
>>> help(id)
Help on built-in function id in module builtins:

id(obj, /)
    Return the identity of an object.

    This is guaranteed to be unique among simultaneously existing objects.
    (CPython uses the object's memory address.)
```

https://docs.python.org/es/3/library/functions.html?highlight=built

Funciones de conversion de tipo

Function	Conversion
int()	string,float->int
float()	string,int->float
str()	int, float, list, tuple, dictionary -> string
list()	string, tuple, dictionary, set -> list
tuple()	string, list, set ->tuple
set()	string, list, tuple -> set
complex()	int, float -> complex

Referencias bibliográficas

Documentación oficial Python.
https://docs.python.org/es/3/tutorial/index.html