



U.
RÍO NEGRO
UNIVERSIDAD
NACIONAL

PYTHON

Lenguaje de
programación

Ing. Pablo E. Argañaras

parganaras@unrn.edu.ar

COIL 2023



ET LUX IN TENEBRIS LUCET
PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ

PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ



1



U.
RÍO NEGRO
UNIVERSIDAD
NACIONAL

PYTHON

Lenguaje de
programación

Generalidades

COIL 2023



ET LUX IN TENEBRIS LUCET
PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ

PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ



2

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

Aprende Python

Tabla 1: Tipos de datos en Python

Nombre	Tipo	Ejemplos
Booleano	bool	True, False
Entero	int	21, 34500, 34_500
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'}

3

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

Reglas para nombrar variables

En Python existen una serie de reglas para los nombres de variables:

- Sólo pueden contener los siguientes caracteres²:
 - Letras minúsculas.
 - Letras mayúsculas.
 - Dígitos.
 - Guiones bajos (_).
- Deben **empezar con una letra o un guión bajo**, nunca con un dígito.
- No pueden ser una **palabra reservada** del lenguaje («keywords»).

² Para ser exactos, si se pueden utilizar otros caracteres, e incluso *emojis* en los nombres de variables, aunque no suele ser una práctica extendida, ya que podría dificultar la legibilidad.

4

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

Aprende Python

Podemos obtener un listado de las palabras reservadas del lenguaje de la siguiente forma:

```
>>> 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	

5

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

Como regla general:

- Usar **nombres** para *variables* (ejemplo article).
- Usar **verbos** para *funciones* (ejemplo get_article()).
- Usar **adjetivos** para *booleanos* (ejemplo available).

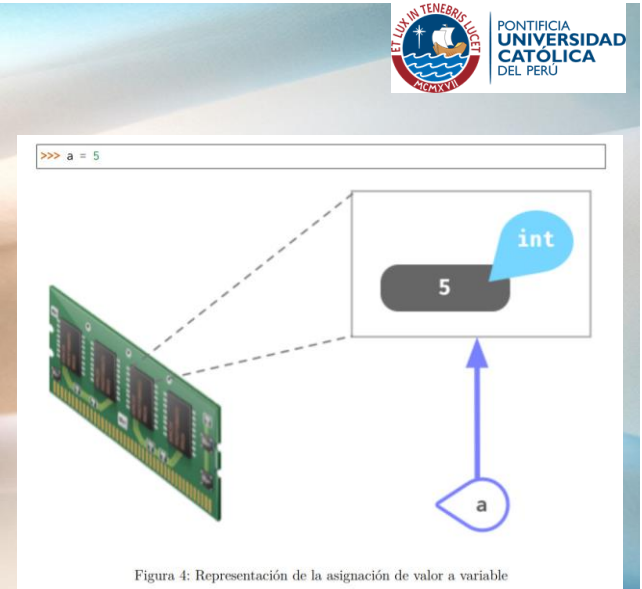
6

PYTHON

Lenguaje de
programación

Generalidades

COIL 2023



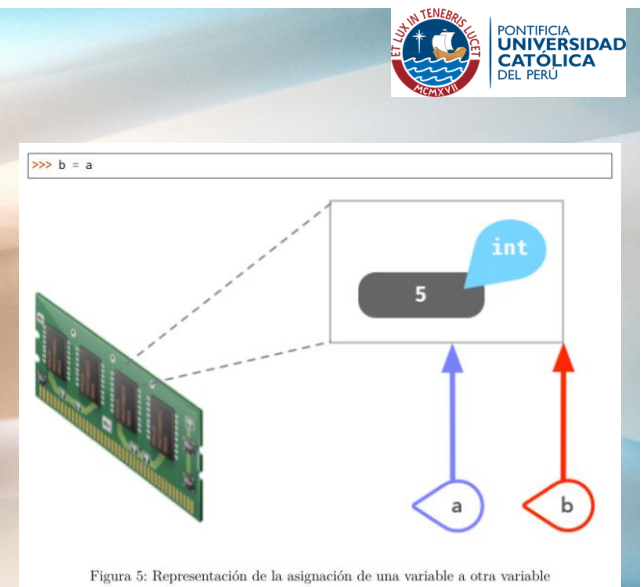
7

PYTHON

Lenguaje de
programación

Generalidades

COIL 2023



8

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

9

Cuando la zona de memoria que ocupa el objeto se puede modificar hablamos de tipos de datos **mutables**. En otro caso hablamos de tipos de datos **inmutables**.

Por ejemplo, las **listas** son un tipo de dato mutable ya que podemos modificar su contenido (aunque la asignación de un nuevo valor sigue generando un nuevo espacio de memoria).

Ejecución **paso a paso** a través de *Python Tutor*:

<https://cutt.ly/lvCyXeL>

Tipos de objetos en Python según su naturaleza de cambio:

Inmutable	Mutable
bool	list
int	set
float	dict
str	
tuple	

PYTHON

Lenguaje de programación

Generalidades



COIL 2023

Tabla 3: Funciones «built-in»

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()	

Los detalles de estas funciones se puede consultar en la documentación oficial de Python.

10

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

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

Supongamos que queremos obtener información sobre `id`. Desde el intérprete de Python ejecutamos lo siguiente:

```
>>> 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.)
```

Existe una *forma alternativa* de obtener ayuda: añadiendo el signo de interrogación `?` al término de búsqueda:

```
>>> id?
Signature: id(obj, /)
Docstring:
Return the identity of an object.

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

11

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

3.2.1 Booleanos

George Boole es considerado 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 discretos: **verdadero** o **falso**.

El tipo de datos `bool` proviene de lo explicado anteriormente y admite dos posibles valores:

¹ Foto original de portada por Brett Jordan en Unsplash.

60

Capítulo 3. Tipos de datos

Aprende Python

- `True` que se corresponde con *verdadero* (y también con `1` en su representación numérica).
- `False` que se corresponde con *falso* (y también con `0` en su representación numérica).

12

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

3.2.2 Enteros

Los números enteros no tienen decimales pero sí pueden contener signo y estar expresados en alguna base distinta de la usual (base 10).

Dos detalles a tener en cuenta:

- No podemos comenzar un número entero por 0.
- Python permite dividir los números enteros con *guiones bajos* `_` para clarificar su lectura/escritura. A efectos prácticos es como si esos guiones bajos no existieran.

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

Tabla 4: Operaciones con enteros en Python

Operador	Descripción	Ejemplo	Resultado
+	Suma	3 + 9	12
-	Resta	6 - 2	4
*	Multiplicación	5 * 5	25
/	División flotante	9 / 2	4.5
//	División entera	9 // 2	4
%	Módulo	9 % 4	1
**	Exponenciación	2 ** 4	16

Es de buen estilo de programación **dejar un espacio** entre cada operador. Además hay que tener en cuenta que podemos obtener errores dependiendo de la operación (más bien de los *operandos*) que estemos utilizando, como es el caso de la *división por cero*.

Igualmente es importante tener en cuenta la **prioridad** de los distintos operadores:

Prioridad	Operador
1 (mayor)	()
2	**
3	~a +a
4	* / // %
5 (menor)	+ -

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

Módulo

La operación **módulo** (también llamado **resto**), cuyo símbolo en Python es %, se define como el resto de dividir dos números. Veamos un ejemplo para entender bien su funcionamiento:

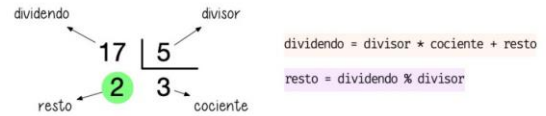


Figura 7: Operador «módulo» en Python

```
>>> dividendo = 17
>>> divisor = 5

>>> cociente = dividendo // divisor # división entera
>>> resto = dividendo % divisor

>>> cociente
3
>>> resto
2
```

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

Exponenciación

Para elevar un número a otro en Python utilizamos el operador de exponenciación **:

```
>>> 4 ** 3
64
>>> 4 * 4 * 4
64
```

Se debe tener en cuenta que también podemos elevar un número entero a un **número decimal**. En este caso es como si estuviéramos haciendo una *raíz*². Por ejemplo:

$$4^{\frac{1}{2}} = 4^{0.5} = \sqrt{4} = 2$$

Hecho en Python:

```
>>> 4 ** 0.5
2.0
```


PYTHON

Lenguaje de programación

Generalidades

COIL 2023

19

3.2.3 Flotantes

Los números en **punto flotante**³ tienen **parte decimal**. Veamos algunos ejemplos de flotantes en Python.

Lista 3: Distintas formas de escribir el flotante 4.0

```
>>> 4.0
4.0
>>> 4.
4.0
>>> 04.0
4.0
>>> 04.
4.0
>>> 4.0
```

(continué en la próxima página)

³ Punto o coma flotante es una notación científica usada por computadores.

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

Conversión de tipos

El hecho de que existan distintos tipos de datos en Python (y en el resto de lenguajes de programación) es una ventaja a la hora de representar la información del mundo real de la mejor manera posible. Pero también se hace necesario buscar mecanismos para convertir unos tipos de datos en otros.

Conversión implícita

Cuando mezclamos enteros, booleanos y flotantes, Python realiza automáticamente una conversión implícita (o **promoción**) de los valores al tipo de «mayor rango». Veamos algunos ejemplos de esto:

```
>>> True + 25
26
>>> 7 * False
0
>>> True + False
1
>>> 21.8 + True
22.8
>>> 10 + 11.3
21.3
```

Podemos resumir la conversión implícita en la siguiente tabla:

Tipo 1	Tipo 2	Resultado
bool	int	int
bool	float	float
int	float	float

20

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

Conversión explícita

Aunque más adelante veremos el concepto de **función**, desde ahora podemos decir que existen una serie de funciones para realizar conversiones explícitas de un tipo a otro:

bool() Convierte el tipo a *booleano*.

int() Convierte el tipo a *entero*.

float() Convierte el tipo a *flotante*.

Veamos algunos ejemplos de estas funciones:

```
>>> bool(1)
True
>>> bool(0)
False
>>> int(True)
1
>>> int(False)
0
>>> float(1)
1.0
>>> float(0)
0.0
>>> float(True)
1.0
>>> float(False)
0.0
```

En el caso de que usemos la función **int()** sobre un valor flotante nos retornará su **parte baja**:

$$\text{int}(x) = \lfloor x \rfloor$$

21

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

Errores de aproximación

Nivel intermedio

Supongamos el siguiente cálculo:

```
>>> (19 / 155) * (155 / 19)
0.9999999999999999
```

Debería dar 1.0, pero no es así puesto que la representación interna de los valores en **coma flotante** sigue el estándar IEEE 754 y estamos trabajando con aritmética finita.

Aunque existen distintas formas de solventar esta limitación, de momento veremos una de las más sencillas utilizando la función «built-in» **round()** que nos permite redondear un número flotante a un número determinado de decimales:

```
>>> pi = 3.14159265359
>>> round(pi)
```



```
3
>>> round(pi, 1)
3.1
>>> round(pi, 2)
3.14
>>> round(pi, 3)
3.142
>>> round(pi, 4)
3.1416
>>> round(pi, 5)
3.14159
```

Para el caso del error de aproximación que nos ocupa:

```
>>> result = (19 / 155) * (155 / 19)
>>> round(result, 1)
1.0
```

Prudencia: **round()** aproxima al valor más cercano, mientras que **int()** obtiene siempre el entero «por abajo».

22

PYTHON

Lenguaje de programación

Generalidades



COIL 2023

Límite de un flotante

A diferencia de los *enteros*, los números flotantes sí que tienen un límite en Python. Para descubrirlo podemos ejecutar el siguiente código:

```
>>> import sys
>>> sys.float_info.min
2.2250738585072014e-308
>>> sys.float_info.max
1.7976931348623157e+308
```

23

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

3.3.1 Creando «strings»

Para escribir una cadena de texto en Python basta con rodear los caracteres con comillas simples¹:

```
>>> 'Mi primera cadena en Python'
'Mi primera cadena en Python'
```

Para incluir *comillas dobles* dentro de la cadena de texto no hay mayor inconveniente:

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

¹ Foto original de portada por Roman Kraft en Unsplash.
⁶ También es posible utilizar comillas dobles. Yo me he decantado por las comillas simples ya que quedan más limpias y suele ser el formato que devuelve el propio intérprete de Python.

24

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

Puede surgir la duda de cómo incluimos *comillas simples* dentro de la propia cadena de texto. Veamos soluciones para ello:

Lista 4: Comillas simples escapadas

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

Lista 5: Comillas simples dentro de comillas dobles

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

En la primera opción estamos **escapando** las comillas simples para que no sean tratadas como caracteres especiales. En la segunda opción estamos creando el «string» con comillas dobles (por fuera) para poder incluir directamente las comillas simples (por dentro). Python también nos ofrece esta posibilidad.

25

PYTHON

Lenguaje de programación

Generalidades

COIL 2023



Comillas triples

Hay una forma alternativa de crear cadenas de texto utilizando *comillas triples*. Su uso está pensado principalmente para **cadenas multilinea**:

```
>>> 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'''
```

Importante: 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.

26

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

Comillas triples

Hay una forma alternativa de crear cadenas de texto utilizando *comillas triples*. Su uso está pensado principalmente para **cadenas multilinea**:

```
>>> 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'''
```



Importante: 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.

Cadena vacía

La cadena vacía es aquella que no contiene ningún carácter. Aunque a priori no lo pueda parecer, es un recurso importante en cualquier código. Su representación en Python es la siguiente:

```
>>> ''
''
```

27

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

3.3.2 Conversión

Podemos crear «strings» a partir de otros tipos de datos usando la función `str()`:

```
>>> str(True)
'True'
>>> str(10)
'10'
>>> str(21.7)
'21.7'
```

Para el caso contrario de convertir un «string» a un valor numérico, tenemos a disposición las funciones ya vistas:

```
>>> int('10')
10
>>> float('21.7')
21.7
```

Pero hay que tener en cuenta un detalle. La función `int()` también admite la **base** en la que se encuentra el número. Eso significa que podemos pasar un número, por ejemplo, en **hexadecimal** (como «string») y lo podríamos convertir a su valor entero:

```
>>> int('FF', 16)
255
```

Nota: La base por defecto que utiliza `int()` para convertir cadenas de texto es la **base decimal**.

28

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

3.3.4 Más sobre print()

Hemos estado utilizando la función `print()` de forma sencilla, pero admite algunos parámetros interesantes:

```
1 >>> msg1 = '¿Sabes por qué estoy acá?'
2 >>> msg2 = 'Porque me apasiona'
3
4 >>> print(msg1, msg2)
5 ¿Sabes por qué estoy acá? Porque me apasiona
6
7 >>> print(msg1, msg2, sep='|')
8 ¿Sabes por qué estoy acá?|Porque me apasiona
9
10 >>> print(msg2, end='!!!')
11 Porque me apasiona!!!
```

Línea 4: Podemos imprimir todas las variables que queramos separándolas por comas.

Línea 7: El *separador por defecto* entre las variables es un *espacio*, podemos cambiar el carácter que se utiliza como separador entre cadenas.

Línea 10: El *carácter de final de texto* es un *salto de línea*, podemos cambiar el carácter que se utiliza como final de texto.

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

3.3.5 Leer datos desde teclado

Los programas se hacen para tener interacción con el usuario. Una de las formas de interacción es solicitar la entrada de datos por teclado. Como muchos otros lenguajes de programación, Python también nos ofrece la posibilidad de leer la información introducida por teclado. Para ello se utiliza la función `input()`:

```
>>> name = input('Introduzca su nombre: ')
Introduzca su nombre: Sergio
```

(continúa en la siguiente diapositiva)

```
>>> name
'Sergio'
>>> type(name)
str

>>> age = input('Introduzca su edad: ')
Introduzca su edad: 41
>>> age
'41'
>>> type(age)
str
```

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

3.3.6 Operaciones con «strings»

Combinar cadenas

Podemos combinar dos o más cadenas de texto utilizando el operador +:

```
>>> proverb1 = 'Cuando el río suena'
>>> proverb2 = 'agua lleva'

>>> proverb1 + proverb2
'Cuando el río suenaagua lleva'

>>> proverb1 + ', ' + proverb2 # incluimos una coma
'Cuando el río suena, agua lleva'
```

Repetir cadenas

Podemos repetir dos o más cadenas de texto utilizando el operador *:

```
>>> reaction = 'Wow'

>>> reaction * 4
'WowWowWowWow'
```

31

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

Obtener un carácter

Los «strings» están **indexados** y cada carácter tiene su propia posición. Para obtener un único carácter dentro de una cadena de texto es necesario especificar su **índice** dentro de corchetes [...].

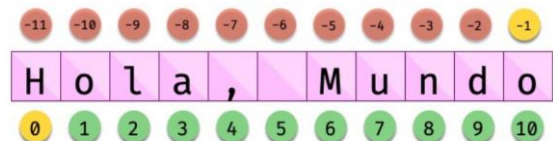


Figura 8: Indexado de una cadena de texto

32

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

```
>>> sentence = 'Hola, Mundo'
>>> sentence[0]
'H'
>>> sentence[-1]
'o'
>>> sentence[4]
','
>>> sentence[-5]
'M'
```

Truco: Nótese que existen tanto **índices positivos** como **índices negativos** para acceder a cada carácter de la cadena de texto. A priori puede parecer redundante, pero es muy útil en determinados casos.

En caso de que intentemos acceder a un índice que no existe, obtendremos un error por *fuera de rango*:

```
>>> sentence[50]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
```

33

PYTHON

Lenguaje de programación

Generalidades

COIL 2023



Advertencia: Téngase en cuenta que el indexado de una cadena de texto siempre empieza en 0 y termina en **una unidad menos de la longitud** de la cadena.

Las cadenas de texto son tipos de datos *inmutables*. Es por ello que no podemos modificar un carácter directamente:

```
>>> song = 'Hey Jude'
>>> song[4] = 'D'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

Truco: Existen formas de modificar una cadena de texto que veremos más adelante, aunque realmente no estamos transformando el original sino creando un nuevo objeto con las modificaciones.

34

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

Trocear una cadena

Es posible extraer «trozos» («rebanadas») de una cadena de texto⁶. Tenemos varias aproximaciones para ello:

[:] Extrae la secuencia entera desde el comienzo hasta el final. Es una especie de *copia* de toda la cadena de texto.

[start :] Extrae desde **start** hasta el final de la cadena.

[: end] Extrae desde el comienzo de la cadena hasta **end** *menos 1*.



[start : end] Extrae desde **start** hasta **end** *menos 1*.

[start : end : step] Extrae desde **start** hasta **end** *menos 1* haciendo saltos de tamaño **step**. Veamos la aplicación de cada uno de estos accesos a través de un ejemplo:

```
>>> proverb = 'Agua pasada no mueve molino'
>>> proverb[:]
'Agua pasada no mueve molino'
>>> proverb[12:]
'no mueve molino'
>>> proverb[:11]
'Agua pasada'
>>> proverb[5:11]
'pasada'
>>> proverb[5:11:2]
'psd'
```

Importante: El troceado siempre llega a una unidad menos del índice final que hayamos especificado. Sin embargo el comienzo sí coincide con el que hemos puesto.

35

PYTHON

Lenguaje de programación

Generalidades

COIL 2023

Reglas para escribir buenos comentarios:⁶

1. Los comentarios no deberían duplicar el código.
2. Los buenos comentarios no arreglan un código poco claro.
3. Si no puedes escribir un comentario claro, puede haber un problema en el código.
4. Los comentarios deberían evitar la confusión, no crearla.
5. Usa comentarios para explicar código no idiomático.
6. Proporciona enlaces a la fuente original del código copiado.

⁶ Referencia: Best practices for writing code comments

36



PYTHON

Lenguaje de
programación

Generalidades

COIL 2023

¿Preguntas?