

Parte I: Intro pensamiento computacional

Clase 04: Ciclo for, break y continue, listas

Diego Caro
dcaro@udd.cl



Basada en presentaciones oficiales de libro Introduction to Programming in Python (Sedgewick, Wayne, Dondero).

Disponible en <https://introcs.cs.princeton.edu/python>

Outline

- Reconocer sintaxis y funcionamiento de ciclo for
- Detener y saltarse una iteración en un ciclo for o while
- Reconocer diagrama de flujo de un condicional y ciclos while/for
- Comprender la utilidad de las listas
- Identificar patrones de uso de procesamiento de datos con listas

Ejercicio clase anterior

```
1 i = 1
2 while i <= 10:
3     print("7 *",i,"=",7*i)
4     i = i + 1
```

```
1 j = 1
2 while j <= 12:
3     print('Tabla del',j)
4     i = 1
5     while i <= 10:
6         print(j,"*",i,"=",j*i)
7         i = i + 1
8     j = j + 1
```

Chequear traza en <https://goo.gl/cdGQx8>

Python 3.6

```
1 j = 1
2 while j <= 12:
3     print('Tabla del',j)
4     i = 1
5     while i <= 10:
6         print(j,"*",i,"=",j*i)
7         i = i + 1
8     j = j + 1
```

[Edit this code](#)

Print output (drag lower right corner to resize)

```
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9
1 * 10 = 10
```

Frames Objects

Global frame	
j	2
i	11

→ line that has just executed
→ next line to execute

Click a line of code to set a breakpoint; use the Back and Forward buttons to jump there.

<< First < Back Step 38 of 422 Forward > Last >>

Human-based python interpretertm

- ¿Qué hace este programa?

```
1 a = 5
2 b = int(input())
3 if a + b < b:
4     print('Si')
5 else:
6     print('No')
```

Típico caso de loop

- Preguntar indefinidamente al usuario.
- Ejemplo: x es un número múltiplo de 7?

```
1 while True:
2     x = int(input('ingrese número entero: '))
3     if x % 7 == 0:
4         print(x, 'es múltiplo de 7')
5     else:
6         print(x, 'NO es múltiplo de 7')
```

Ciclo for

- Ejecutar código mientras se recorre una secuencia de elementos.
 - La secuencia se recorre en orden.
 - El término está garantizado.

Variable usada
para recorrer la
secuencia

Secuencia de enteros
hasta n - 1

```
for i in range(4):  
    print('Hola número', i)
```

Salida:

```
$ python3 holas.py  
Hola número 0  
Hola número 1  
Hola número 2  
Hola número 3
```

Inicio

Fin

```
for i in range(4,8):  
    print('Hola número', i)
```

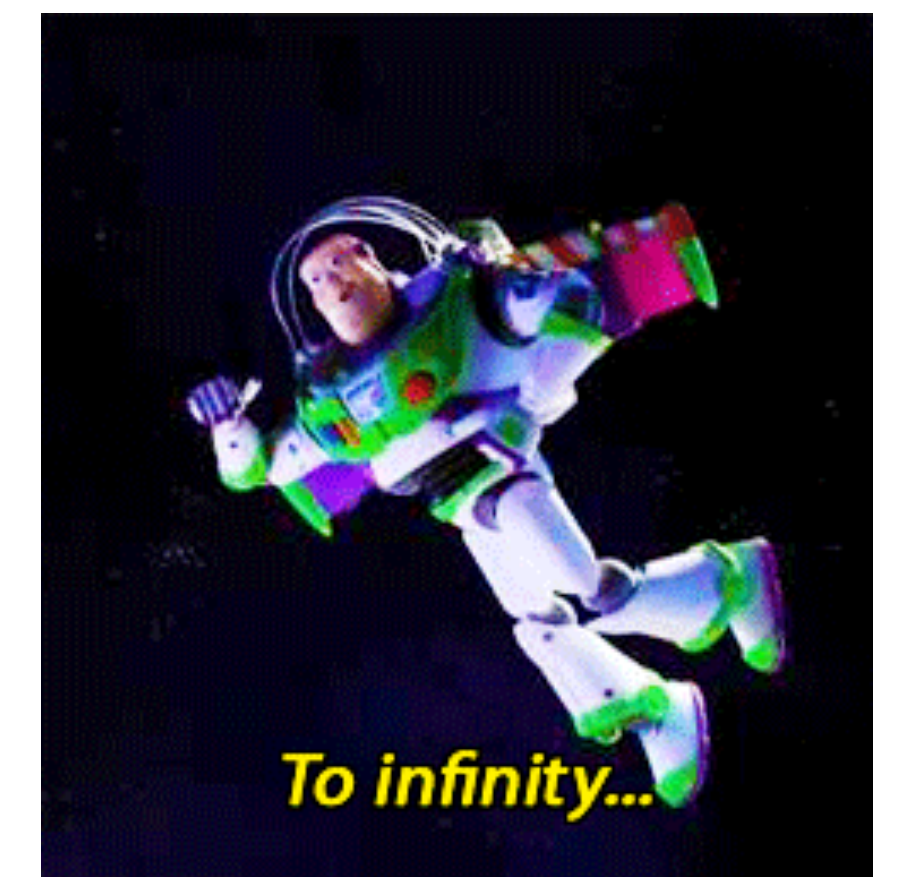
Salida:

```
$ python3 holas2.py  
Hola número 4  
Hola número 5  
Hola número 6  
Hola número 7
```

Ejemplo de un ciclo **while** que **nunca termina**.
La condición de detención siempre es **True**!

```
x = 1  
while True:  
    print("Al infinito y más allá! Ya vamos en {:d}!".format(x))  
    x += 1
```

```
1. bash  
Al infinito y más allá! Ya vamos en 93523!  
Al infinito y más allá! Ya vamos en 93524!  
Al infinito y más allá! Ya vamos en 93525!  
Al infinito y más allá! Ya vamos en 93526!  
Al infinito y más allá! Ya vamos en 93527!  
Al infinito y más allá! Ya vamos en 93528!  
Al infinito y más allá! Ya vamos en 93529!  
Al infinito y más allá! Ya vamos en 93530!  
Al infinito y más allá! Ya vamos en 93531!  
Al infinito y más allá! Ya vamos en 93532!  
Al infinito y más allá! Ya vamos en 93533!  
Al infinito y más allá! Ya vamos en 93534!  
Al infinito y más allá! Ya vamos en 93535!  
Al infinito y más allá! Ya vamos en 93536!  
Al infinito y más allá! Ya vamos en 93537!  
Al infinito y más allá! Ya vamos en 93538!  
Al infinito y más allá! Ya vamos en 93539!  
Al infinito y más allá! Ya vamos en 93540!  
Al infinito y más allá! Ya vamos en 93541!  
Al infinito y más allá! Ya vamos en 93542!  
Al infinito y más allá! Ya vamos en 93543!
```



while vs for

- Imprima todos los números impares menores que n mayores o iguales a cero.

```
1 n = int(input('ingrese n: '))
2 if n <= 0:
3     print('Debe ingresar un número mayor a cero')
4 i = 0
5 while i < n:
6     if i % 2 == 1:
7         print(i)
8     i = i+1
```

```
1 n = int(input('ingrese n: '))
2 for i in range(n):
3     if i % 2 == 1:
4         print(i)
```

```
1 n = int(input('ingrese n: '))
2 for i in range(1, n, 2):
3     print(i)
```

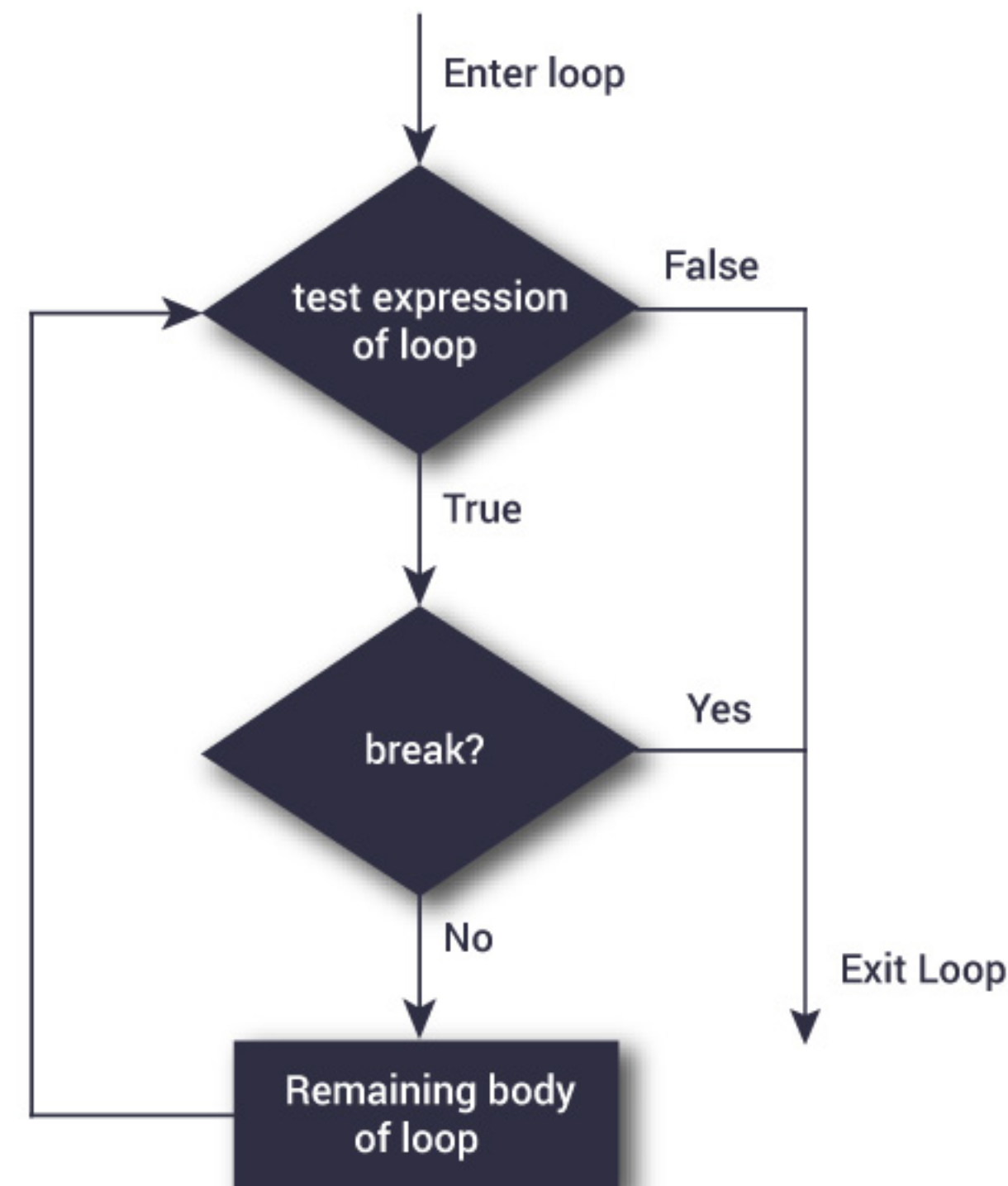
break: detener un ciclo

- Los ciclos se pueden detener antes de que recorran la secuencia o la condición en while no se cumpla. Keyword: **break**
- **Ventaja:** podemos ahorrar tiempo de procesador (muuuuuuy poco).
- **Desventaja:** código más complejo.

```
for var in secuencia:
    # código dentro del ciclo for
    if condicion:
        break # detiene el ciclo for
    # código dentro del ciclo for
#código fuera del ciclo for

--

while test expresión:
    # código dentro del ciclo while
    if condicion:
        break # detiene el ciclo while
    # código dentro del ciclo while
#código fuera del ciclo while
```



```
1 for e in 'hola':
2     if e == 'l':
3         break
4     print(e)
```

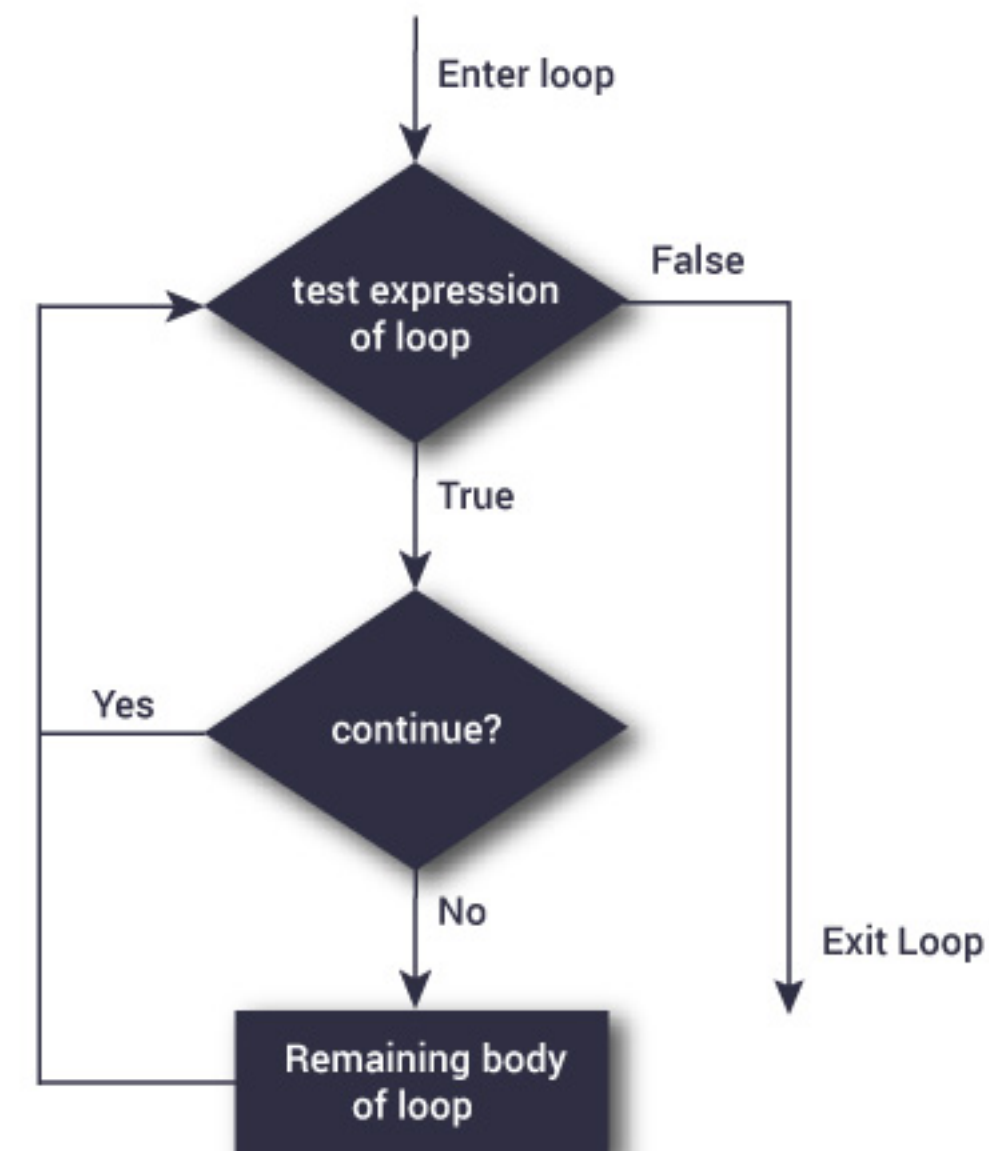
```
$ python3 simple-break.py
h
o
```

Nota: si necesitas usar **break**, verifica que sea la alternativa más sencilla.

continue: saltar a siguiente iteración

- A veces queremos saltarnos alguna iteración (ej.: ignorar elementos negativos). Puedes saltarlos usando **continue**.
- **Ventaja:** podemos ahorrar tiempo de procesador (muuuuuy poco).
- **Desventaja:** código más complejo.

```
for var in secuencia:  
    # código dentro del ciclo for  
    if condicion:  
        continue # salta a siguiente iteración  
    # código dentro del ciclo for  
  
#código fuera del ciclo for  
  
while test expresión:  
    # código dentro del ciclo while  
    if condicion:  
        continue # salta a siguiente iteración  
    # código dentro del ciclo while  
  
#código fuera del ciclo while
```



```
1 for e in 'hola':  
2     if e == 'l':  
3         continue  
4     print(e)
```

```
$ python3 simple-continue.py  
h  
o  
l  
a
```

Nota: si necesitas usar **continue**, verifica que sea la alternativa más sencilla.

Había una vez una investigación...

- ¿Qué tanto varía tu tiempo de viaje a la universidad?
- La variación la podemos cuantificar con la desviación estándar. Permite calcular cuanto se aleja cada medición al promedio.

Día	Tiempo de viaje en minutos
1	67
2	45
3	84
s	19,553

20 minutos de variación

Día	Tiempo de viaje en minutos
1	70
2	70
3	70
s	0

Sin variación

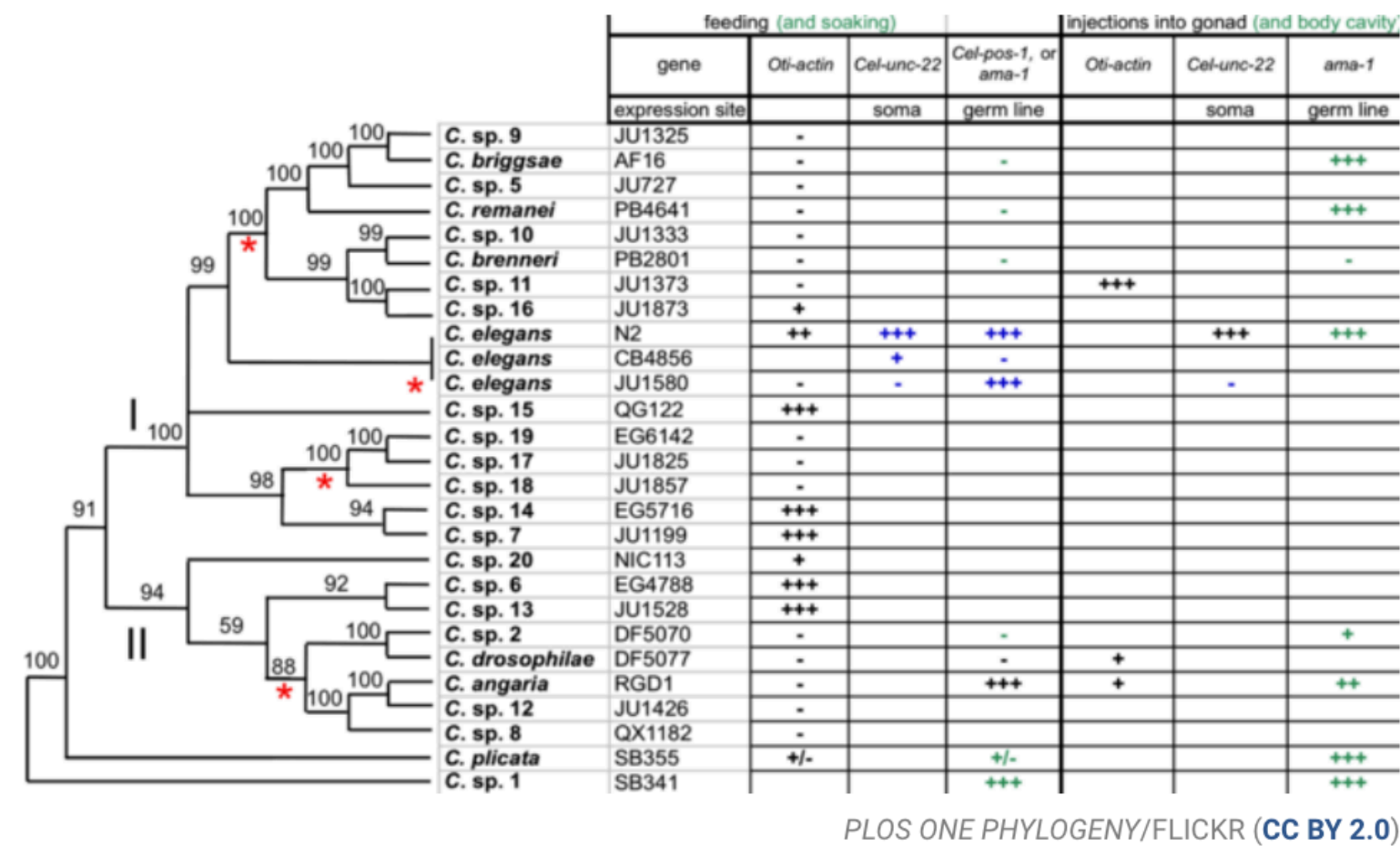
Desviación estándar

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}$$

↑
Promedio

P1: ¿Cómo harías un programa que calcule la desviación estándar?

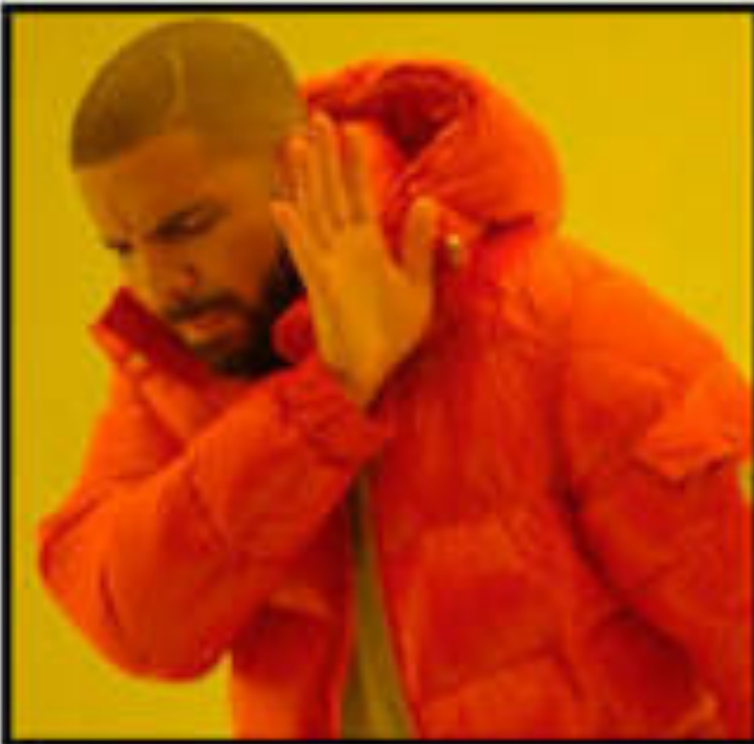
P2: ¿Y si el número de días es 1000?



One in five genetics papers contains errors thanks to Microsoft Excel

By **Jessica Boddy** | Aug. 29, 2016 , 1:45 PM

Autoformatting in Microsoft Excel has caused many a headache—but now, a new study shows that one in five genetics papers in top scientific journals **contains errors from the program**, *The Washington Post* reports. The errors often arose when gene names in a spreadsheet **were automatically changed** to calendar dates or numerical values. For example, one gene called *Septin-2* is commonly shortened to *SEPT2*, but is changed to 2-SEP and stored as the date 2 September 2016 by Excel. The researchers, who published their analysis in *Genome Biology*, say the issue can be fixed by formatting Excel columns as text and remaining vigilant—or switching to Google Sheets, where gene names are stored exactly as they’re entered.



**ANALIZAR DATOS
CON EXCEL**



**ANALIZAR DATOS
CON PYTHON**

imgflip.com

<http://www.sciencemag.org/news/2016/08/one-five-genetics-papers-contains-errors-thanks-microsoft-excel>

Listas

- Lista: secuencia de elementos de cualquier tipo.
- **Propósito:** facilitar el almacenamiento y procesamiento de datos.

Corchetes crean una lista Elementos separados por coma Tercer elemento en la lista

```
cartas = ['Diamante', 'Corazón', 'Pica', 'Trébol']  
print(cartas[2])
```

Acceso al i-ésimo elemento


```
$ python3.7 lista.py  
Pica
```

Importante: El primer elemento está en la posición 0

Ejemplos:

- 52 cartas en un mazo
- 27 alumnos en una clase
- 8 millones de píxeles en una imagen
- 4 mil millones de nucleótidos en una base de ADN
- 86 mil millones de neuronas en el cerebro
- $6.02 \cdot 10^{23}$ partículas en un mol

index	value
0	2♥
1	6♠
2	A♦
3	A♥
...	
49	3♣
50	K♣
51	4♠



Procesando muchos valores

```
a0 = 0
a1 = 0
a2 = 0
a3 = 0
a4 = 0
a5 = 0
a6 = 0
a7 = 0
a8 = 0
a9 = 0

a4 = 3.1
...
a8 = 5.2

x = a4 + a8
```

↑
**Tedioso y propenso
a generar errores**

**No es una multiplicación.
Repite 10 veces la lista [0]**

↓

```
a = 10*[0]
a[4] = 3.1
a[8] = 5.2

x = a[4] + a[8]
```

↑
Alternativa sencilla

```
a = 1000000*[0]
a[234567] = 3.1
a[891234] = 5.2

x = a[234567] + a[891234]
```

↑
**Y además puede escalar
a millones de elementos!**

Procesando elementos en listas

Utilizando ciclo for

```
1 meses = ['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio',  
2         'Julio', 'Agosto', 'Septiembre', 'Octubre', 'Noviembre',  
3         'Diciembre']  
4  
5 for mes in meses:  
6     print(mes)
```

Simple, menos propenso
a creación de bugs

Utilizando ciclo for + generación de posiciones

```
1 meses = ['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio',  
2         'Julio', 'Agosto', 'Septiembre', 'Octubre', 'Noviembre',  
3         'Diciembre']  
4  
5 n = len(meses) # tamaño lista meses  
6  
7 for i in range(n):  
8     print(meses[i])
```

1. Genera las posiciones en el arreglo
2. Recupera el elemento en la posición i

P: ¿Cuándo usar for, o for + range?
R: La opción más simple es la adecuada ;)

```
1 # calcula producto punto  
2 x = [0.30, 0.60, 0.10]  
3 y = [0.50, 0.10, 0.40]  
4 total = 0.0  
5 for i in range(len(x)):  
6     total += x[i]*y[i]  
7 print(total)
```




David Winterbottom
@codeinthehole

Follow



Desirable developer skills:

- 1 Ability to ignore new tools and technologies
- 2 Taste for simplicity
- 3 Good code deletion skills
- 4 Humility

9:17 AM - 3 Dec 2014

3,387 Retweets 3,382 Likes



88

3.4K

3.4K



<https://twitter.com/codeinthehole/status/540117725604216832>

Habilidades deseables para programadores:

1. Habilidad para ignorar nuevas herramientas y tecnologías
2. Gusto por la simplicidad
3. Buenas habilidades para eliminar código
4. Humildad

Resumen

Conceptos

- **for**: ejecutar código al recorrer una secuencia. La secuencia se puede generar con la función range(...)
- **Lista**: secuencia de elementos
- **String**: secuencia de caracteres (texto)
- **Continue**: saltar una iteración en ciclo while/for
- **Break**: detener un ciclo for/while

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

https://docs.python.org/3/reference/lexical_analysis.html

Funciones

- **range(stop)**: secuencias de enteros hasta stop-1
- **range(start, stop[, step])**: secuencia de enteros desde start hasta stop-1, saltándose step pasos.
- **len(lista)**: tamaño de una lista o de un string

		Built-in Functions		
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	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/3/library/functions.html>