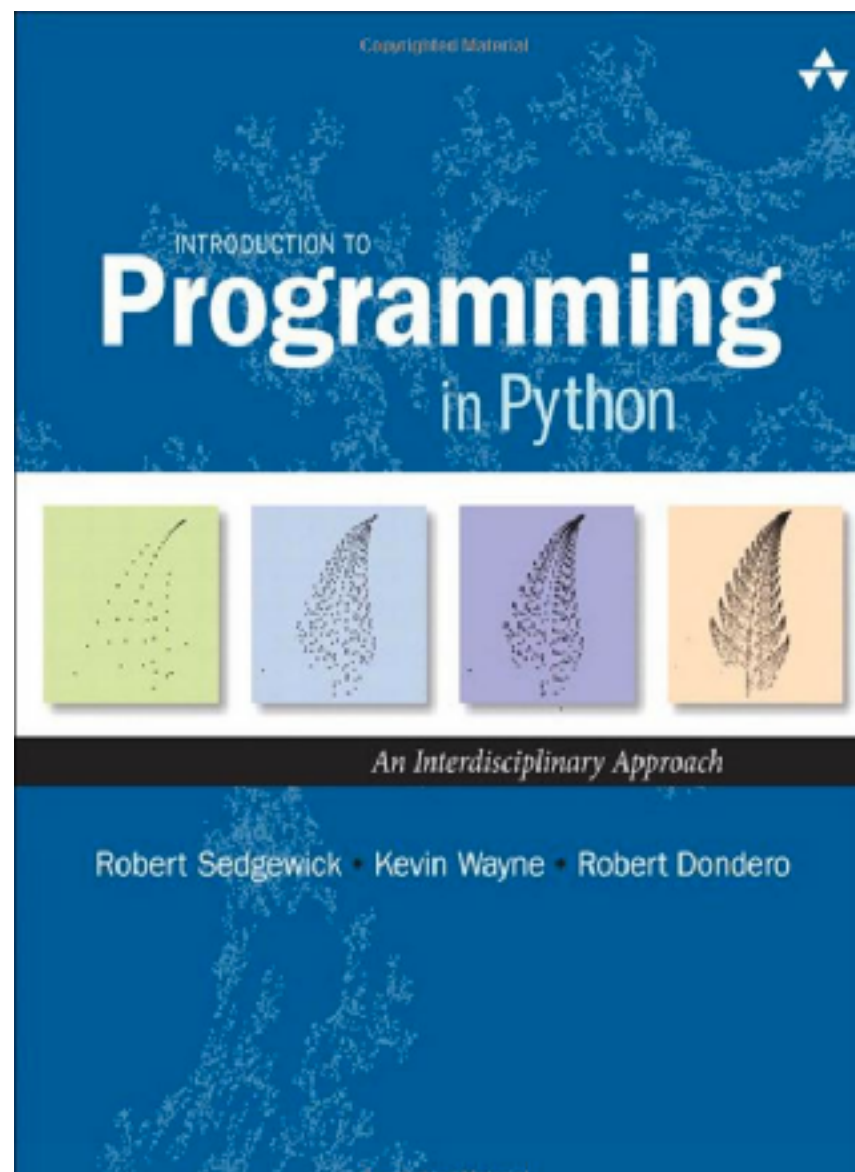


Parte I: Intro pensamiento computacional

Clase 04: Listas y Strings

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

- Administrivia: tarea 1 y detalles sobre la entrega
- Comprender la utilidad de las listas
- Identificar patrones de uso de procesamiento de datos con listas
- Conocer las operaciones básicas sobre strings (secuencias de textos)
- Detener y saltarse una iteración en un ciclo

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.

Desviación estándar

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

↑
Promedio

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

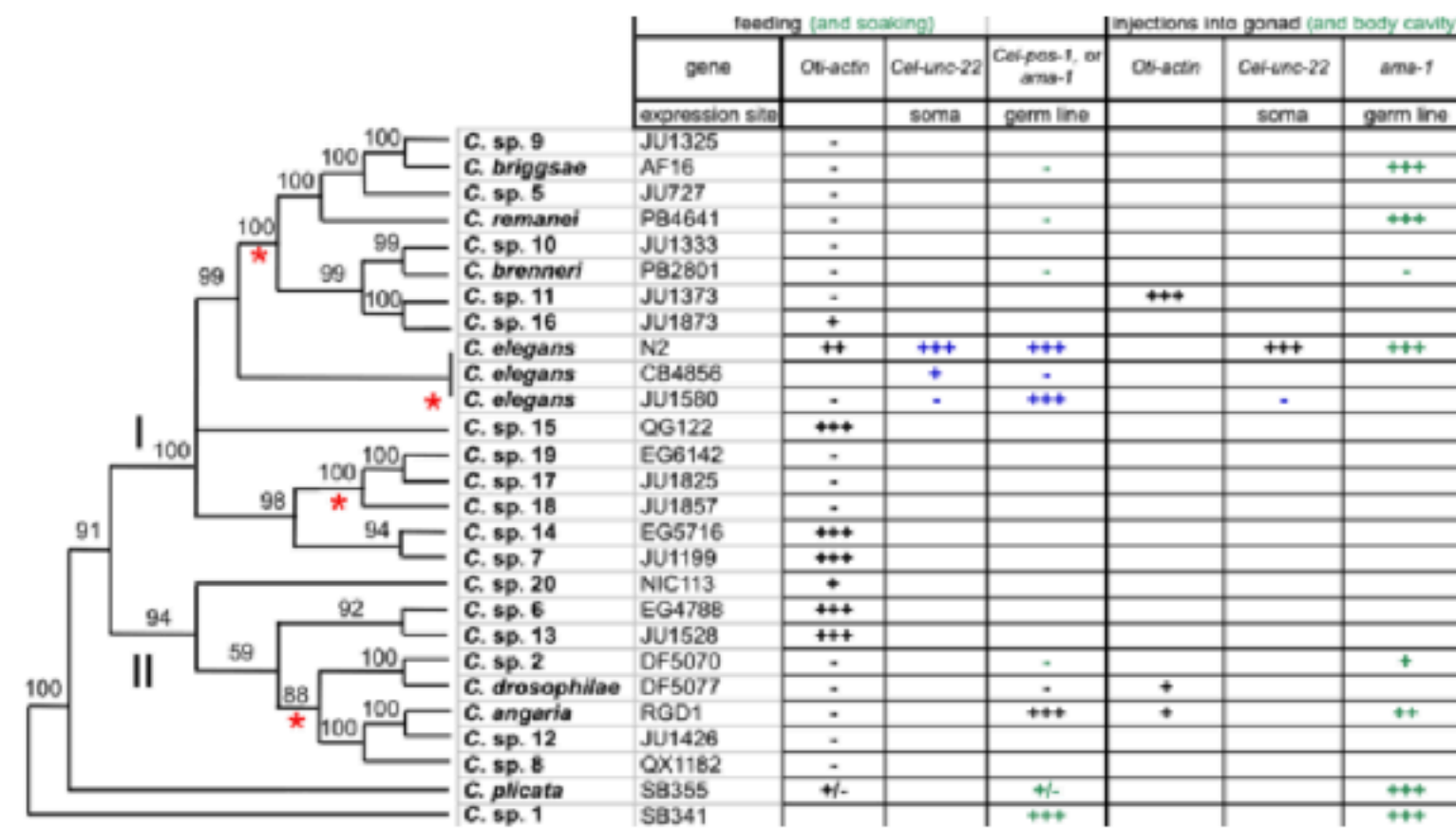
20 minutos de variación

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

Sin variación

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

P2: ¿Y si el número de días es 1000?



PLOS ONE PHYLOGENY/FICKER (CC BY 2.0)

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**



imgflip.com

**ANALIZAR DATOS
CON PYTHON**

<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)
```

Más operaciones con listas

- Append: agregar nuevo elemento a la lista
- Concatenar: unir dos listas
- Obtener sublista: L[inicio:fin]
- Contiene: elem in L (devuelve True o False)

```
>>> 'a' in ['b','c','d','a']  
True
```

```
1 L = [11, 3, 5, 7, 2]  
2 print('L', L)  
3  
4 if 5 in L:  
5     print('cinco está en L')  
6  
7 # Actualizar elemento  
8 L[4] = 9999  
9 print('L[4]=9999', L)  
10  
11 # Agregar elemento a listas  
12 L.append(100) #modifica lista  
13 print('L.append(100)', L)  
14  
15 # Concatenar lista  
16 L2 = L + [19, 17, 13] # crea lista nueva  
17 print('L+[19, 17, 13]', L2)  
18  
19 # Sublista  
20 L3 = L[2:5] # Elementos 2,3 y 4  
21 print('L[2:5]', L3)
```

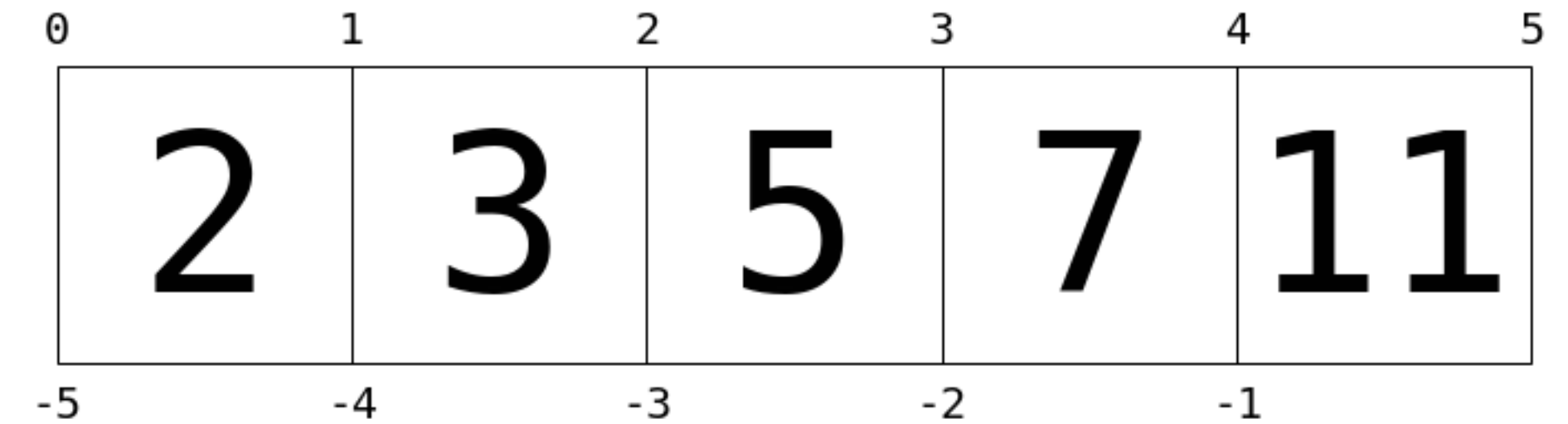
```
$ python3 ops.py  
L [11, 3, 5, 7, 2]  
cinco está en L  
L[4]=9999 [11, 3, 5, 7, 9999]  
L.append(100) [11, 3, 5, 7, 9999, 100]  
L+[19, 17, 13] [11, 3, 5, 7, 9999, 100, 19, 17, 13]  
L[2:5] [5, 7, 9999]
```


Desafíos

Crear lista con valores de teclado	<pre>L = [] #lista vacía for i in range(N): v = int(input()) L.append(v)</pre>	Obtener el promedio	<pre># promedio suma = 0.0 for elem in L: suma = suma + elem prom = suma/N</pre>
Imprimir valores en lista (uno por uno)	<pre>for elem in L: print(elem) # alternativa for i in range(N): print(L[i])</pre>	Copiar elementos a otra lista	<pre>L2 = [] for elem in L: L2.append(elem)</pre>
Encontrar el máximo valor en una lista	<pre>maxi = L[0] for elem in L: if elem > maxi: maxi = elem print(maxi)</pre>	Crear nueva listas con elementos invertidos	<pre>N = len(L) R = [] for i in range(N): j = N-i-1 R.append(L[j])</pre>
Encontrar el mínimo valor en una lista	<pre>mini = L[0] for elem in L: if elem < mini: mini = elem print(mini)</pre>	Invertir elementos del arreglo	<pre>for i in range(N): temp = L[i] L[i] = L[N-i-1] L[N-i-1] = temp</pre>

Más sobre acceso en listas

- Los elementos de la lista se pueden acceder con el operador corchete []
- Si la posición del elemento es negativo, se accede desde el final.
- Si accedes una posición que no existe: **ERROR!**



```
1 L = [2, 3, 5, 7, 11]
2 print('L[0]', L[0])
3 print('L[1]', L[1])
4
5 print('L[-1]', L[-1])
6 print('L[-2]', L[-2])
7 print('L[99]', L[99])
```

```
L[0] 2
L[1] 3
L[-1] 11
L[-2] 7
```

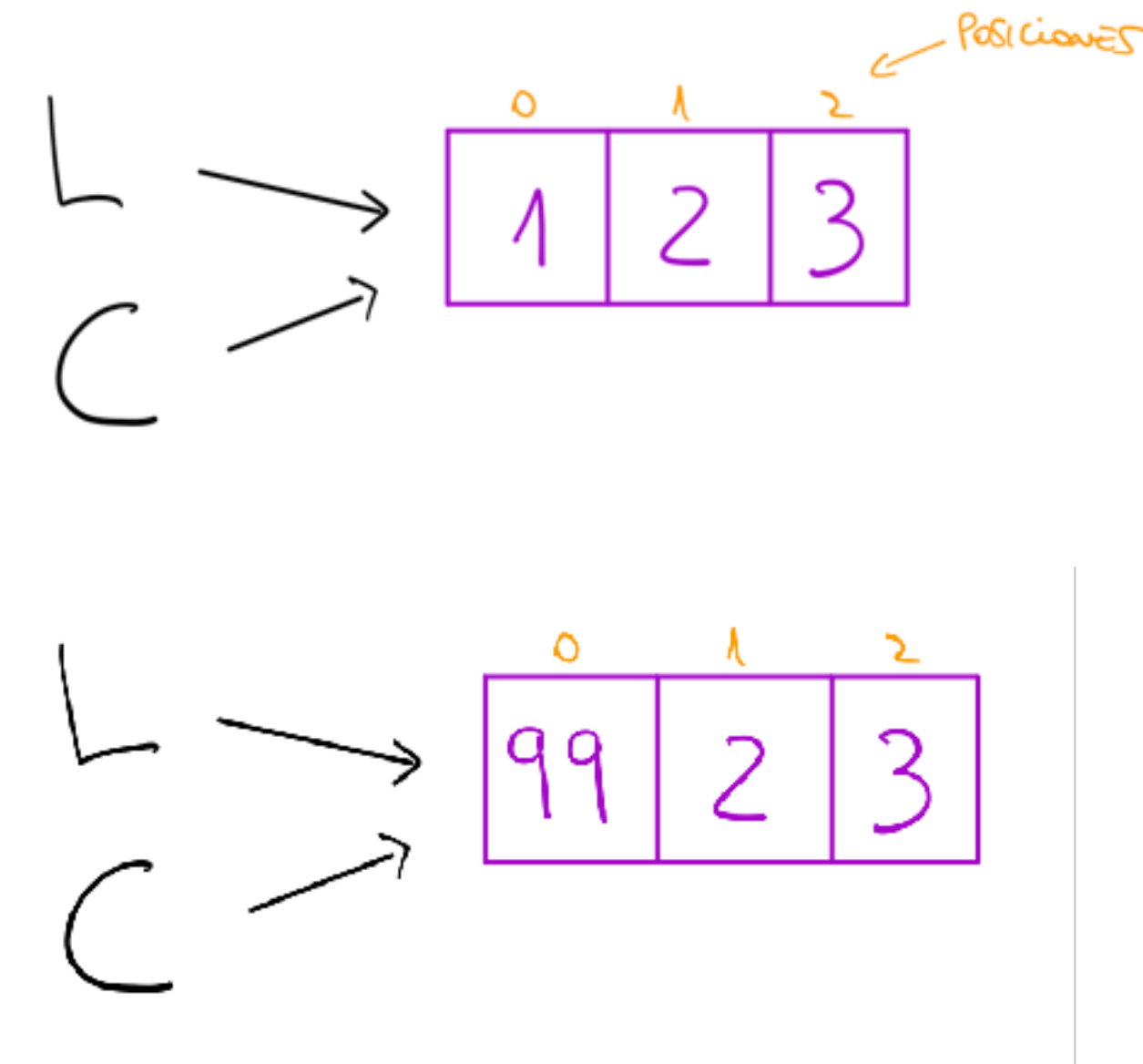
```
Traceback (most recent call last):
  File "lista-neg.py", line 7, in <module>
    print('L[99]', L[99])
IndexError: list index out of range
```

Error, programa se caerá.
Lista L tiene 5 elementos.

Variable alias

- Si ambos elementos son listas, el operador de asignación crea un nuevo nombre a la variable

```
1 L = [1, 2, 3]
2 C = L      Crea un alias de la lista
3 L[0] = 99
4
5 print(L)
6 print(C)
```



Importante: El operador de asignación '=' crea un alias (dos nombres para una misma variable).
Si quieres copiar una lista usa la función `.copy()`

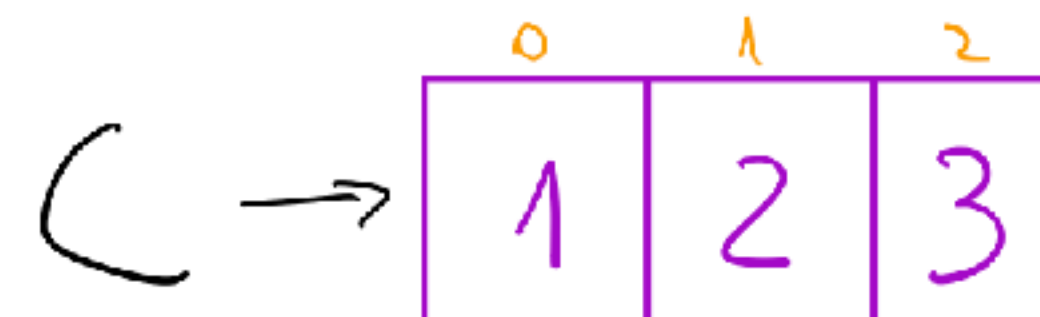
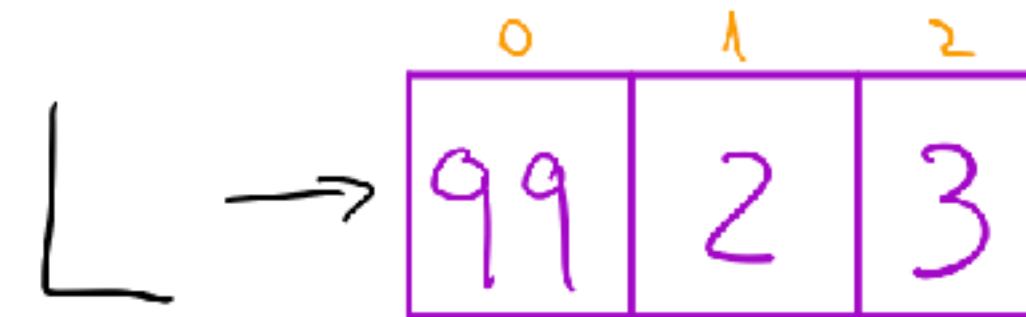
Variable alias

- Si ambos elementos son listas, el operador de asignación crea un nuevo nombre a la variable
- Para crear una copia usa lista.copy()

```
1 L = [1, 2, 3]
2 C = L
3 L[0] = 99
4
5 print(L)
6 print(C)
```

Reemplazar por

```
2 C = L.copy()
```



Importante: El operador de asignación '=' crea un alias (dos nombres para una misma variable).
Si quieres copiar una lista usa la función .copy()

Human-based python interpreter™

Genera números aleatorios
dentro de un rango



```
1 from random import randrange
2 TRAJES = ['Picas', 'Diamantes', 'Treboles', 'Corazones']
3 VALORES = ['2', '3', '4', '5', '6', '7', '8', '9', '10',
4            'Jota', 'Reina', 'Rey', 'As']
5
6 valor = randrange(0, len(VALORES))
7 traje = randrange(0, len(TRAJES))
8 print(VALORES[valor], 'de', TRAJES[traje])
```

```
1 from random import randrange
2 TRAJES = ['Picas', 'Diamantes', 'Treboles', 'Corazones']
3 VALORES = ['2', '3', '4', '5', '6', '7', '8', '9', '10',
4            'Jota', 'Reina', 'Rey', 'As']
5
6 mazo = []
7 for traje in TRAJES:
8     for valor in VALORES:
9         mazo.append(valor + ' de ' + traje)
10
11 print(mazo)
```

Strings

- Secuencia de caracteres, pero que no se puede actualizar.
- Operaciones básicas: tamaño, acceso, concatenación y obtener substring.

```
1 s = "Hola mundo!"
2 print('tamaño s', len(s))
3
4 # concatenar
5 s1 = 'Hola'
6 s2 = 'Chao'
7 s3 = s1 + s2
8 print(s3)
9
10 # acceso, la primera posición comienza en 0
11 print(s1[3]) # imprime el 4to element
12
13 # substring
14 s4 = s[1:6]
15 print('s[1:6] = ', s4)
16
17 # actualizar string: concatenar
18 nuevo = "m" + s[1:]
19 print(nuevo)
20
21 # actualizar string: reemplazar
22 nuevo2 = "m{}".format(s[1:])
23 print(nuevo2)
```

Si no se indica el fin del substring, se asume que llega hasta el final del string.
Si no se indica el inicio, se asume que comienza desde la posición 0.

```
$ python3 string.py
tamaño s 11
HolaChao
a
s[1:6] = ola m
mola mundo!
mola mundo!
```

Los strings son inmutables, es decir, no se pueden actualizar. Debes crear uno nuevo.

```
>>> s = 'Mi super texto'
>>> s[0] = 'm'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

Operaciones básicas sobre strings

Leer string desde entrada estándar	<code>s = input()</code>
Tamaño del string	<code>len(s)</code>
Obtener carácter en posición i	<code>ch = s[i]</code>
Copiar string	<code>b = s # aquí si funciona la copia!</code>
Comparar dos strings	<pre>if c == "gatito": print("c es igual a mensaje") else: print("c es distinto a mensaje")</pre>
Concatenar dos o más strings	<code>b = s + "más texto";</code>
Extraer j caracteres desde posición i	<code>c = s[i:i+j]</code>
Convertir string a int	<code>j = int(s)</code>
Convertir int a string	<pre>i = 9543 numero = str(i)</pre>
Encontrar substring dentro de string	<pre>mensaje = "la udd la lleva" if 'udd' in mensaje: print('todo bien!') else: print('buuuu')</pre>
Concatenar una lista de strings	<pre>L = ['uno', 'dos', 'tres'] s = ','.join(L) print(s) # imprime: 'uno,dos,tres'</pre>

Desafíos

Cuando se puede leer
igual de atrás hacia
adelante. Ej: ana



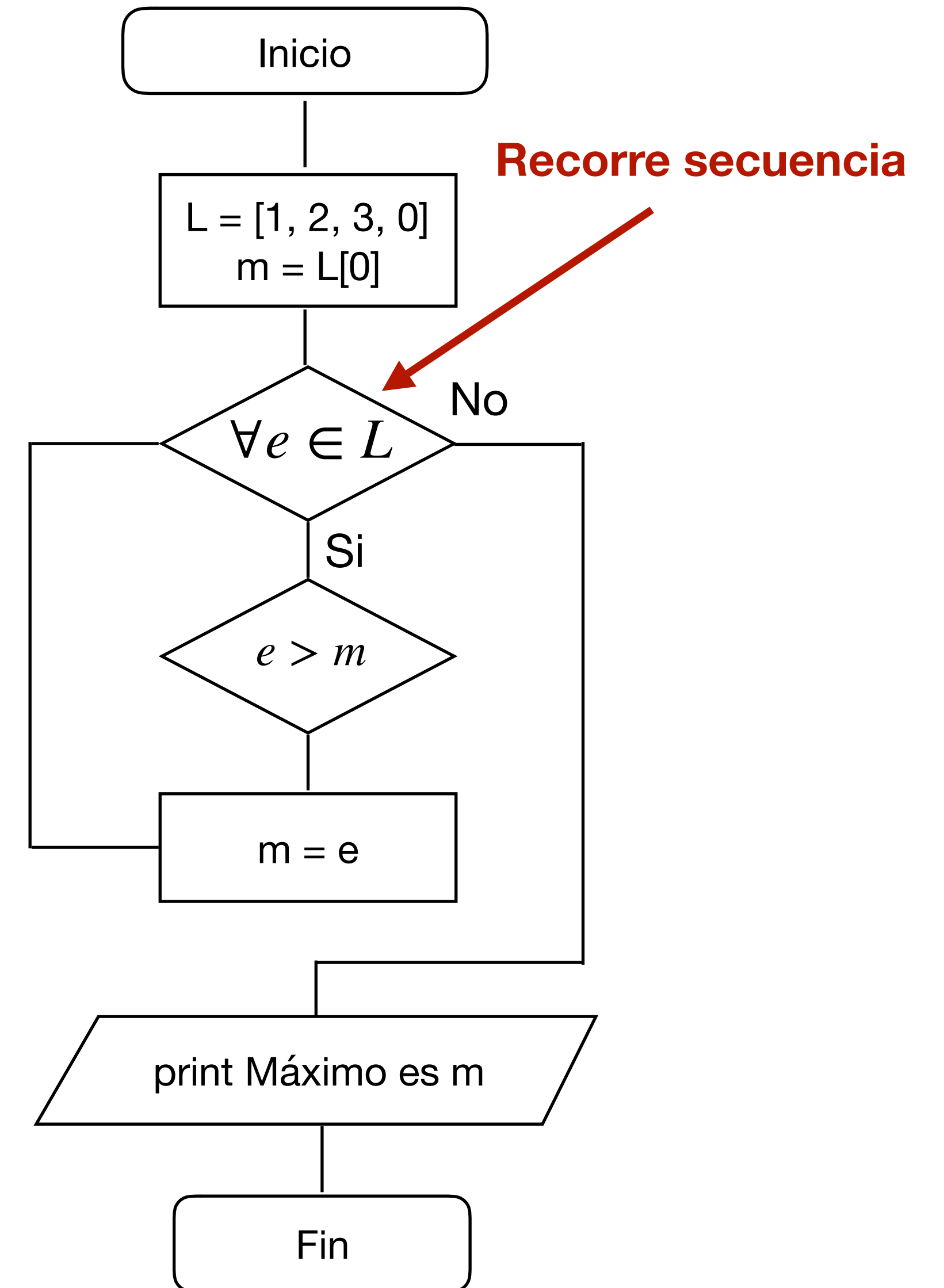
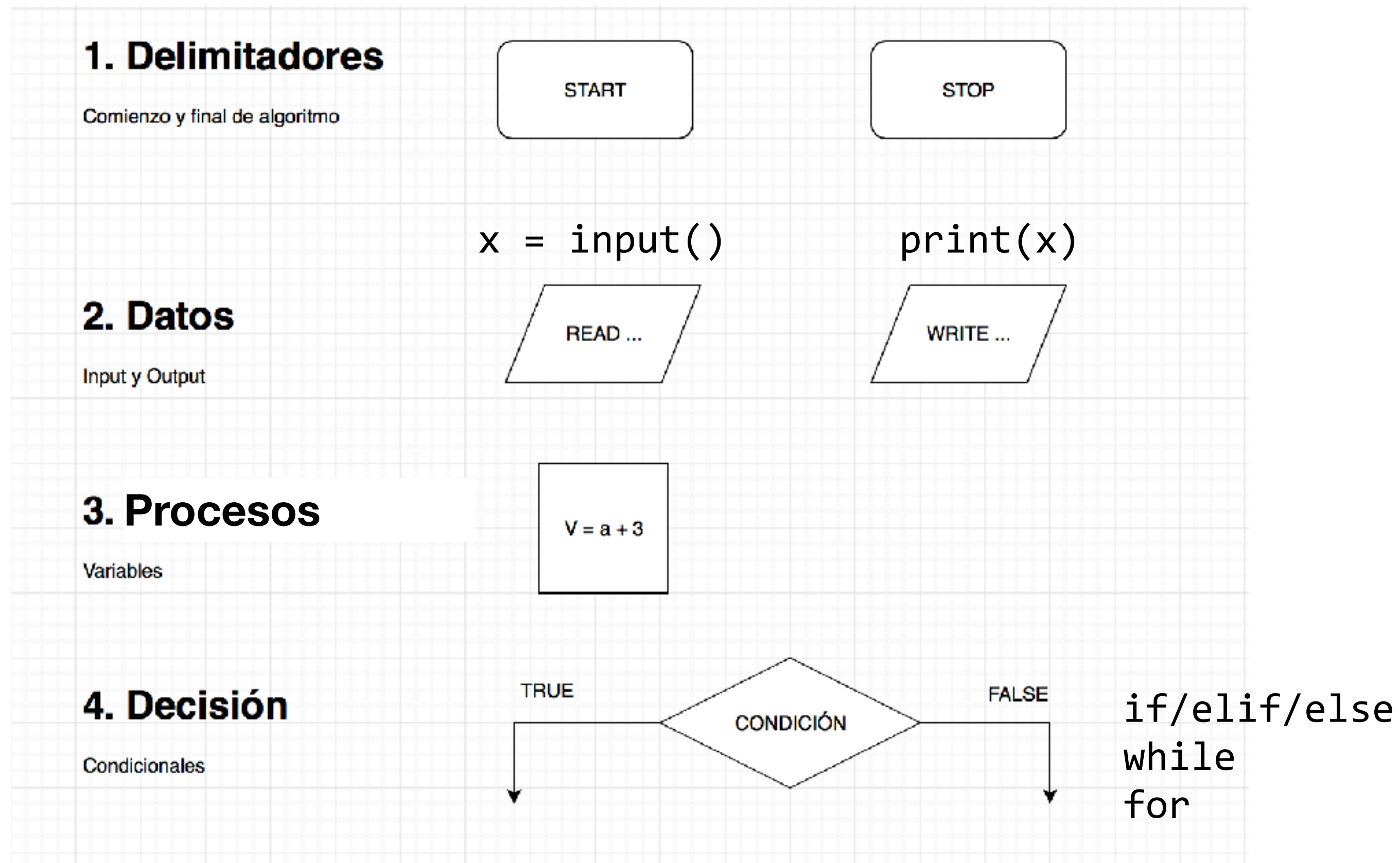
**Chequear si una palabra es
palíndromo**

```
palabra = 'anitalavalatina'
N = len(palabra)
palindromo = True
for i in range(N//2):
    if palabra[i] != palabra[N-i-1]:
        palindromo = False
        break #detiene ciclo for
print(palindromo)
```

**Extraer nombre y extensión de un
archivo**

```
nombre = 'hola.py'
partes = []
for i in range(len(nombre)):
    if nombre[i] == '.':
        n = nombre[0:i]
        e = nombre[i+1:]
        partes.append(n)
        partes.append(e)
        break
print('nombre archivo:', partes[0])
print('extension archivo:', partes[1])
```


Recordatorio: diagramas de flujo



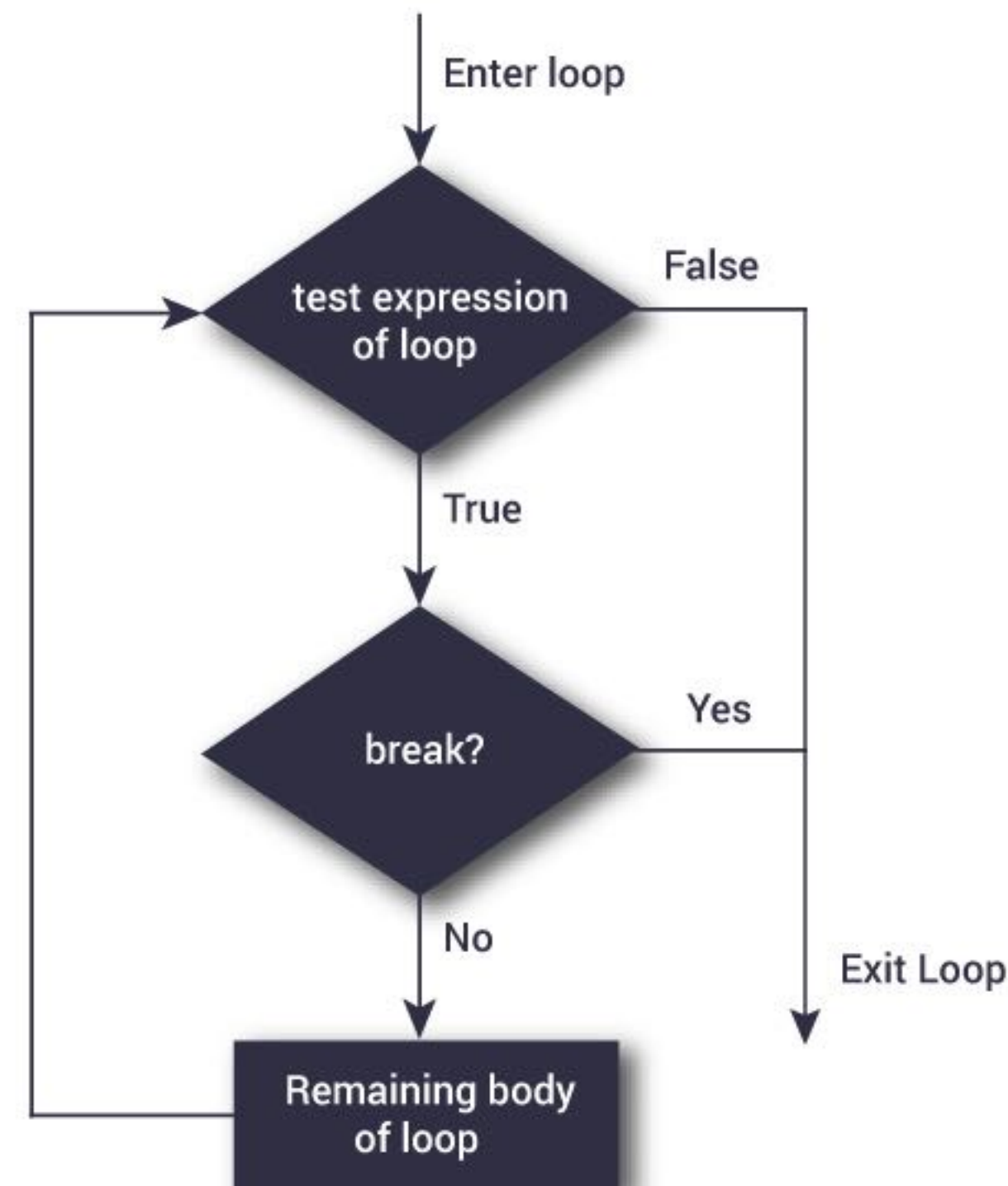
Ciclos II: break

- 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 (muuuuuy 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.

¿Qué hacen los programas a, b, c y d?

a)

```
1 L = 1000000*[0, -1, 3, 5, 9, 10, 12, 99, 33]
2 print('len(L):', len(L))
3
4 a = False
5 for e in L:
6     if e < 0:
7         a = True
8         break
9 print(a)
```

b)

```
1 L = 1000000*[0, -1, 3, 5, 9, 10, 12, 99, 33]
2 print('len(L):', len(L))
3
4 a = False
5 for e in L:
6     if e < 0:
7         a = True
8 print(a)
```

c)

```
1 L = 1000000*[0, -1, 3, 5, 9, 10, 12, 99, 33]
2 print('len(L):', len(L))
3
4 a = False
5 i = 0
6 while i < len(L):
7     if L[i] < 0:
8         a = True
9         break
10    i += 1
11 print(a)
```

d)

```
1 L = 1000000*[0, -1, 3, 5, 9, 10, 12, 99, 33]
2 print('len(L):', len(L))
3
4 a = False
5 i = 0
6 while i < len(L):
7     if L[i] < 0:
8         a = True
9     i += 1
10 print(a)
```

P: ¿Cuál de todos te gusta más? ¿Por qué?



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

Ciclos II: continue

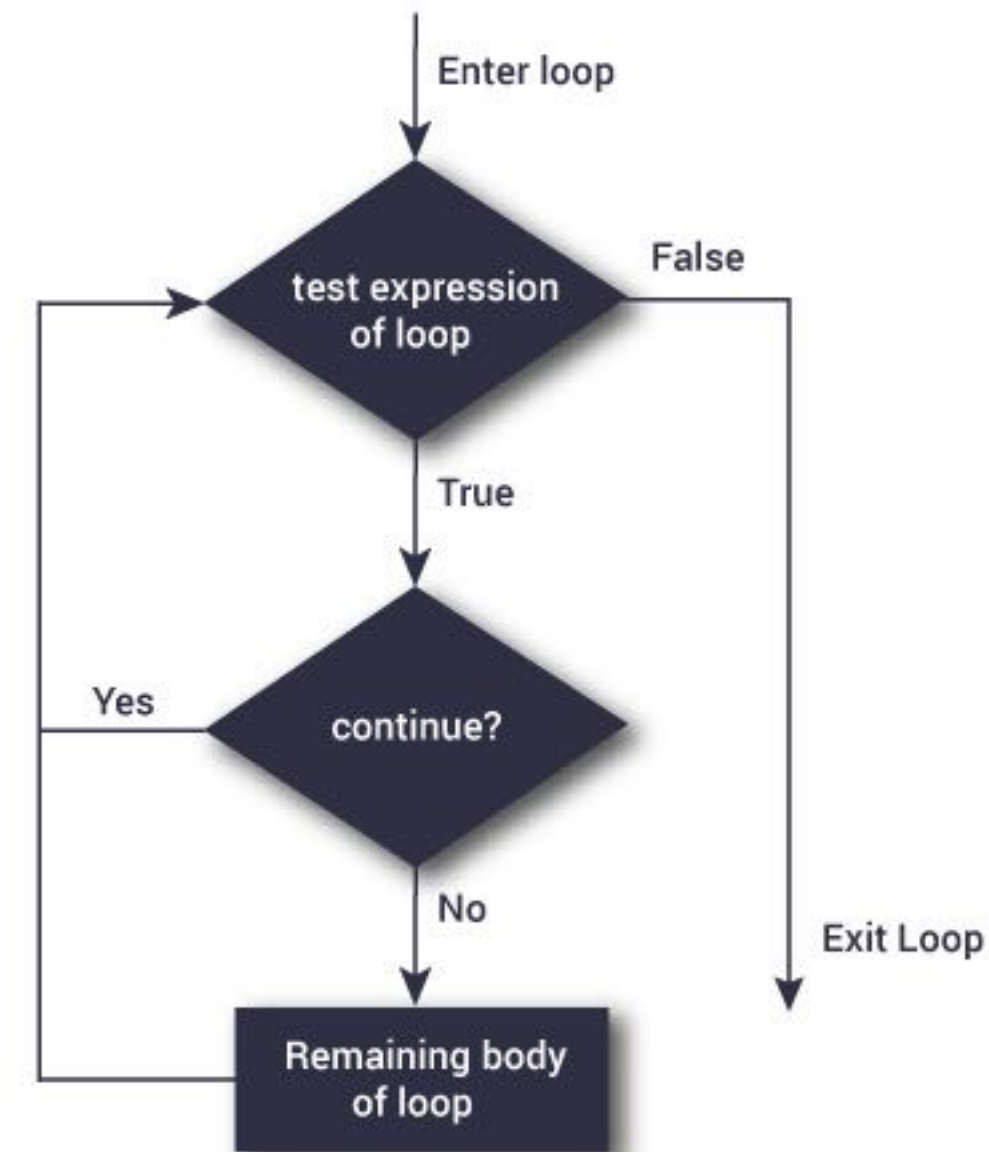
- A veces queremos saltarnos alguna iteración (ej.: ignorar elementos negativos). Puedes saltarlos usando **continue**.
- **Ventaja:** podemos ahorrar tiempo de procesador (muuuuuy poco).
- **Desventaja:** código más complejo.

```
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.

¿Qué hacen los programas a, b, c y d?

a)

```
1 L = [0, -1, 3, 5, 9, 10, 12, 99, 33]
2 t = 0
3 for e in L:
4     if e < 0:
5         continue
6     t += e
7 print(t)
```

Salta a siguiente iteración

b)

```
1 L = [0, -1, 3, 5, 9, 10, 12, 99, 33]
2 t = 0
3 for e in L:
4     if e >= 0:
5         t += e
6 print(t)
```

c)

```
1 L = [0, -1, 3, 5, 9, 10, 12, 99, 33]
2 t = 0
3 i = 0
4 while i < len(L)
5     e = L[i]
6     i += 1
7     if e < 0:
8         continue
9     t += e
10 print(t)
```

Salta a siguiente iteración

d)

```
1 L = [0, -1, 3, 5, 9, 10, 12, 99, 33]
2 t = 0
3 i = 0
4 while i < len(L)
5     e = L[i]
6     i += 1
7     if e >= 0:
8         t += e
9 print(t)
```

P: ¿Cuál de todos te gusta más? ¿Por qué?



David Winterbottom
@codainthole

Follow

Desirable developer skills:

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

Ejercicio 1

El profesor Rossa tiene problemas para usar el computador, así que normalmente calcula el promedio de notas de sus alumnos usando papel y lápiz. Cuando el profesor entregó los promedios, Guru-guru se dió cuenta que su promedio no correspondía a sus calificaciones.

La coordinadora académica de la Facultad le pidió a usted diseñar un programa que permita calcular el promedio de notas, para ayudar al profesor Rossa.

Si usted desea ayudar al profesor Rossa, y hacer justicia con Guru-Guru, resuelva lo siguiente:

1. Escriba un programa que calcule el promedio. Asuma que se le entrega una lista con n números, cada uno de ellos representando una nota y que todas las notas tienen la misma ponderación.
2. Calcule la desviación estándar del promedio de notas del curso.
3. Calcule el promedio ponderado, asuma que le entregan otra lista con n números flotantes representando el porcentaje que representa cada nota.

Resumen

Conceptos

- **Lista:** secuencia de elementos
- **String:** secuencia de caracteres (texto)
- **Alias:** nuevo nombre a una variable. Si modifico el contenido en una, se modifica en la otra también.
- **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

- **len(lista):** tamaño de una lista o de un string
- **elem.copy():** crear copia de variable elem

		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>