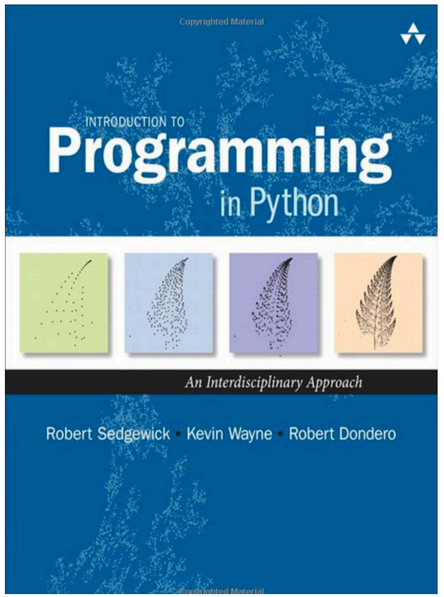
Taller de Programación

Clase 06: Break y Continue

Daniela Opitz, Diego Caro
dopitz@udd.cl



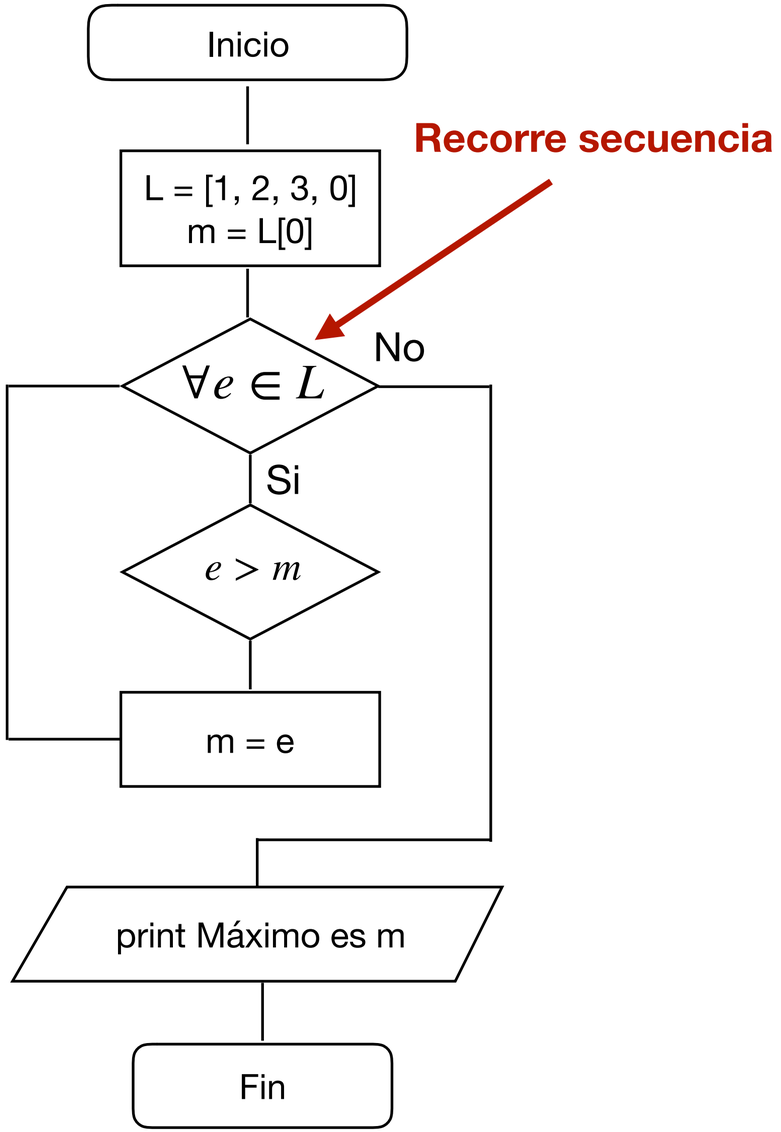
Basada en presentaciones oficiales de libro Introduction to Programming in Python (Sedgewick, Wayne, Dondero).

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

Outline

- Repaso Listas
- Break
- Continue

Listas en Diagramas Lógicos



Ciclos II: `break`

- **`break`**: Sirve parar detener ciclos antes de que se recorra una secuencia o la condición en **`while`** no se cumpla.
  - **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
```

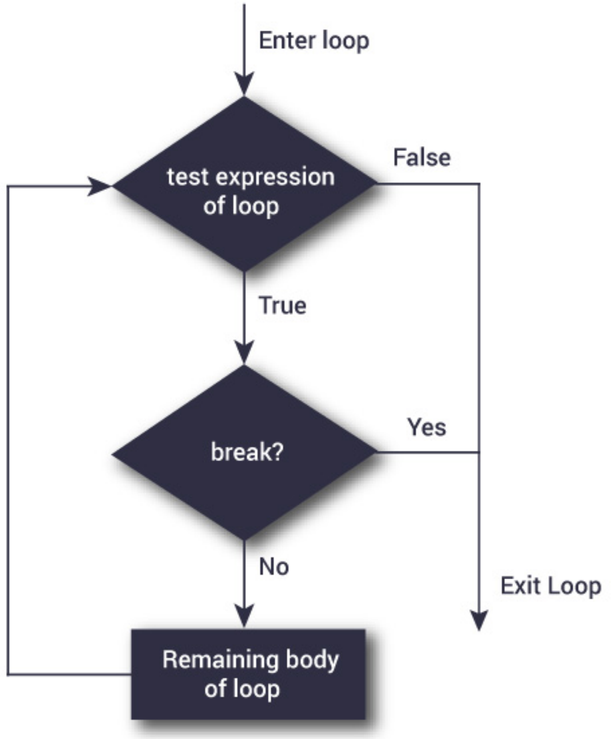


Imagen extraída desde https://www.programiz.com/python-programming/break-continue

Ciclos II: `break`

```
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)

```
L = 10000000*[0, -1, 3, 5, 9, 10, 12, 99, 33]
print('len(L):', len(L))

a = False
for e in L:
    if e < 0:
        a = True
        break
print(a)
```

b)

```
L = 10000000*[0, -1, 3, 5, 9, 10, 12, 99, 33]
print('len(L):', len(L))

a = False
for e in L:
    if e < 0:
        a = True
print(a)
```

c)

```
L = 10000000*[0, -1, 3, 5, 9, 10, 12, 99, 33]
print('len(L):', len(L))

a = False
i = 0
while i < len(L):
    if L[i] < 0:
        a = True
        break
    i += 1
print(a)
```

d)

```
L = 10000000*[0, -1, 3, 5, 9, 10, 12, 99, 33]
print('len(L):', len(L))

a = False
i = 0
while i < len(L):
    if L[i] < 0:
        a = True
    i += 1
print(a)
```

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

Actividad

1. Extraer nombre y extensión de un archivo

   Al recibir archivo.py el programa dede retornar nombre: archivo, extensión: py

Actividad 2

2. Escribir un código que chequea si una palabra es palíndromo, es decir una palabra que se lee igual tanto de derecha a izquierda como de izquierda a derecha.

Ciclos II: `continue`

- **`continue`:** Sirve para saltar alguna iteración (ej.: ignorar elementos negativos).
  - **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
```

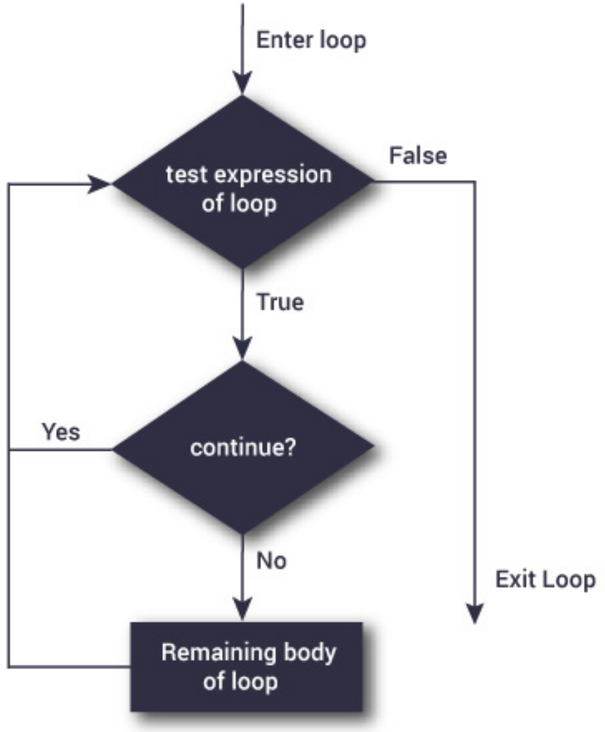


Imagen extraída desde https://www.programiz.com/python-programming/break-continue

Ciclos II: `continue`

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

```
$ python3 simple-continue.py
h
o
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?

Actividad

El profesor Rossa tiene problemas para usar el computador, así que normalmente calcula el promedio de notas de 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 carácteres (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