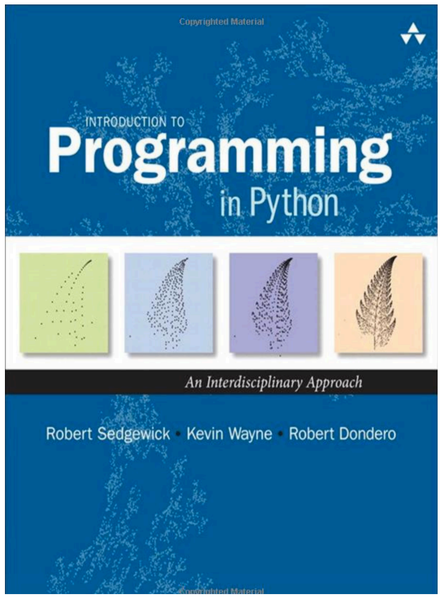
Taller de Programación

Clase 08: Break, Continue & Strings

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

- Ciclos: Break
- Ciclos: Continue
- Operaciones con Strings

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
```

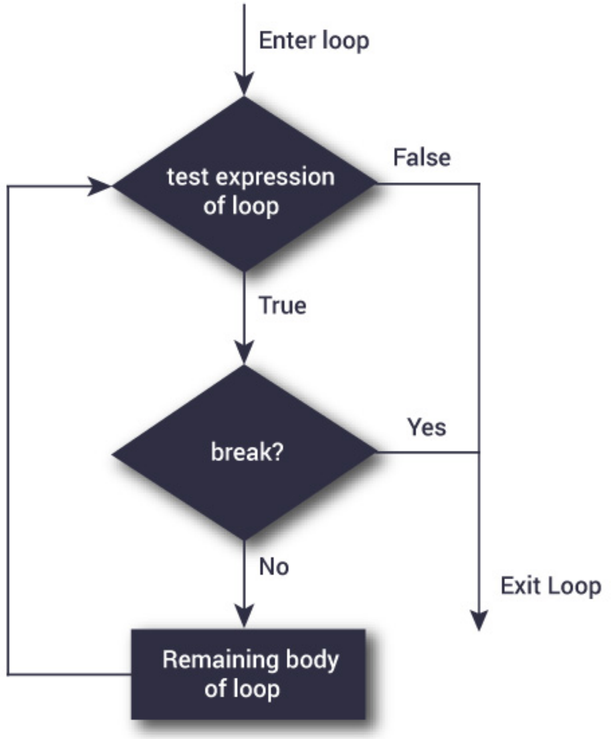


Imagen extraída desde https://www.programiz.com/python-programming/break-continue

Ciclos II: `break`

```
L=[1,3,4,0,8]

for e in L:
    if e == 0:
        break
    print(e)
```

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?

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
```

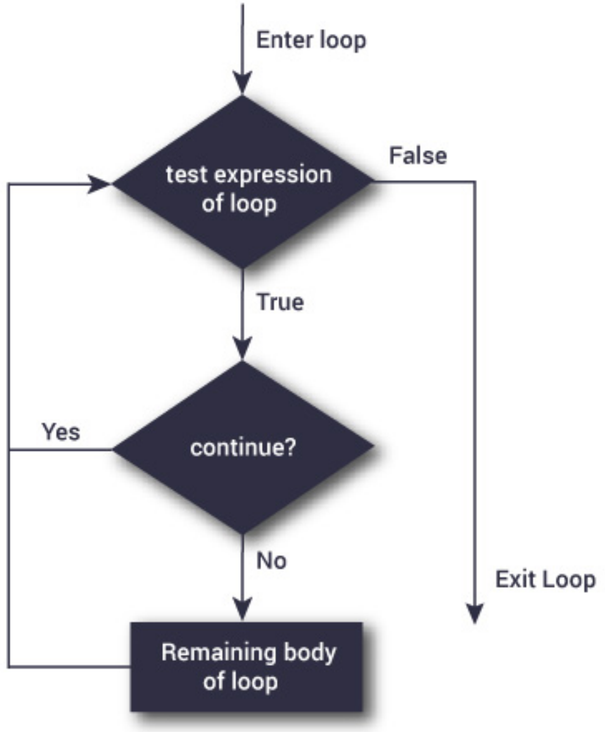


Imagen extraída desde https://www.programiz.com/python-programming/break-continue

Ciclos II: `continue`

```
L=[1,3,4,0,8]

for e in L:
    if e == 0:
        continue
    print(e)
```

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?

Strings

- Secuencia de caracteres
- 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)
```

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

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.

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

Leer string desde entrada estándar	`s = input()`
Tamaño del string	`len(s)`
Obtener carácter en posición i	`ch = s[i]`
Copiar string	`b = s # aquí si funciona la copia!`

Comparar dos strings	`if c == "gatito":`
	`  print("c es igual a mensaje")`
	`else:`
	`  print("c es distinto a mensaje)`

Concatenar dos o más strings	`b = s + "más texto";`
Extraer j caracteres desde posición i	`c = s[i:i+j]`
Convertir string a int	`j = int(s)`
Convertir int a string	`i = 9543` `numero = str(i)`

Encontrar substring dentro de string	`mensaje = "la udd la lleva"`
	`if 'udd' in mensaje:`
	`  print('todo bien!')`
	`else`
	`  print('buuuu')`

Concatenar una lista de strings	`L = ['uno', 'dos', 'tres']`
	`s = ','.join(L)`
	`print(s) # imprime: 'uno,dos,tres'`

Invertir un `string`

- Hay muchas formas de hacerlo. Busque en Google diversas alternativas

- Dos caminos:

```
s1= '' #str vacío, no es un espacio
s2='Python'
for c in s2:
    s1 = c + s1
print(s1)
```

```
s3='Python'
print(s3[::-1])
```

Use la que usted quiera!

Codificación

- ASCII: Metodo para representar caracteres utilizando 7 bytes

Binario	Dec	Hex	Representación
0010 0000	32	20	espacio ( )
0010 0001	33	21	!
0010 0010	34	22	"
0010 0011	35	23	#
0010 0100	36	24	$
0010 0101	37	25	%
0010 0110	38	26	&
0010 0111	39	27	'
0010 1000	40	28	(
0010 1001	41	29	)
0010 1010	42	2A	*
0010 1011	43	2B	+
0010 1100	44	2C	,
0010 1101	45	2D	-
0010 1110	46	2E	.

Binario	Dec	Hex	Representación
0100 0000	64	40	@
0100 0001	65	41	A
0100 0010	66	42	B
0100 0011	67	43	C
0100 0100	68	44	D
0100 0101	69	45	E
0100 0110	70	46	F
0100 0111	71	47	G
0100 1000	72	48	H
0100 1001	73	49	I
0100 1010	74	4A	J
0100 1011	75	4B	K
0100 1100	76	4C	L
0100 1101	77	4D	M
0100 1110	78	4E	N

Binario	Dec	Hex	Representación
0110 0000	96	60	`
0110 0001	97	61	a
0110 0010	98	62	b
0110 0011	99	63	c
0110 0100	100	64	d
0110 0101	101	65	e
0110 0110	102	66	f
0110 0111	103	67	g
0110 1000	104	68	h
0110 1001	105	69	i
0110 1010	106	6A	j
0110 1011	107	6B	k
0110 1100	108	6C	l
0110 1101	109	6D	m
0110 1110	110	6E	n

https://es.wikipedia.org/wiki/ASCII

Codificación

Unicode es un estándar de codificación de caracteres diseñado para tratar, transmitir y visualizar textos de numerosos idiomas y disciplinas técnicas.

Unicode define tres formas de codificación bajo el nombre **UTF** (Unicode transformation format: formato de transformación Unicode):

- UTF-8:
- UTF-16:
- UTF-32:

La forma de codificación más utilizada es el UTF-8 (Unicode Transformation Format-8) en la cual un carácter puede ser representado por uno hasta cuatro bits.

Actividad

1. Extraer nombre y extensión de un archivo. Al recibir archivo.py el programa debe 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.

```
Ingrese una palabra: anitalavalatina

¿Es palíndromo?: True



Ingrese una palabra: programacion
¿Es palíndromo?: False
```

Resumen

Conceptos

- **Lista:** secuencia de elementos
- **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
- **String:** secuencia de carácteres (texto)

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