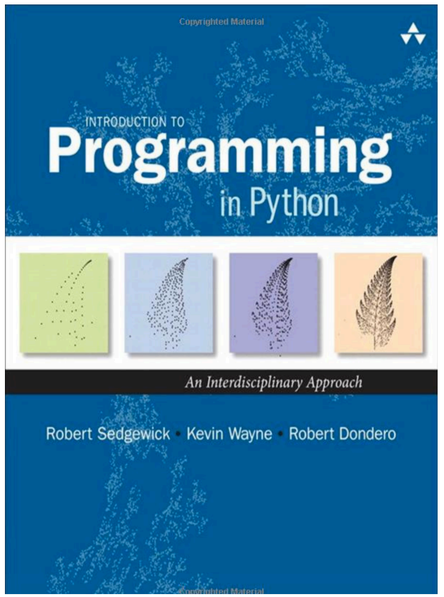
Taller de Programación

Clase 06: 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

- Operaciones con Strings
- Ciclos: Break & Continue

Strings

- Secuencia de caracteres
- Operaciones básicas: tamaño, acceso, concatenación y obtener substring.

```python
s = 'Hola mundo!'
print('Tamaño s', len(s))

# concatenar
s1 = 'Hola'
s2 = 'Chao'
s3 = s1 + s2
print(s3)

# acceso, la primera posición comienza en 0
print(s1[3]) # imprime el 4to element

# substring
s4 = s[1:6]
print('s[1:6] = ', s4)

# actualizar string: concatenar
nuevo = "m" + s[1:]
print(nuevo)

# actualizar string: reemplazar
nuevo2 = "m{}".format(s[1:])
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

- 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!

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
```

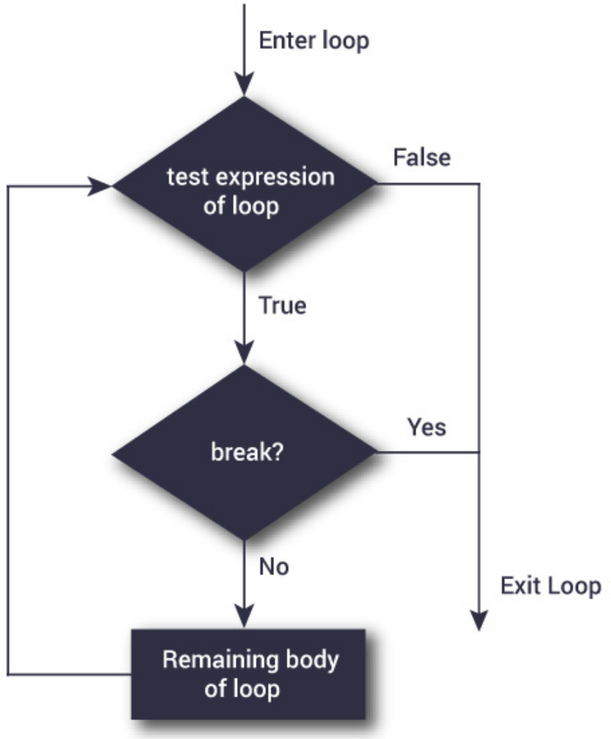


Imagen extraída desde https://www.programiz.com/python-programming/break-continue

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.

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
```

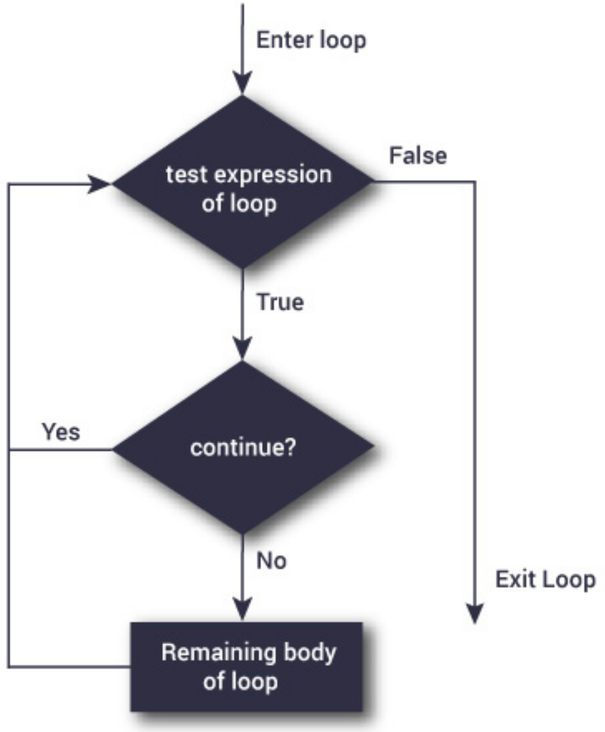


Imagen extraída desde https://www.programiz.com/python-programming/break-continue

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.

Métodos para Trabajar con Strings

- `str.strip()`: Borra todo lo que está antes y después de la cadena
- `str.lstrip()`: Borra lo que está al inicio de la cadena
- `str.rstrip()`: Borra lo que está al final de la cadena
- `str.replace(str1, str2, n)`: Reemplaza la cadena str1 por la cadenas str2 una máximo de n veces.
- `str.split(separador, maxsplit)`: Retorna una lista de strings después de romper la cadena por un separador específico.

Métodos para Trabajar con Strings

- `s.isdigit()`: verifica si un string s es un digito, devuelve `True` si es un dígito y `False` si no lo es.
- `''.join(L)`: convierte una lista L en un string en donde cada elemento es separado por string entre las comillas, en este caso un espacio. Por ejemplo si `L=['1', '2', '3']`, `''.join(L)` devuelve `'1 2 3'`
- `s1.index(s2)`: retorna la posición en s1 donde esta el carácter s2. Por ejemplo si `s1='Hola'`, `s1.index('H')` devuelve 0.

Métodos para Trabajar con Strings

```
cadena = "- - - programaresmuydivertido - - -"

#Usando strip() para borrar todos los '-'
print ("String después de remover '-' antes y después: ")
print (cadena.strip('-'))

# Usando lstrip para borrar los '-' anteriores
print ("String después de remover todos '-' anteriores: ")
print (cadena.lstrip('-'))

# Usando rstrip para borrar los '-' posteriores
print ("String después de remover todos los '-' posteriores: ")
print (cadena.rstrip('-'))
```

Métodos para Trabajar con Strings

```
# Usando split() para separar
x = 'azul,rojo,verde'
print('String para separar por comas: ' + x + '\n')

print('String para separar por comas: ' + str(x.split(",")) + '\n')
print(x.split(",", 1))

#Usando replace() para reemplazar
print("String despues de reemplazar 'divertido' por 'aburrido'")
print(cadena.replace('divertido', 'aburrido', 1))
```

Actividad 1 y 2

1. Extraer nombre y extensión de un archivo. Al recibir archivo.py el programa dede retornar nombre: archivo, extensión:

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.

Actividad 3

Programe un código que encripte una palabra ingresada por teclado e imprima la versión encriptada. El programa debe recibir la palabra a encriptar y retornar la palabra encriptada. El proceso de encriptación que debe implementar se describe a continuación:

1. Existe la palabra mágica “**murcielago**”, que actúa como clave de encriptación.
2. La palabra a encriptar se compara, carácter por carácter con la palabra mágica, y dependiendo del resultado de la comparación, ciertos caracteres de la palabra a encriptar son reemplazados. Para esto existen dos casos:
   - Si el carácter de la palabra a encriptar existe en la palabra mágica, este se reemplazará por la posición en la palabra mágica donde el carácter se encuentra.
     Si el carácter buscado no existe en la palabra mágica, el caracter de la palabra a encriptar no se cambiará.
   - El resultado final de los reemplazos de caracteres, dan origen a la palabra encriptada. **Ejemplo:** el programa al recibir la palabra **‘mundo’** retorna **‘01nd9’** . Esto es porque m está en la posición 0 de la pálabra mágica, u está en la posición 1 , n y d no se encuentran, y o está en la posición 9 .

3.

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