Programación en Python I

Ing. Wilmer Garzón Alfonso

wilmer.garzon@escuelaing.edu.co

www.wilmergarzon.com.co

Agenda

- Condicionales No Repetitivos: if; if – else; if - elif - else
- Condicionales Repetitivos: while; for
- Estructuras anidadas
- Listas - Operaciones
- Strings
- Ord() – Chr()
- Caracteres de escape
- Split() – Join()
- Referencias

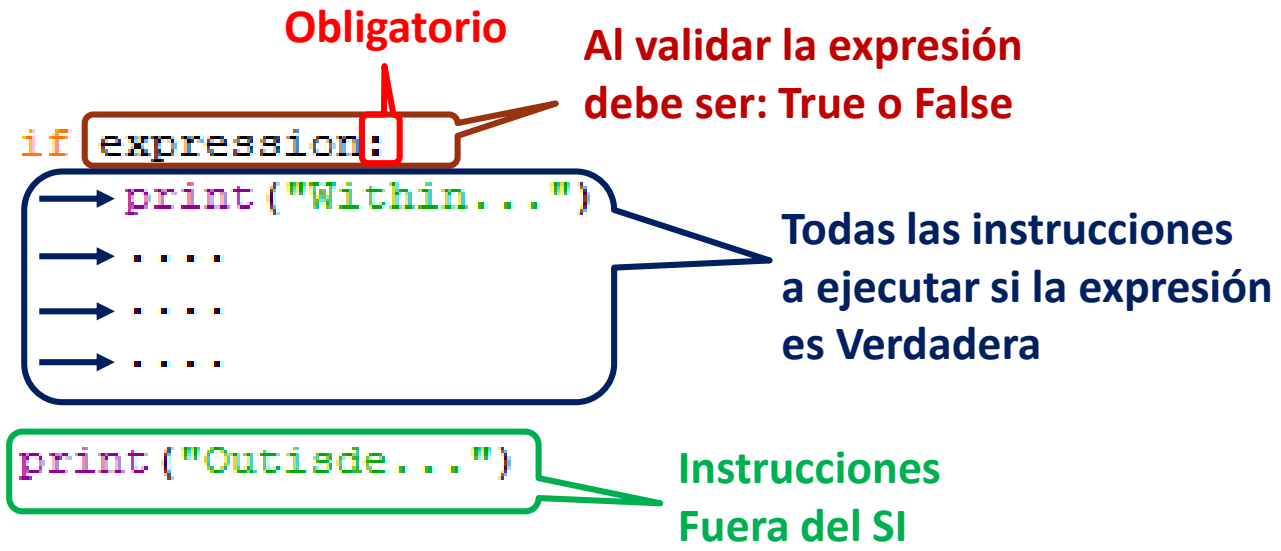
Condicionales

- Dos tipos de condicionales: **NO** repetitivos y **SI** repetitivos
- Día a día llevamos a cabo acciones con base en algunas condiciones.
  - Si tengo sed, entonces tomo agua
  - Si no estudio, entonces tendré malas notas
  - Mientras llueva, llevo la sombrilla abierta
  - Mientras tengo hambre, debo comer

…..

NO repetitivos - if

- Las sentencias terminan en nueva línea
- Los bloques son indicados por una tabulación



if

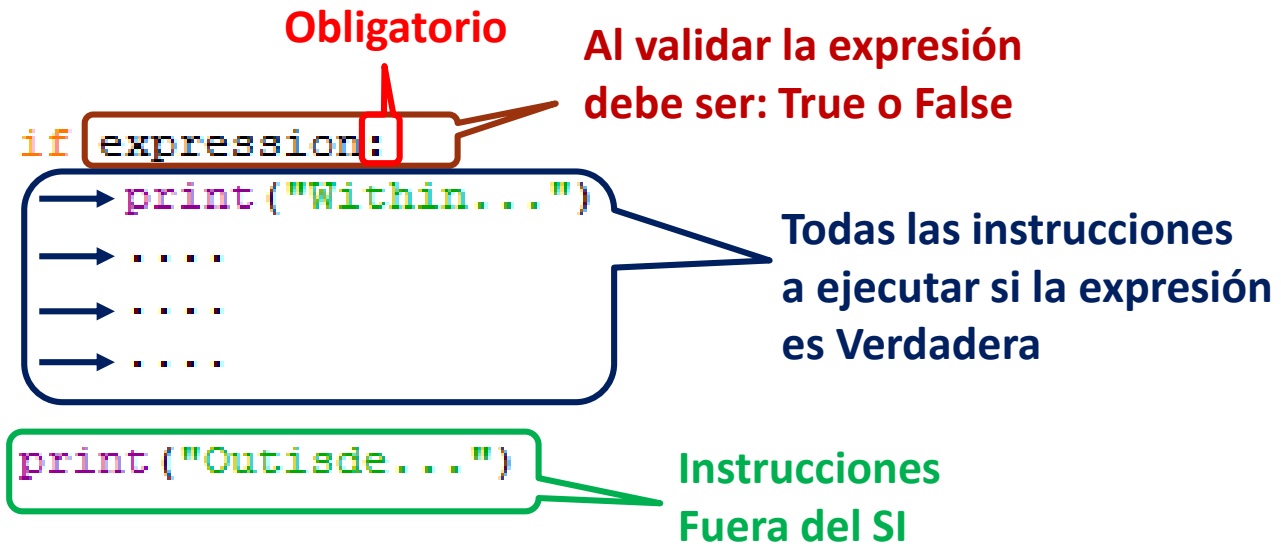
```
x = int(input("Ingresa un número:"))

if x%2==0:
    print("El número", x, " es PAR")
    print("Estuve por ACA")

if x%2!=0:
    print("Estuve por AQUÍ")
    print("El número", x, " es IMPAR")

print("Byee")
```

```
>>>
Ingresa un número:20
El número 20  es PAR
Estuve por ACA
Byee
```

```
>>>
Ingresa un número:11
Estuve por AQUÍ
El número 11  es IMPAR
Byee
```

if

- Las sentencias terminan en nueva línea
- Los bloques son indicados por una tabulación



if - else

```
x = int(input('Ingrese un entero: '))
if x%2 == 0:
    print('Número Par')
else:
    print('Número Impar')

print('Programa realizado con Condicionales')
```

```
>>>
Ingrese un entero: 25
Número Impar
Programa realizado con Condicionales
```

```
>>>
Ingrese un entero: 58
Número Par
Programa realizado con Condicionales
```

if – elif -else

```
x = int(input("Ingresa un número:"))

if x>0:
    print("El número", x, " es POSITIVO")
elif x<0:
    print("El número", x, " es NEGATIVO")
else:
    print("El número", x, " es CERO")

print("Byee")
```

```
>>>
Ingresa un número:25
El número 25  es POSITIVO
Byee
```

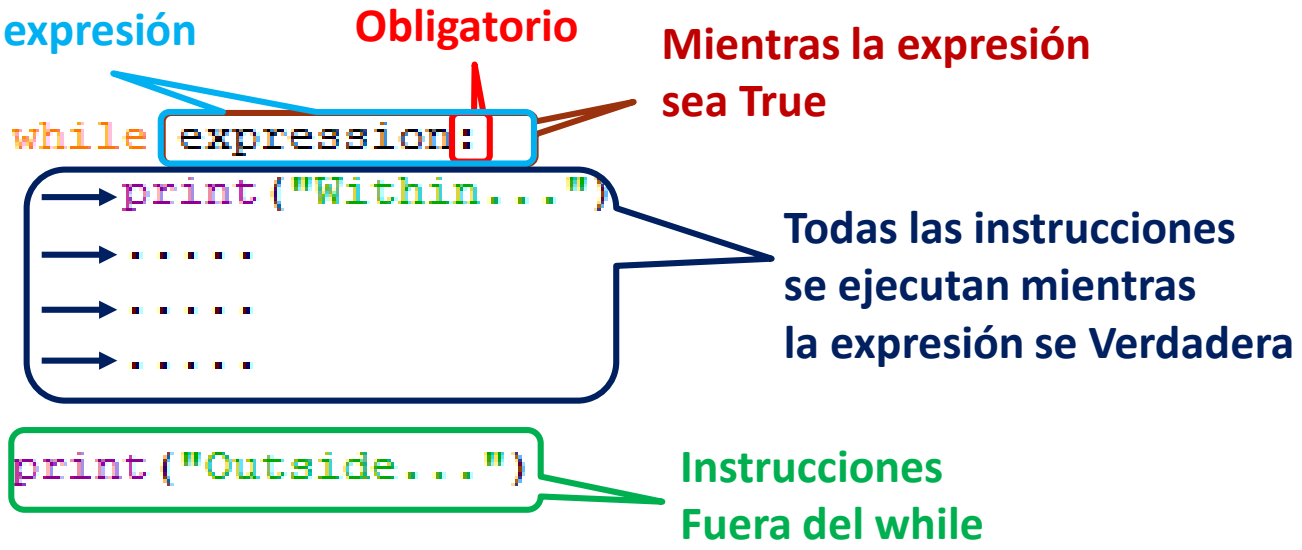
```
>>>
Ingresa un número:0
El número 0  es CERO
Byee
```

```
>>>
Ingresa un número:-2
El número -2  es NEGATIVO
Byee
```

Repetitivos - while

- Permiten ejecutar varias veces la misma instrucción.
- La cantidad de veces debe ser finita, y el usuario debe garantizar que algún momento termine la ejecución.

IMPORTANTE:
En algún momento expresión debe ser False



while

```
x = 10
cont = 1
while (cont != x):
    print (cont)
    cont = cont + 1
```

```
>>>
1
2
3
4
5
6
7
8
9
```

```
info = 'repite'
while info == 'repite':
   print ('Holaaaa')
   info = input('Introduce "repite" para hacerlo de nuevo: ')
```

```
>>>
Holaaaa
Introduce "repite" para hacerlo de nuevo: repite
Holaaaa
Introduce "repite" para hacerlo de nuevo: repite
Holaaaa
Introduce "repite" para hacerlo de nuevo: no mas
```

while

Datos

file.in

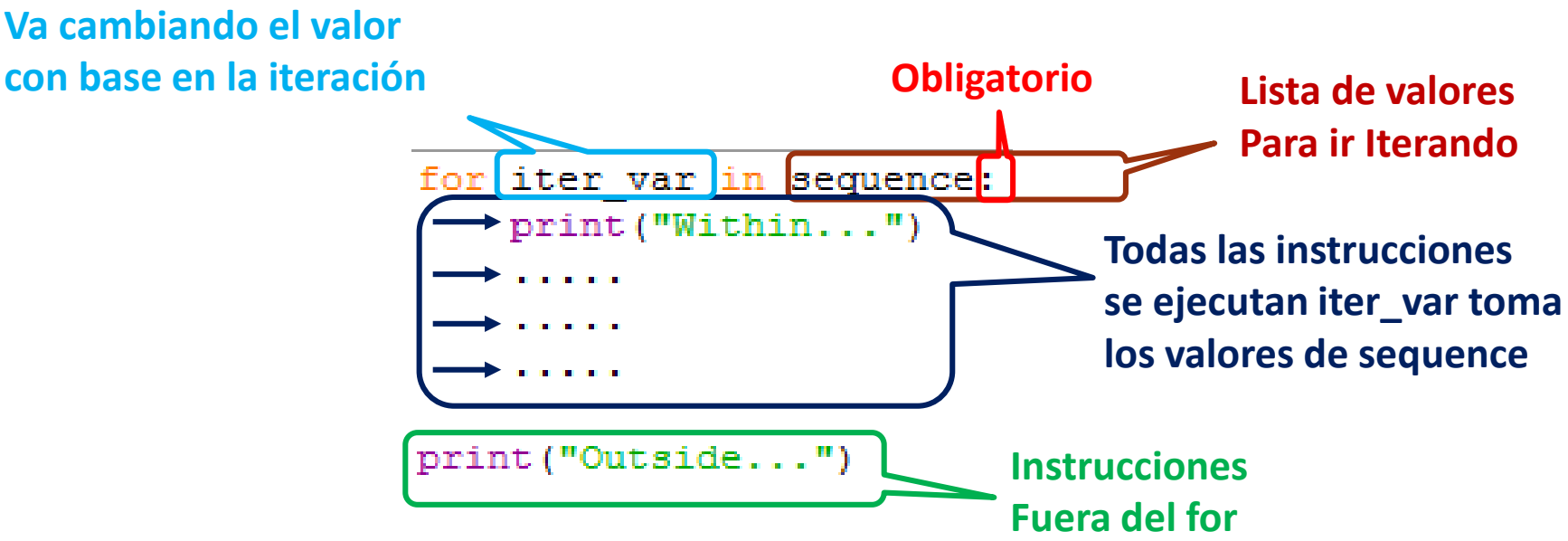
```
Pedro
4
-23
2
1
5
-22
10
120
10
```

```
from sys import stdin

name=stdin.readline()
casos=int(stdin.readline())

print("Hola ", name)
while casos>0:
    a = int(stdin.readline())
    b = int(stdin.readline())
    s = a+b
    print("La suma es:", s)
    casos -=1

print("Chaooo")
```

```
C:\Python34>python prueba.py < file.in
Hola  Pedro

La suma es: -21
La suma es: 6
La suma es: -12
La suma es: 130
Chaooo
```

Repetitivos - for

- Desea ejecutar n veces las mismas instrucciones.
- Ejemplo: mostrar 10 veces la frase: “Hola Mundo”



for

```
vals=[1,2,3,4,5]
for x in vals:
    print("Iteración",x)
    print("Hola Mundo")

print("Bye...")
```

```
>>>
Iteración 1
Hola Mundo
Iteración 2
Hola Mundo
Iteración 3
Hola Mundo
Iteración 4
Hola Mundo
Iteración 5
Hola Mundo
Bye...
```

```
for x in range(1,6):
    print("Iteración",x)
    print("Hola Mundo")

print("Bye...")
```

```
>>>
Iteración 1
Hola Mundo
Iteración 2
Hola Mundo
Iteración 3
Hola Mundo
Iteración 4
Hola Mundo
Iteración 5
Hola Mundo
Bye...
```

for

```
for x in range(1,10,2):
    print(x)

print("Bye...")
```

```
>>>
1
3
5
7
9
Bye...
```

```
for x in range(10,1,-1):
    print(x)

print("Bye...")
```

```
>>>
10
9
8
7
6
5
4
3
2
Bye...
```

Estructuras Anidadas

```
for val1 in sequence1:
    for val2 in sequence2:
        if expression:
            ....
            ....
        else:
            ....
            ....

print("Bye...")
```

```
for val1 in sequence1:
    if expression:
        for val2 in sequence2:
            ....
            ....

print("Bye...")
```

```
while expression1:
    for val1 in sequence1:
        if expression2:
            ....
            ....
            ....

print("Bye...")
```

- etc etc....

Listas

- Son un conjunto que contiene cierta cantidad de objetos.

- La lista de delimita por [ ], y los elementos esta separados por comas “,”:
  - [1,2,3,4,5]
  - [“Hola”,1234,”Pedro”, 5.32, False]

Tamaño de la lista es 5

	-5	-4	-3	-2	-1	: Negativos
	“Hola”	1234	“Pedro”	5.32	False	
Index:	**0**	**1**	**2**	**3**	**4**	

Listas

- Comienzan desde 0:

```
>>> dias = ["Ln","Mr","Mc","Jv","Vr","Sb","Dm"]
>>> print (dias[2])
Mc
>>> print(dias)
['Ln', 'Mr', 'Mc', 'Jv', 'Vr', 'Sb', 'Dm']

>>> dias = ["Ln","Mr","Mc","Jv","Vr","Sb",3,4,5]
>>> print (dias[7])
4
>>> type(dias)
<class 'list'>
```

- Obtener una parte de la lista ‘:’

```
>>> print (dias[2:6])
['Mc', 'Jv', 'Vr', 'Sb']
```

Listas

- Adicionar un elemento:

```
>>> dias.append("Festivo")
>>> print(dias)
['Ln', 'Mr', 'Mc', 'Jv', 'Vr', 'Sb', 3, 4, 5, 'Festivo']
```

- Listas de listas:

```
>>> dias = ["Ln","Mr","Mc","Jv","Vr","Sb",[3,4,5]]
>>> print(dias[6])
[3, 4, 5]
```

- Adicionar un elemento en cualquier posición:

```
>>> dias.insert(3,"otro")
>>> print(dias)
['Ln', 'Mr', 'Mc', 'otro', 'Jv', 'Vr', 'Sb', [3, 4, 5]]
```

Operaciones

`x in s`	Indica si la variable `x` se encuentra en `s`
`s + t`	Concantena las secuencias `s` y `t`
`s * n`	Concatena `n` copias de `s`
`s[i]`	Elemento `i` de `s`, empezando por `0`
`s[i:j]`	Porción de la secuencia `s` desde `i` hasta `j` (no inclusive)
`s[i:j:k]`	Porción de la secuencia `s` desde `i` hasta `j` (no inclusive), con paso `k`
`len(s)`	Cantidad de elementos de la secuencia `s`
`min(s)`	Mínimo elemento de la secuencia `s`
`max(s)`	Máximo elemento de la secuencia `s`

Strings

- Un conjunto de caracteres delimitados por comilla sencilla o doble. Las cadenas son una lista:

```
>>> "Colombia es Pasion"
'Colombia es Pasion'
>>> 'Colombia es Pasion'
'Colombia es Pasion'
```

- Algunas funciones para string:

```
>>> "perro y carro".upper()
'PERRO Y CARRO'
>>> "perro y carro".find("Y")
-1
>>> "perro y carro".find("y")
6
>>> "perro y carro".replace("carro","moto")
'perro y moto'
```

Strings

- Algunas operaciones:

```
>>> 3 * 'b'
'bbb'
>>> 'b'+'c'
'bc'
>>> 'a'+str(23)
'a23'
>>> len('colombia')
8
```

- Indexación:

```
>>> 'casa'[0]
'c'
>>> 'casa'[3]
'a'
```

Strings

- Subcadena:

```
>>> x="colombia es pasion"
>>> x[4:10]
'mbia e'
```

- Formateo:

```
>>> "La capital de %s es %s " % ("Colombia","Bogotá")
'La capital de Colombia es Bogotá '
```

```
a = "Juan"
b = 25

print("El Sr. %s tiene %d años" % (a,b))
```

```
>>>
El Sr. Juan tiene 25 años
```

Ord()

- **Ord:** Devuelve el valor ASCII de una de un carácter o un carácter Unicode.

```
a = "Python@.."
for x in a:
    print(ord(x))


print("Byeee")
```

```
>>>
80
121
116
104
111
110
64
46
46
Byeee
```

Chr(k)

- **Chr:** Devuelve una cadena de un carácter cuyo código ASCII es el entero k

```
rta=""
for x in range(10500,11000):
    rta=rta+chr(x)

print(rta)
print("Byeee")
```

```
>>>
<many rows of unusual Unicode characters>
Byeee
```

Caracteres de Escape

Escape Sequence	hex value	meaning
\0	0x00	end-of-string
\a	0x07	bell (alert)
\b	0x08	backspace
\t	0x09	horizontal tab
\n	0x0A	linewline
\v	0x0B	vertical tab
\f	0x0C	form feed (newpage)
\r	0x0D	return
\'	0x27	single quote
\"	0x22	double quote
\\	0x5D	Backslash
\uxxxx	16-bit hex value Unicode character	
\Uxxxxxxxx	32-bit hex value Unicode character	
\xhh	Prints character based on its hex value	

Imagen copiada de: https://courses.edx.org/courses/course-v1:UTAx+CSE1309x+2016T1/courseware/cdd624dd14794a72bcc093e756123f74/bc43a1dfa2e247e6a93de793d572a3b1/

Caracteres de Escape

```
a="Colombia"
rta=""
for x in a:
    rta=rta+x+"\n"

print("La \"salida\" es:")
print(rta)
```

```
>>>
La "salida" es:
C
o
l
o
m
b
i
a

>>>
```

Split()

- Convertir una cadena en una lista a partir de un carácter:

```
a="Colombia"
l=a.split("o")
print(l)
```

```
>>>
['C', 'l', 'mbia']
```

```
a="Colombia es pasión"
l=a.split()
print(l)
```

```
>>>
['Colombia', 'es', 'pasión']
```

```
a="Hola;Mundo;Python"
l=a.split(";")
print(l)
```

```
>>>
['Hola', 'Mundo', 'Python']
```

Join()

- Es la función inversa a Split(). Hace la unión de una lista de cadenas en una cadena

```
x = ["1","2","3","4","5"]
rta = "-".join(x)
print(rta)


>>>
1-2-3-4-5
```

```
x = ["hola","colombia","python"]
rta = "\n".join(x)
print(rta)


>>>
hola
colombia
python
```

```
x = ["a","b","c"]
rta = "@,".join(x)
print(rta)


>>>
a@,b@,c
```

Referencias

- Introduction to Computation and Programming Using Python, revised and expanded edition, John V Guttag, MIT Press.
- Python Programming: An Introduction to Computer Science, John Zelle.
- MITx's Introduction to Computer Science and Programming Using Python.
- http://paginaspersonales.deusto.es/dipina/teaching.html
- https://courses.edx.org/courses/course-v1:UTAx+CSE1309x+2016T1/
- https://www.python.org/
- http://www.camilorocha.info/