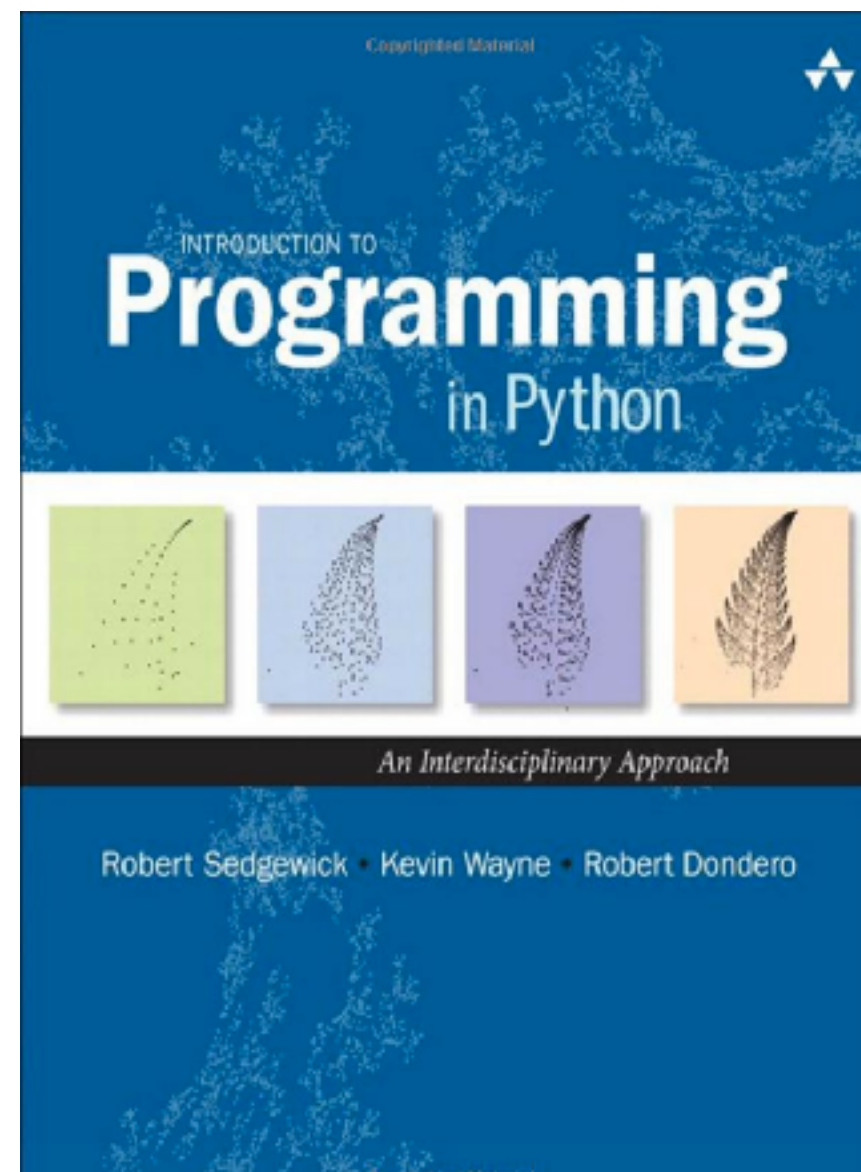


Parte I: Intro pensamiento computacional

Clase 03: Ciclos

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>

Conversión de tipos

- Explícito: directamente en el código
 - Convertir texto a entero
 - Convertir entero a float
 - Convertir flotante a entero
- Implícito: automático por Python*
 - Operaciones entre números de distinto tipo
 - Multiplicación entre un entero n y un string s devuelve el string s concatenado n veces.

Salida:

```
1 # tipo-explicit.py
2 print(type('2'))          <class 'str'>
3 print(type(int('2')))     <class 'int'>
4 print(type(float('2')))   <class 'float'>
5 print(type(float(2)))      <class 'float'>
6 print(type(str(2)))        <class 'str'>
7 print(type(str(2.0)))      <class 'str'>
```

P: ¿Cuál es el resultado de la línea 6 y 7?

Salida:

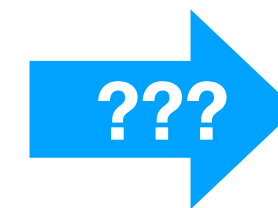
```
1 # tipo-implicit.py
2 print(type(2+2.0))
3 print(type(2+float(2)))
4 print(type(2*'hola'))
```

Salida:

```
<class 'float'>
<class 'float'>
<class 'str'>
```

P: ¿Cuál es el resultado de la línea 4?

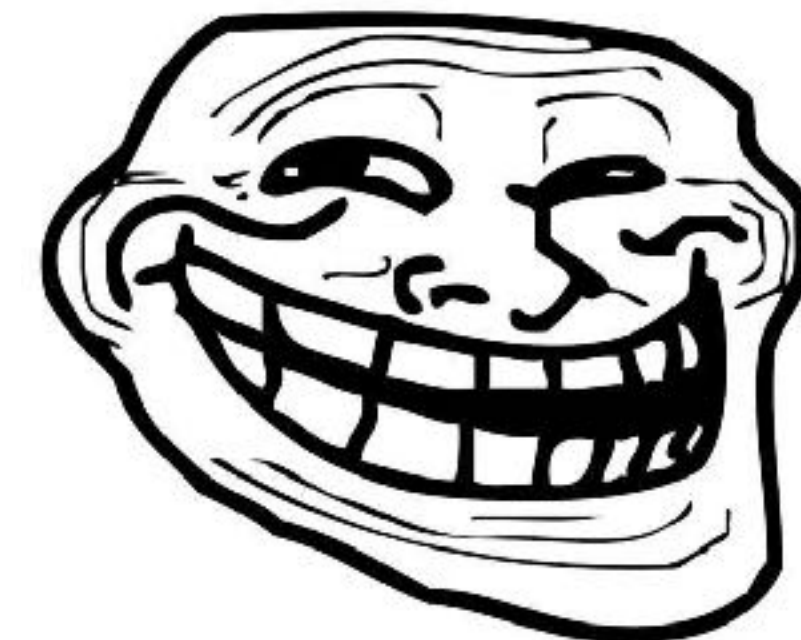
```
1 # pizzas.py
2 i = input('¿Cuántas pizzas individuales desea?: ')
3 m = input('¿Cuántas pizzas mediantas desea?: ')
4 f = input('¿Cuántas pizzas familiares desea?: ')
5 total = 4600*i + 7850*m + 10750*f
6 print('Total a pagar:', total)
```

[illegible]

***Nota:** en la programación no existe la magia, existe un mecanismo definido conversión de tipos implícitos

Ejemplo de uso para if: chequeo de errores

```
$ python3 pizzas2.py
¿Cuántas pizzas individuales desea?: 2
¿Cuántas pizzas medianas desea?: -1
¿Cuántas pizzas familiares desea?: 0
Error en el número de pizzas
Total a pagar: 1350
```



problem?

```
1 # pizzas2.py
2 i = int(input('¿Cuántas pizzas individuales desea?: '))
3 m = int(input('¿Cuántas pizzas medianas desea?: '))
4 f = int(input('¿Cuántas pizzas familiares desea?: '))
5
6 if i < 0 or m < 0 or f < 0:
7     print('Error en el número de pizzas')
8 else:
9     total = 4600*i + 7850*m + 10750*f
10    print('Total a pagar:', total)
11
12 total = 4600*i + 7850*m + 10750*f
13 print('Total a pagar:', total)
```



Human-based python interpreter™

- ¿Qué hace este programa?

```
1 # argv contiene los argumentos entregados
2 # por el usuario en la consola
3 from sys import argv
4 a = int(argv[1])
5 b = int(argv[2])
6 if b < a:
7     t = b
8     b = a
9     a = t
10 print(a)
11 print(b)
```

P: ¿Cómo ordenaría tres números?

← a, b = b, c

<http://www.pythontutor.com/visualize.html>

I need
this

Can you solve the logical list riddle?

1. In this list, exactly 1 statement is false.
2. In this list, exactly 2 statements are false.
3. In this list, exactly 3 statements are false.
4. In this list, exactly 4 statements are false.
5. In this list, exactly 5 statements are false.
6. In this list, exactly 6 statements are false.
7. In this list, exactly 7 statements are false.
8. In this list, exactly 8 statements are false.
9. In this list, exactly 9 statements are false.
10. In this list, exactly 10 statements are false.

Which of these statements, if any, are false?

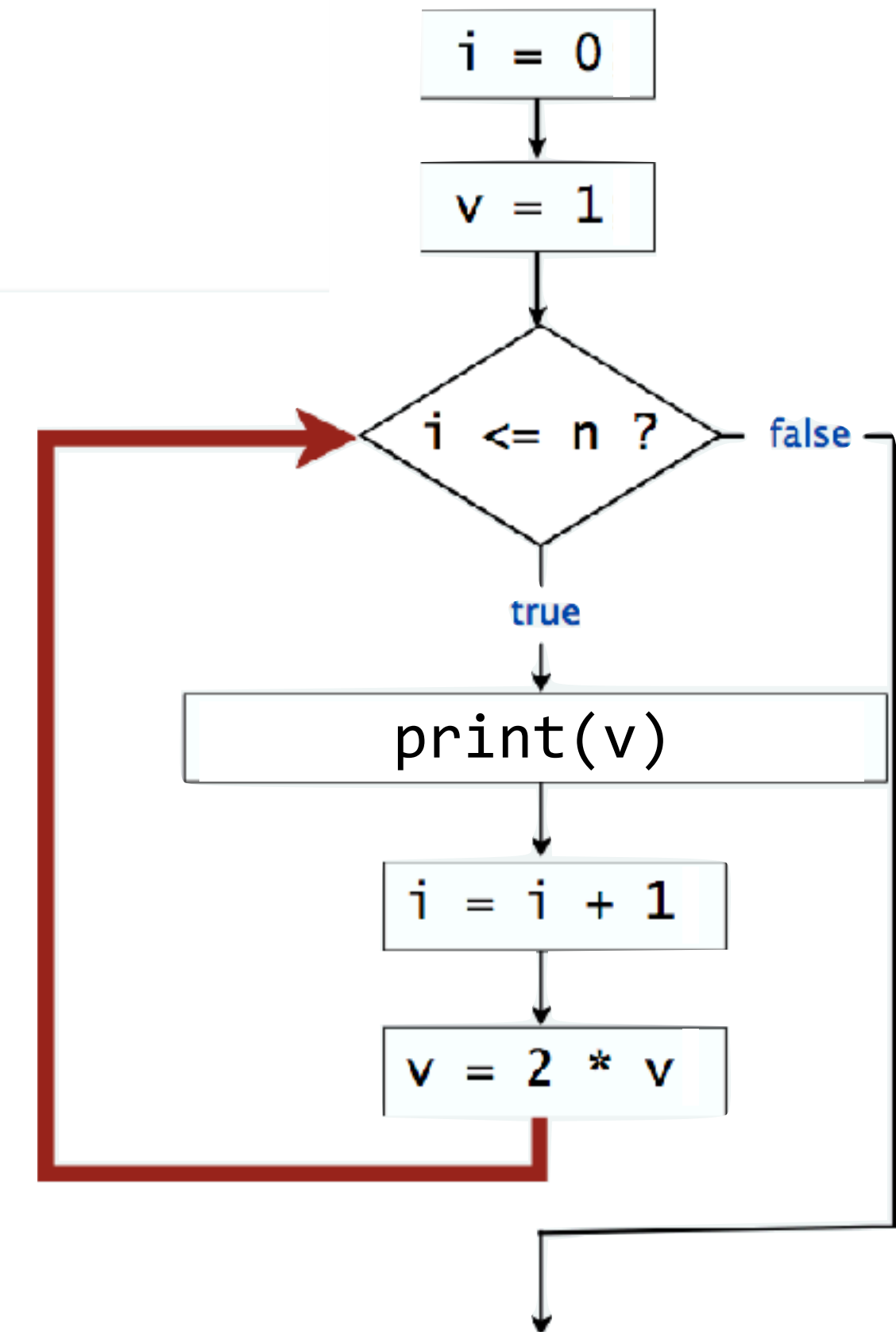
Ciclo while

- Ejecutar código mientras se cumple una condición:
 - Evaluar una expresión booleana
 - Si la expresión es True, ejecutar un bloque de código
 - **Repetir**

Expresión booleana

```
i = 0
v = 1
while i <= n:
    print(v)
    i = i + 1
    v = 2*v
```

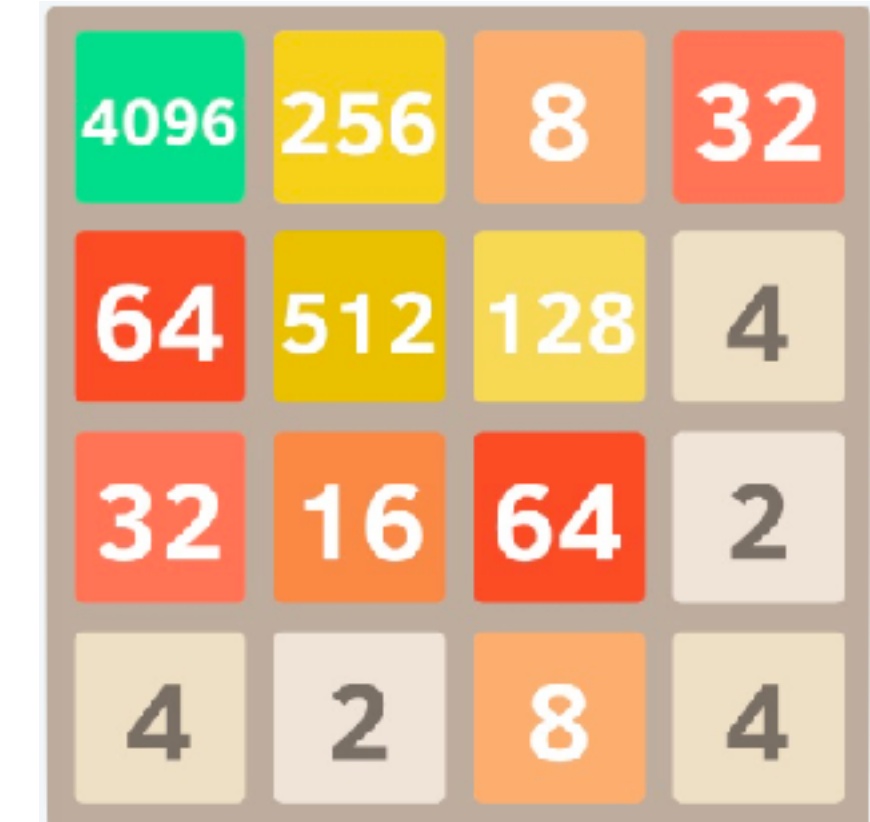
Imprime las potencias de dos desde 2^0 a 2^n



Imprimir potencias de dos

```
1 from sys import argv
2 n = int(argv[1])
3 i = 0
4 v = 1
5 while i <= n:
6     print(v)
7     i = i + 1
8     v = 2*v
```

i	v	i <= n
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true
5	32	true
6	64	true
7	128	false



4096	256	8	32
64	512	128	4
32	16	64	2
4	2	8	4

```
$ python3 potencia2.py 7
1
2
4
8
16
32
64
128
```

Implementación raíz cuadrada con while

- **Objetivo:** implementar raíz cuadrada usando el método Newton-Rhapson.
- Para calcular \sqrt{c} :
 - Inicializar $t = c$ (primera aproximación)
 - Repetir hasta que $t = c/t$ tiene suficiente precisión
 - Actualizar t con el promedio de t y c/t

i	t_i	$2/t_i$	<i>average</i>
0	2	1	1.5
1	1.5	1.3333333	1.4166667
2	1.4166667	1.4117647	1.4142157
3	1.4142157	1.4142114	1.4142136
4	1.4142136	1.4142136	

Ejemplo raíz cuadrada de 2

```

1 # sqrt.py
2 from sys import argv
3 EPS = 1e-15
4 c = float(argv[1])
5 t = c
6 while abs(t - c/t) > t*EPS:
7     t = (c/t + t)/2.0
8 print(t)
    
```

```

$ python3 sqrt.py 60481729
7777.0
$ python3 sqrt.py 9
3.0
$ python3 sqrt.py 2
1.414213562373095
    
```

Y mucha matemática omitida... pero es algo como esto

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \quad x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Wanted $f(x) = x^2 - c$

$$x_1 = x_0 - \frac{x_0^2 - c}{2x_0}$$

$$x_1 = x_0 - \left[\frac{x_0}{2} - \frac{c}{x_0} \right]$$

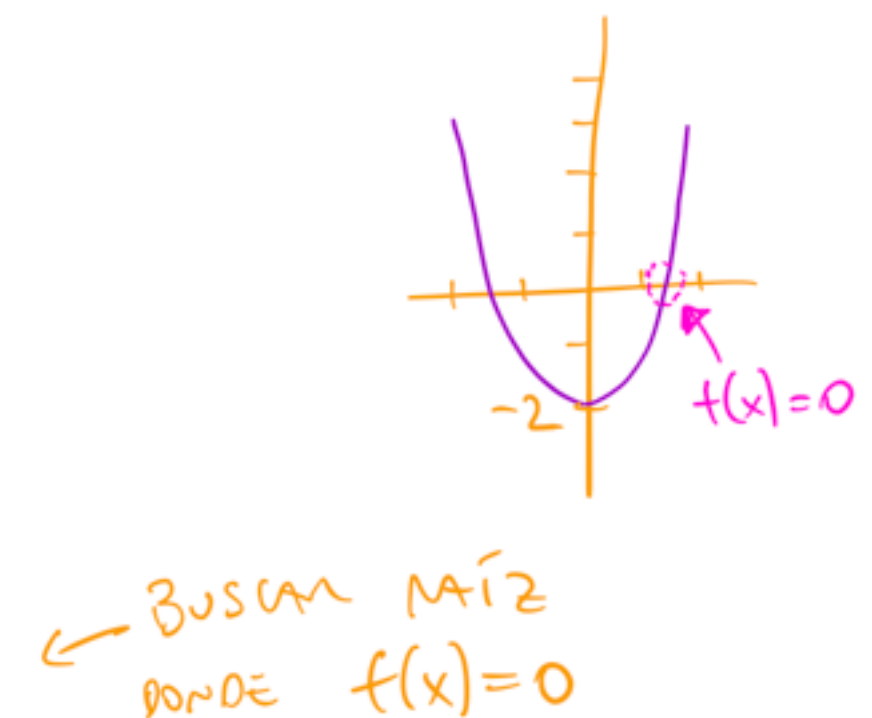
$$x_1 = x_0 + \left[\frac{c}{x_0} - \frac{x_0}{2} \right]$$

$$\sqrt{c} = x$$

$$c = x^2$$

$$f(x) = x^2 - c$$

$$f'(x) = 2x$$



Ciclo for

- Ejecutar código mientras se recorre una secuencia de elementos.
 - La secuencia se recorre en orden.
 - El término está garantizado.

Variable usada
para recorrer la
secuencia

Secuencia de enteros
hasta n - 1

```
for i in range(4):  
    print('Hola número', i)
```

Salida:

```
$ python3 holas.py  
Hola número 0  
Hola número 1  
Hola número 2  
Hola número 3
```

Inicio

Fin

```
for i in range(4,8):  
    print('Hola número', i)
```

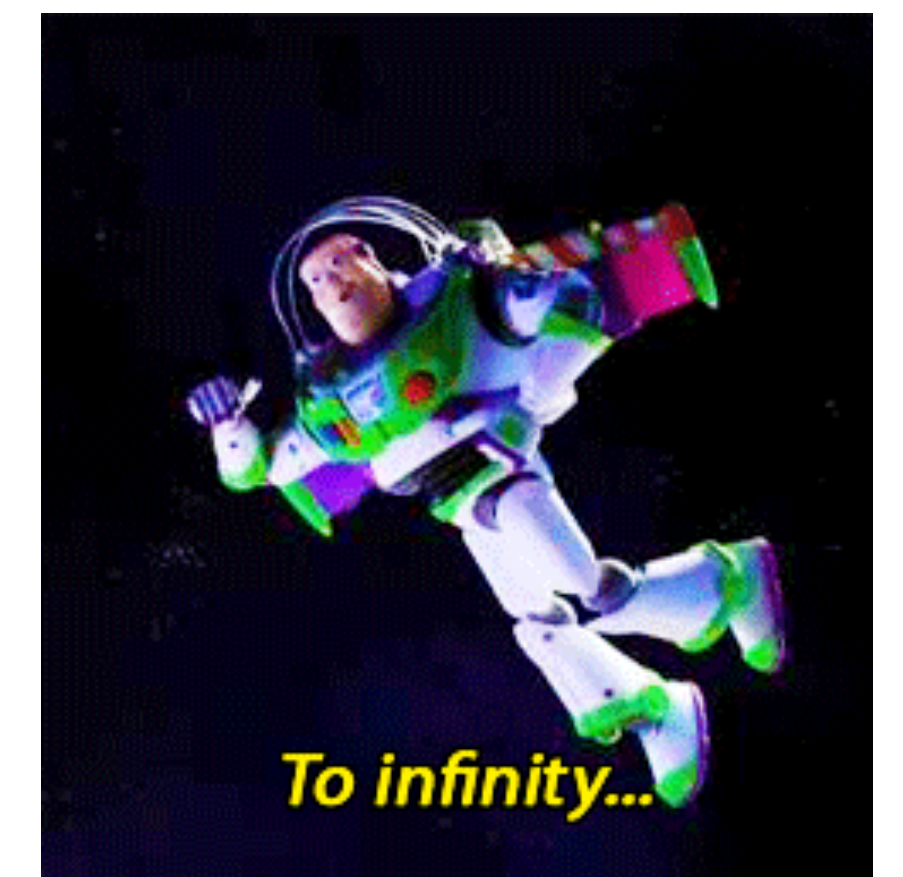
Salida:

```
$ python3 holas2.py  
Hola número 4  
Hola número 5  
Hola número 6  
Hola número 7
```

Ejemplo de un ciclo **while** que **nunca termina**.
La condición de detención siempre es **True**!

```
x = 1  
while True:  
    print("Al infinito y más allá! Ya vamos en {:d}!".format(x))  
    x += 1
```

```
1. bash  
Al infinito y más allá! Ya vamos en 93523!  
Al infinito y más allá! Ya vamos en 93524!  
Al infinito y más allá! Ya vamos en 93525!  
Al infinito y más allá! Ya vamos en 93526!  
Al infinito y más allá! Ya vamos en 93527!  
Al infinito y más allá! Ya vamos en 93528!  
Al infinito y más allá! Ya vamos en 93529!  
Al infinito y más allá! Ya vamos en 93530!  
Al infinito y más allá! Ya vamos en 93531!  
Al infinito y más allá! Ya vamos en 93532!  
Al infinito y más allá! Ya vamos en 93533!  
Al infinito y más allá! Ya vamos en 93534!  
Al infinito y más allá! Ya vamos en 93535!  
Al infinito y más allá! Ya vamos en 93536!  
Al infinito y más allá! Ya vamos en 93537!  
Al infinito y más allá! Ya vamos en 93538!  
Al infinito y más allá! Ya vamos en 93539!  
Al infinito y más allá! Ya vamos en 93540!  
Al infinito y más allá! Ya vamos en 93541!  
Al infinito y más allá! Ya vamos en 93542!  
Al infinito y más allá! Ya vamos en 93543!
```



while vs for

- Imprima todos los números impares menores que n mayores o iguales a cero.

```
1 n = int(input('ingrese n: '))
2 if n <= 0:
3     print('Debe ingresar un número mayor a cero')
4 i = 0
5 while i < n:
6     if i % 2 == 1:
7         print(i)
8     i = i+1
```

```
1 n = int(input('ingrese n: '))
2 for i in range(n):
3     if i % 2 == 1:
4         print(i)
```

```
1 n = int(input('ingrese n: '))
2 for i in range(1, n, 2):
3     print(i)
```




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

9:17 AM - 3 Dec 2014

3,387 Retweets 3,382 Likes



88

3.4K

3.4K



<https://twitter.com/codeinthehole/status/540117725604216832>

Habilidades deseables para un programador:

1. Habilidad para ignorar nuevas herramientas y tecnologías
2. Gusto por la simplicidad
3. Buenas habilidades para eliminar código
4. Humildad

Human-based python interpretertm

- ¿Qué hace este programa?

```
1 a = 5
2 b = int(input())
3 if a + b < b:
4     print('Si')
5 else:
6     print('No')
```


Resumen

Conceptos

- **while**: ejecutar código mientras una condición se cumple
- **for**: ejecutar código al recorrer una secuencia. La secuencia se puede generar con la función `range(...)`

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

- **range(stop)**: secuencias de enteros hasta stop-1
- **range(start, stop[, step])**: secuencia de enteros desde start hasta stop-1, saltándose step pasos.

		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>