

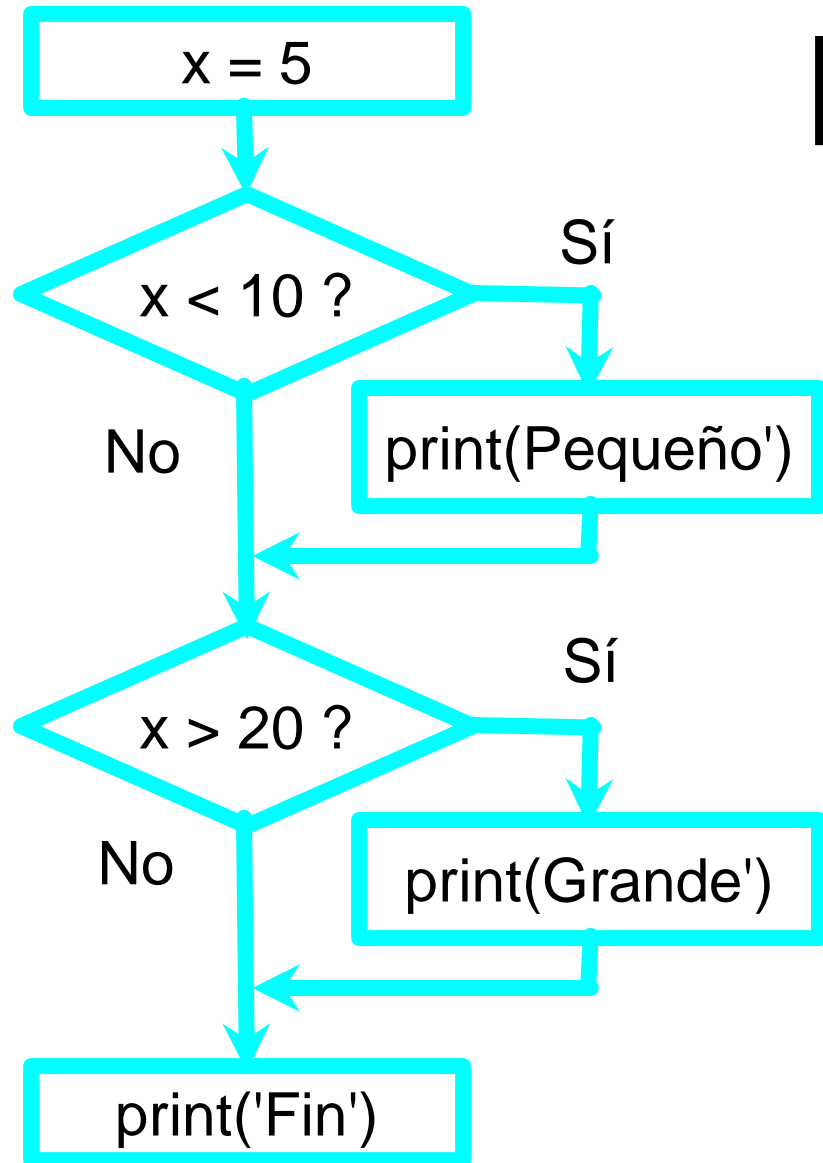
Ejecución Condicional

Unidad 3

Python para principiantes
dgm@uma.es



Pasos Condicionales



Programa:

```
x = 5
if x < 10:
    print('Pequeño')
if x > 20:
    print('Grande')

print('Fin')
```

Salida:

Pequeño
Fin

Operadores de Comparación

- Una expresión booleana hace una pregunta y produce un resultado **Sí** o **No** que puede usarse para controlar la ejecución del programa
- Las expresiones booleanas usan operadores de comparación que evalúan a **True / False** o Sí / No
- True y False son palabras reservadas de Python
- Los operadores de comparación miran el valor de las variables pero no las cambian

Python	Significado
<	Menor que
<=	Menor que o igual a
==	Igual a
>=	Mayor que o Igual a
>	Mayor que
!=	Distinto

Recuerda “=” se utiliza para la asignación.

http://es.wikipedia.org/wiki/George_Boole

Operadores de Comparación

```
x = 5
if x == 5 :
    print('Igual a 5')
if x > 4 :
    print('Mayor que 4')
if x >= 5 :
    print('Igual que o mayor a 5')
if x < 6 : print('Menor que 6')
if x <= 5 :
    print('Menor que o igual a 5')
if x != 6 :
    print('Distinto de 6')
```

Igual a 5

Mayor que 4

Igual que o mayor a 5

Menor que 6

Menor que o igual a 5

Distinto de 6

Decisiones Simples

```
x = 5
print('Antes de 5')
if x == 5 :
    print('Es 5')
    print('Todavía 5')
    print('Tercer 5')
print('Después de 5')
print('Antes de 6')
if x == 6 :
    print('Es 6')
    print('Todavía 6')
    print('Tercer 6')
print('Después de 6')
```

Antes de 5

Es 5

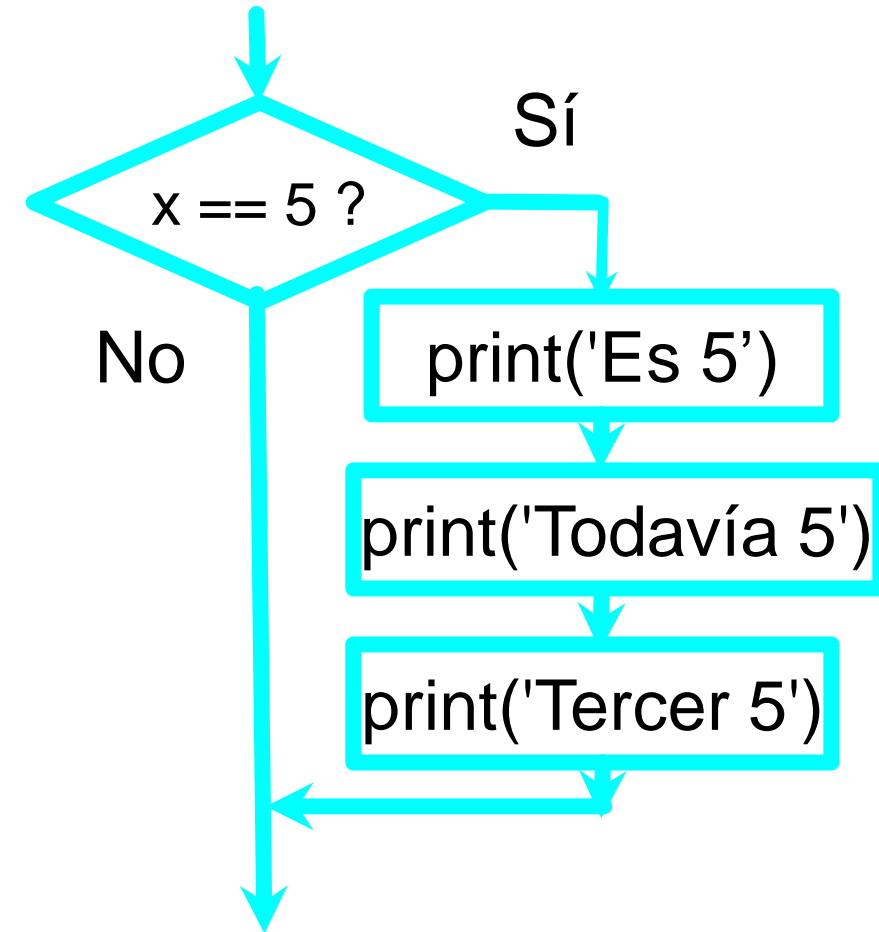
Todavía 5

Tercer 5

Después de 5

Antes de 6

Después de 6

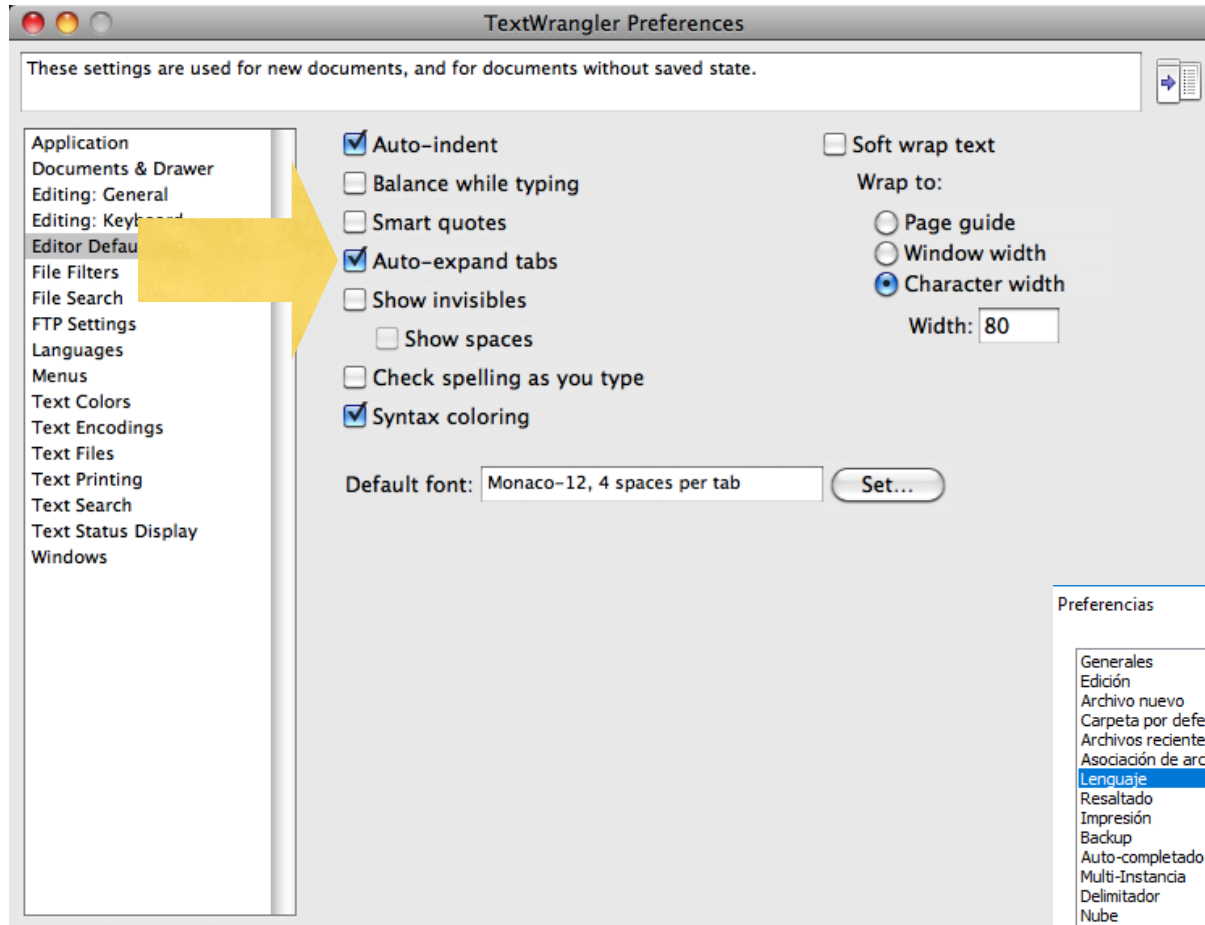


Indentación

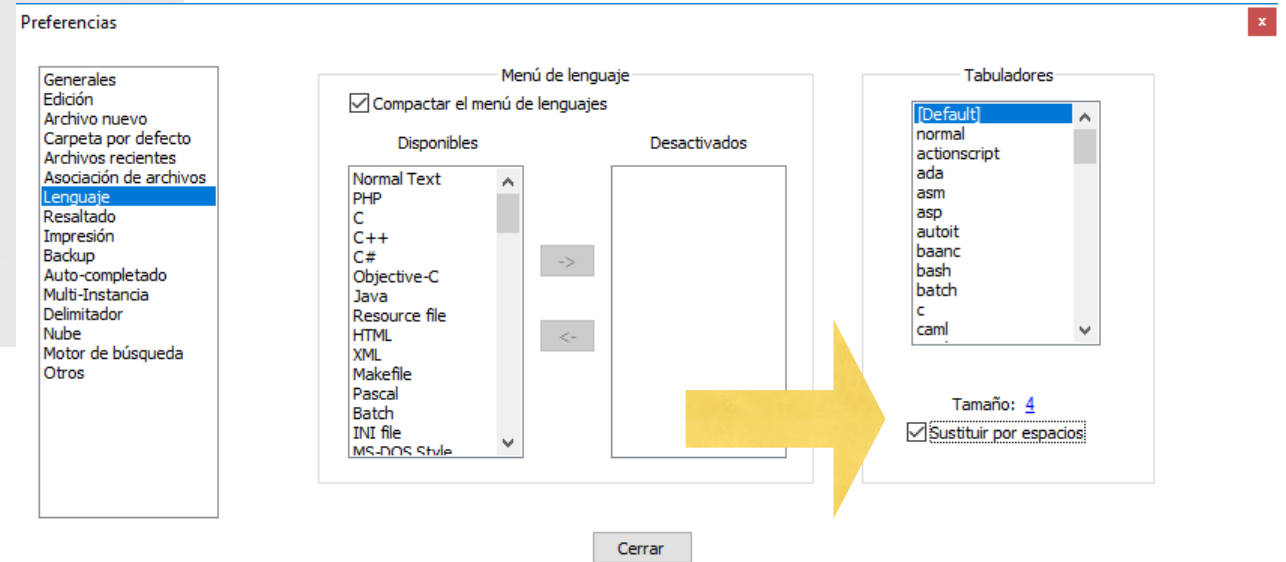
- La indentación (escribir más a la derecha o más a la izquierda que la línea anterior) se utiliza en Python para **agrupar** bloques de código
- Hay que **incrementar** la indentación después de una sentencia if o for (después de :)
- Hay que **mantener** la indentación para indicar el ámbito del bloque (si seguimos dentro del if/for)
- Hay que **reducir** la indentación al mismo nivel que la sentencia if o for para indicar la finalización del bloque
- Las líneas en blanco se ignoran – no afectan a la indentación
- Los comentarios se ignoran también con respecto a la indentación

Advertencia: ¡¡Desactivar la tabulación!!

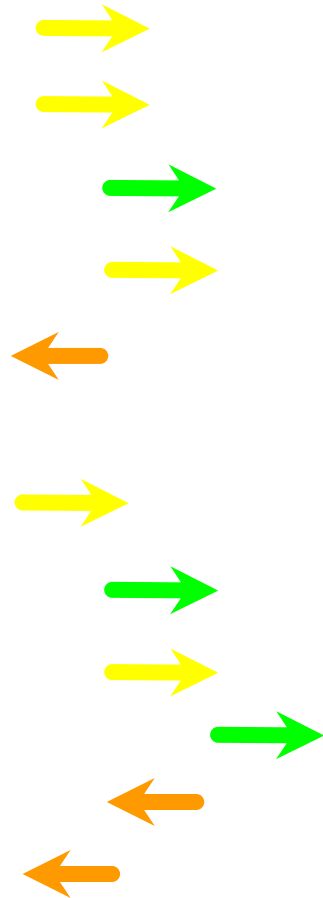
- El editor Atom utiliza espacios automáticamente en archivos con extensión ".py" (bueno!)
- La mayoría de los editores pueden **convertir las tabulaciones en espacios** – **asegúrate** de habilitarlo
 - - Notepad++: Settings -> Preferences -> Language Menu/Tab Settings o bien Configuración -> Preferencias -> Lenguaje -> Tabuladores
 - - TextWrangler: TextWrangler -> Preferences -> Editor Defaults
- Python se preocupa ***mucho*** sobre la indentación de las líneas. Si mezclas tabulaciones y espacios, puedes obtener “**indentation errors**” incluso si todo parece correcto



Esto te ahorrará
muchos problemas
si usas Atom.



incrementar / mantener después de if o for
decrementar para indicar el fin de bloque



```
x = 5
if x > 2 :
    print('Mayor que 2')
    print('Todavía mayor')
print('Listo con el 2')

for i in range(5) :
    print(i)
    if i > 2 :
        print('Mayor que 2')
    print('Listo con i', i)
print('Fin')
```

Pensar sobre el inicio/fin de los bloques

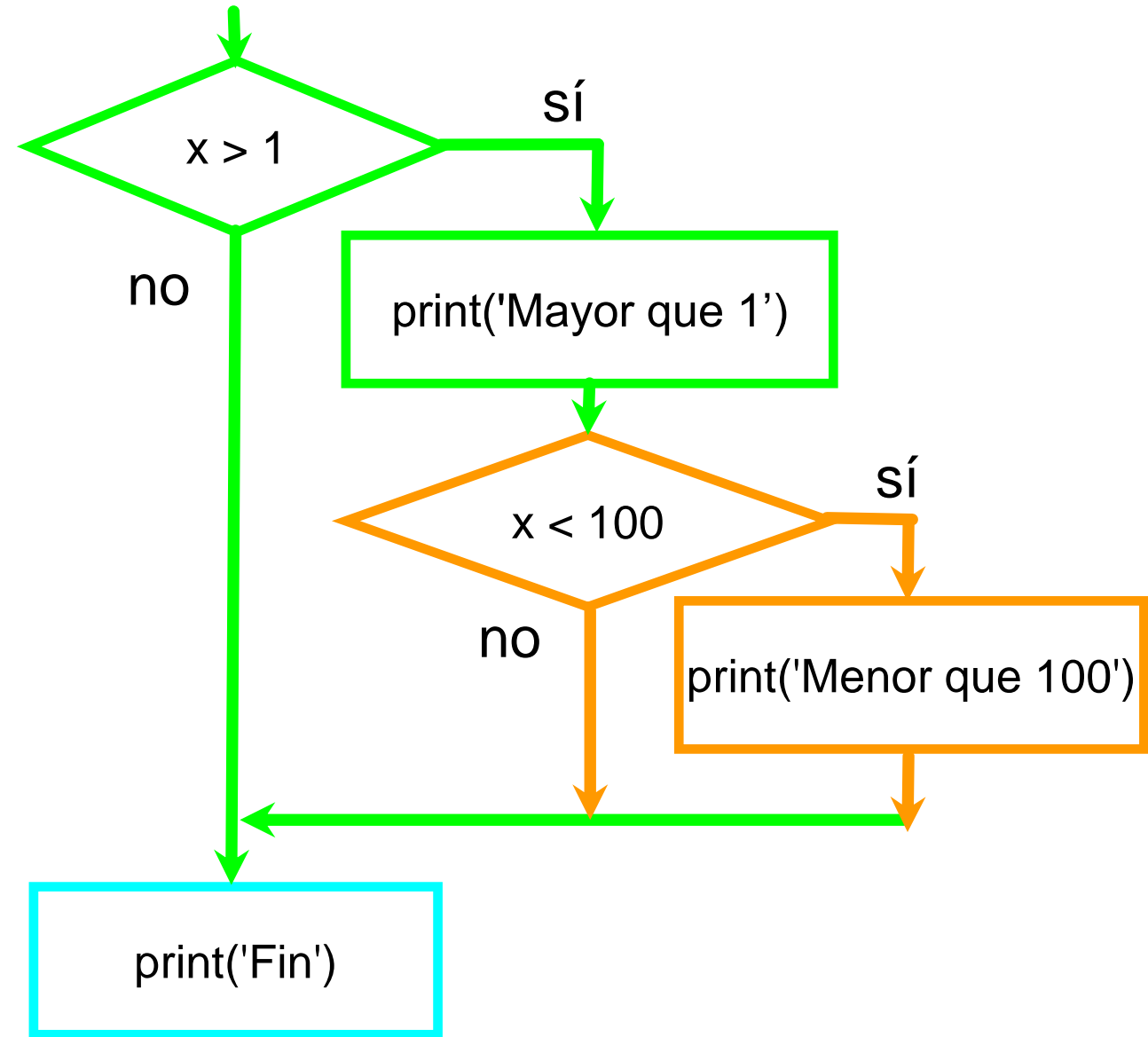
```
x = 5
```

```
if x > 2 :  
    print('Mayor que 2')  
    print('Todavía mayor')  
print('Listo con el 2')
```

```
for i in range(5) :  
    print(i)  
    if i > 2 :  
        print('Mayor que 2')  
    print('Listo con i', i)  
print('Fin')
```

Decisiones Anidadas

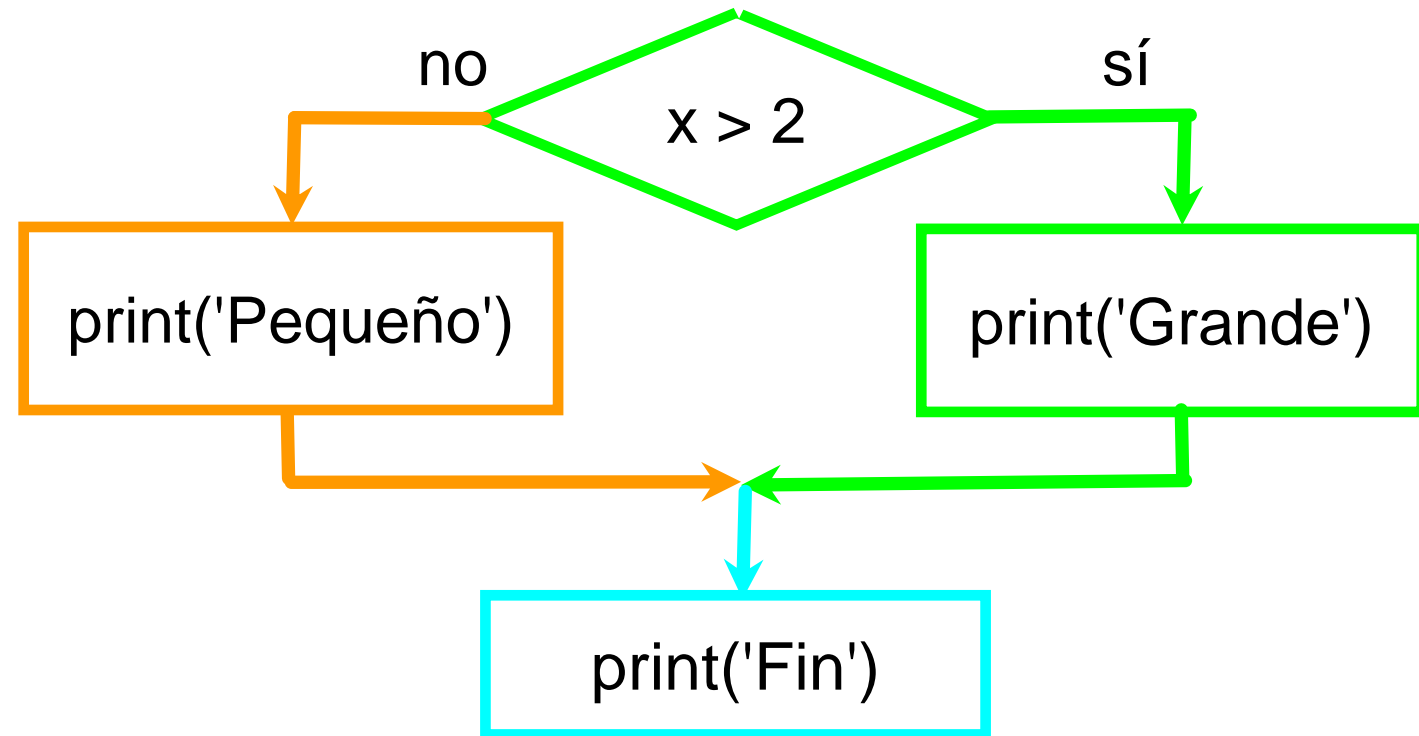
```
x = 42
if x > 1 :
    print('Mayor que 1')
    if x < 100 :
        print('Menor que 100')
print('Fin')
```



Decisiones con alternativa (else)

$x = 4$

- A veces queremos hacer algo si una expresión lógica es cierta y otra cosa si la expresión es falsa
- Es como una bifurcación en un camino – debemos **elegir** uno u otro pero no los dos

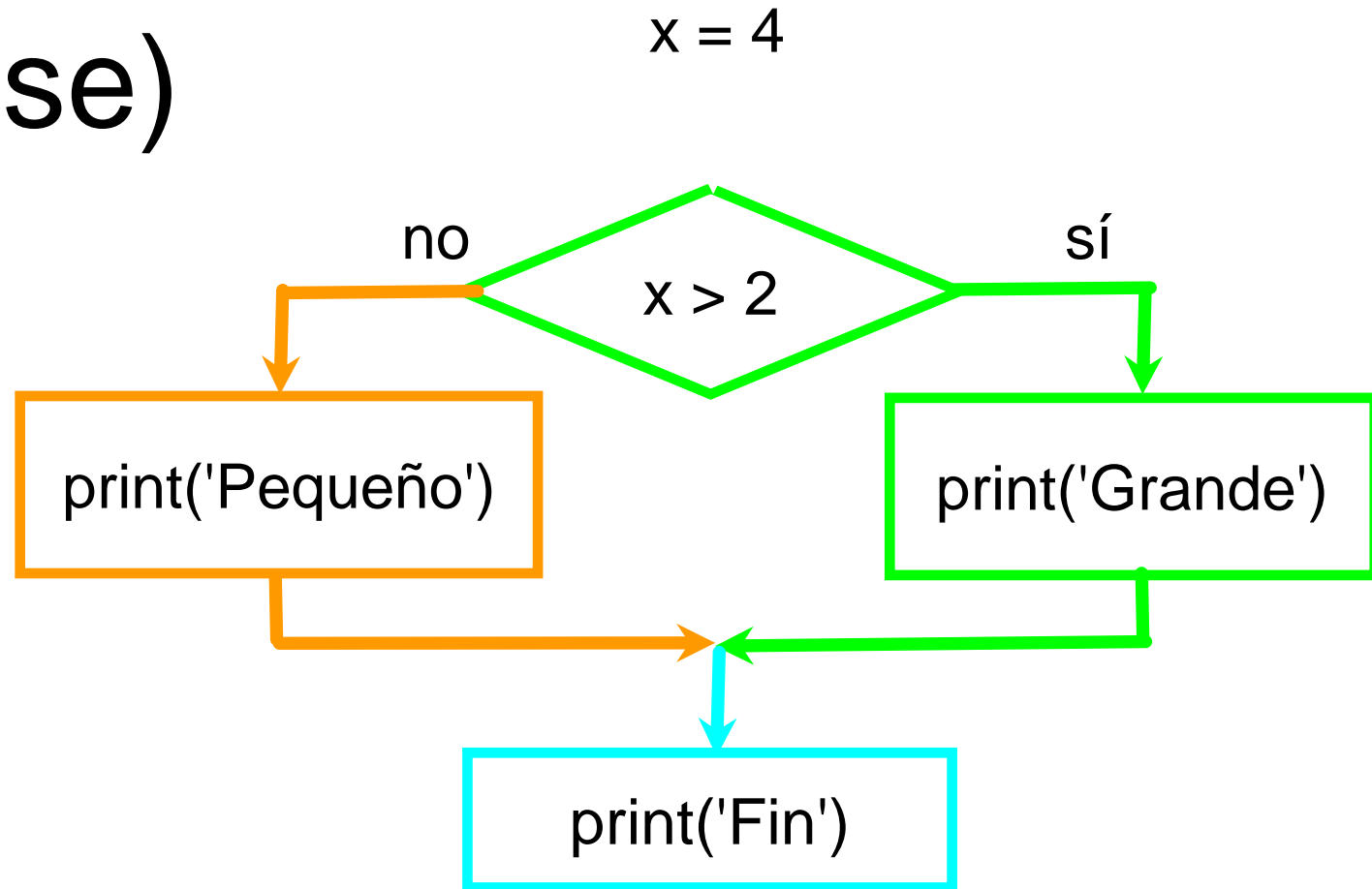


Decisiones con alternativa (else)

x = 4

```
if x > 2 :  
    print('Grande')  
else :  
    print('Pequeño')
```

```
print('Fin')
```



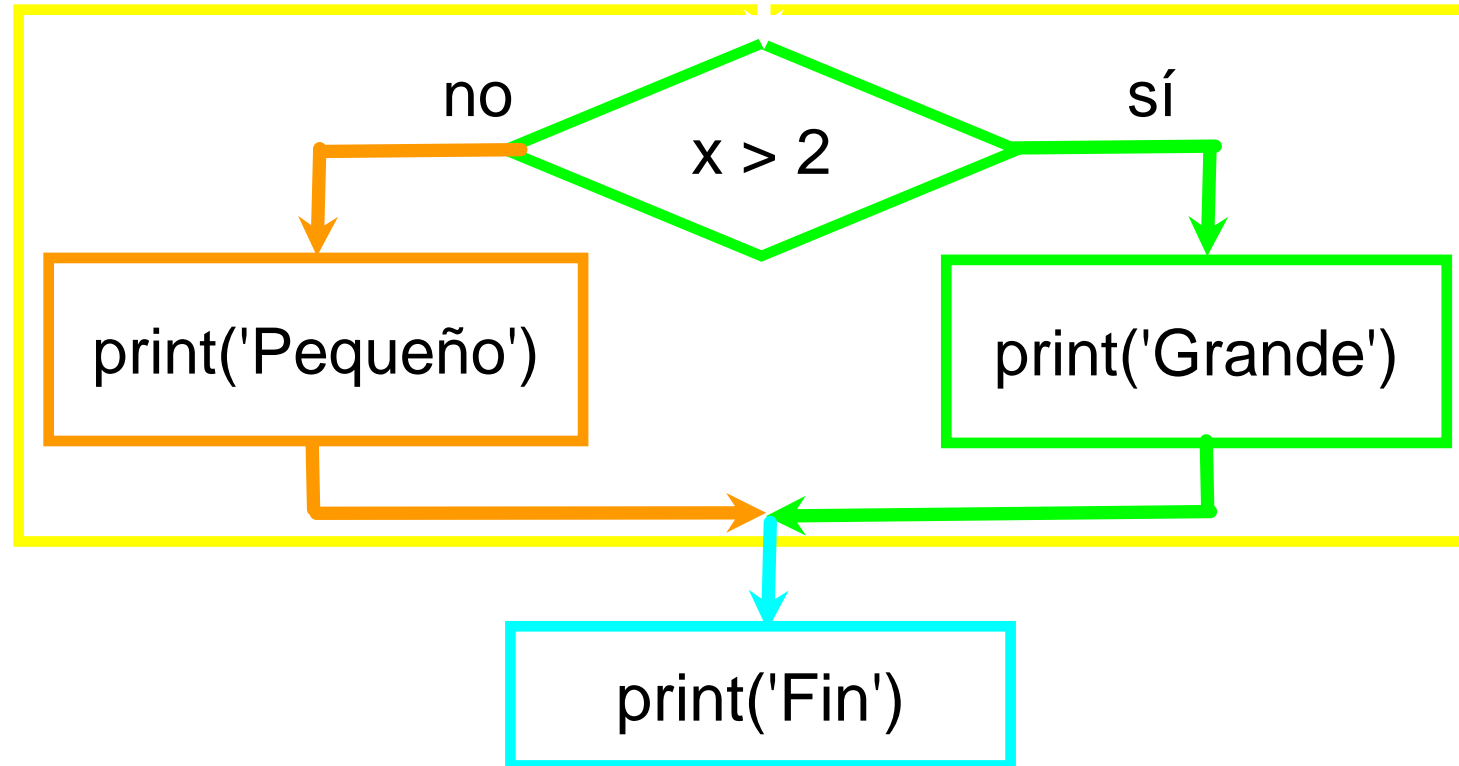
Visualiza los bloques

x = 4

```
if x > 2 :  
    print('Grande')  
else :  
    print('Pequeño')
```

```
print('Fin')
```

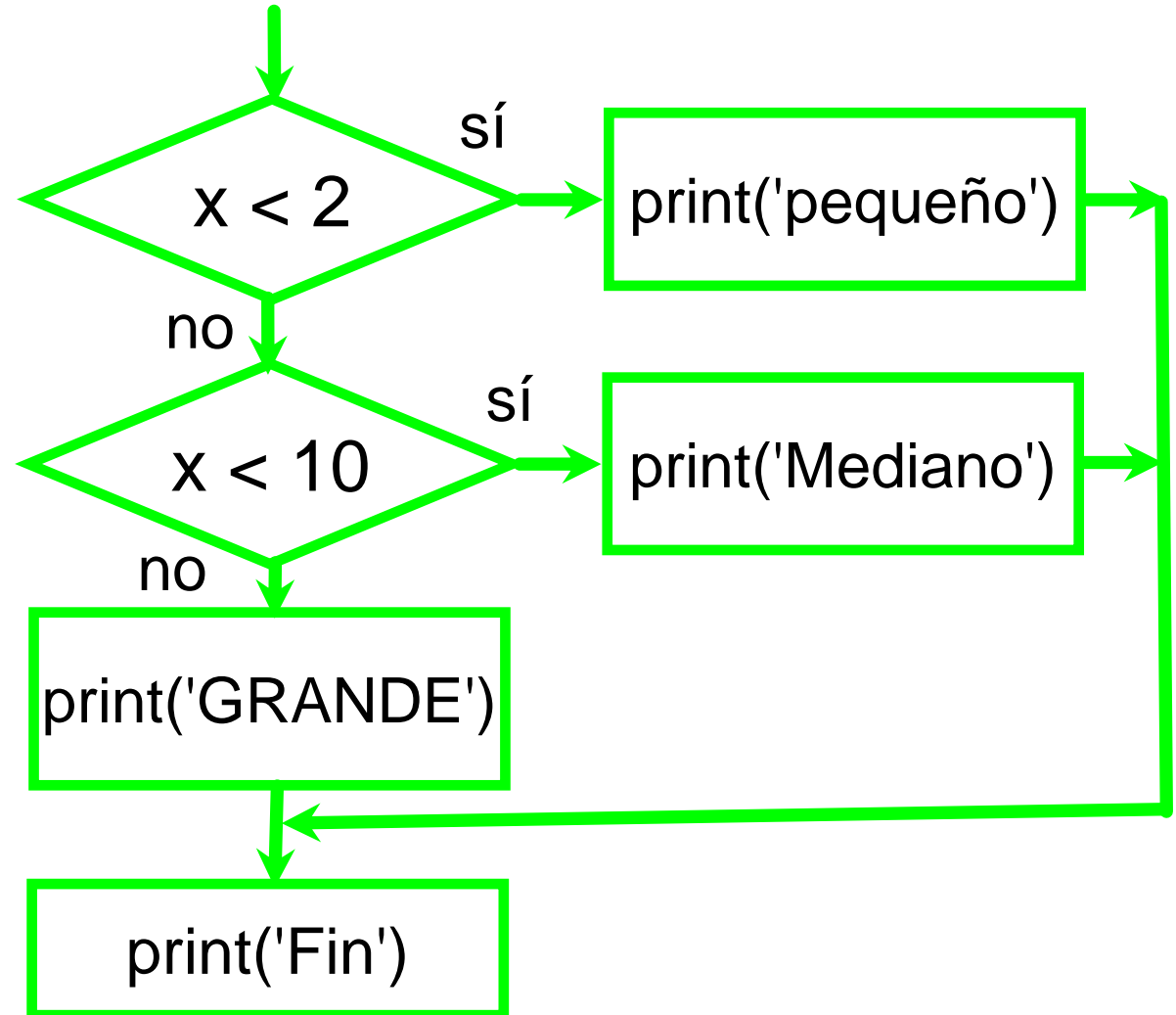
x = 4



Más Estructuras Condicionales...

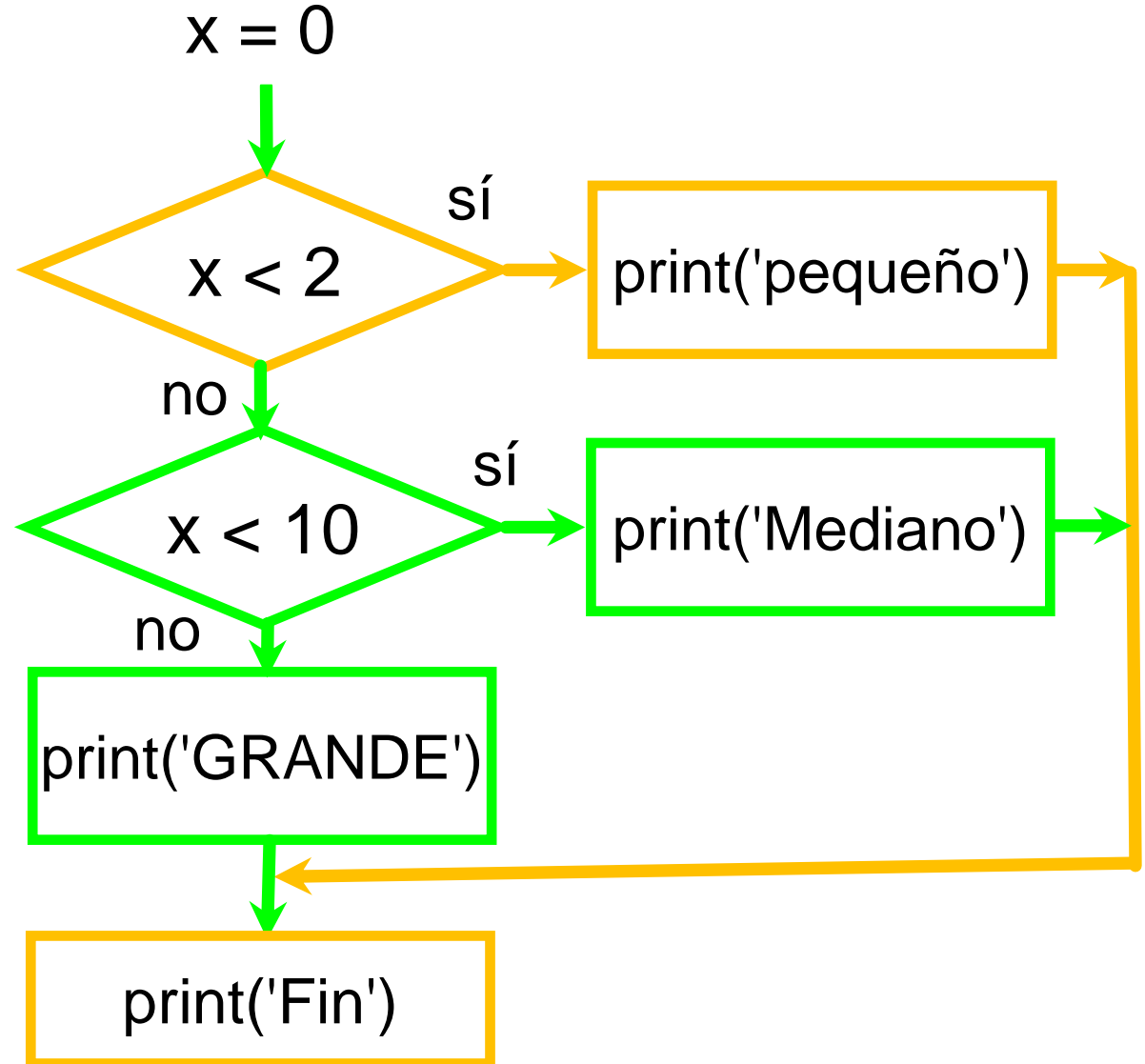
Múltiples rutas

```
if x < 2 :  
    print('pequeño')  
elif x < 10 :  
    print('Mediano')  
else :  
    print('GRANDE')  
print('Fin')
```



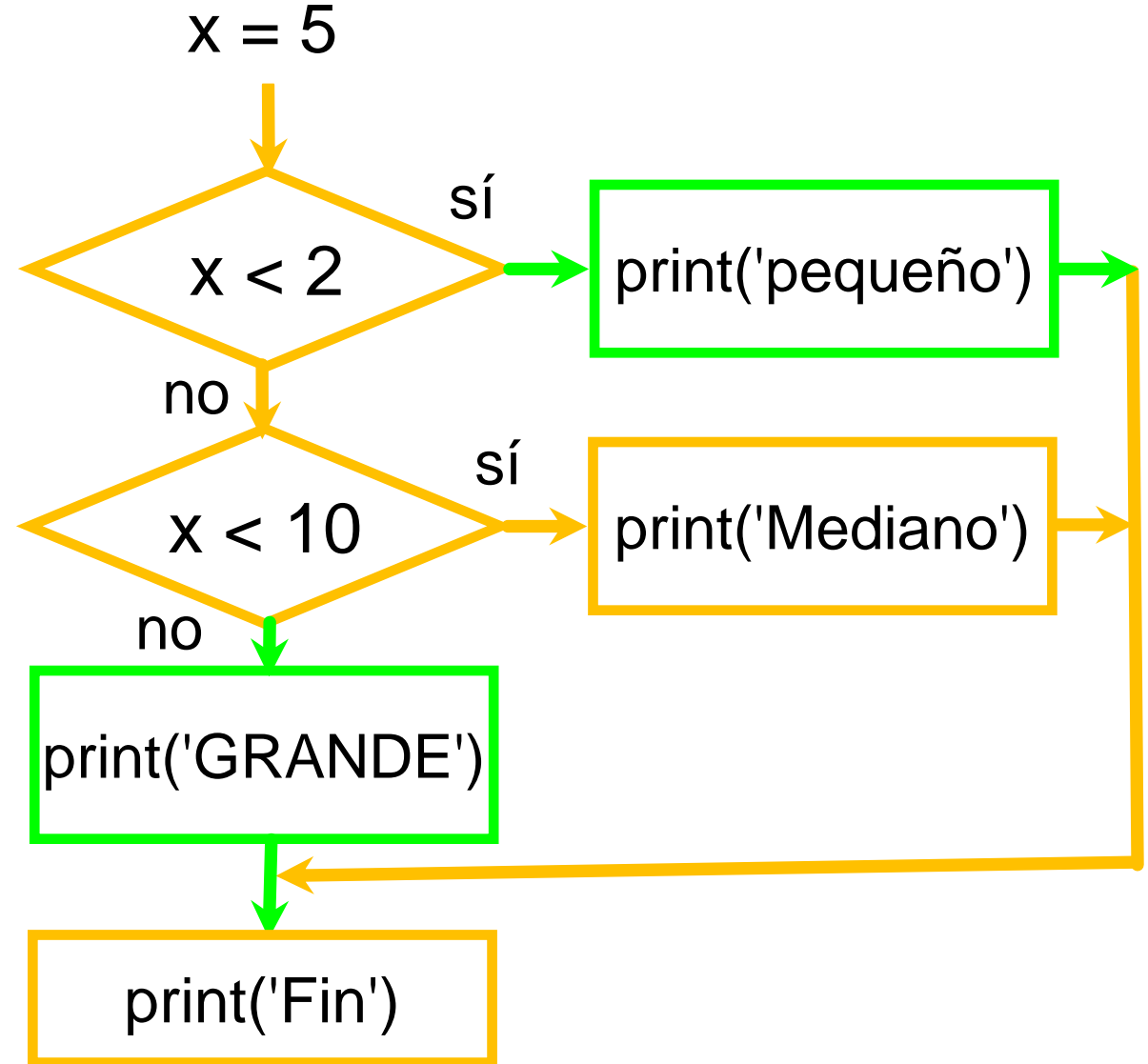
Múltiples rutas

```
x = 0
if x < 2 :
    print('pequeño')
elif x < 10 :
    print('Mediano')
else :
    print('GRANDE')
print('Fin')
```



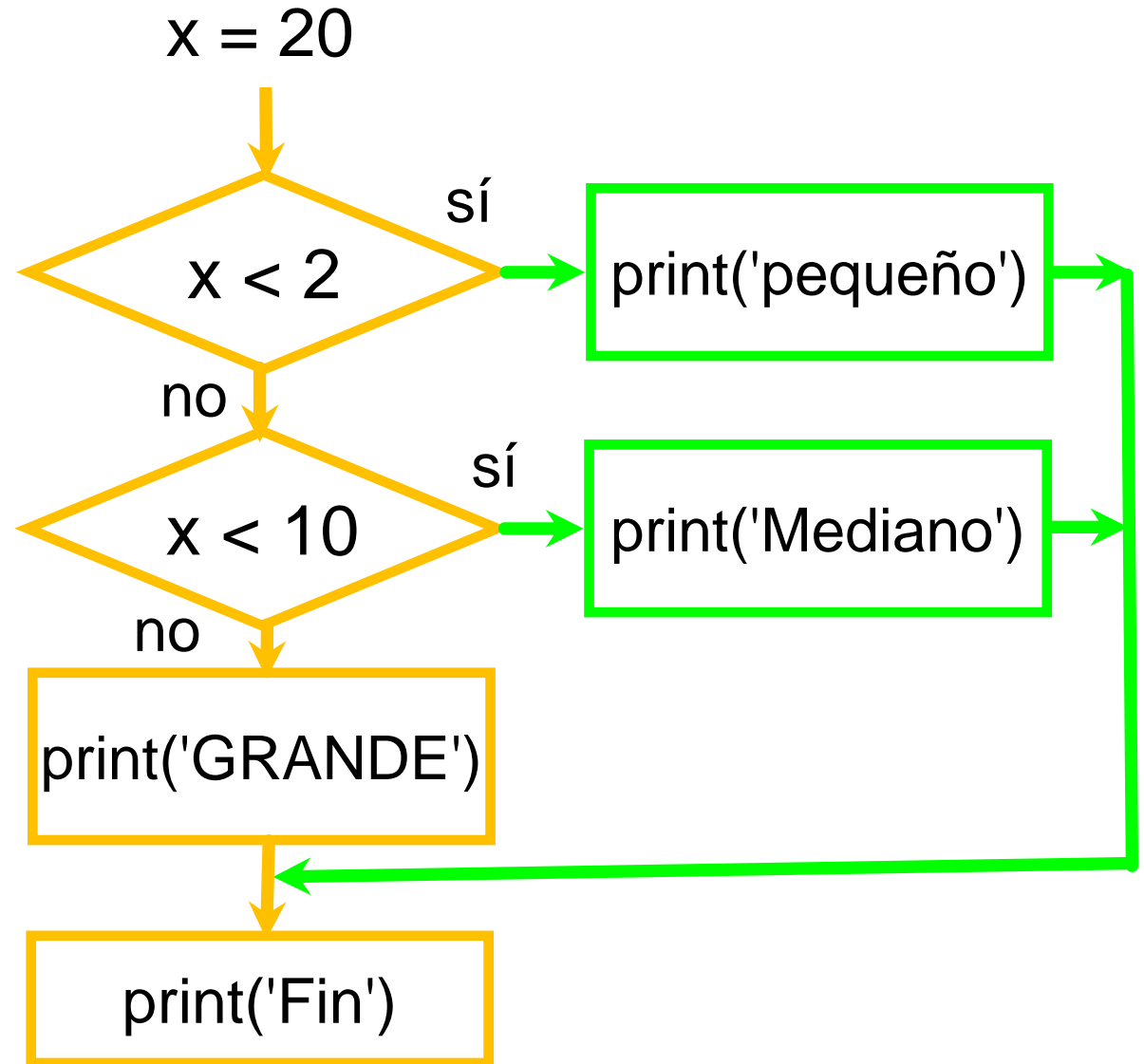
Múltiples rutas

```
x = 5
if x < 2 :
    print('pequeño')
elif x < 10 :
    print('Mediano')
else :
    print('GRANDE')
print('Fin')
```



Múltiples rutas

```
x = 20
if x < 2 :
    print('pequeño')
elif x < 10 :
    print('Mediano')
else :
    print('GRANDE')
print('Fin')
```



Múltiples rutas

```
# Sin else
x = 5
if x < 2 :
    print('Pequeño')
elif x < 10 :
    print('Mediano')

print('Fin')
```

```
if x < 2 :
    print('Pequeño')
elif x < 10 :
    print('Mediano')
elif x < 20 :
    print('Grande')
elif x < 40 :
    print('Enorme')
elif x < 100:
    print('Prodigioso')
else :
    print('Gigantesco')
```

Puzzles con múltiples rutas

¿Qué mensaje **nunca** se imprimirá independientemente del valor de x?

```
if x < 2 :  
    print('Menor que 2')  
elif x >= 2 :  
    print('2 o más')  
else :  
    print('Algo más')
```

```
if x < 2 :  
    print('Menor que 2')  
elif x < 20 :  
    print('Menor que 20')  
elif x < 10 :  
    print('Menor que 10')  
else :  
    print('Algo más')
```

Operadores and y or

¿Y si queremos comprobar que se cumplen varias condiciones al mismo tiempo? Operador **and**

```
if x > 2 and x < 8:  
    print('Entre 2 y 8')
```

¿Y si queremos comprobar que se cumple alguna condición? Operador **or**

```
if x < 2 or x > 8:  
    print('Menor que 2 o mayor que 8')
```

Operador not

Cuando buscamos que una condición no se cumpla

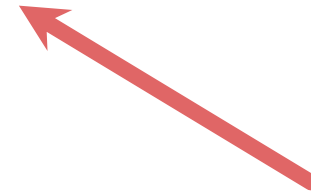
```
if not x > 2:  
    print('No mayor que 2')
```

La estructura try / except

- Las secciones de código "**peligrosas**" se rodean con try y except
- Si el código dentro del try funciona – el except se **salta**
- Si el código dentro del try **falla** – se **ejecuta** el bloque del except


```
$ cat notry.py
astr = 'Hola Bob'
istr = int(astr)
print('First', istr)
astr = '123'
istr = int(astr)
print('Second', istr)
```


```
$ python3 notry.py
Traceback (most recent call last):
File "notry.py", line 2, in <module>
istr = int(astr)ValueError: invalid literal
for int() with base 10: 'Hola Bob'
```



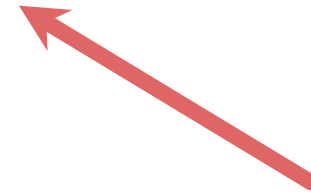
Fin

El
programa
para aquí

```
$ cat notry.py  
astr = 'Hola Bob'  
istr = int(astr)
```



```
$ python3 notry.py  
Traceback (most recent call last):  
File "notry.py", line 2, in <module>  
istr = int(astr)ValueError: invalid literal  
for int() with base 10: 'Hola Bob'
```



Fin

```
astr = 'Hola Bob'
try:
    istr = int(astr)
except:
    istr = -1

print('Primero', istr)
```

```
astr = '123'
try:
    istr = int(astr)
except:
    istr = -1

print('Segundo', istr)
```

Cuando la primera conversion falla
– salta al except y el programa
sigue.

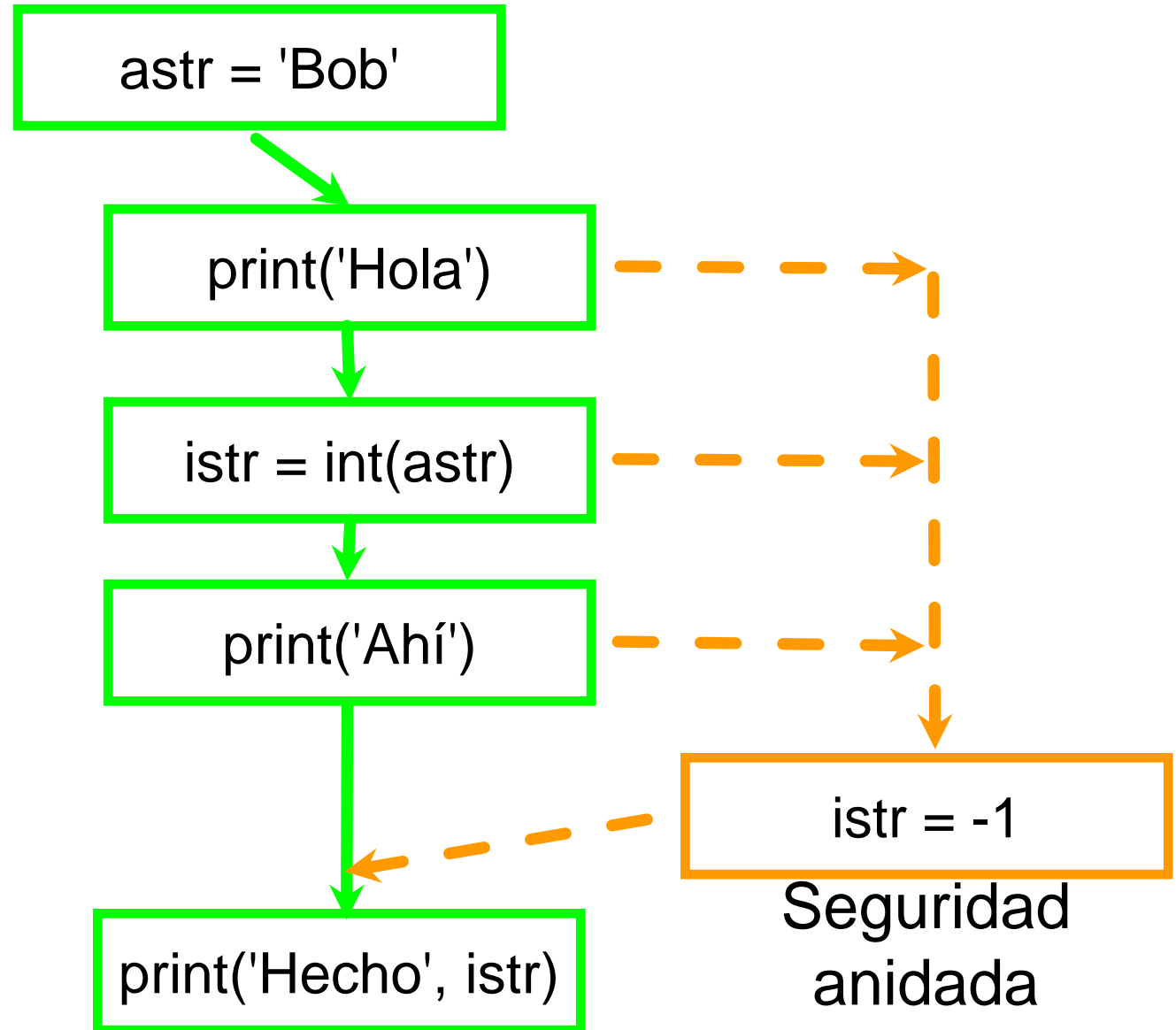
```
$ python tryexcept.py
Primero -1
Segundo 123
```

Cuando la segunda conversion
funciona – salta el except y el
programa sigue.

try / except

```
astr = 'Bob'
try:
    print('Hola')
    istr = int(astr)
    print('Ahí')
except:
    istr = -1

print('Hecho', istr)
```



Ejemplo try / except

```
rawstr = input('Dime un número:')
try:
    ival = int(rawstr)
except:
    ival = -1

if ival > 0 :
    print('Bien hecho')
else:
    print('No es un número')
```

```
$ python3 trynum.py
Dime un número:42
Bien hecho
$ python3 trynum.py
Dime un número:forty-two
No es un número
$
```

Resumen

- Operadores de comparación
== <= >= > < !=
- Indentación
- Decisiones simples
- Decisiones con alternativa:
if: and else:
- Decisiones anidadas
- Decisiones con múltiples ramas usando elif
- try / except para compensar errores

Ejercicio 1

Reescribe el programa de pagos para incrementar en un factor de 1.5 las horas que se trabajen por encima de 40 horas (hasta 40 horas todas se cobran a precio normal)

Introduzca las horas: 45

Introduzca el precio/hora: 10

Total: 475.0

$$475 = 40 * 10 + 5 * 15$$

Ejercicio 2

Reescribe el programa de pagos usando try y except para que se manejen adecuadamente entradas no numéricas (nota: no hace falta que se vuelvan a pedir los números si falla)

```
Introduzca las horas: 20
```

```
Introduzca el precio/hora: nueve
```

```
Error, por favor introduzca un  
número
```

```
Introduzca las horas: cuarenta
```

```
Error, por favor introduzca un  
número
```


Acknowledgements / Contributions



These slides are Copyright 2010- Charles R. Severance (www.dr-chuck.com) of the University of Michigan School of Information and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

Continue...

Initial Development: Charles Severance, University of Michigan School of Information

... Insert new Contributors and Translators here
Spanish Version: Daniel Garrido (dgm@uma.es)