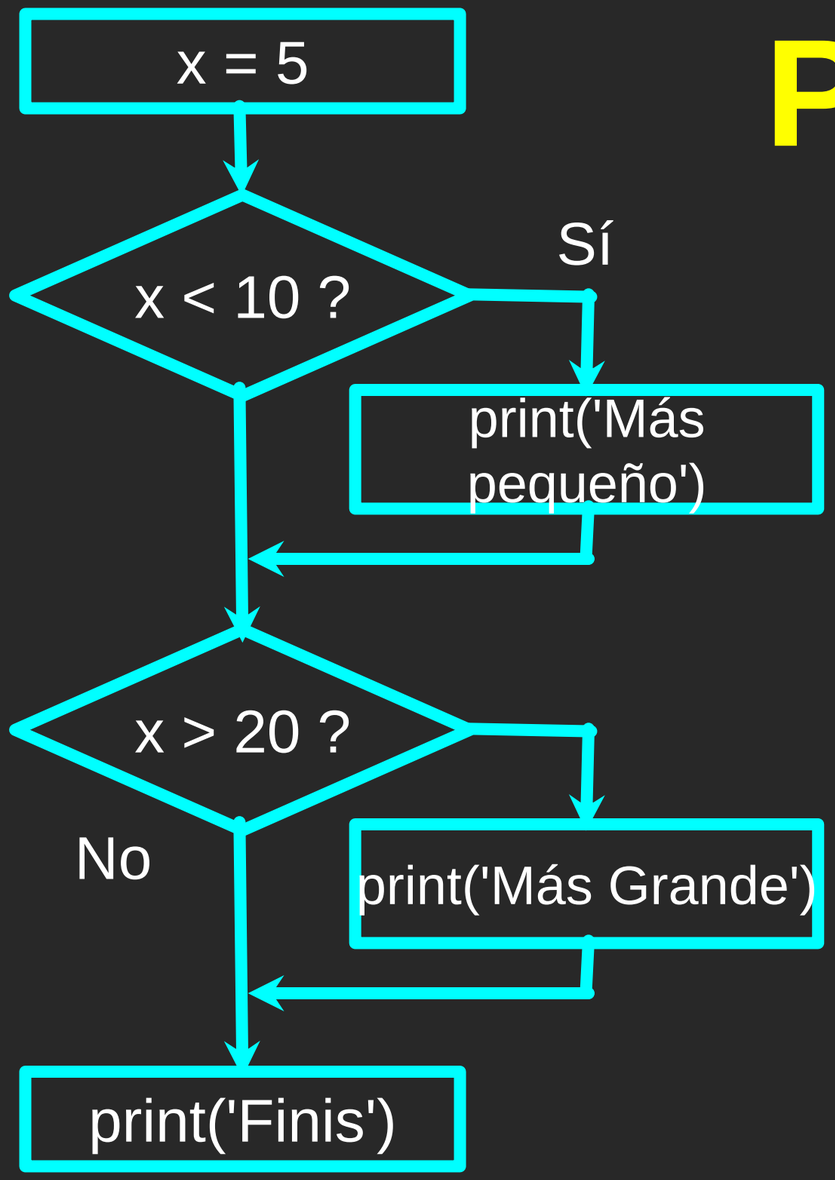
Ejecución Condicional

Pasos Condicionales



Programa:

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

print('Finis')
```

Resultado:

```
Más pequeño
Finis
```

Operadores de Comparación

- Las expresiones booleanas formulan una pregunta y generan un resultado Yes (afirmativo) o No (negativo) que utilizamos para controlar el flujo del programa

- Las expresiones booleanas utilizan operadores de comparación para evaluar si es True (Verdadero) / False (Falso) o Yes (Sí) / No

- Los operadores de comparación observan las variables pero no las modifican

Python	Significado
<	Menor que
<=	Menor que o Igual a
==	Igual a
>=	Mayor que o igual a
>	Mayor que
!=	No igual a

Recuerde: “=” se usa para asignación.

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

Operadores de Comparación

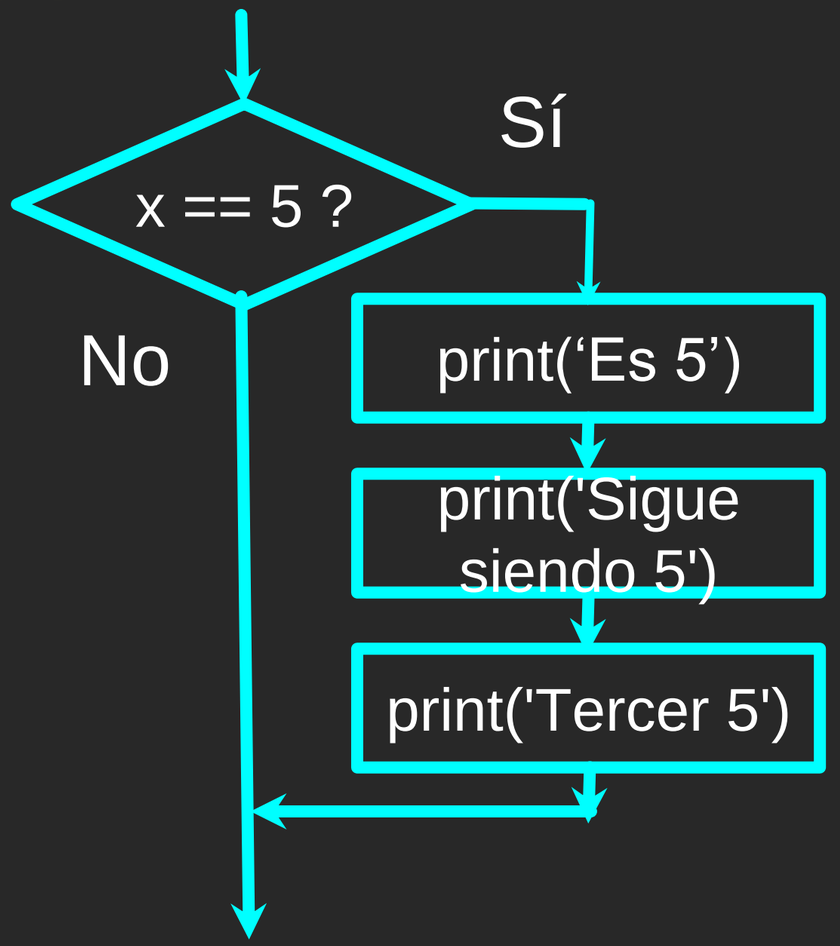
```
x = 5
if x == 5 :
    print('Igual a 5')
if x > 4 :
   print('Mayor que 4')
if  x >= 5 :
    print('Mayor que o Igual a 5')
if x < 6 : print('Menor que 6')
if x <= 5 :
    print('Menor que o Igual a 5')
if x != 6 :
    print('No igual a 6')
```

```
Igual a 5
Mayor que 4
Mayor que o Igual a 5
Menor que 6
Menor que o Igual a 5
No igual a 6
```

Decisiones Unidireccionales

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

```
Antes de 5
Es 5
Sigue siendo 5
Tercer 5
Después de 5
Antes de 6
Después de 6
```



Indentación

- Aumentar la indentación sirve para indentar luego de un enunciado if o for (después:)
- Mantener la indentación sirve para indicar el alcance del bloque (qué líneas son afectadas por if/for)
- Reducir la indentación permite regresarla al nivel del enunciado if o for para indicar el final del bloque
- Las líneas en blanco son ignoradas y no afectan la indentación
- Los comentarios en una línea en sí mismos se ignoran en lo que respecta a la indentación

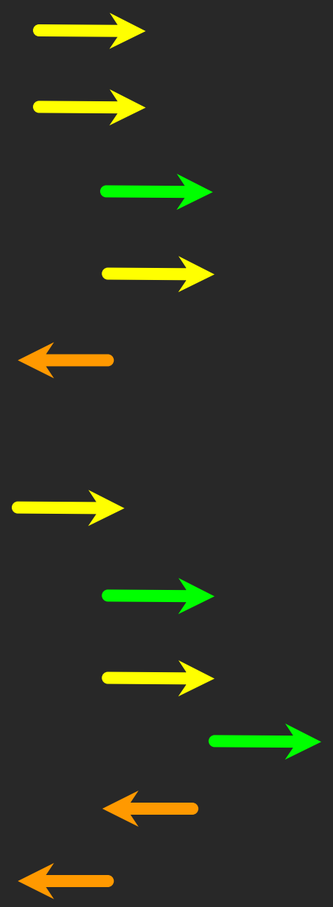
Advertencia: ¡Deshabilite las Tabulaciones!

Atom automáticamente usa los espacios para los archivos con la extensión ".py" (¡genial!)

- La mayoría de los editores de texto pueden convertir las tabulaciones en espacios – asegúrese de habilitar esta funcionalidad
  - NotePad++: Settings -> Preferences -> Language Menu/Tab Settings (Configuración -> Preferencias -> Menú de Idiomas/Configuración de Tabulación)
  - TextWrangler: TextWrangler -> Preferences -> Editor Defaults (TextWrangler: TextWrangler -> Preferencias -> Valores Predeterminados del Editor)
- A Python le importa *mucho* cuánta indentación tiene una línea. Si usted mezcla tabulaciones y espacios, tal vez obtenga “indentation errors” (errores de indentación) incluso aunque todo se vea bien

aumentar / mantener después de if o for

reducir para indicar el final del bloque



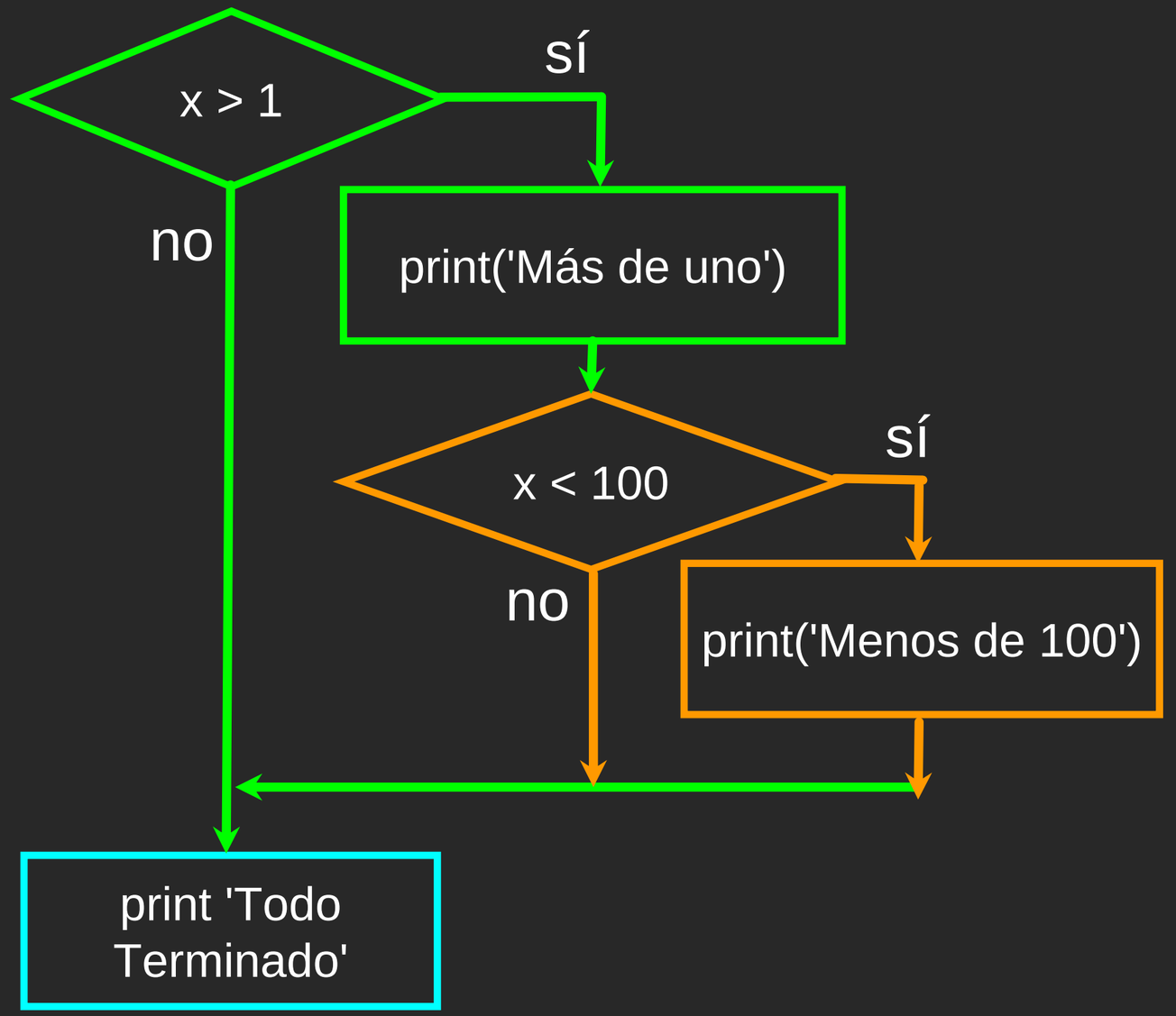
```
x = 5
if x > 2 :
    print('Mayor que 2')
    print('Sigue siendo mayor')
print('Terminado con 2')

for i in rango(5) :
    print(i)
    if i > 2 :
        print('Mayor que 2')
    print('Terminado con i', i)
print('Todo Terminado')
```

Piense en los bloques de inicio/fin

```
x = 5
if x > 2 :
    print('Mayor que 2')
    print('Sigue siendo mayor')
print('Terminado con 2’)

for i in rango(5) :
    print(i)
    if i > 2 :
        print('Mayor que 2')
 print('Terminado con i', i)

print('Todo Terminado')
```

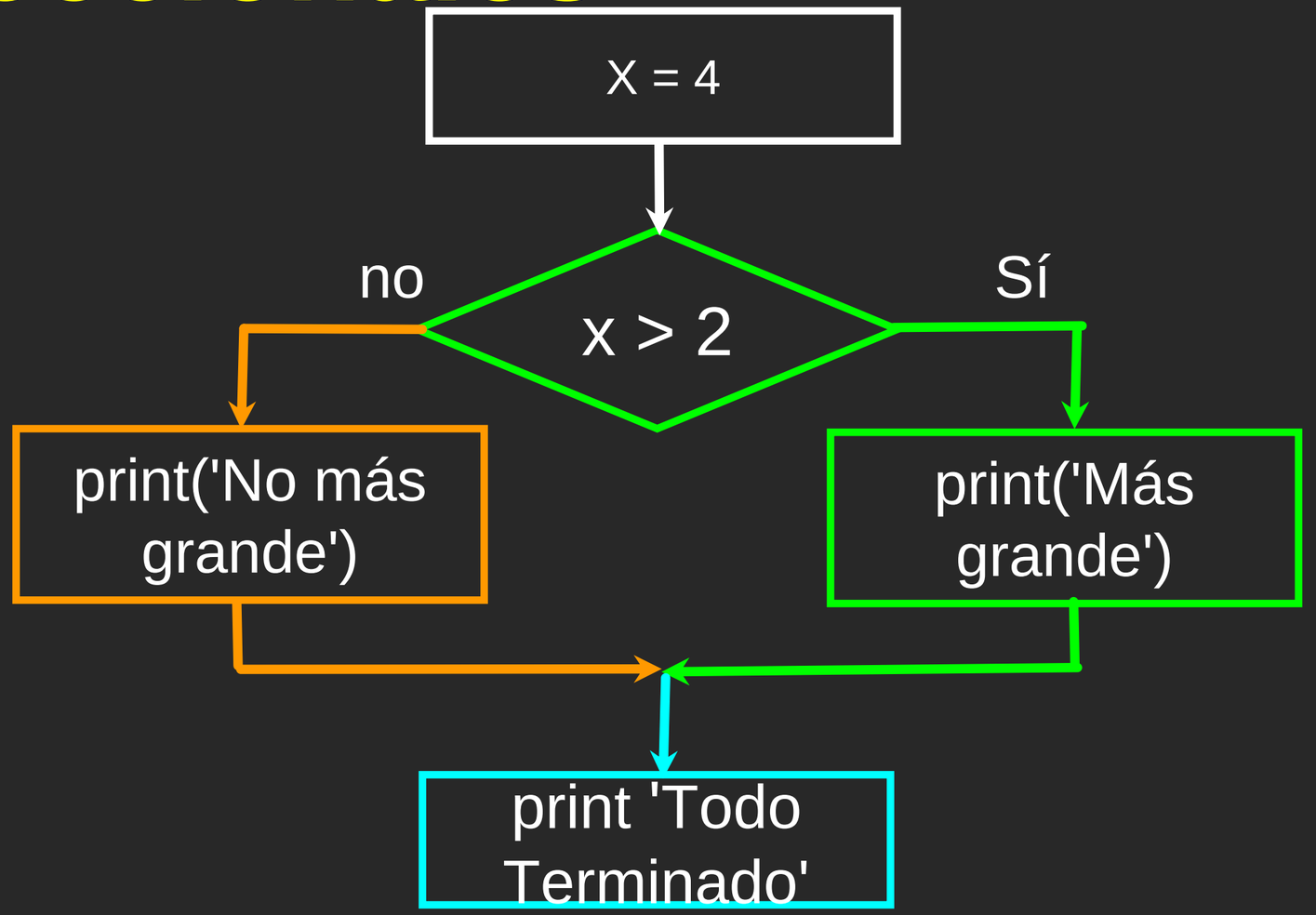
Decisiones Anidadas

```
x = 42
if x > 1 :
    print('Más de 1')
    if x < 100 :
        print('Menos de 100')
print('Todo Terminado')
```



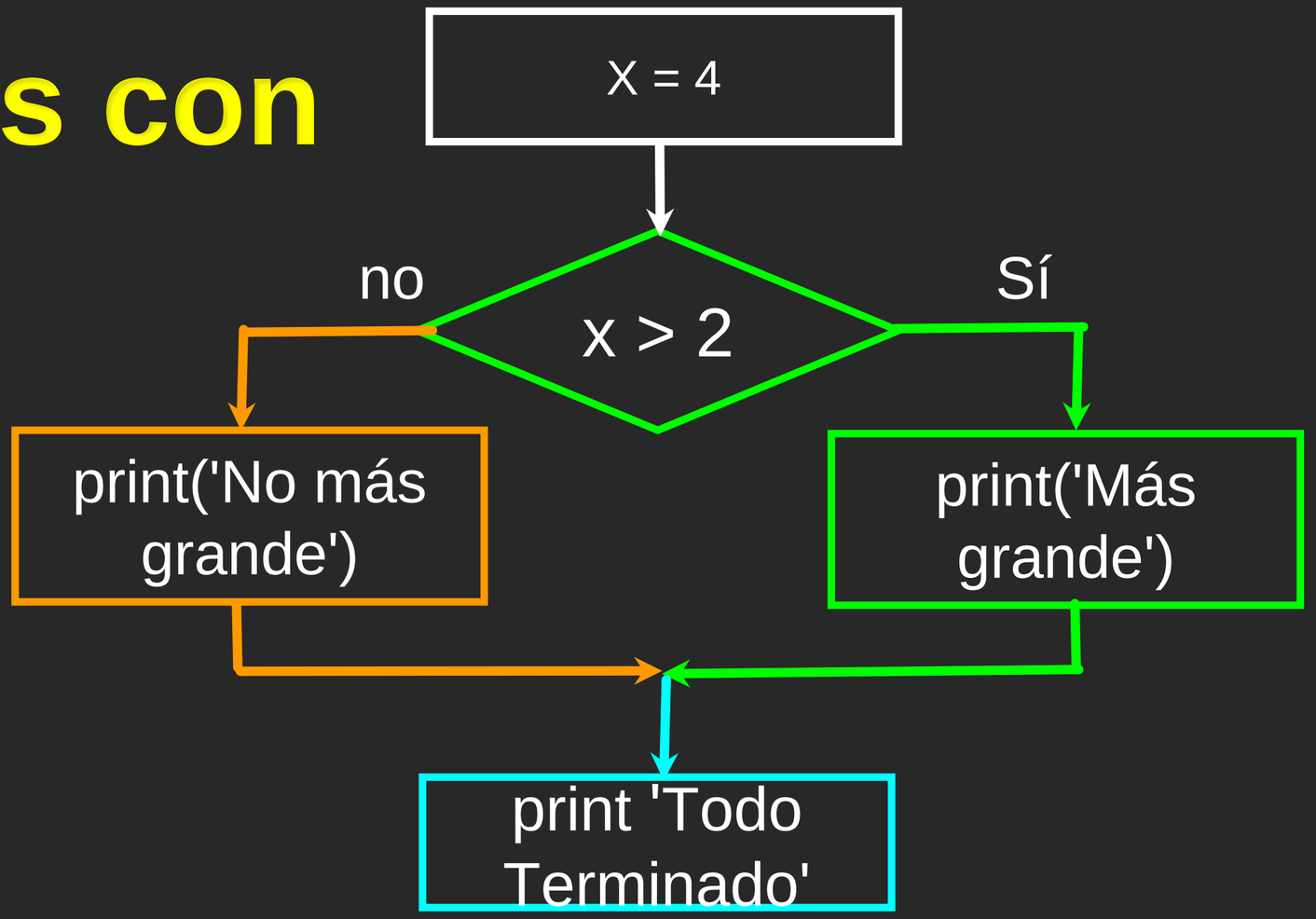
Decisiones Bidireccionales

- **A veces, queremos hacer una cosa si una expresión lógica es verdadera y otra cosa si la expresión es falsa**
- **Es como una encrucijada – debemos elegir un camino u otro pero no podemos elegir ambos**



Decisiones Bidireccionales con else:

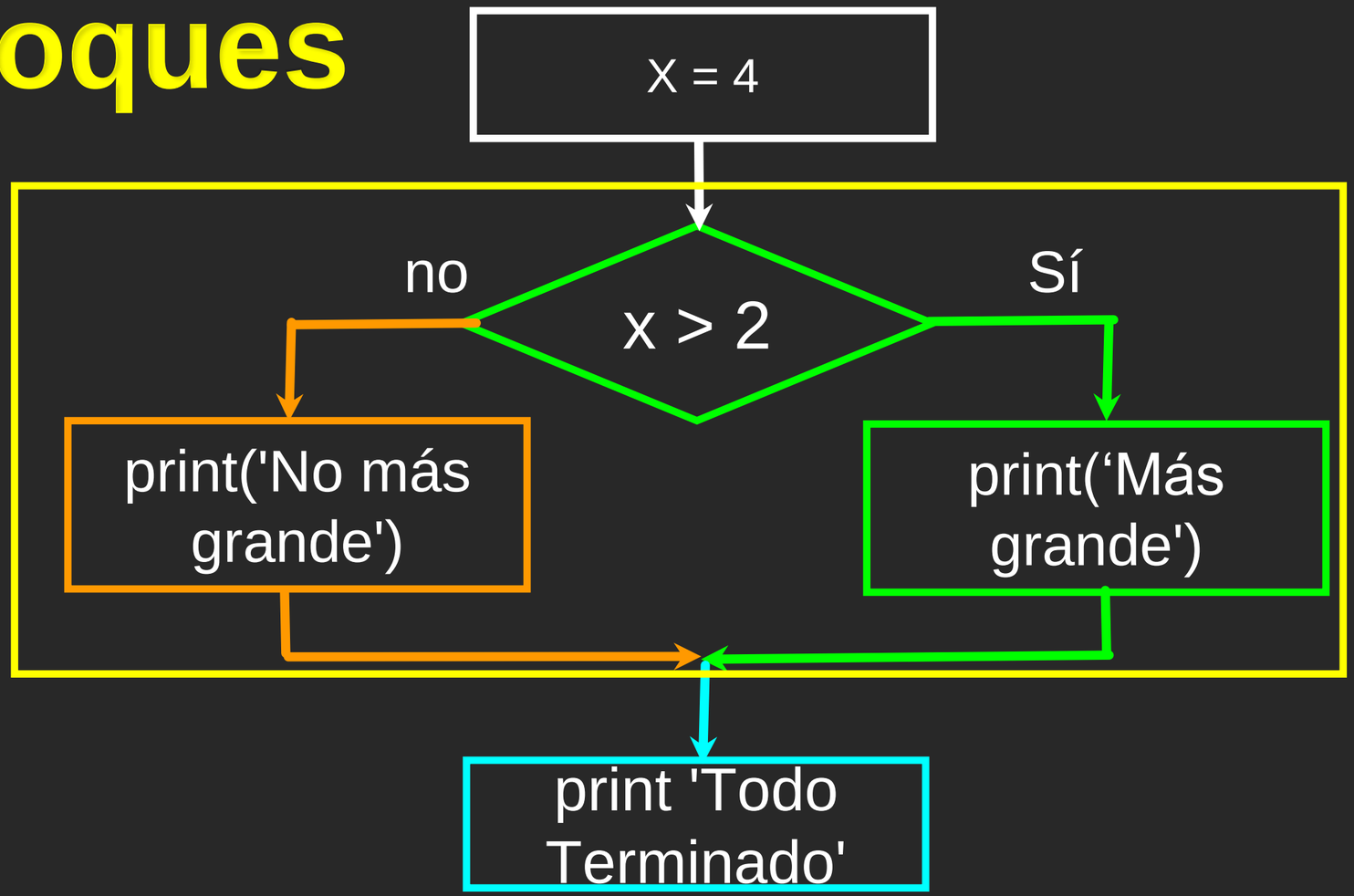
```
x = 4

if x > 2 :
    print('Más grande')
else :
    print('Más pequeño')

print 'Todo Terminado'
```



Más Patrones de Ejecución Condicional

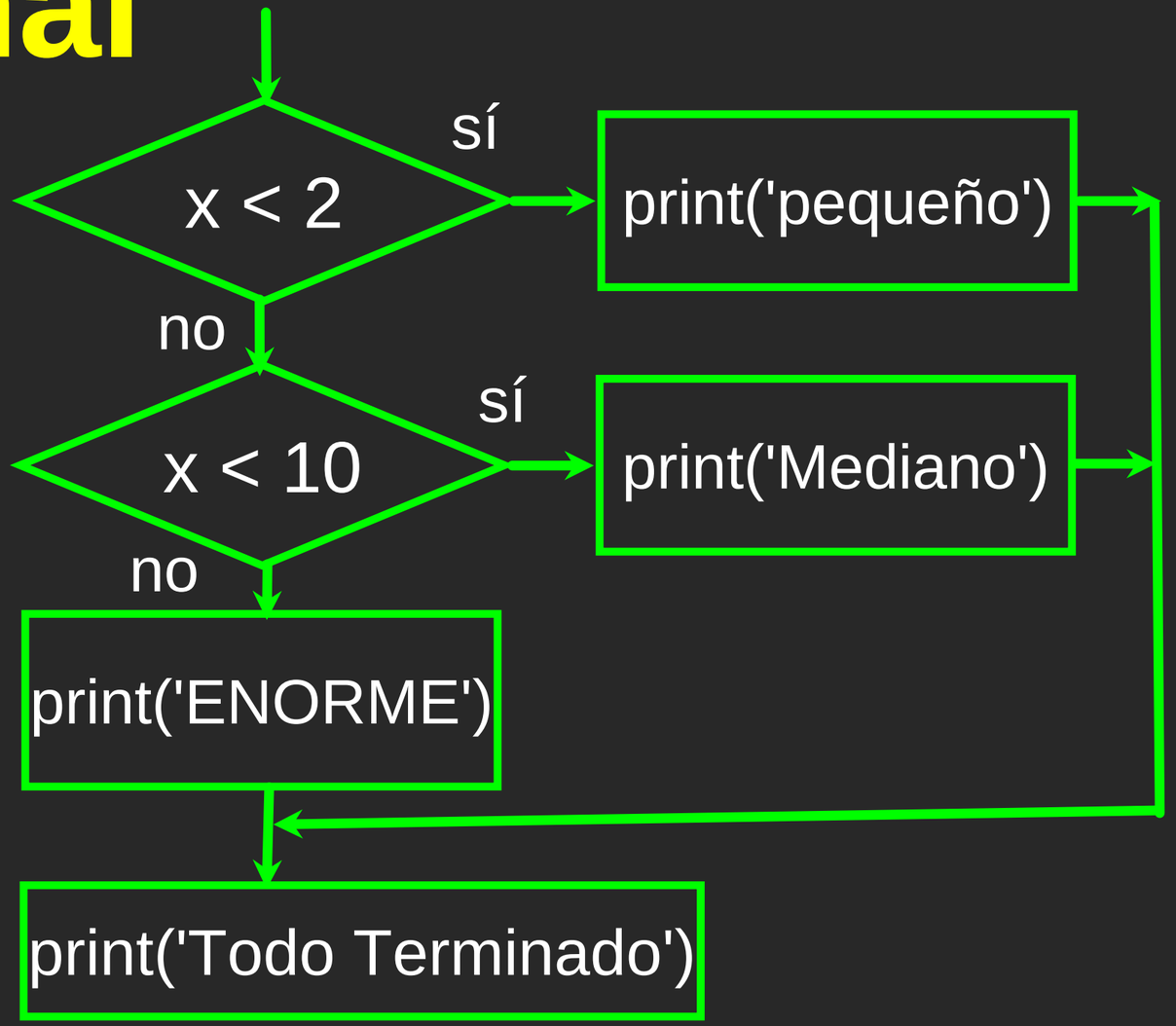
Visualizar Bloques

```
x = 4

if x > 2 :
    print('Más grande')
else :
    print('Más pequeño')

print 'Todo Terminado'
```



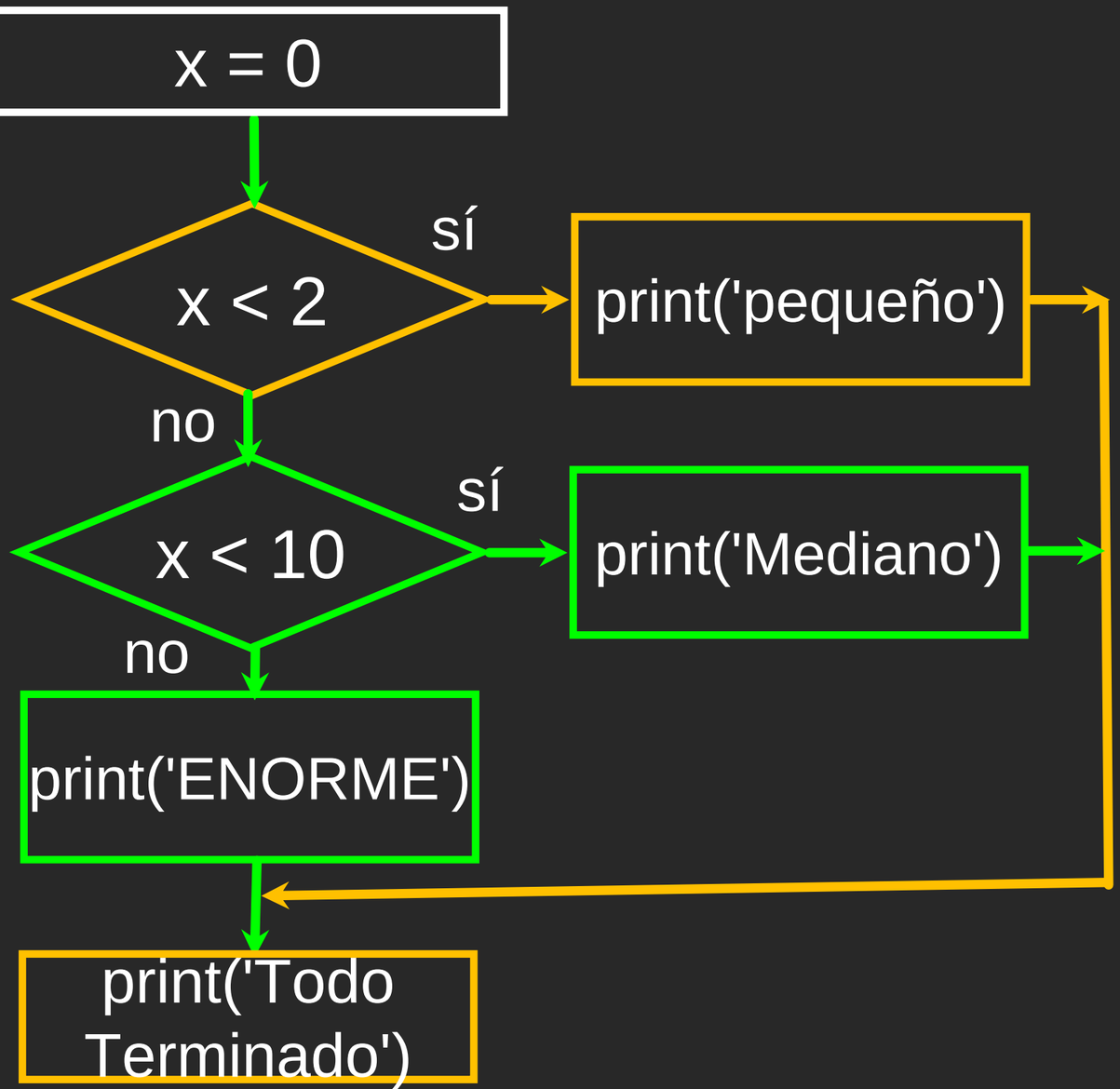
Multidireccional

```
if x < 2 :
    print('Pequeño')
elif x < 10 :
    print('Mediano')
else :
    print('ENORME')
print('Todo terminado')
```



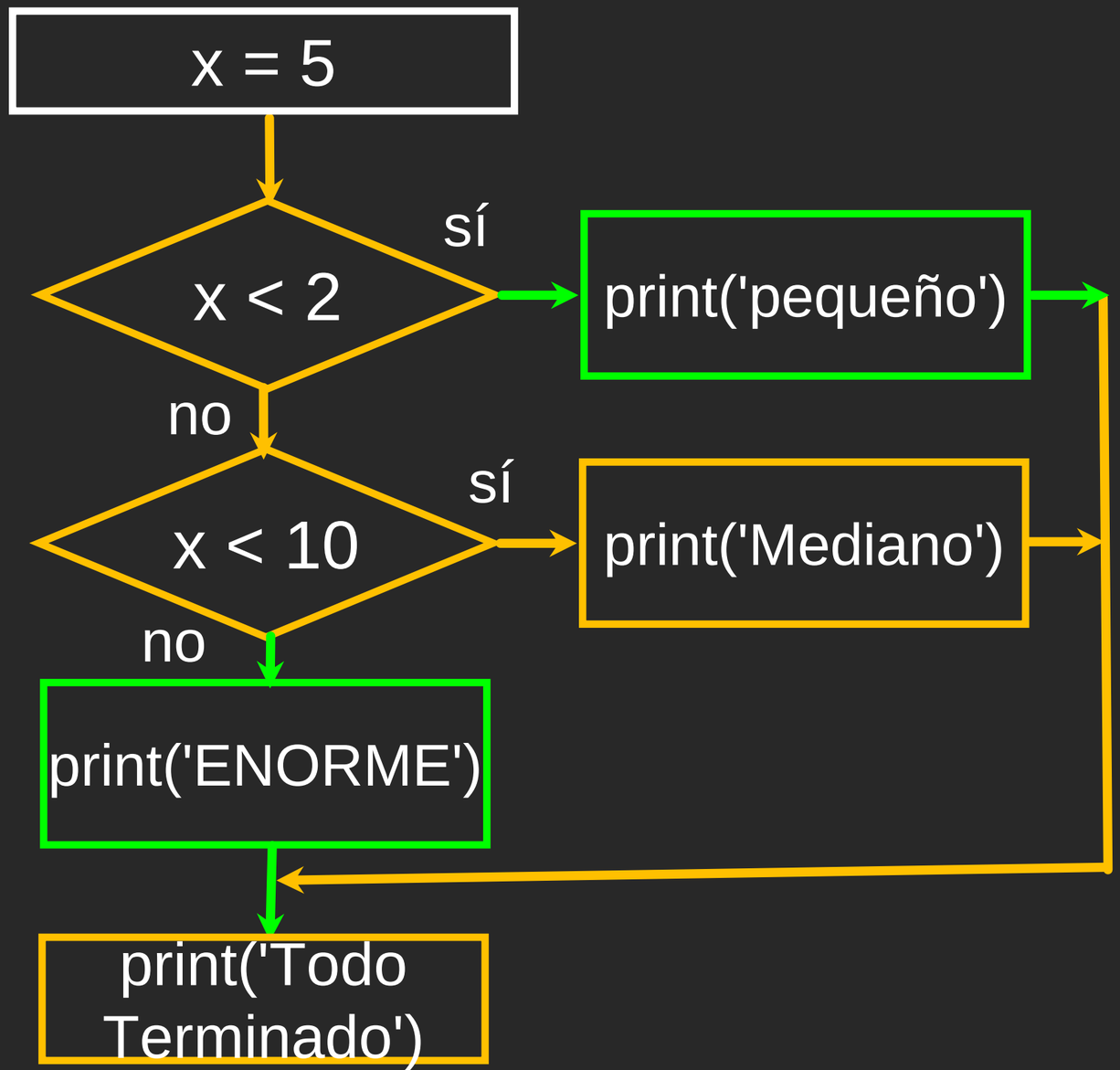
Multidireccional

```
x = 0
if x < 2 :
    print('pequeño')
elif x < 10 :
    print('Mediano')
else :
    print('ENORME')
print('Todo
terminado')
```



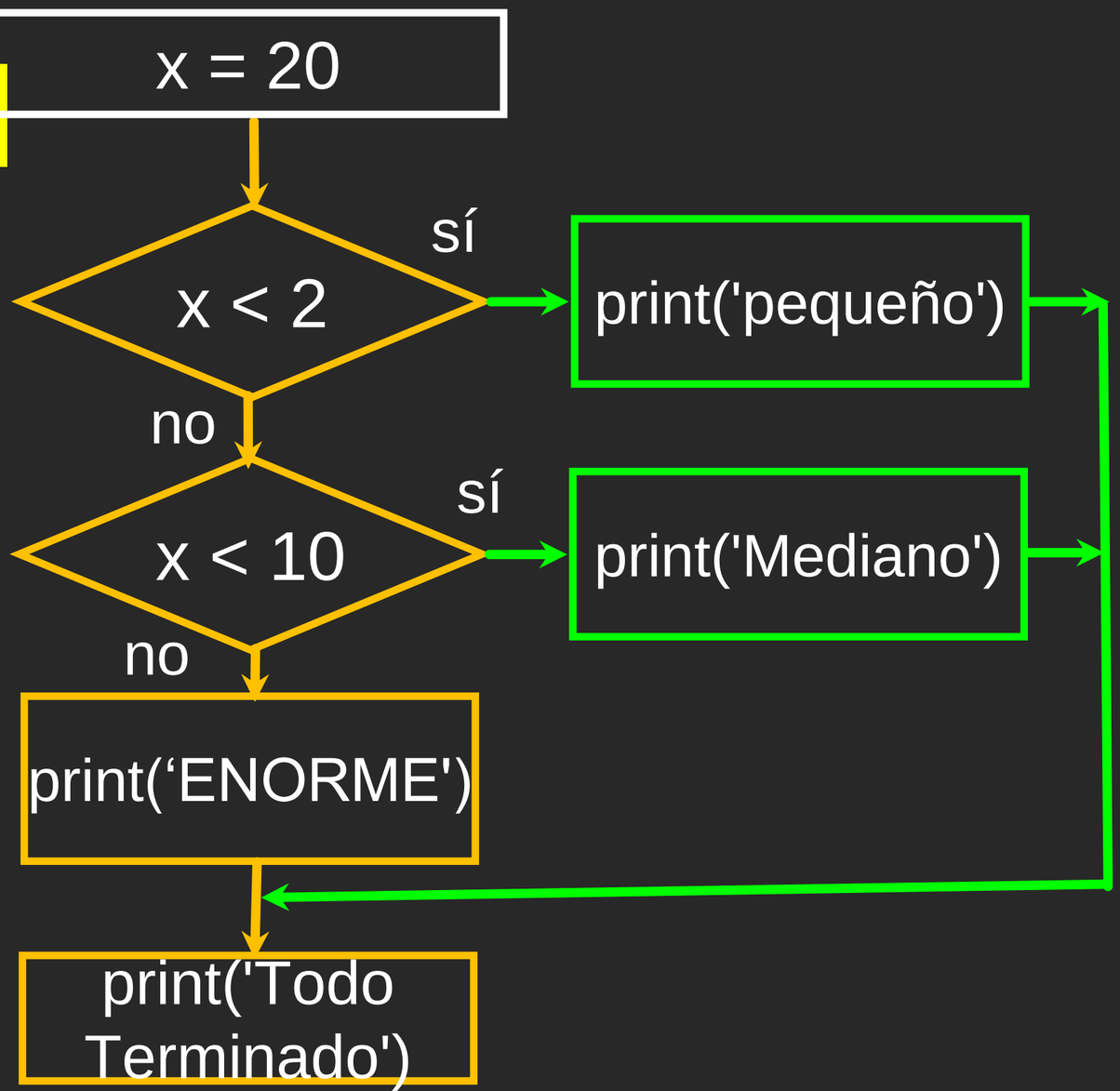
Multidireccional

```
x = 5
if x < 2 :
    print('pequeño')
elif x < 10 :
    print('Mediano')
else :
    print('ENROME')
print('Todo
terminado')
```



Multidireccional

```
x = 20
if x < 2 :
    print('pequeño')
elif x < 10 :
    print('Mediano')
else :
    print('ENORME')
print('Todo
terminado')
```



Multidireccional

```
# No Else
x = 5
if x < 2 :
    print('Pequeño')
elif x < 10 :
    print('Mediano')

print 'Todo terminado'
```

```
if x < 2 :
    print('pequeño')
elif x < 10 :
    print('Mediano')
elif x < 20 :
    print('Grande')
elif x < 40 :
    print('Enorme')
elif x < 100:
    print('Gigante')
else :
    print('Descomunal')
```

Enigmas Multidireccionales

¿Cuál es el que nunca se imprimirá independientemente del valor de x?

```
if x < 2 :
    print('Debajo de 2')
elif x >= 2 :
    print('Dos o más')
else :
    print('Otro')
```

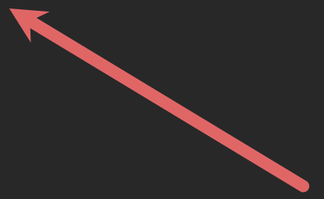
```
if x < 2 :
    print('Debajo de 2')
elif x < 20 :
    print('Debajo de 20')
elif x < 10 :
    print('Debajo de 10')
else :
    print('Otro')
```

La Estructura try / except

- Usted rodea una sección peligrosa del código con try y except
- Si el código en try funciona – except es omitido
- Si el código en try falla – pasa a la sección except 5

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

```
$ python3 notry.py
Traza de rastreo (llamada más reciente a la último):  Archivo "notry.py", línea 2, in <module>    istr = int(astr)ValueError: invalid literal for int() with base 10: 'Hola Bob'
```



Todo Terminado

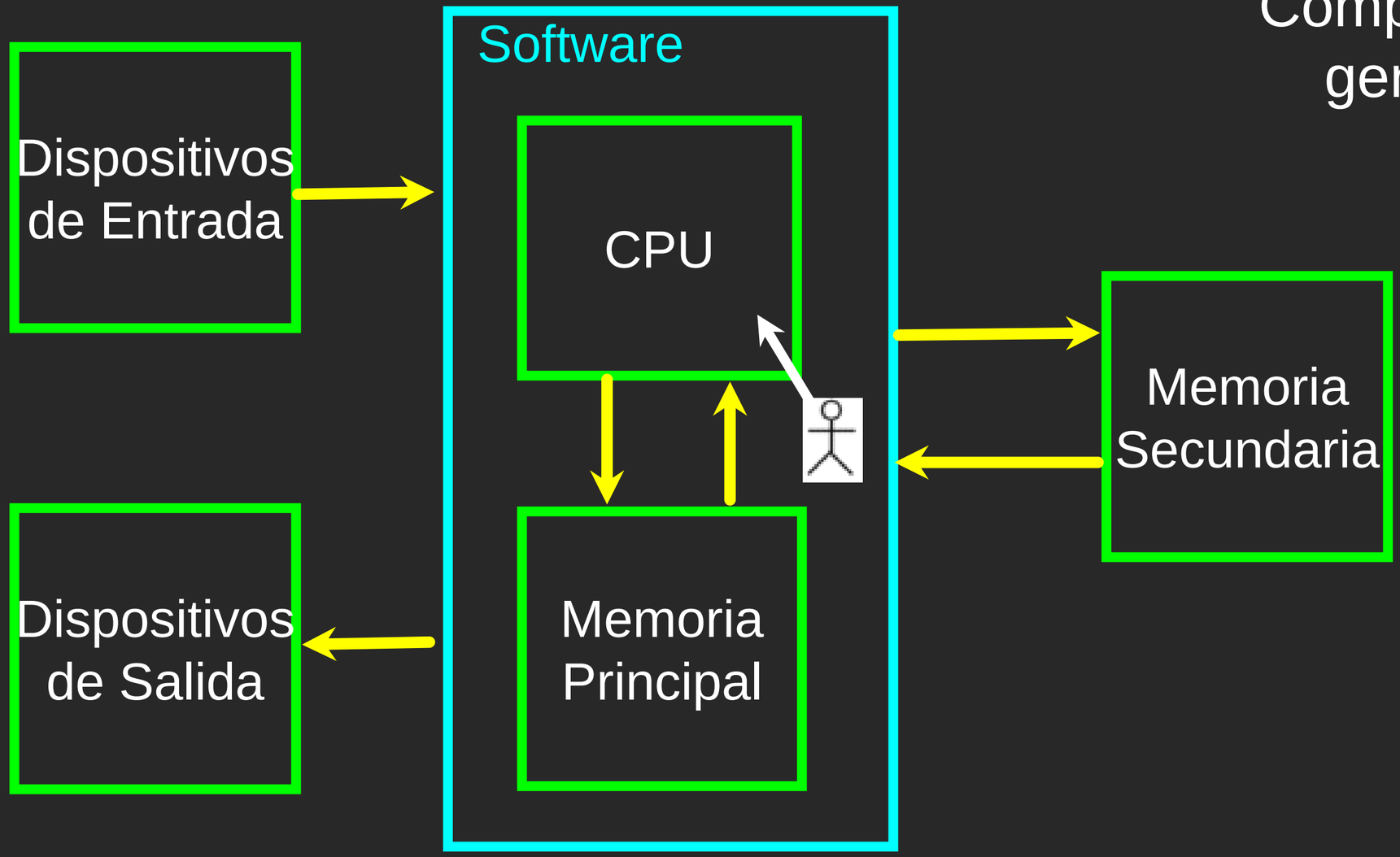
El programa se detiene aquí

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

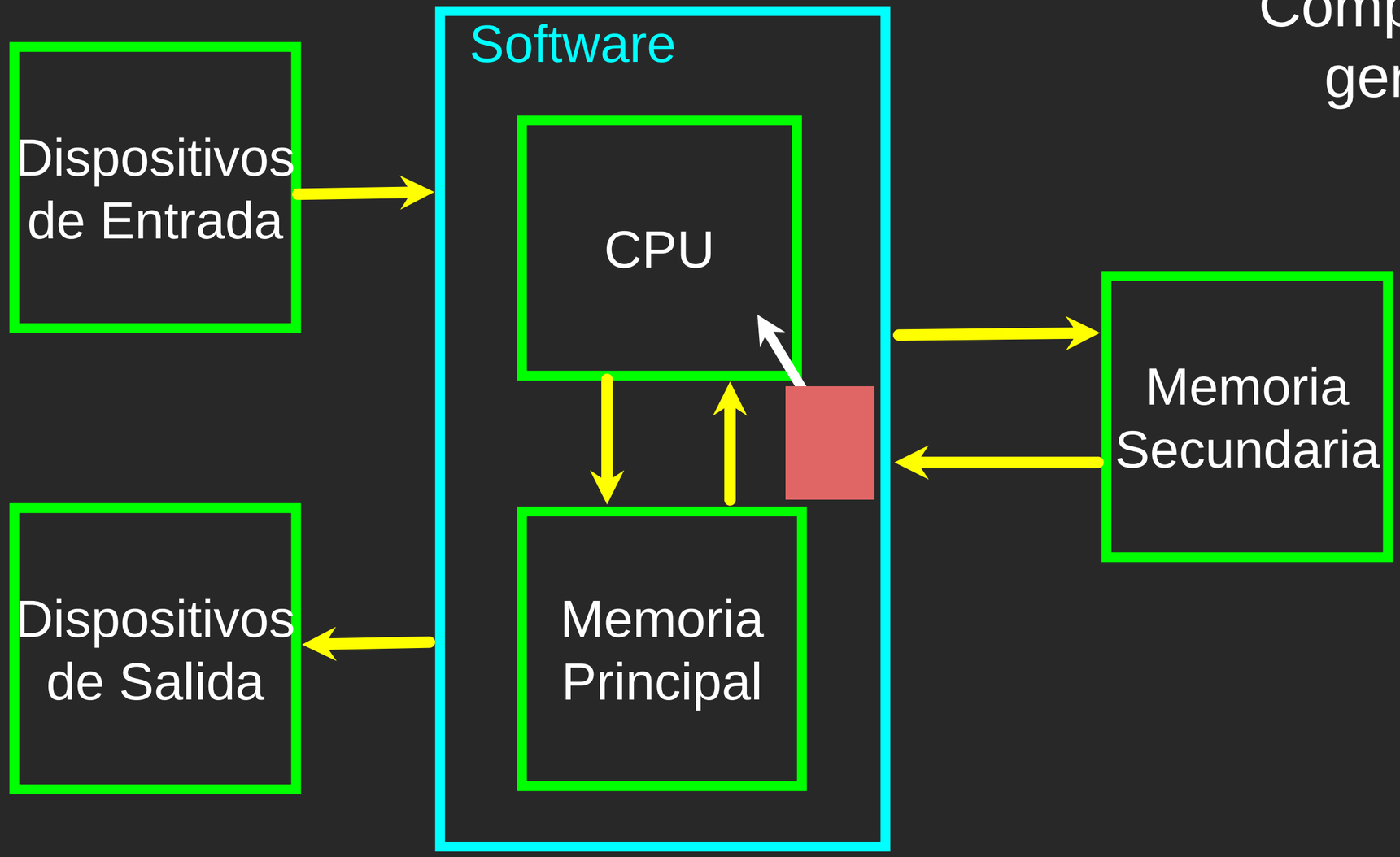
```
$ python3 notry.py
Trazas de rastreo (llamada más reciente a lo último):  Archivo "notry.py", línea 2, in <module>    istr = int(astr)ValueError: invalid literal for int() with base 10: 'Hola Bob'
```

Todo Terminado

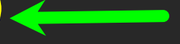
Computadora genérica



Computadora genérica



```
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)
```



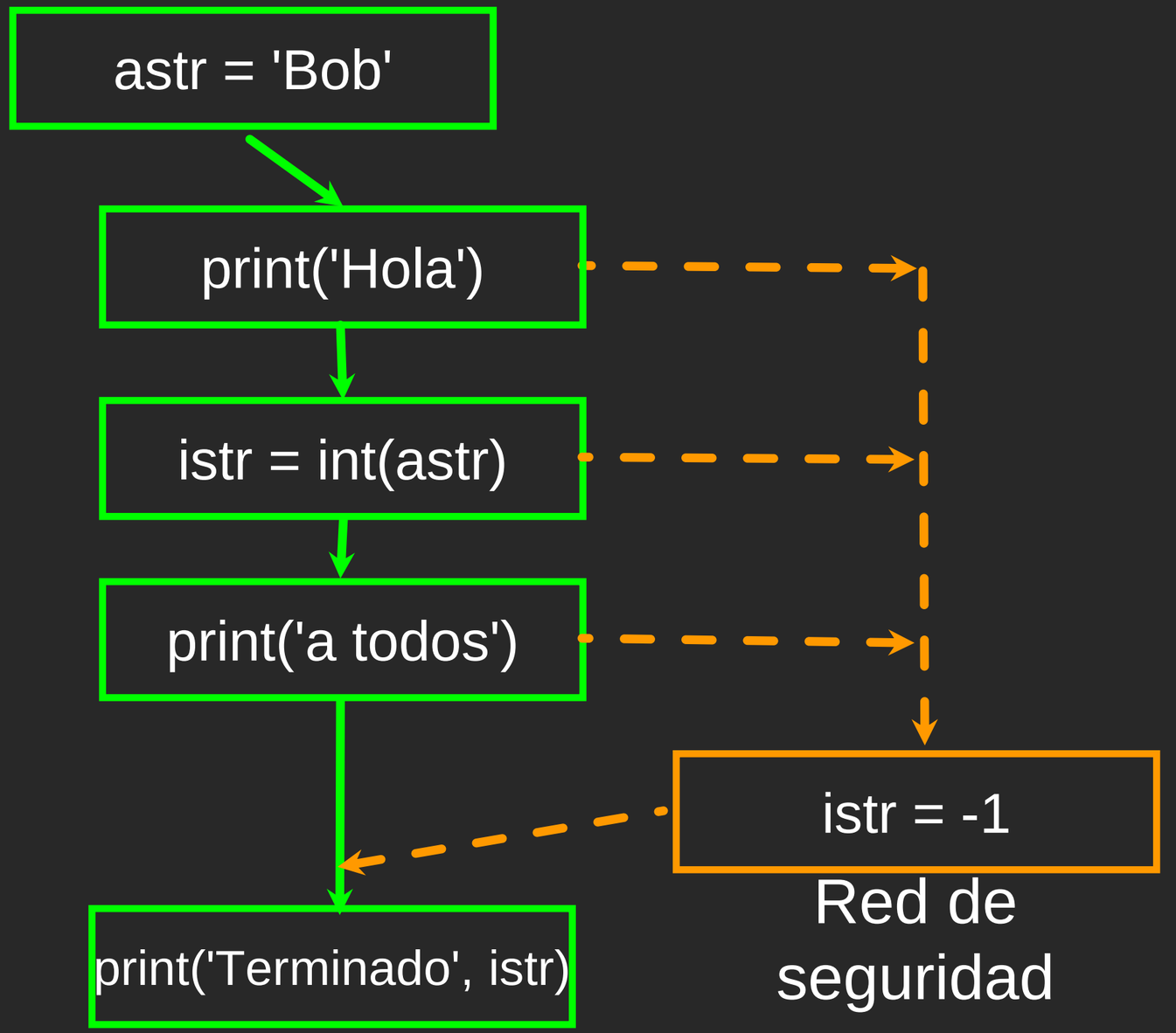
Cuando la primera conversión falla – simplemente cae en except (excepción): clausula, y el programa continúa.

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

Cuando la segunda conversión es exitosa – solo omite except (excepción): clausula, y el programa continúa.

try / except

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

print('Terminado',
istr)
```



Muestra de try / except

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

if ival > 0 :
    print('Buen trabajo')
else:
    print('No es un número')
```

```
$ python3 trynum.py
Ingresar un número:42
Buen trabajo
$ python3 trynum.py
Ingresar un
número:cuarenta-y-dos
No es un número
$
```

Ejercicio

Reescriba su cálculo del salario para darle al empleado 1,5 veces la tarifa por hora para las horas trabajadas que excedan las 40 horas.

```
Ingresar Horas: 45
Ingresar Tarifa: 10

Salario: 475.0
```

475 = 40 * 10 + 5 * 15

Ejercicio

Reescriba su programa de salarios usando try y except de modo que su programa maneje input (entradas) no numéricas de forma correcta.

```
Ingresar Horas: 20
Ingresar Tarifa: nueve
Error, por favor, ingresar un valor numérico

Ingresar Horas: cuarenta
Error, por favor, ingresar un valor numérico
```

Síntesis

- Operadores de comparación
  == <= >= > < ! =
- Indentación
- Decisiones Unidireccionales
- Decisiones Bidireccionales:
  if: y else:
- Decisiones Anidadas
- Decisiones Multidireccionales usando elif
- try / except para compensar errores