

# 조건문 실행

## 제3장

모두를 위한 파이썬

[www.py4e.com](http://www.py4e.com)



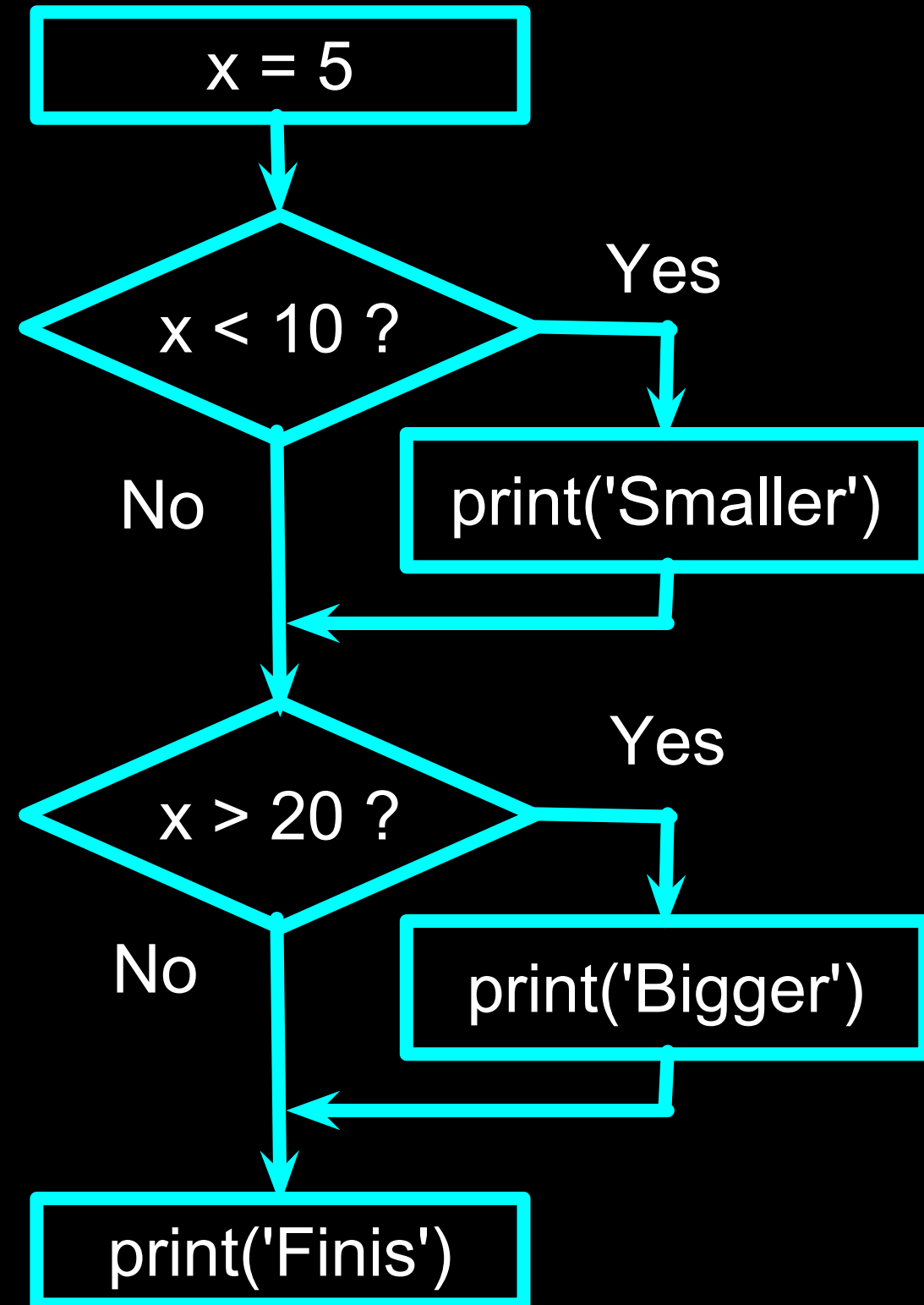
# 조건문

프로그램:

```
x = 5
if x < 10:
    print('Smaller')
if x > 20:
    print('Bigger')
print('Finis')
```

결과:

Smaller  
Finis



# 비교 연산자

- 부울 표현식은 네/아니오 질문을 통해 프로그램의 흐름을 결정
- 부울 표현식은 비교 연산자로 참/거짓, 네 / 아니오를 판단
- 비교 연산자는 변수를 바꾸지 않음

파이썬	의미
<	작다
<=	작거나 같다
==	같다
>=	크거나 같다
>	크다
!=	같지 않다

“=”는 대입문에 쓰임

[http://en.wikipedia.org/wiki/George\\_Boole](http://en.wikipedia.org/wiki/George_Boole)

# 비교 연산자

```
x = 5
```

```
if x == 5 :
```

```
    print('Equals 5')
```

Equals 5

```
if x > 4 :
```

```
    print('Greater than 4')
```

Greater than 4

```
if x >= 5 :
```

```
    print('Greater than or Equals 5')
```

Greater than or Equals 5

```
if x < 6 : print('Less than 6')
```

Less than 6

```
if x <= 5 :
```

```
    print('Less than or Equals 5')
```

Less than or Equals 5

```
if x != 6 :
```

```
    print('Not equal 6')
```

Not equal 6

# 한 갈래 분기

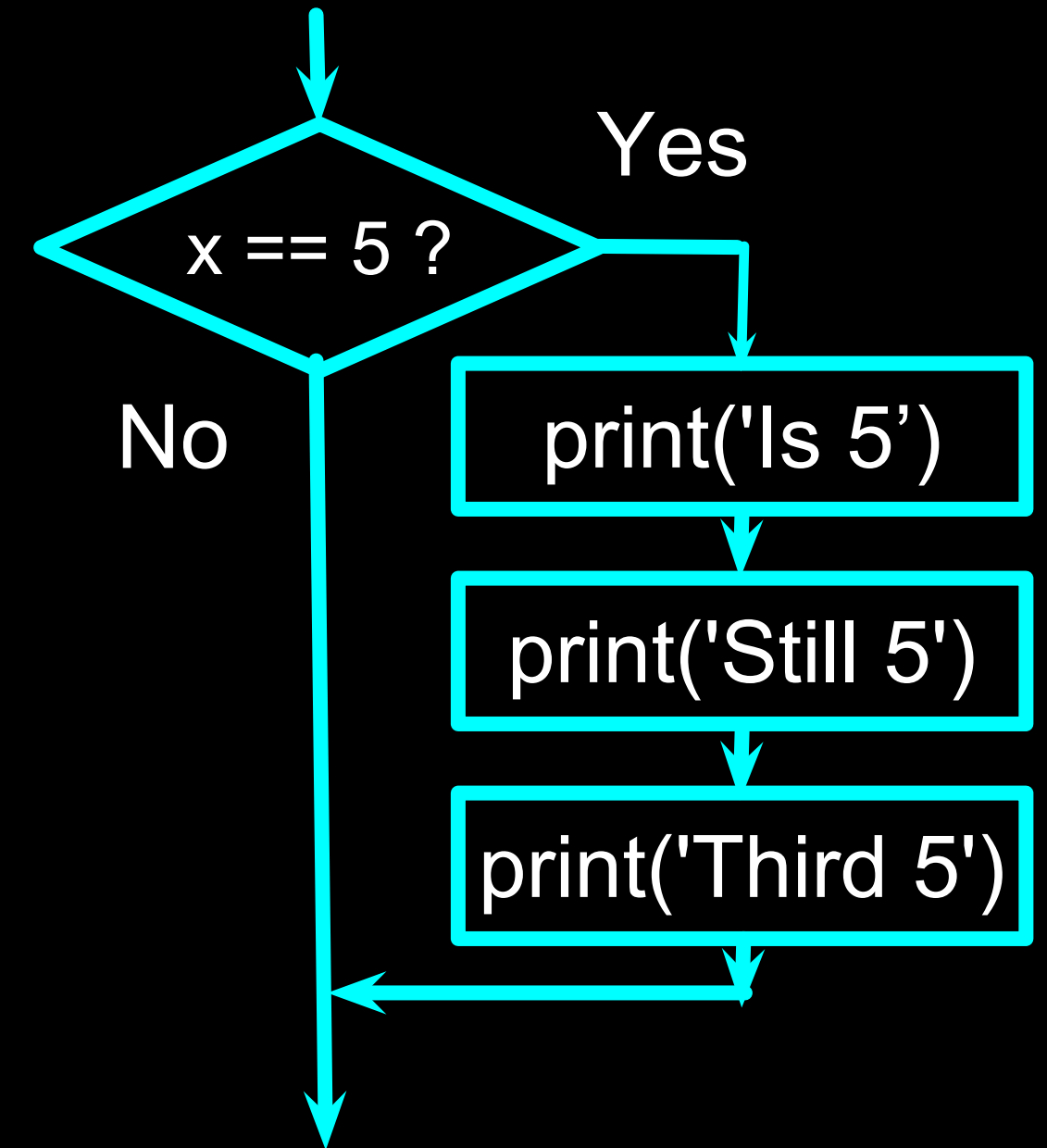
```
x = 5
print('Before 5')
if x == 5 :
    print('Is 5')
    print('Is Still 5')
    print('Third 5')
print('Afterwards 5')
print('Before 6')
if x == 6 :
    print('Is 6')
    print('Is Still 6')
    print('Third 6')
print('Afterwards 6')
```

Before 5

Is 5  
Is Still 5  
Third 5

Afterwards 5  
Before 6

Afterwards 6

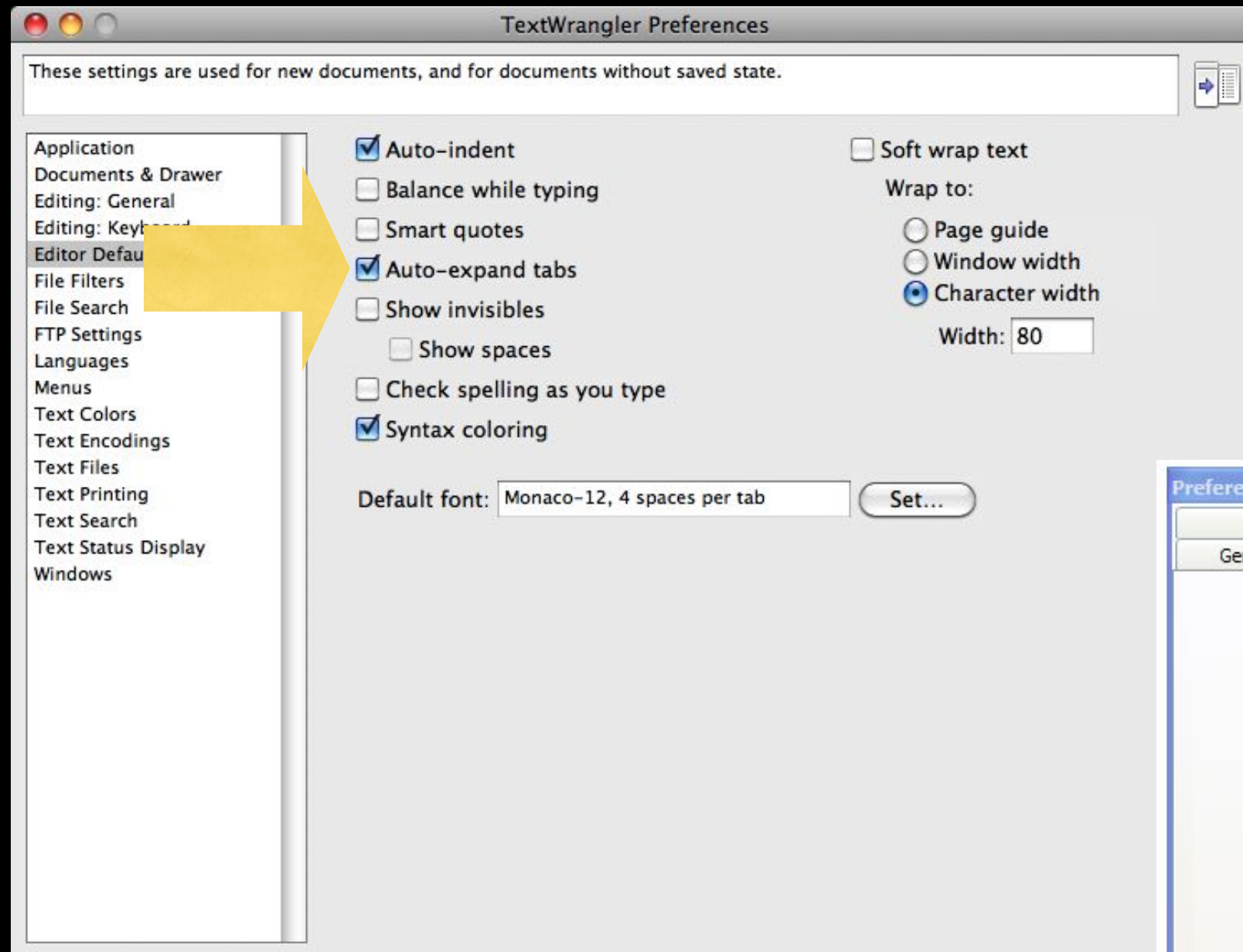


# 들여쓰기

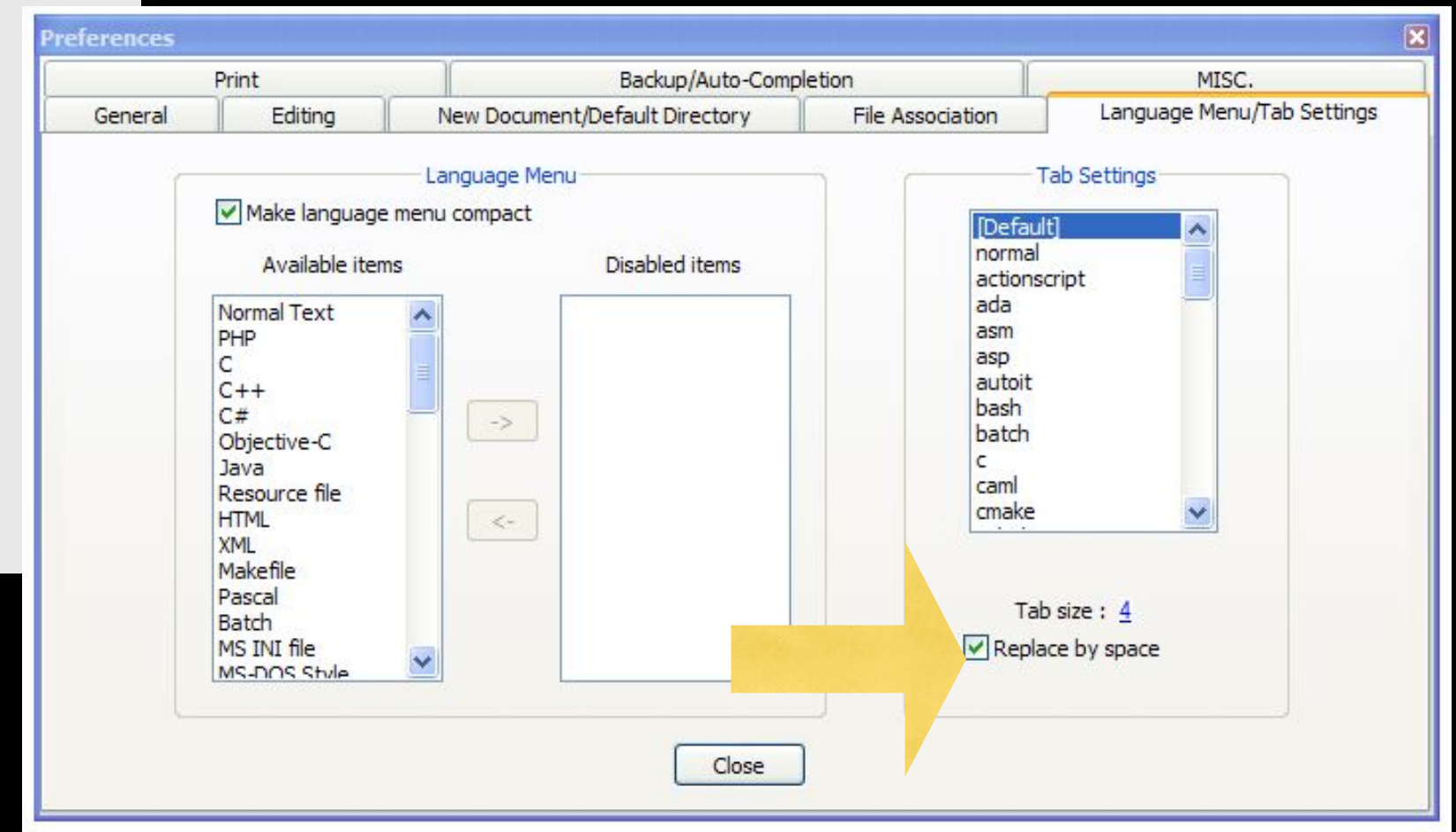
- **if** 문과 **for** 문 ( : ) 다음에 들여쓰기를 함
- 들여쓰기를 유지해서 블록의 **범위**를 표시 (**if**와 **for** 블록의 범위)
- **if** 문과 **for** 문에 맞춰 내어쓰기를 해서 블록의 끝을 표시
- 빈 줄은 **들여쓰기**에 상관없이 무시됨
- 주석은 **들여쓰기**에 상관없이 무시됨

# 주의: 탭을 끄기!

- Atom은 ".py" 파일의 탭을 자동으로 스페이스로 변환
- 대부분 텍스트 에디터는 탭을 스페이스로 변환한다 - 기능을 활성화
  - Notepad++: 설정-> Preferences -> 언어 메뉴/탭 설정
  - TextWrangler: TextWrangler -> Preferences -> Editor Defaults
- 파이썬은 들여쓰기를 중요시하여 탭과 스페이스를 혼동하면 “들여쓰기  
에러”가 발생

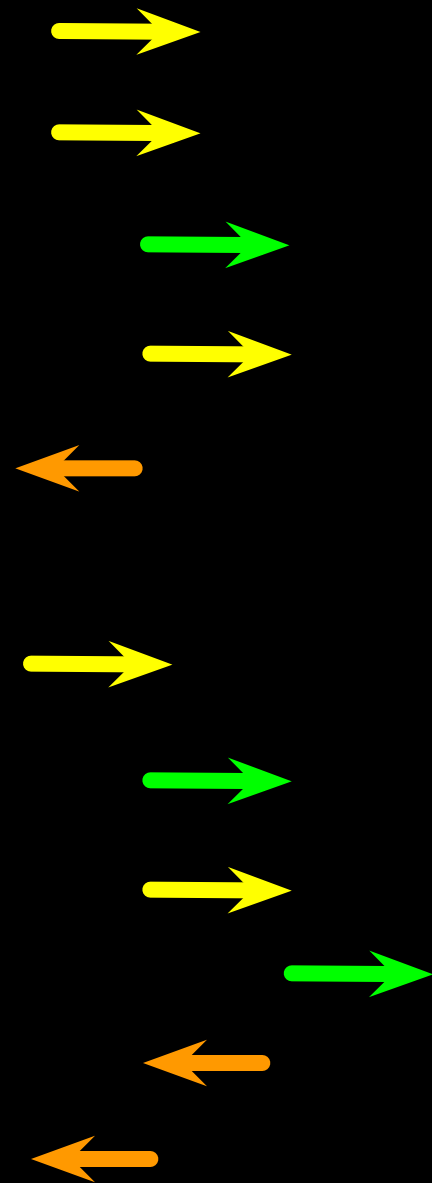


설정을 바꾸면 훨씬 수월





if 와 for문 후 들여쓰기를 하거나 유지  
내어쓰기로 블록의 끝을 표시



```
x = 5
if x > 2 :
    print('Bigger than 2')
    print('Still bigger')
print('Done with 2')

for i in range(5) :
    print(i)
    if i > 2 :
        print('Bigger than 2')
    print('Done with i', i)
print('All Done')
```

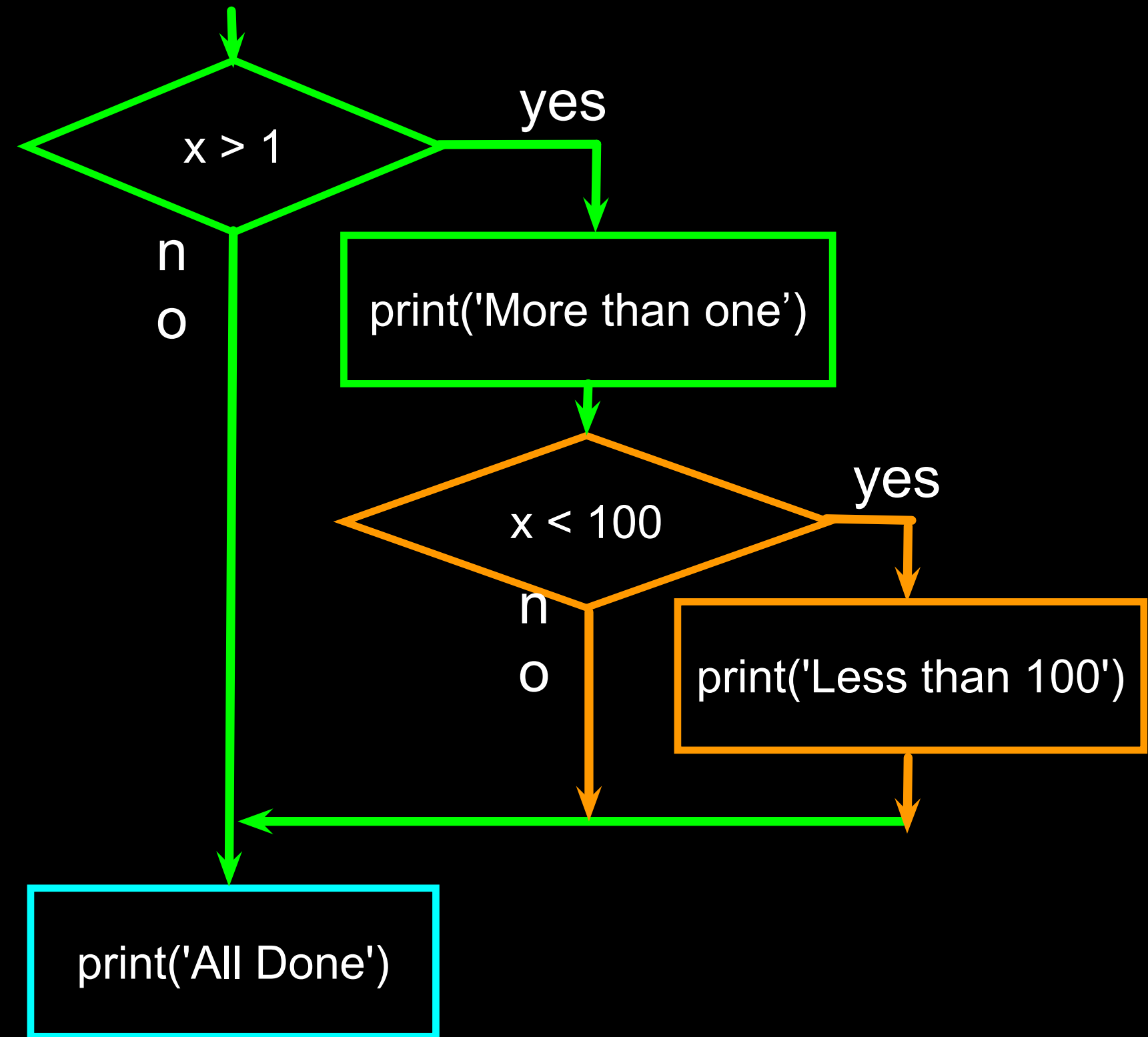
# 블록의 시작과 끝

```
x = 5
if x > 2 :
    print('Bigger than 2')
    print('Still bigger')
print('Done with 2')
```

```
for i in range(5) :
    print(i)
    if i > 2 :
        print('Bigger than 2')
    print('Done with i', i)
print('All Done')
```

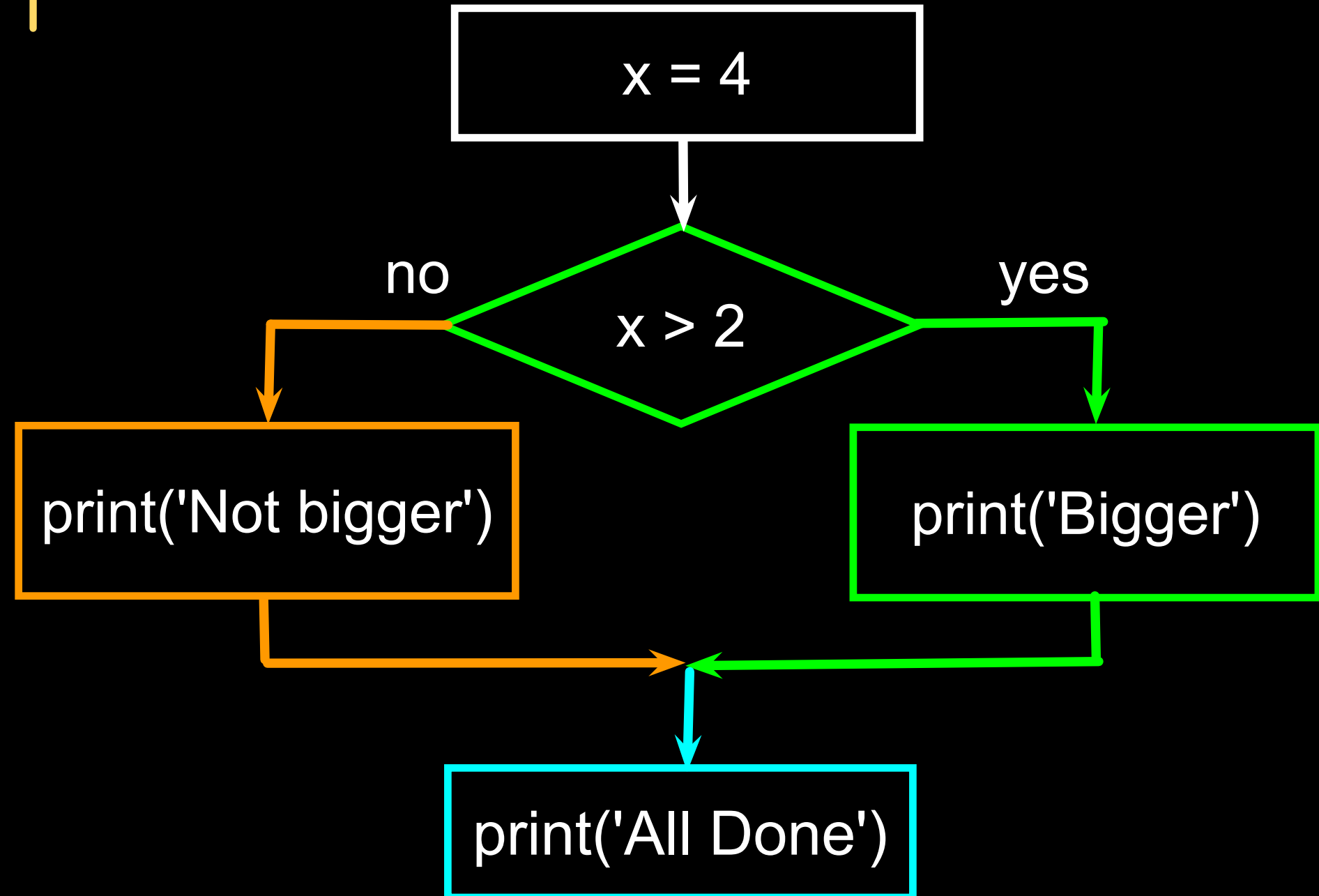
# 중첩된 분기

```
x = 42
if x > 1 :
    print('More than one')
    if x < 100 :
        print('Less than 100')
print('All done')
```



# 두 갈래 분기

- 논리식이 참일 때와 거짓일 때 각각 다른 일을 할 수 있다
- 갈림길과 같이 **둘 중 하나**만 선택할 수 있다

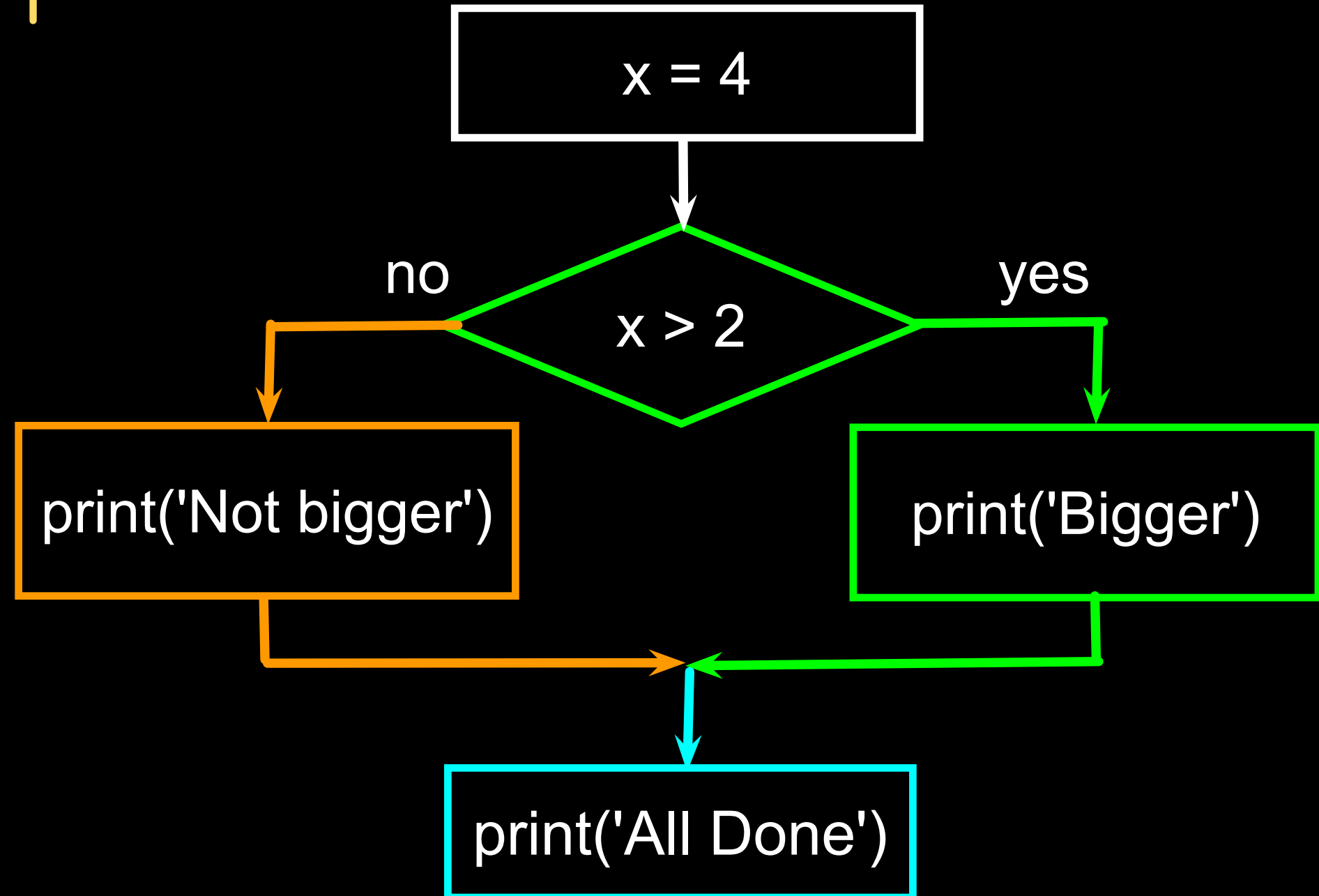


# 두 갈래 분기 else 포함:

```
x = 4
```

```
if x > 2 :  
    print('Bigger')  
else :  
    print('Smaller')
```

```
print('All done')
```

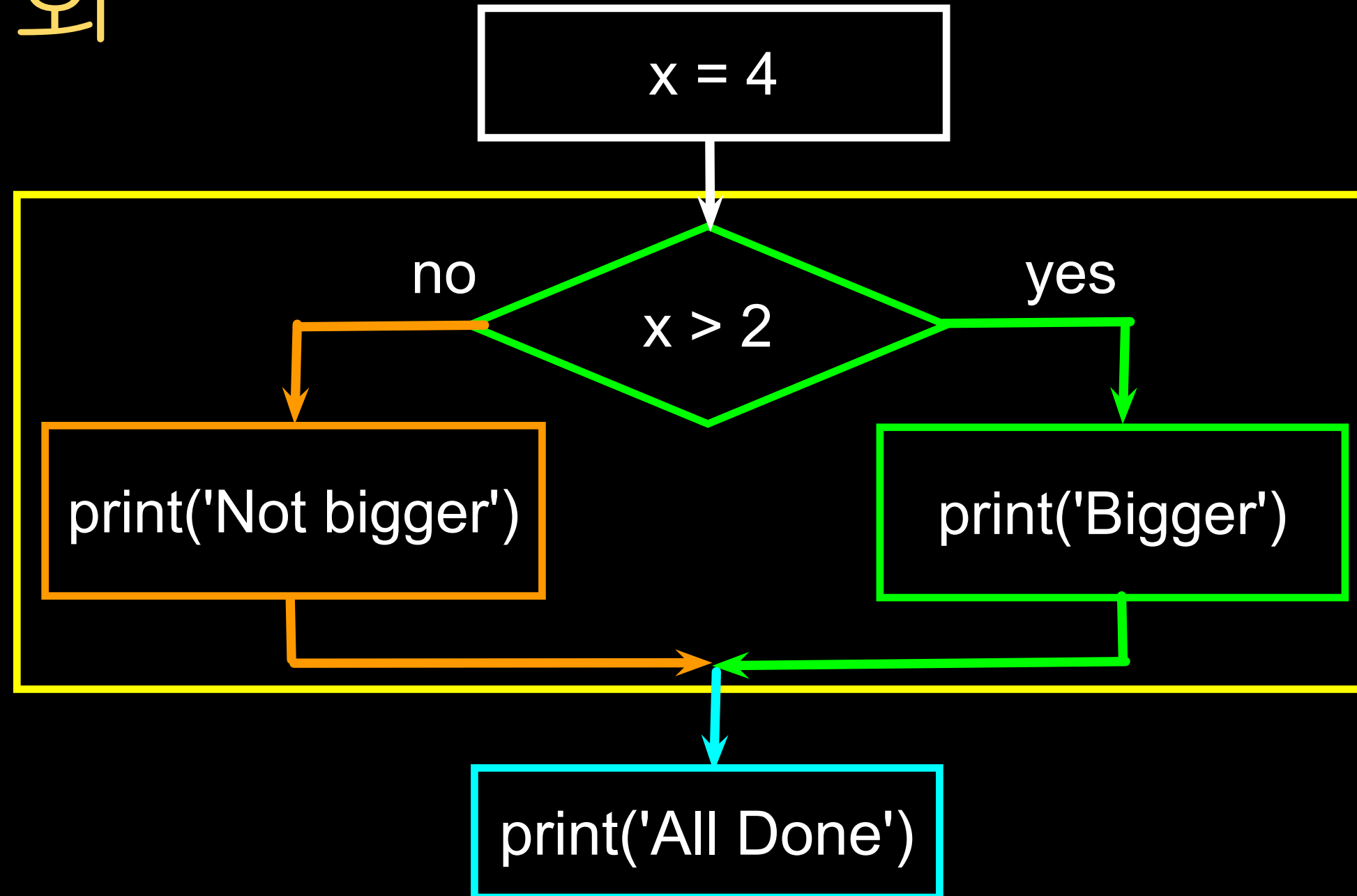


# 블록의 시각화

x = 4

```
if x > 2 :  
    print('Bigger')  
else :  
    print('Smaller')
```

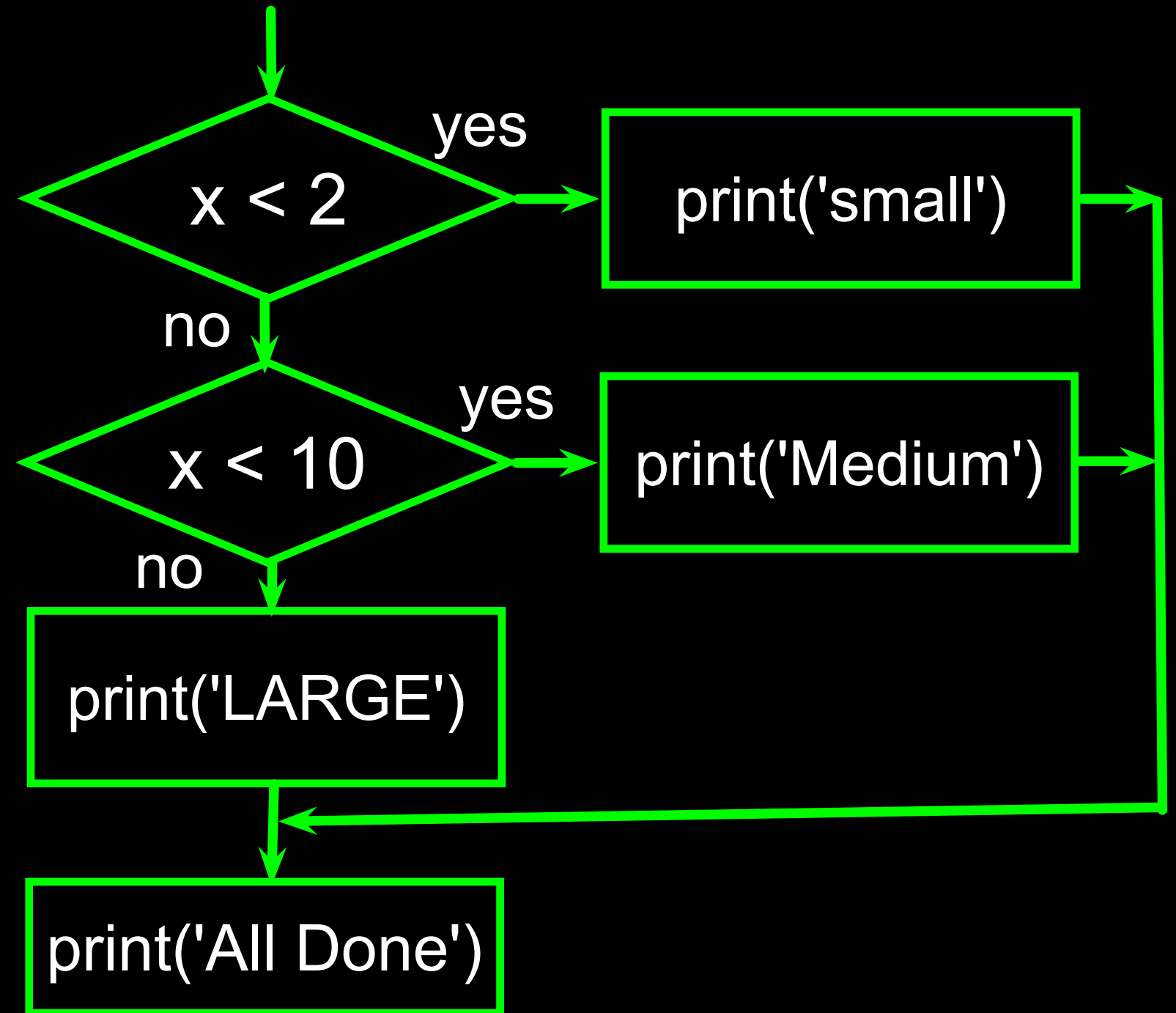
```
print('All done')
```



다른 조건문 구조

# 여러 갈래

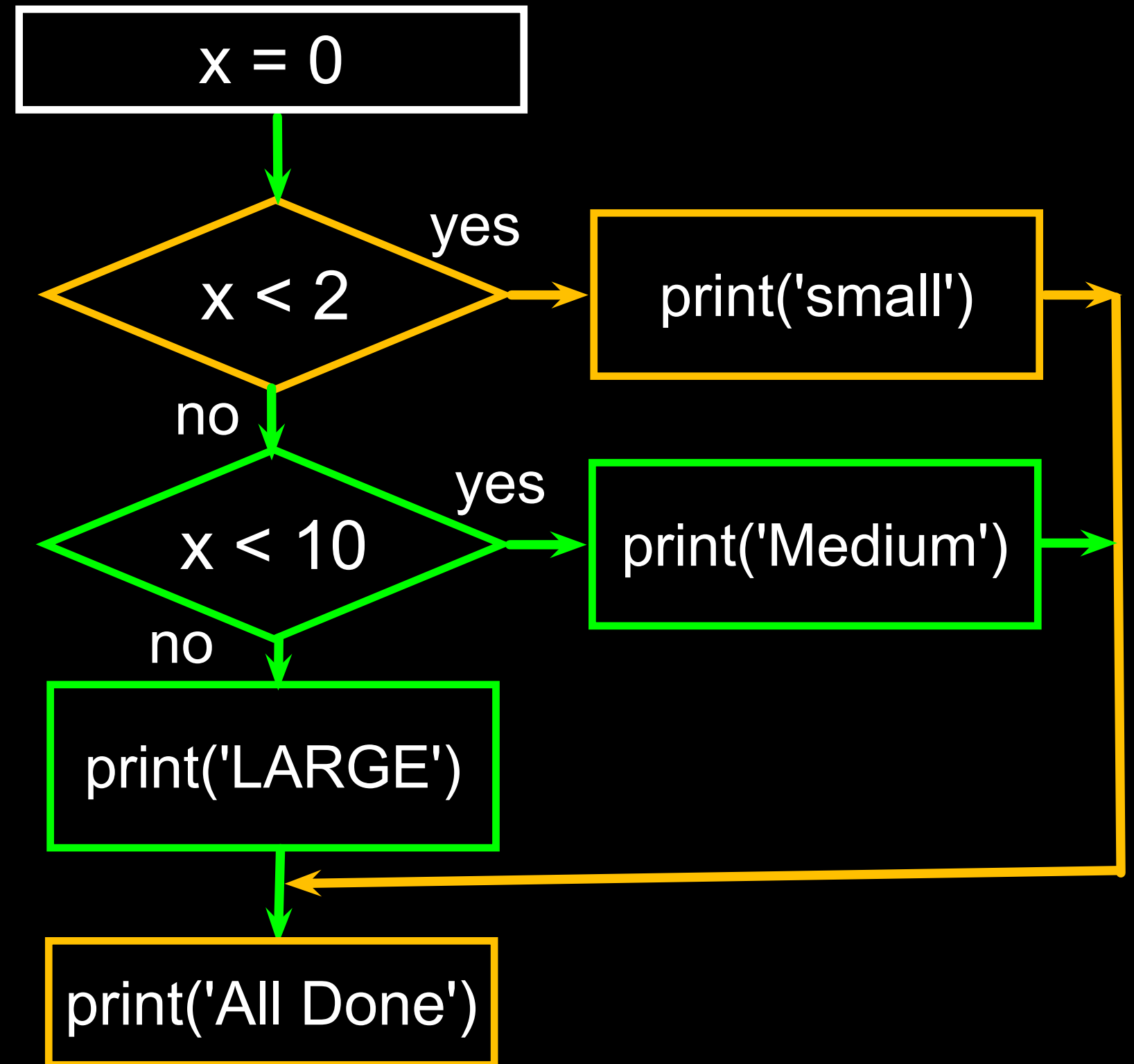
```
if x < 2 :  
    print('small')  
elif x < 10 :  
    print('Medium')  
else :  
    print('LARGE')  
print('All done')
```





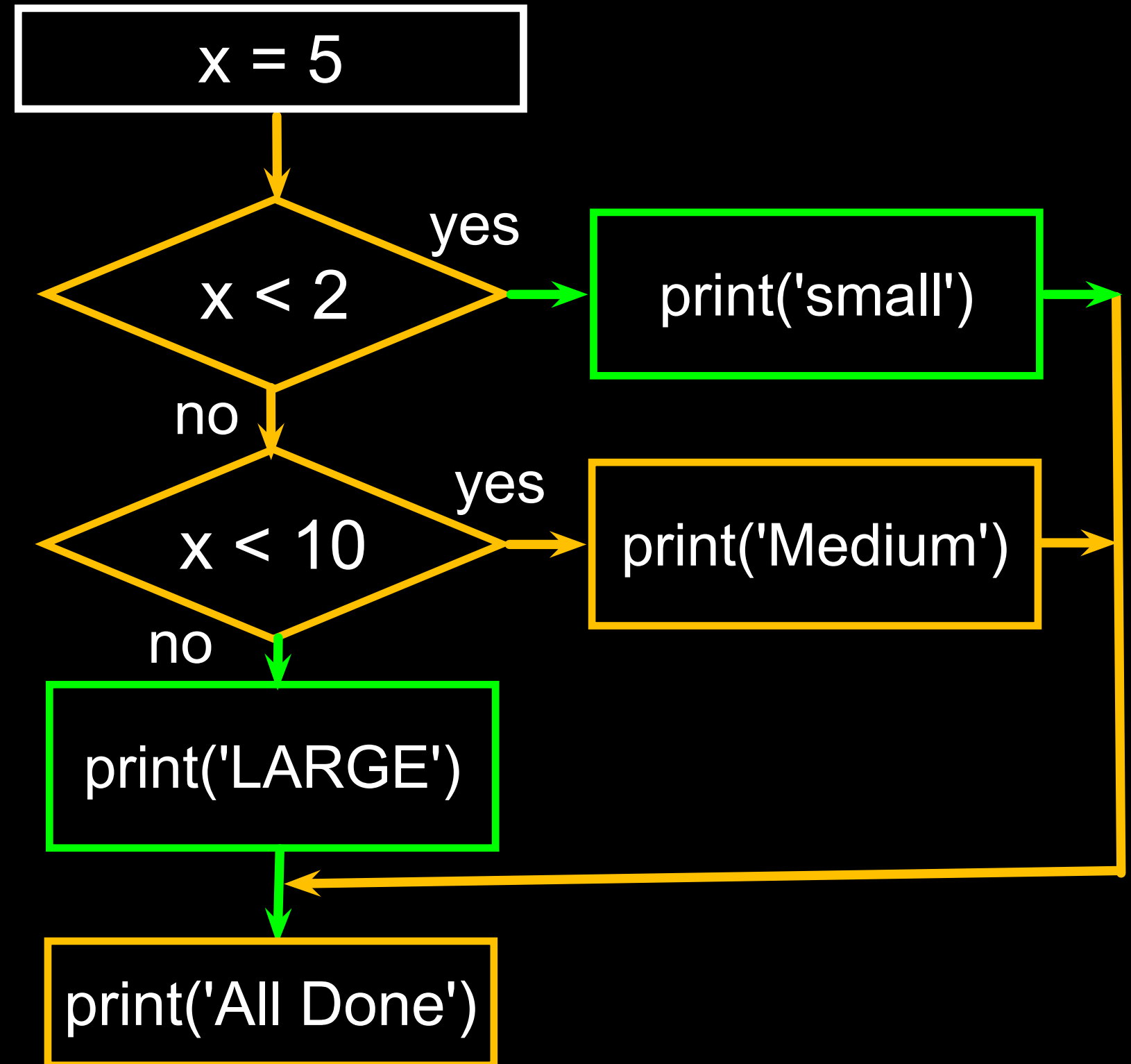
# 여러 갈래

```
x = 0
if x < 2 :
    print('small')
elif x < 10 :
    print('Medium')
else :
    print('LARGE')
print('All done')
```



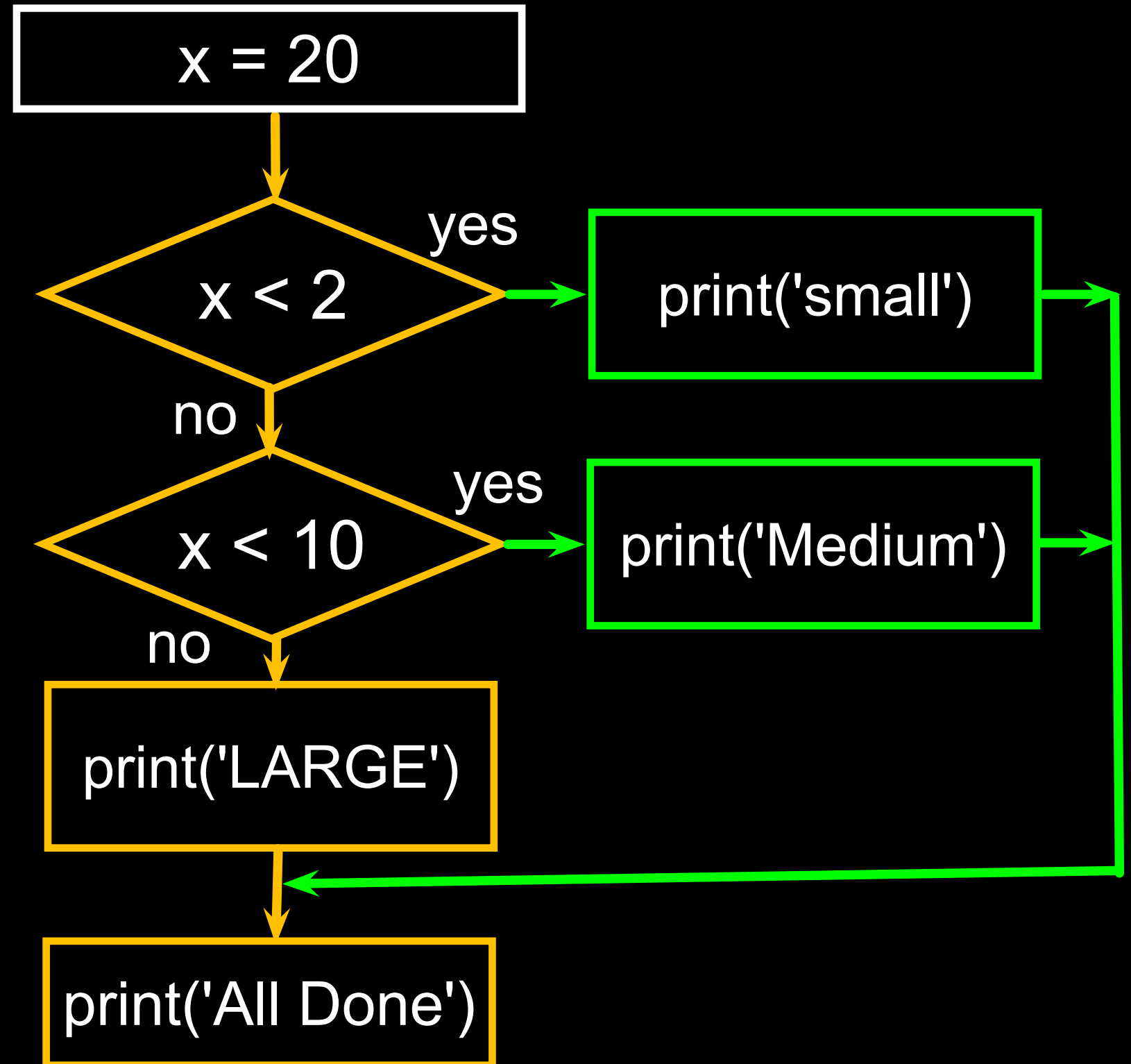
# 여러 갈래

```
x = 5
if x < 2 :
    print('small')
elif x < 10 :
    print('Medium')
else :
    print('LARGE')
print('All done')
```



# 여러 갈래

```
x = 20
if x < 2 :
    print('small')
elif x < 10 :
    print('Medium')
else :
    print('LARGE')
print('All done')
```



# 여러 갈래

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

print('All done')
```

```
if x < 2 :
    print('Small')
elif x < 10 :
    print('Medium')
elif x < 20 :
    print('Big')
elif x < 40 :
    print('Large')
elif x < 100:
    print('Huge')
else :
    print('Ginormous')
```

# 여러 갈래 문제

x 값과 상관없이 무엇이 출력?

```
if x < 2 :  
    print('Below 2')  
elif x >= 2 :  
    print('Two or more')  
else :  
    print('Something else')
```

```
if x < 2 :  
    print('Below 2')  
elif x < 20 :  
    print('Below 20')  
elif x < 10 :  
    print('Below 10')  
else :  
    print('Something else')
```

# try / except 구조

- 위험한 코드를 **try/except** 을 사용해 처리
- **try** 블록에 있는 코드가 성공하면 - **except** 블록을 건너뛴다
- **try** 블록에 있는 코드가 실패하면 - **except** 블록을 실행

```
$ cat notry.py
astr = 'Hello 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: 'Hello Bob'
```



다 끝남

여기서  
멈춤

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



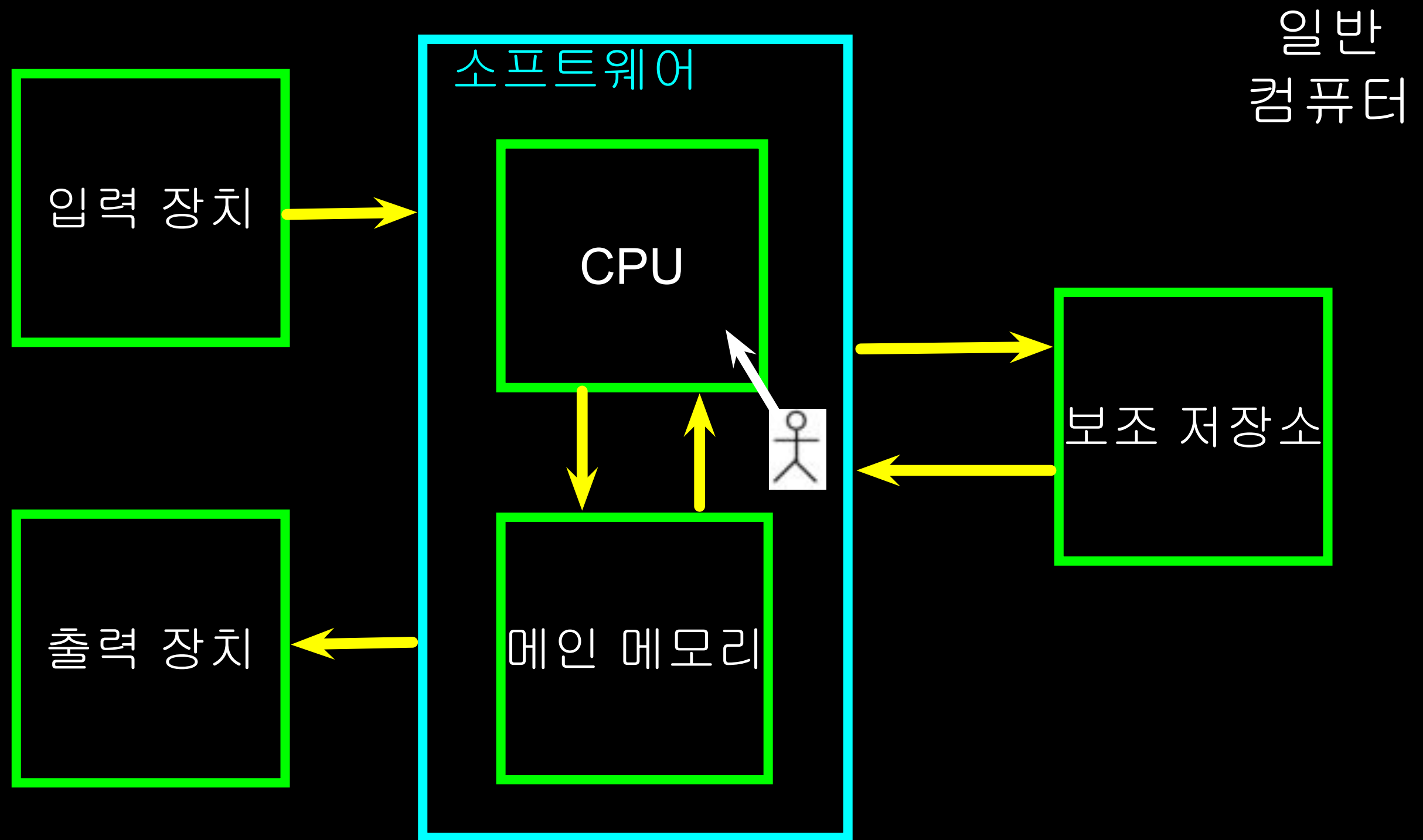
```
$ python3 notry.py
```

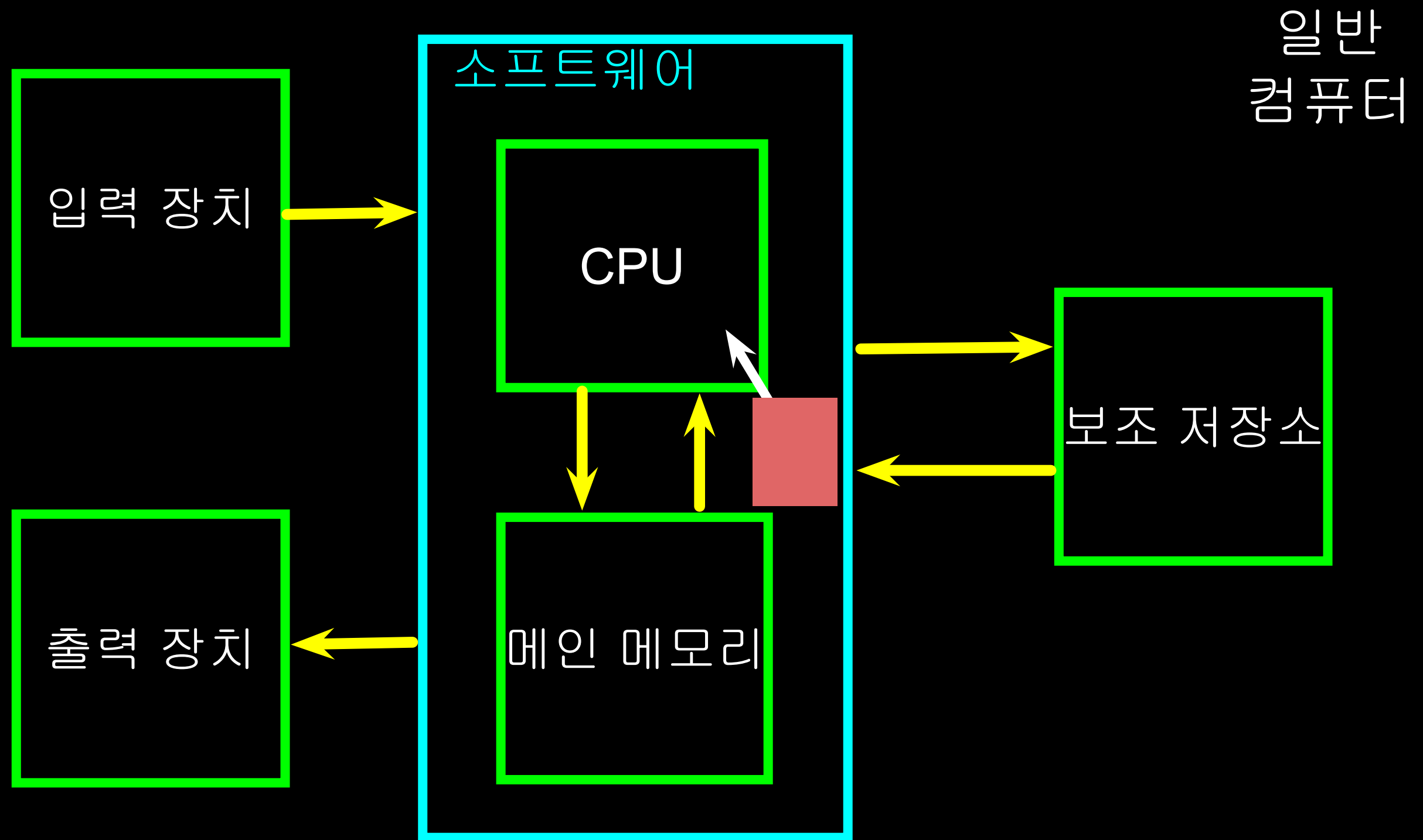
```
Traceback (most recent call last):  
File "notry.py", line 2, in <module>  
istr = int(astr)ValueError: invalid literal  
for int() with base 10: 'Hello Bob'
```

다 끝남









```
astr = 'Hello Bob'
try:
    istr = int(astr)
except:
    istr = -1
print('First', istr)
```

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

첫 번째 변환이 실패하면  
**except** 블록을 실행

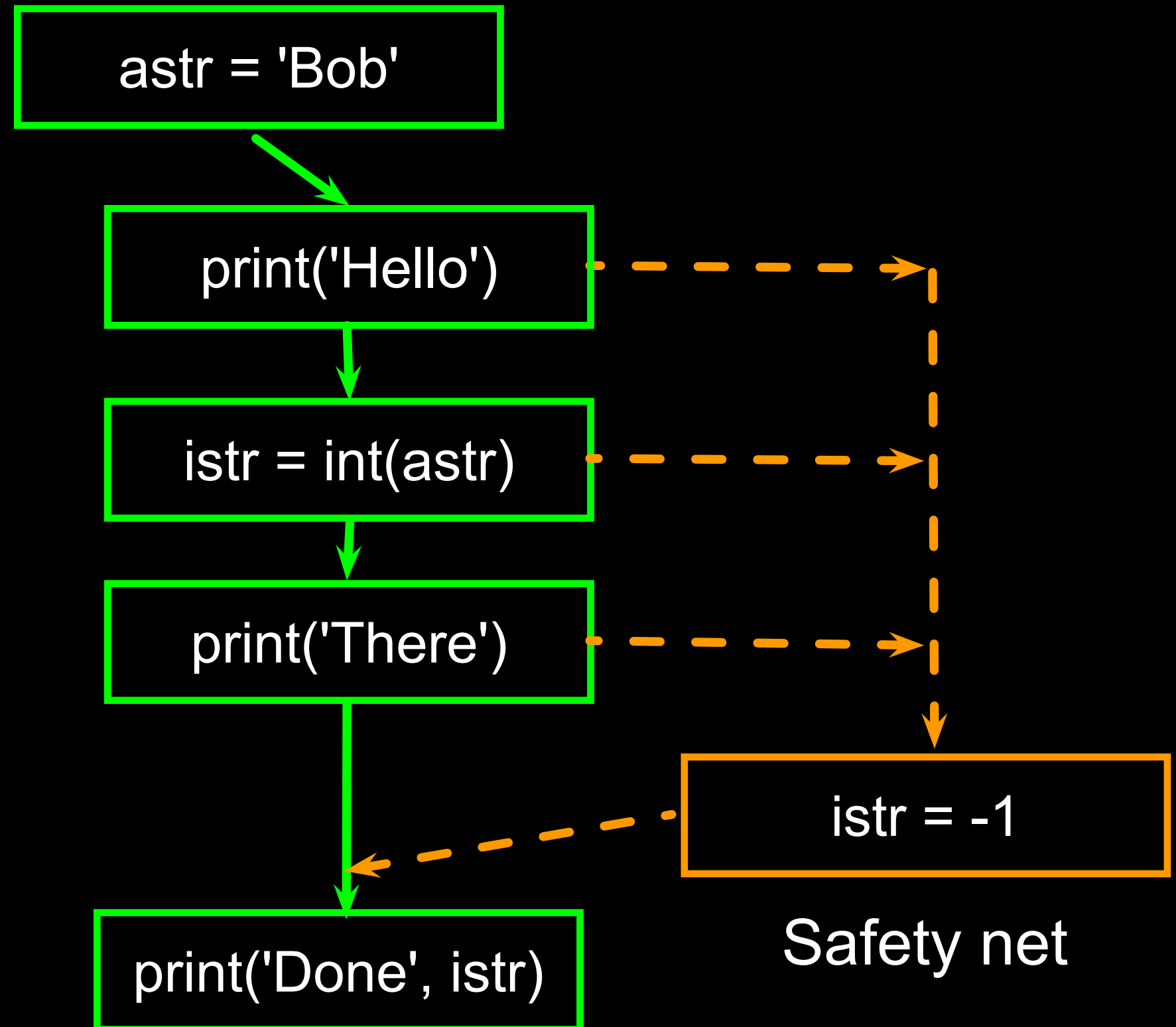
```
$ python tryexcept.py
First -1
Second 123
```

두 번째 변환이 성공하면  
**except** 블록을 건너뛴다

# try / except

```
astr = 'Bob'
try:
    print('Hello')
    istr = int(astr)
    print('There')
except:
    istr = -1

print('Done', istr)
```



# try / except 예시

```
rawstr = input('Enter a number:')
try:
    ival = int(rawstr)
except:
    ival = -1

if ival > 0 :
    print('Nice work')
else:
    print('Not a number')
```

```
$ python3 trynum.py
Enter a number:42
Nice work
$ python3 trynum.py
Enter a number:forty-two
Not a number
$
```

# 요약

- 비교 연산자  
`==` `<=` `>=` `>` `<` `!=`
- 들여쓰기
- 한 갈래 결정
- 두 갈래 결정:  
`if:` 와 `else:`
- 중첩된 결정
- `elif`을 이용한 여러 갈래 결정
- `try` / `except`을 이용한 에러 처리

## 연습 문제

40시간을 초과한 시간의 시급을 1.5배하여 급여  
계산 프로그램을 만들기

Enter Hours: 45

Enter Rate: 10

Pay: 475.0

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

## 연습 문제

`try/except`를 사용해 숫자가 아닌 입력값을 받는  
급여 계산 프로그램으로 수정

Enter Hours: 20

Enter Rate: nine

Error, please enter numeric input

Enter Hours: forty

Error, please enter numeric input





# Acknowledgements / Contributions



These slides are Copyright 2010- Charles R. Severance ([www.dr-chuck.com](http://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.

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

Contributor:

- Seung-June Lee ([plusjune@gmail.com](mailto:plusjune@gmail.com))
- Connect Foundation

Translator:

- Hakyong Kim
- Jeungmin Oh ([tangza@gmail.com](mailto:tangza@gmail.com))