

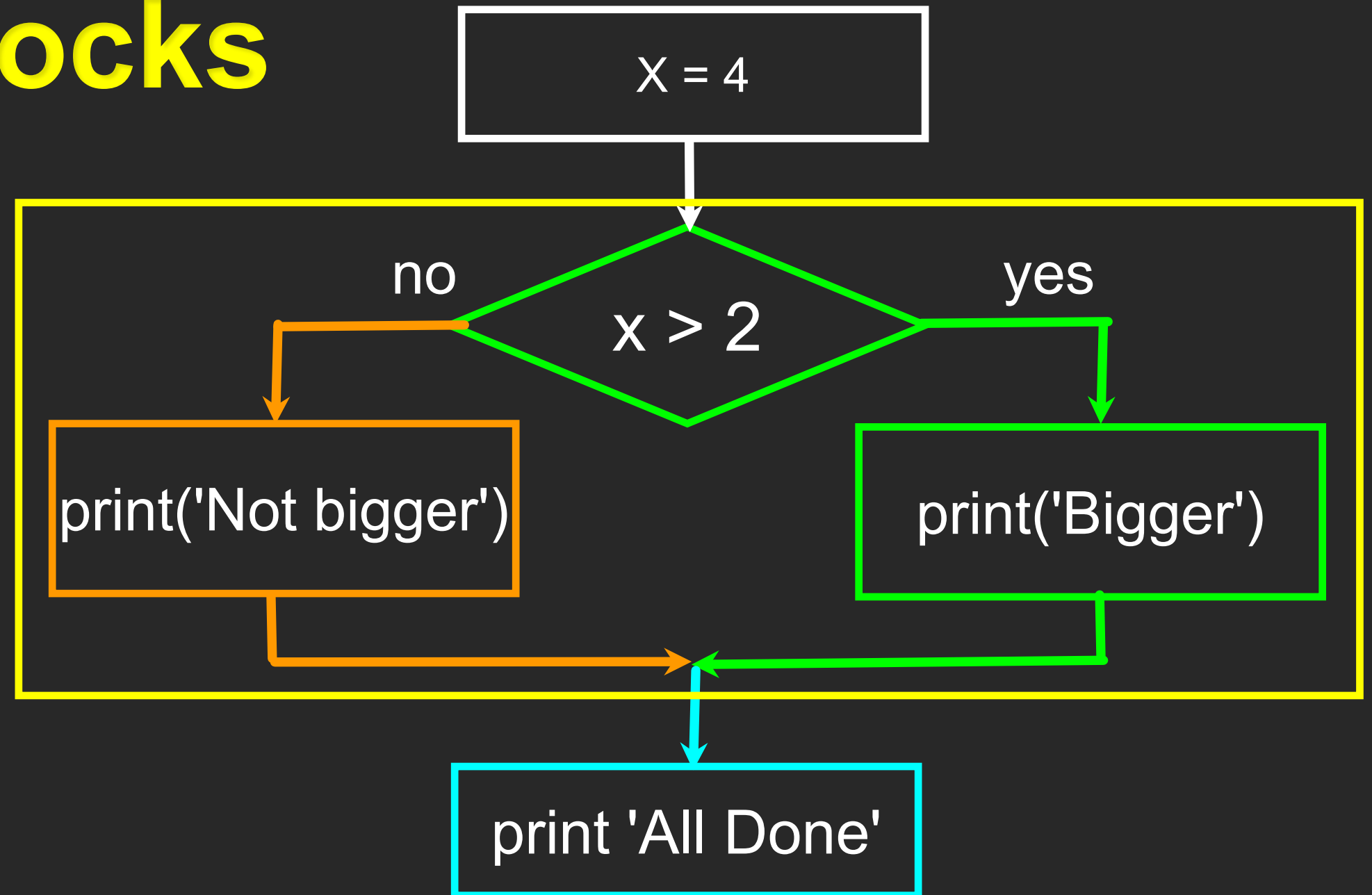
# More Conditional Execution Patterns

# Visualize Blocks

`x = 4`

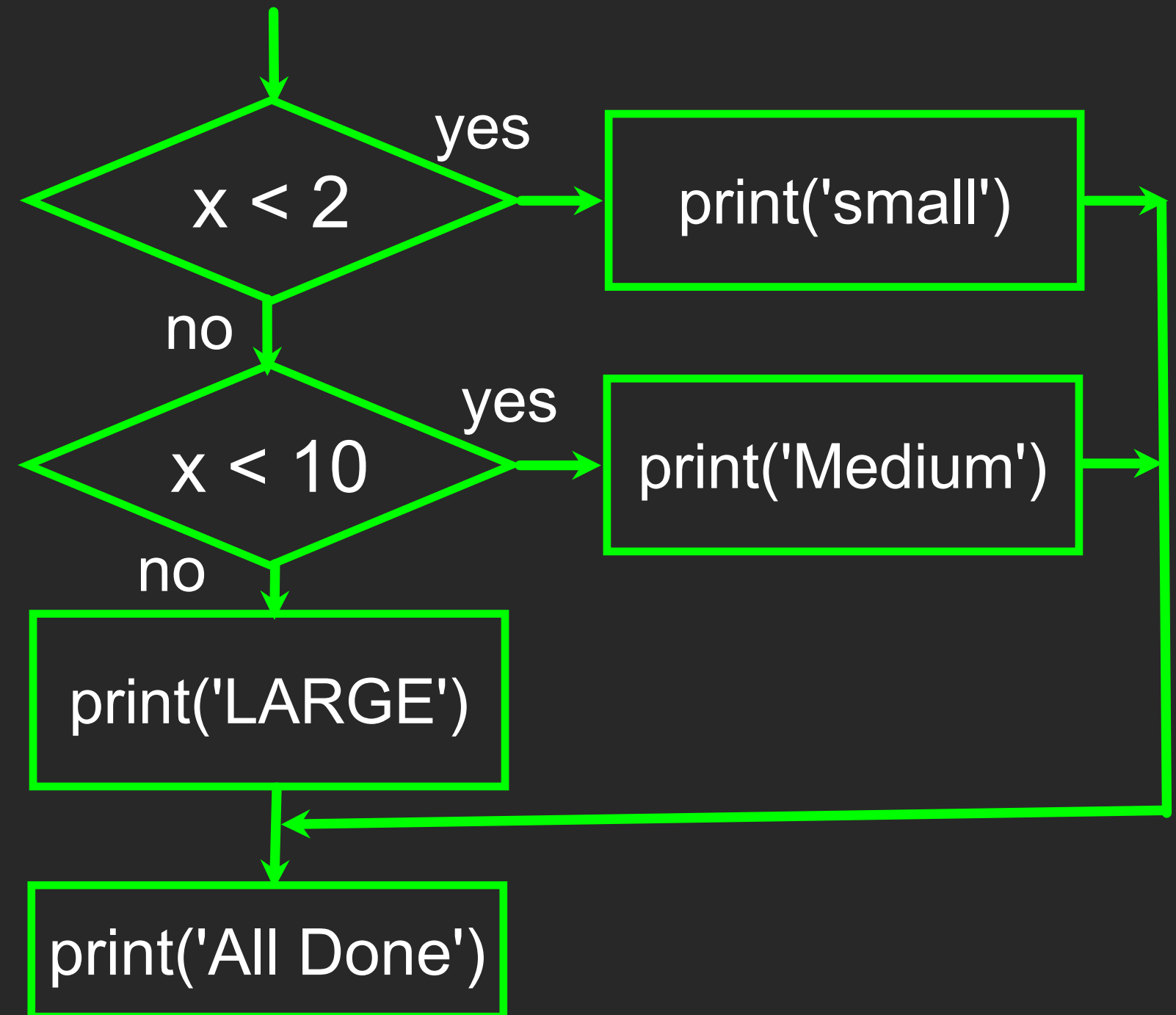
```
if x > 2 :  
    print('Bigger')  
else :  
    print('Smaller')
```

`print 'All done'`



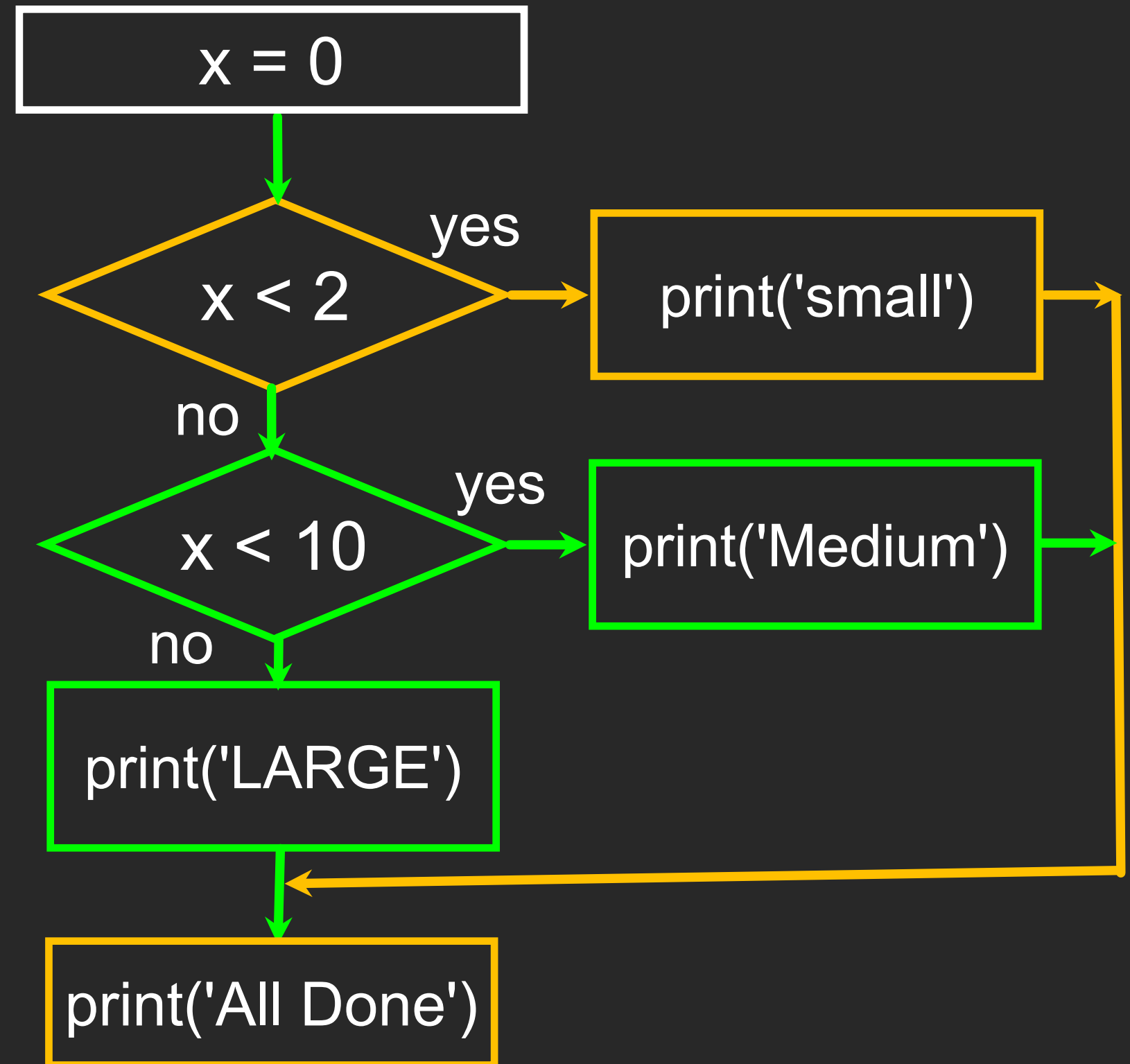
# Multi-way

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



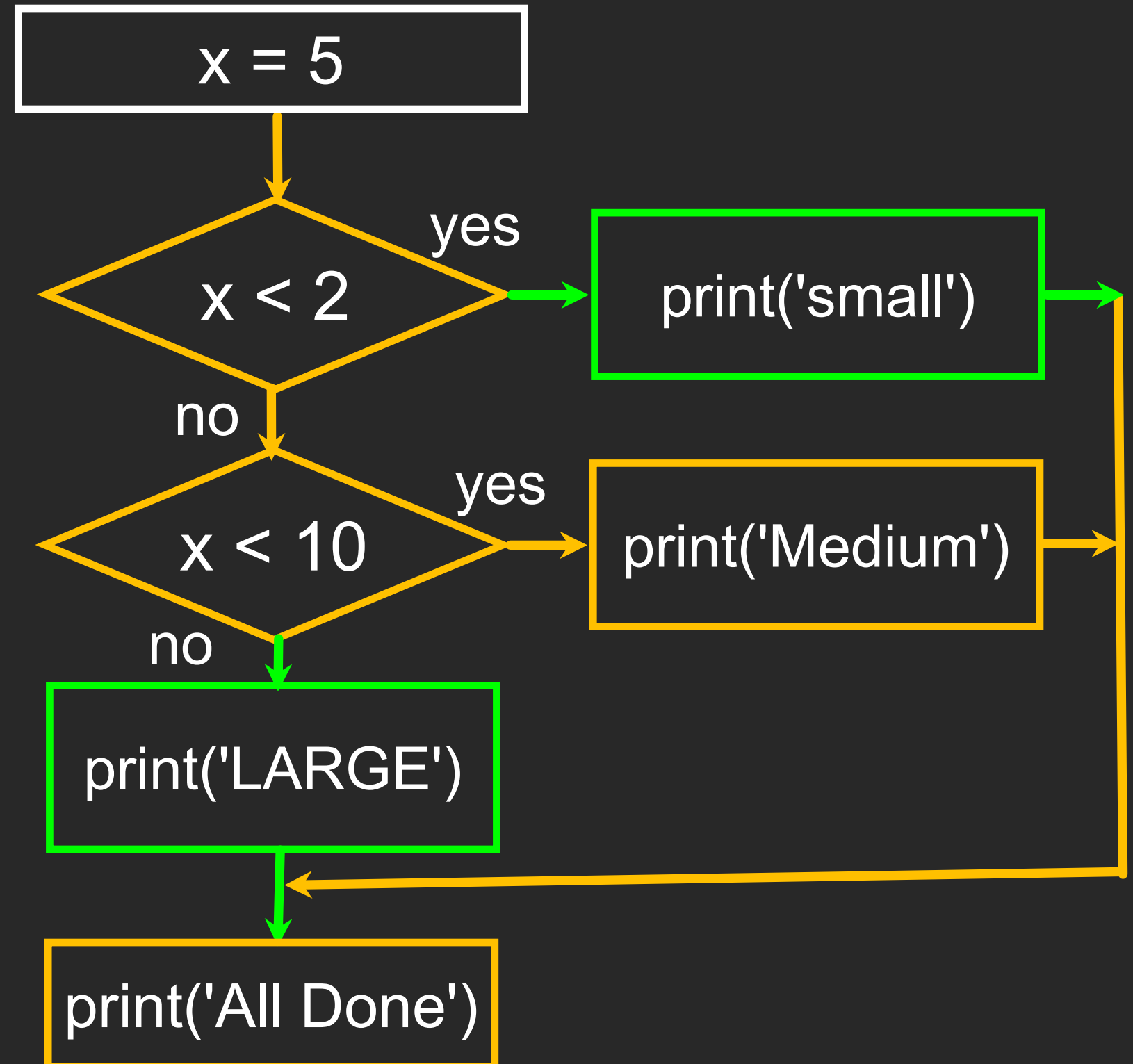
# Multi-way

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



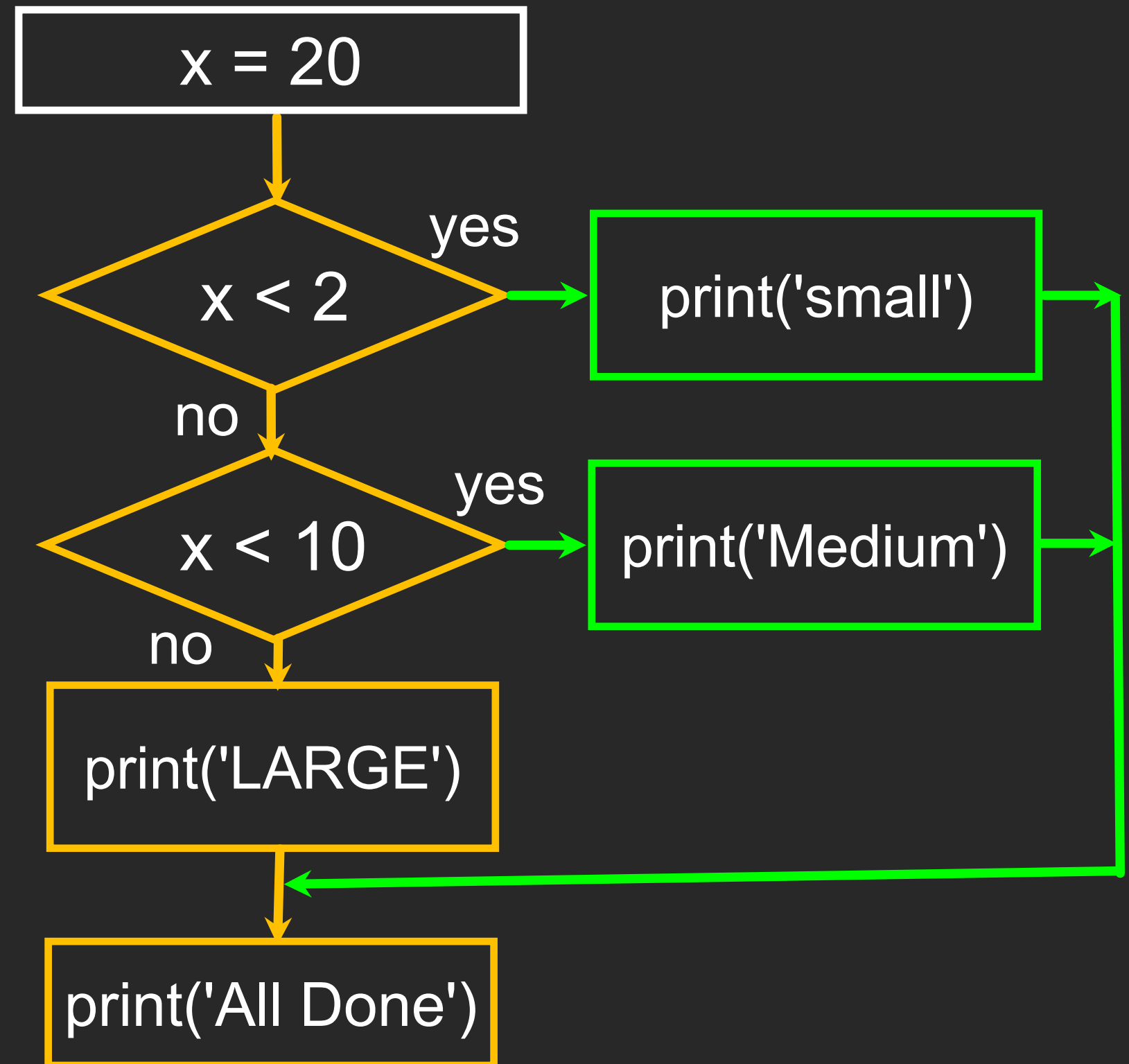
# Multi-way

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



# Multi-way

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



# Multi-way

```
# 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')
```

# Multi-way Puzzles

Which will never print  
regardless of the value for x?

```
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')
```



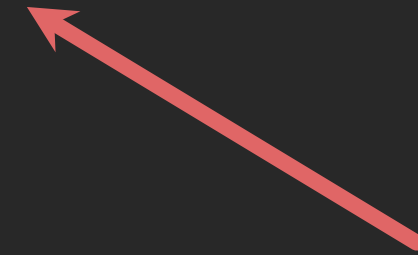
# The try / except Structure

- You surround a dangerous section of code with `try` and `except`
- If the code in the `try` works - the `except` is skipped
- If the code in the `try` fails - it jumps to the `except` section

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

```
$ python3 notry.py
```

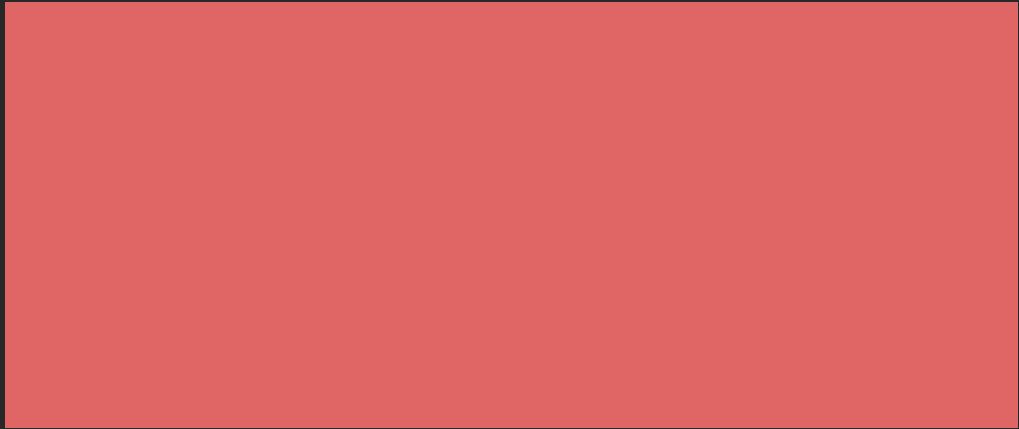
```
Traceback (most recent call last):
File "notry.py", line 2, in <module>
istr = int(astr)ValueError: invalid literal
for int() with base 10: 'Hello Bob'
```



All  
Done

The  
program  
stops  
here

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

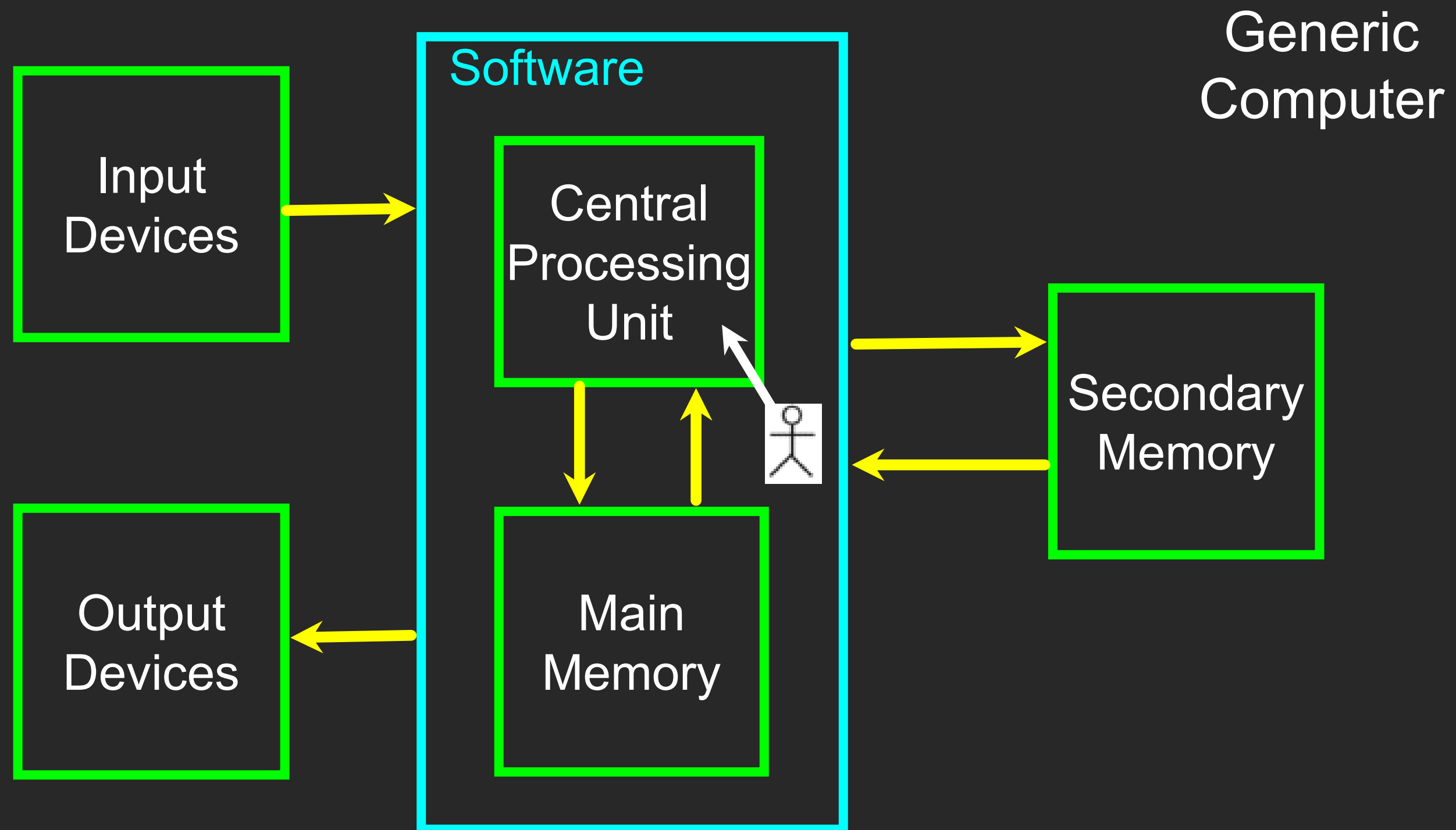


```
$ python3 notry.py
```

```
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'
```

All  
Done





```
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)
```

When the first conversion fails - it just drops into the except: clause and the program continues.

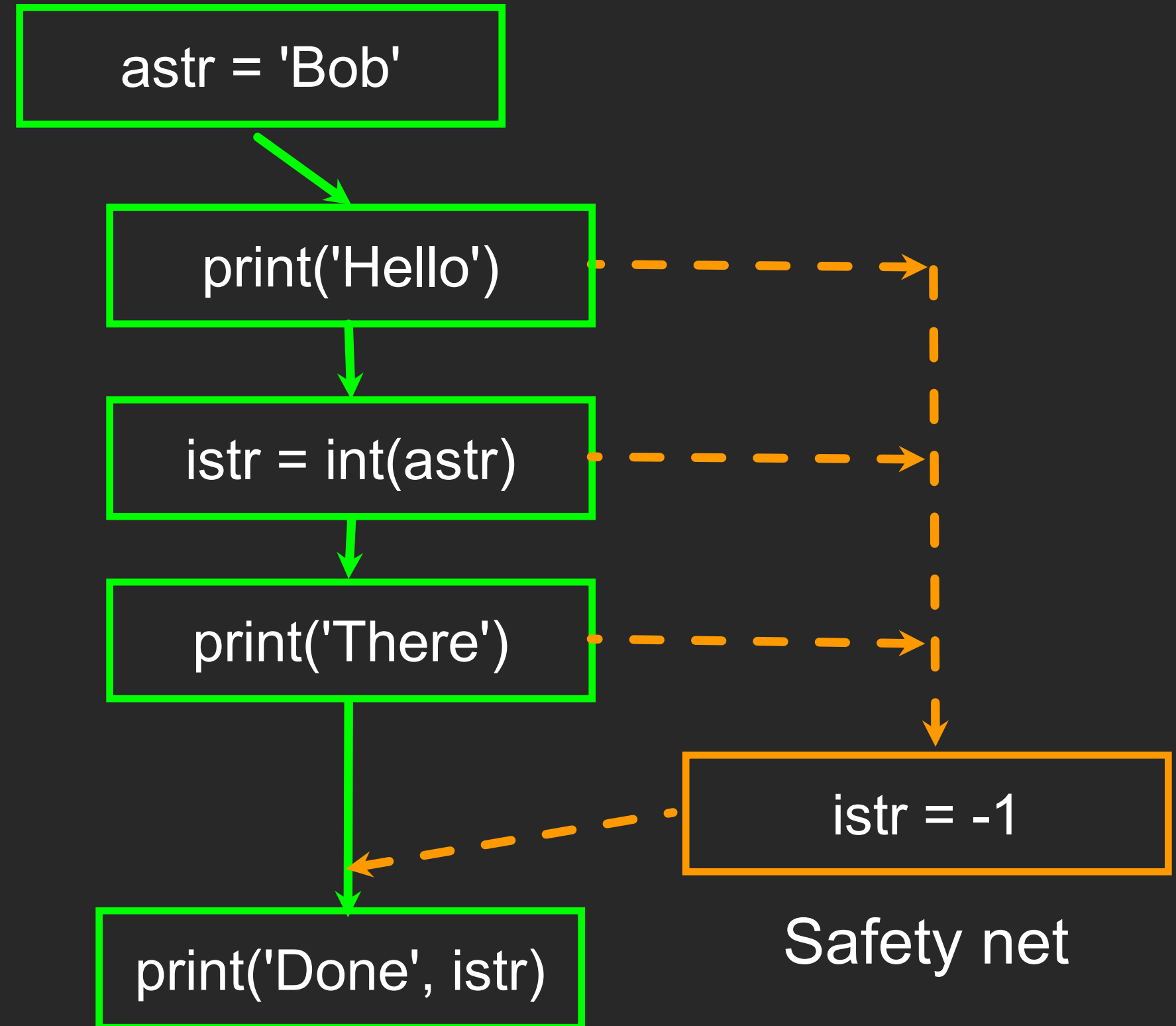
```
$ python tryexcept.py
First -1
Second 123
```

When the second conversion succeeds - it just skips the except: clause and the program continues.

# try / except

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

print('Done', istr)
```



# Sample try / except

```
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
$
```

## Exercise

Rewrite your pay computation to give the employee 1.5 times the hourly rate for hours worked above 40 hours.

```
Enter Hours: 45
```

```
Enter Rate: 10
```

```
Pay: 475.0
```

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



## Exercise

Rewrite your pay program using try and except so that your program handles non-numeric input gracefully.

```
Enter Hours: 20
```

```
Enter Rate: nine
```

```
Error, please enter numeric input
```

```
Enter Hours: forty
```

```
Error, please enter numeric input
```

# Summary

- Comparison operators  
`==` `<=` `>=` `>` `<` `!=`
- Indentation
- One-way Decisions
- Two-way decisions:  
`if:` and `else:`
- Nested Decisions
- Multi-way decisions using `elif`
- `try` / `except` to compensate for errors



## 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

... Insert new Contributors and Translators here

...