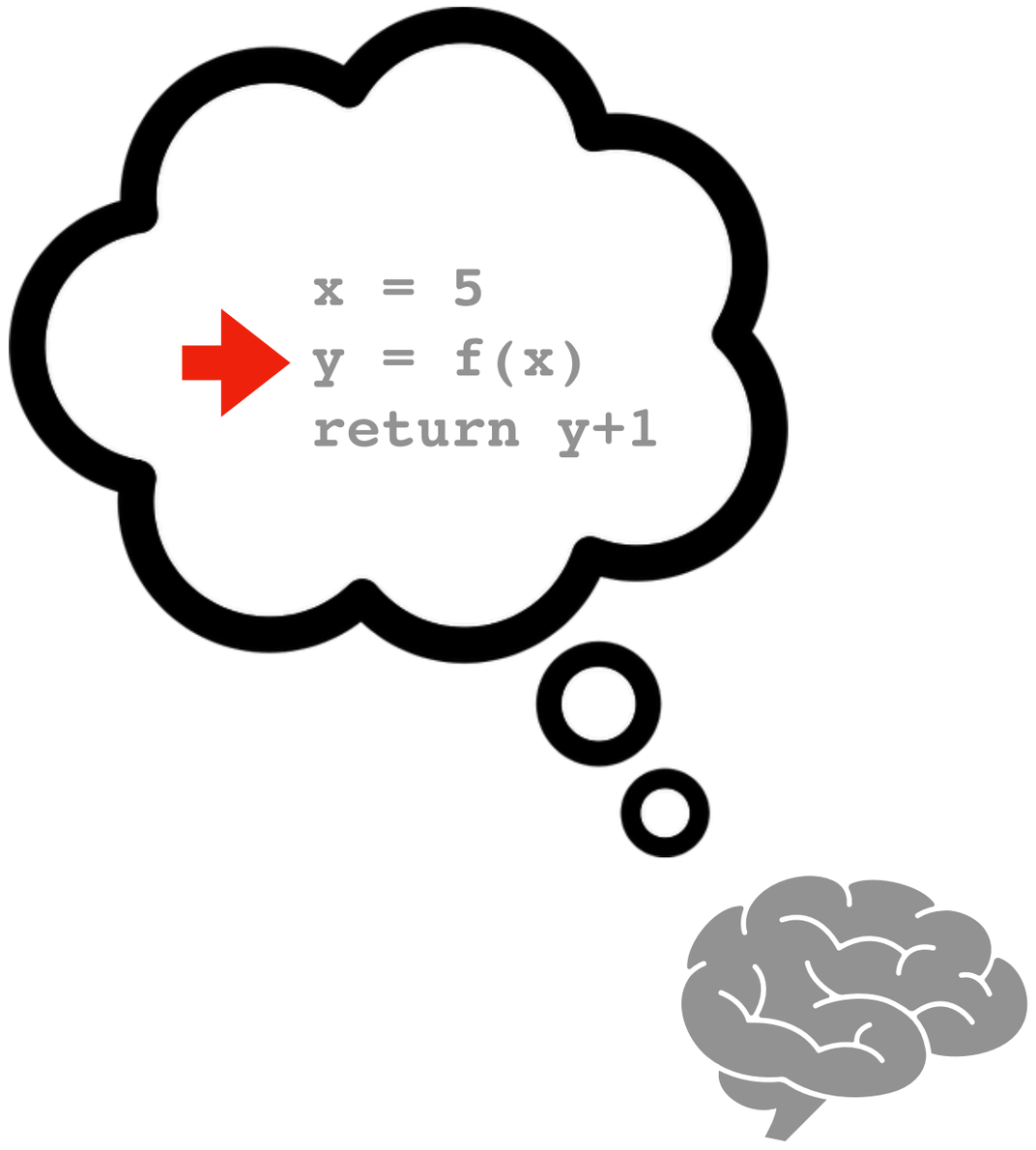
[301] Conditionals

Tyler Caraza-Harter

Mental Model of Control Flow

Code:

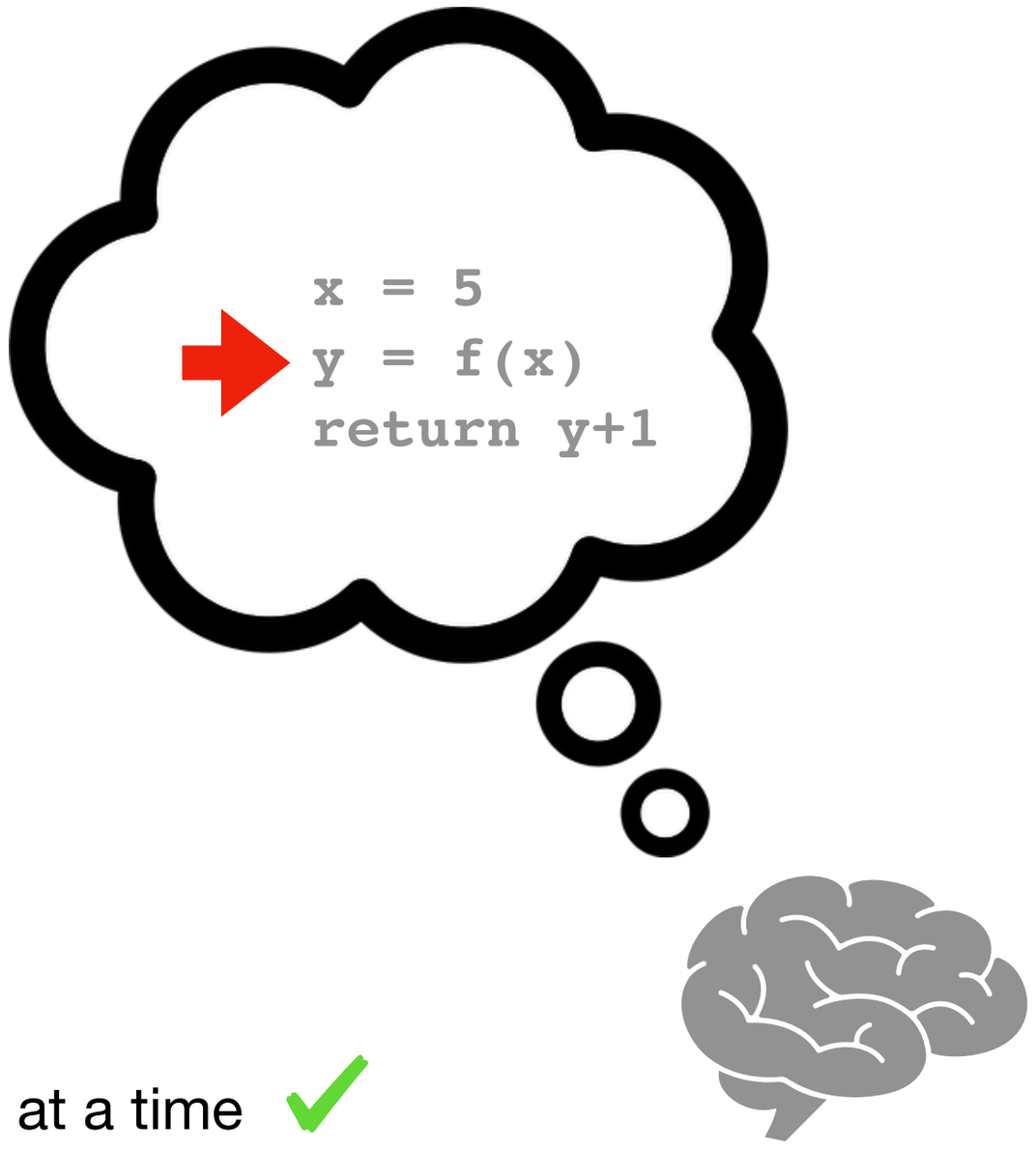
```
...
x = 5
y = f(x)
return y+1
...
```



Mental Model of Control Flow

Code:

```
...
x = 5
y = f(x)
return y+1
...
```



three exceptions

1. do statements in order, one at a time ✓
2. **functions**: jump in and out of these ✓
3. **conditionals**: sometimes skip statements ← **TODAY**
4. **loops**: sometimes go back to previous

Learning Objectives Today

Reason about conditionals

- Conditional execution
- Alternate execution
- Chained conditionals
- Nested conditionals

Chapter 5 of Think Python
(skip "Recursion" sections)

Do PythonTutor Practice!
(posted on schedule)

Understand code blocks

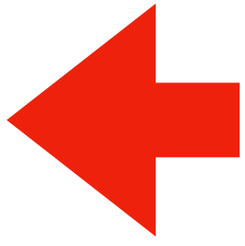
- Be able to identify the lines of code in the same block

Sanity checking

- Recognize errors
- Sanitize bad data automatically

Today's Outline

Review



Control Flow Diagrams

Basic syntax for “if”

Identifying code blocks

Demos

Review 1: Indentation Example

what does it print?

```
print(“A”)
print(“B”)

def print_letters():
    print(“C”)
    print(“D”)

print(“E”)
print(“F”)

print_letters()
```

Review 1: Indentation Example

what does it print?

```
print(“A”)
print(“B”)

def print_letters():
    print(“C”)
    print(“D”)

print(“E”)
print(“F”)

print_letters()
```

A
B
E
F
C
D

Review 1: Indentation Example

what does it print?

```
print(“A”)
print(“B”)

def print_letters():
    print(“C”)
    print(“D”)

print(“E”)
print(“F”)

print_letters()
```

indented, so “inside” print_letters function

A
B
E
F
C
D

Review 1: Indentation Example

what does it print?

```
print("A")
print("B")

def print_letters():
    print("C")
    print("D")

print("E")
print("F")

print_letters()
```

indented, so "inside" print_letters function

printed last because print_letters is called last

A
B
E
F
C
D

Review 1: Indentation Example

what does it print?

```
print(“A”)
print(“B”)

def print_letters():
    print(“C”)
    print(“D”)

print(“E”)
print(“F”)

print_letters()
```

indented, so “inside” print_letters function

A
B
E
F
C
D

Review 1: Indentation Example

```
print(“A”)
print(“B”)
```

not indented, so “outside” any function

```
def print_letters():
    print(“C”)
    print(“D”)

print(“E”)
print(“F”)

print_letters()
```

indented, so “inside” print_letters function

what does it print?

A
B
E
F
C
D

Review 1: Indentation Example

what does it print?

```
print(“A”)
print(“B”)
```

not indented, so
“outside” any function

```
def print_letters():
    print(“C”)
    print(“D”)
```

indented, so “inside”
print_letters function

```
print(“E”)
print(“F”)
```

also not indented, so
“outside” any function.
Runs BEFORE
print_letters is called

```
print_letters()
```

A
B
E
F
C
D

Review 1: Indentation Example

what does it print?

```
print(“A”)
print(“B”)
```

not indented, so
“outside” any function

```
def print_letters():
    print(“C”)
    print(“D”)
```

indented, so **“inside”**
print_letters function

```
print(“E”)
print(“F”)
```

also not indented, so
“outside” any function.
Runs BEFORE
print_letters is called

```
print_letters()
```

A
B
E
F
C
D

We use **indenting** to tell Python which code is **inside** or **outside** of a function (or other things we’ll learn about soon).

Review 1: Indentation Example

what does it print?

```
print("A")
print("B")

def print_letters():
    print("C")
    print("D")

print("E")
print("F")

print_letters()
```

not indented, so "outside" any function

indented, so "inside" print_letters function

blank lines are **irrelevant**

also not indented, so "outside" any function. Runs BEFORE print_letters is called

A
B
E
F
C
D

We use **indenting** to tell Python which code is **inside** or **outside** of a function (or other things we'll learn about soon).

Review 1: Indentation Example

what does it print?

```
print(“A”)
print(“B”)

def print_letters():
    print(“C”)
    print(“D”)

print(“E”)
print(“F”)

print_letters()
```

we’ll often call the lines of code **inside** something a **“block”** of code

A
B
E
F
C
D

Review 1: Indentation Example

what does it print?

```
print(“A”)
print(“B”)

def print_letters():
    print(“C”)

     print(“D”)
print(“E”)
print(“F”)

print_letters()
```

horizontal spaces
identify blocks
(not vertical space)

A
B
E
F
C
D

Review 2: Argument Passing

```
def h(x=1, y=2):
    print(x, y)  # what is printed?

def g(x, y):
    print(x, y)  # what is printed?
    h(y)
```

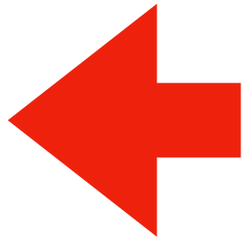
```
def f(x, y):
    print(x, y)  # what is printed?
    g(x=x, y=y+1)

x = 10
y = 20
f(y, x)
```

Today's Outline

Review

Control Flow Diagrams

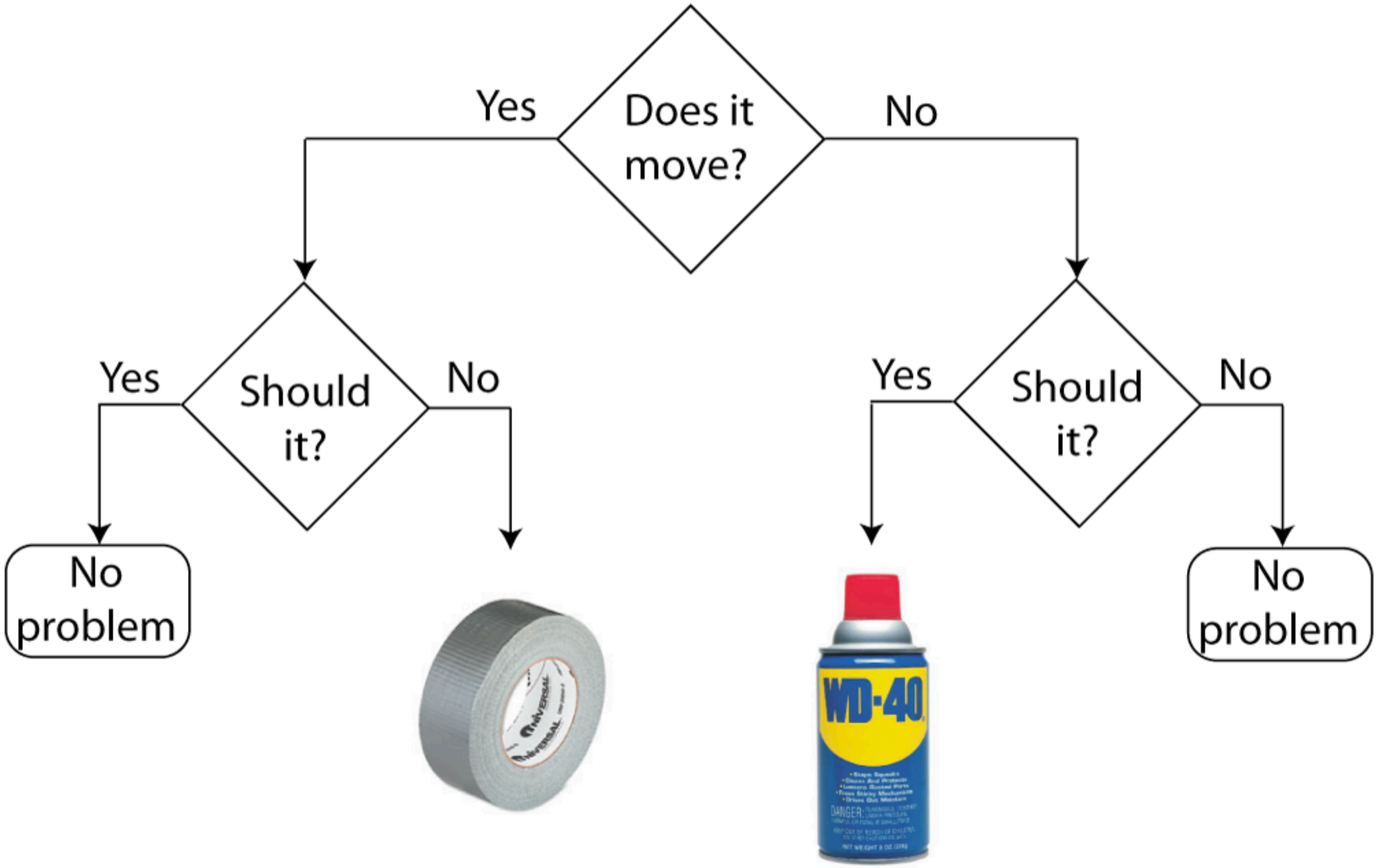


Basic syntax for “if”

Identifying code blocks

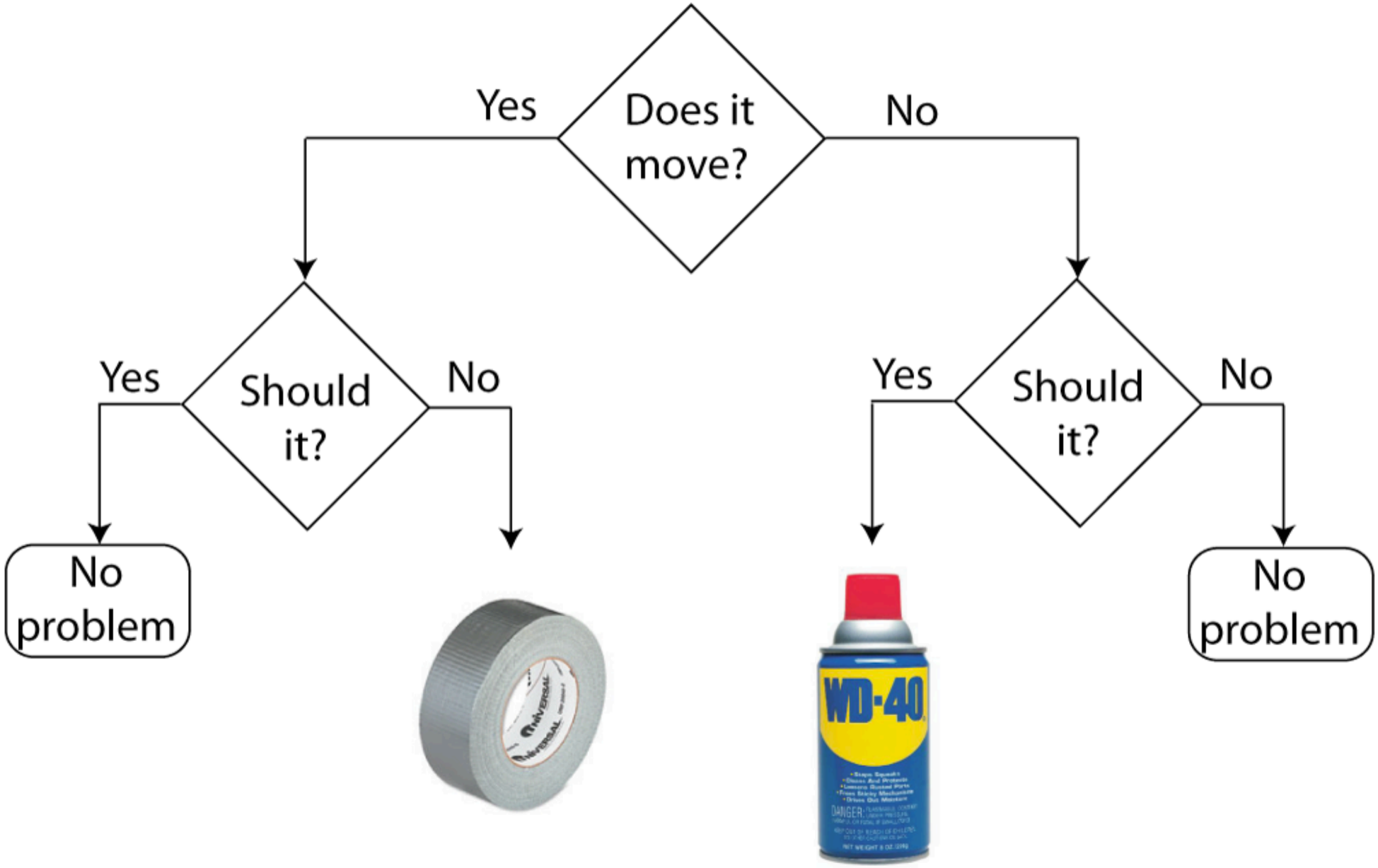
Demos

Laboratory Troubleshooting Flowchart



https://www.drphysics.com/prayer/flowchart.html

Laboratory Troubleshooting Flowchart

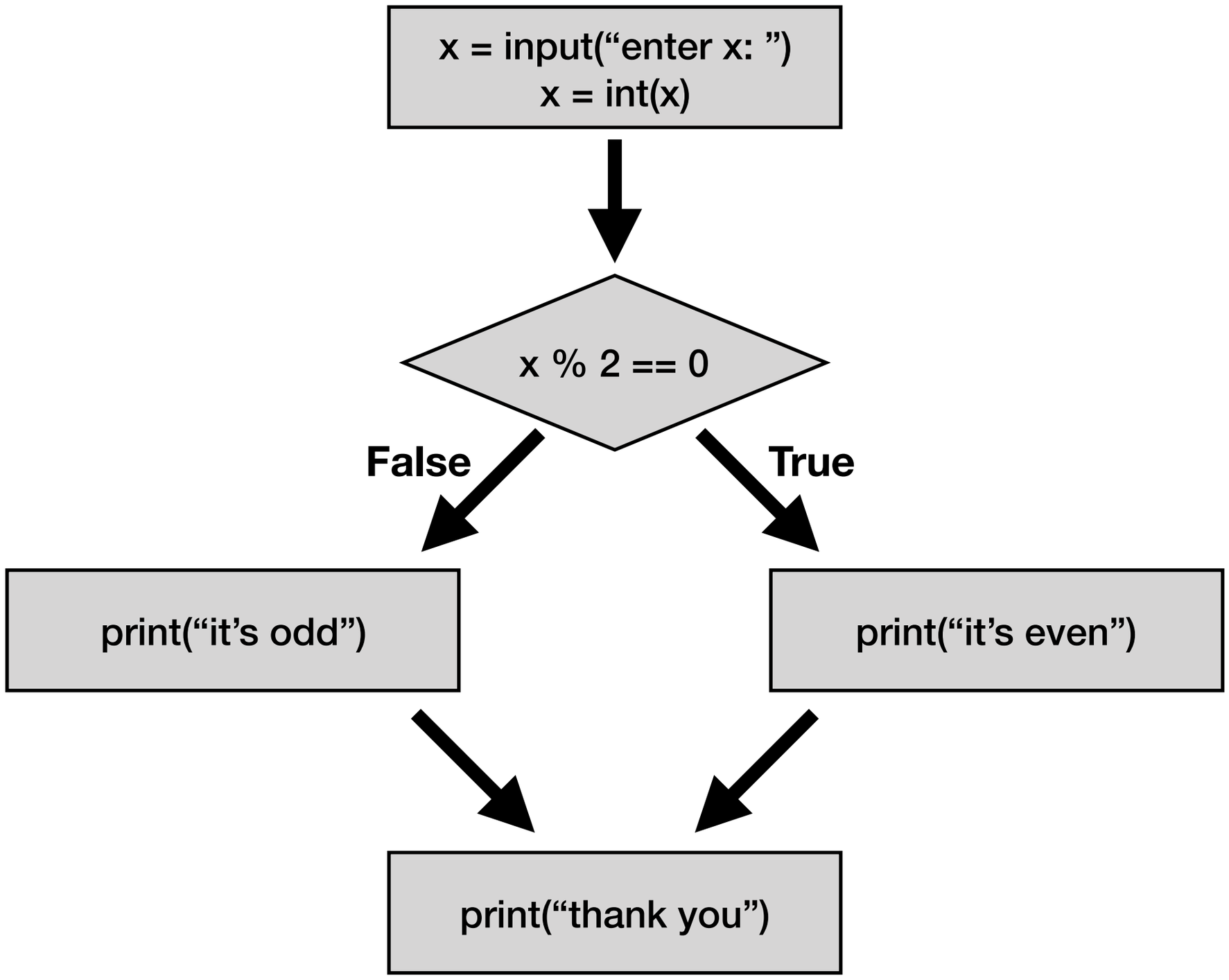


in programming:

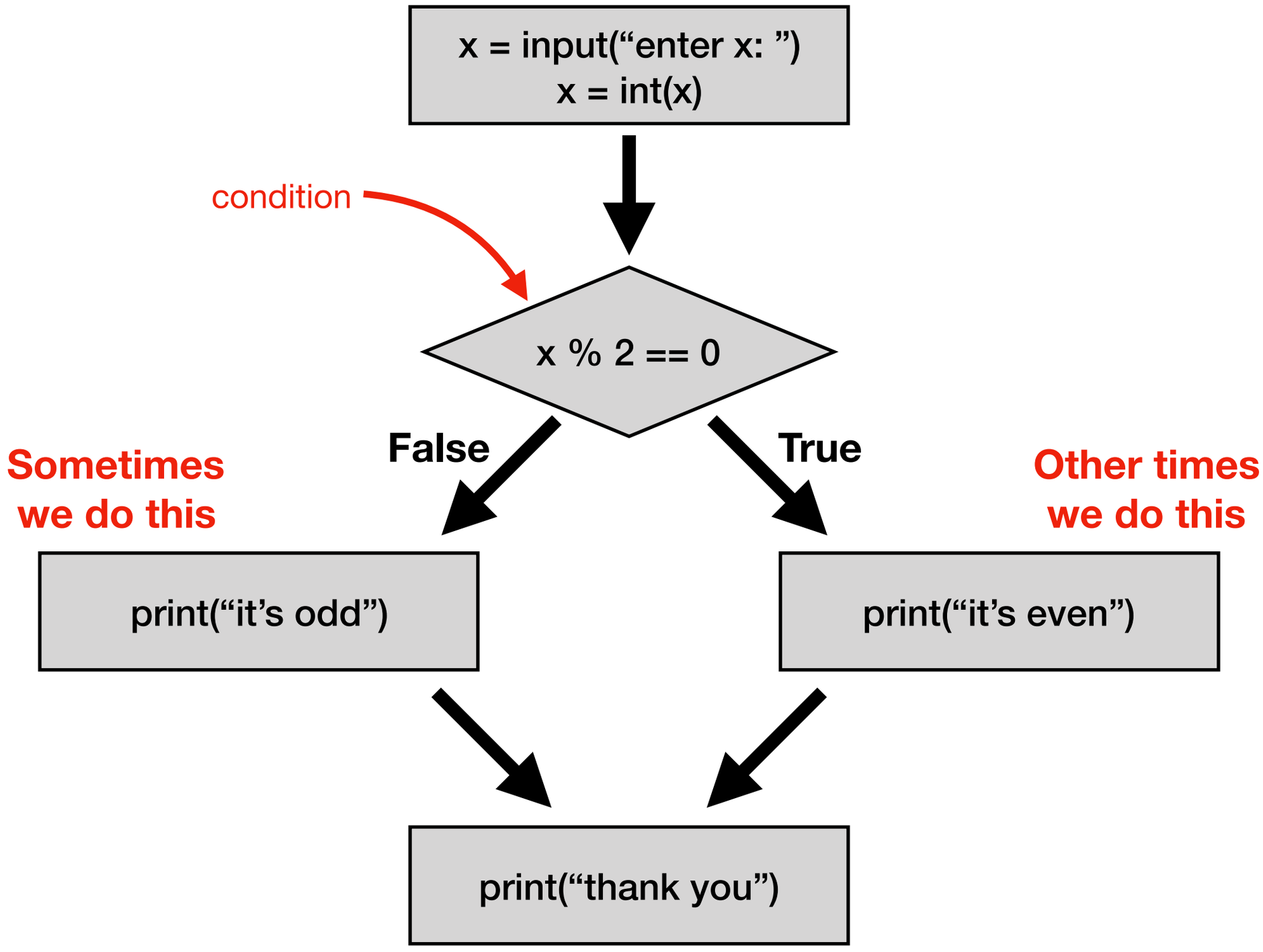
- questions are phrased as *boolean expressions*
- actions are *code/statements*

https://www.drphysics.com/prayer/flowchart.html

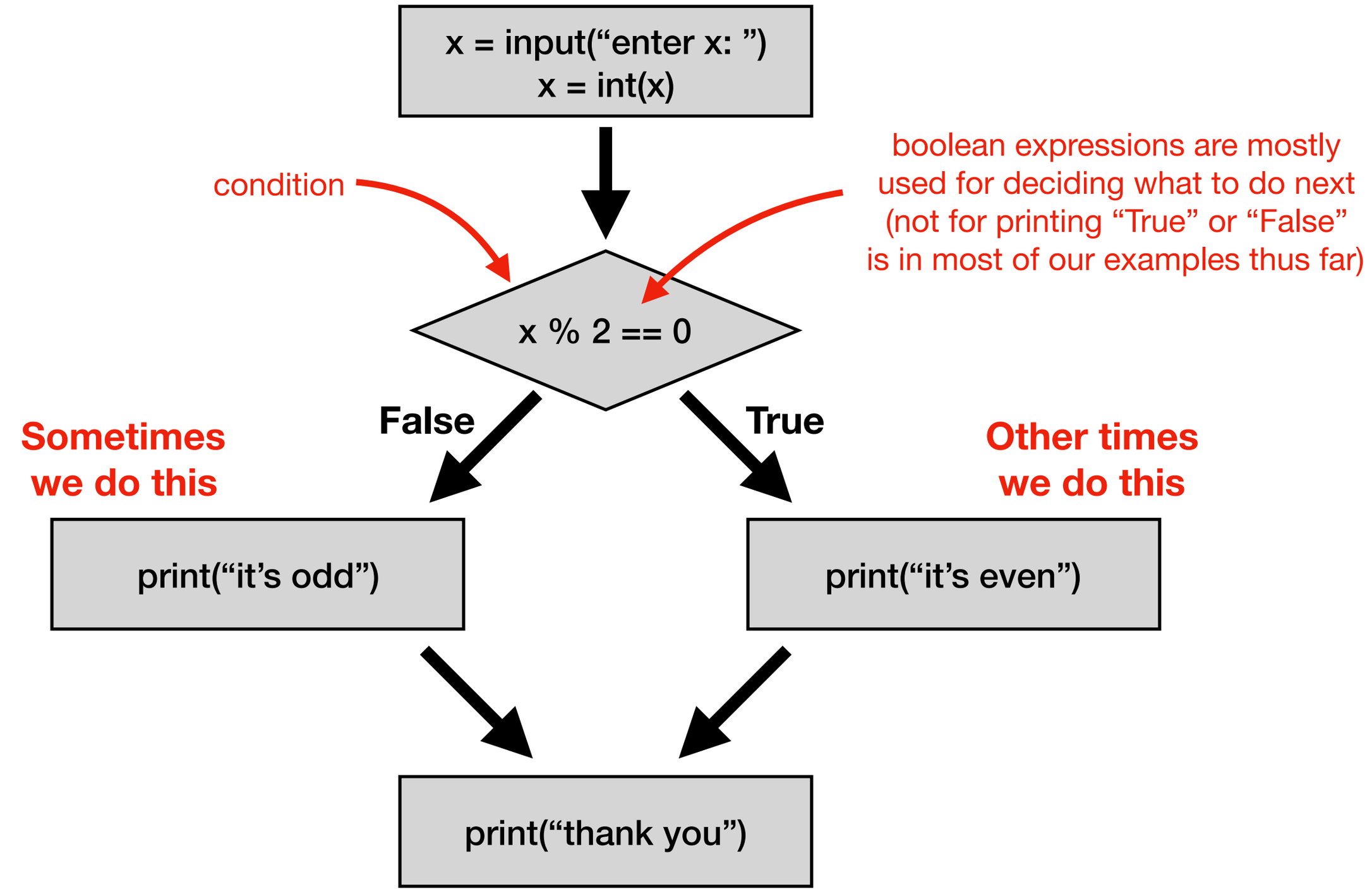
Control Flow Diagrams (Flowcharts for Code)



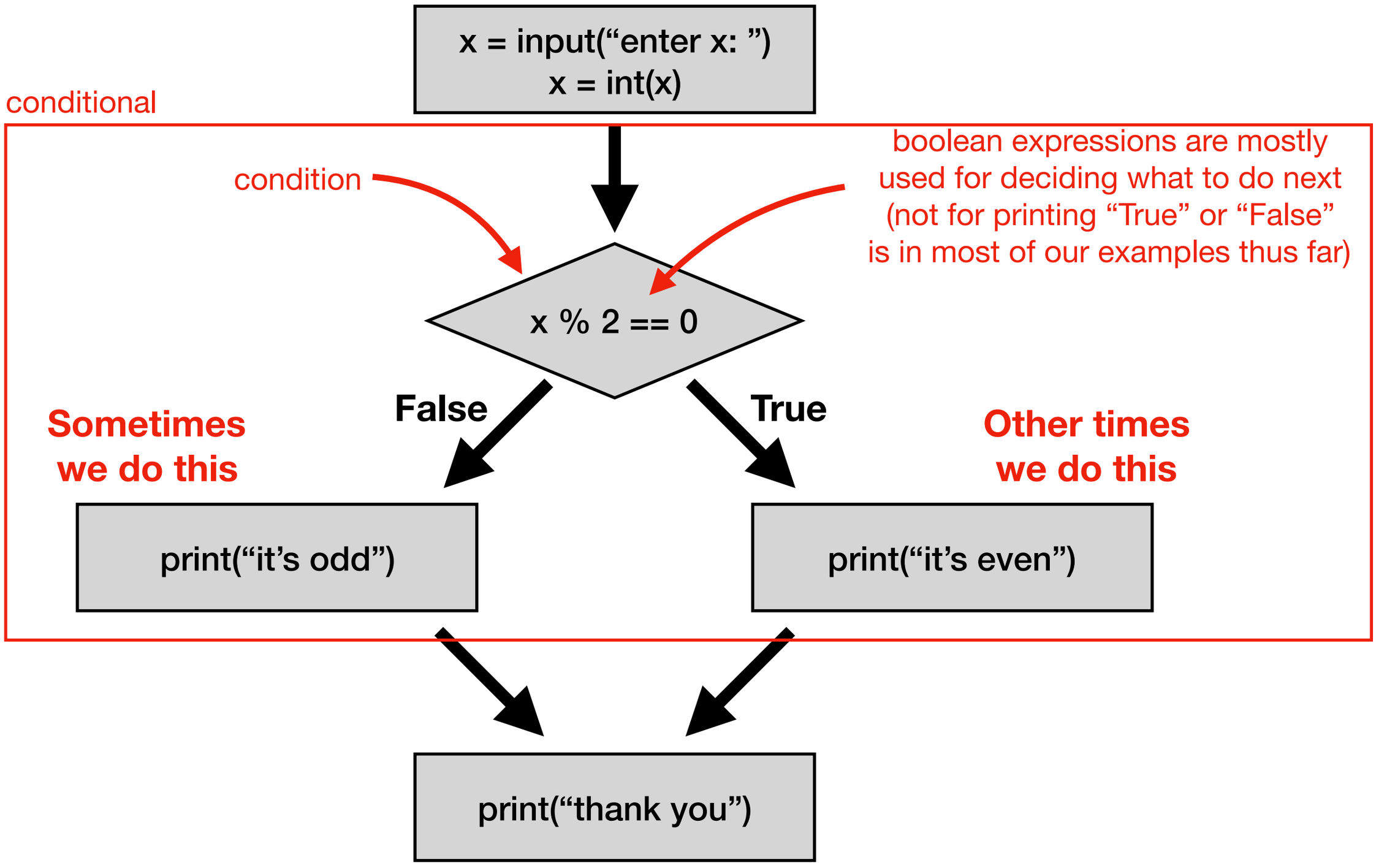
Control Flow Diagrams (Flowcharts for Code)



Control Flow Diagrams (Flowcharts for Code)



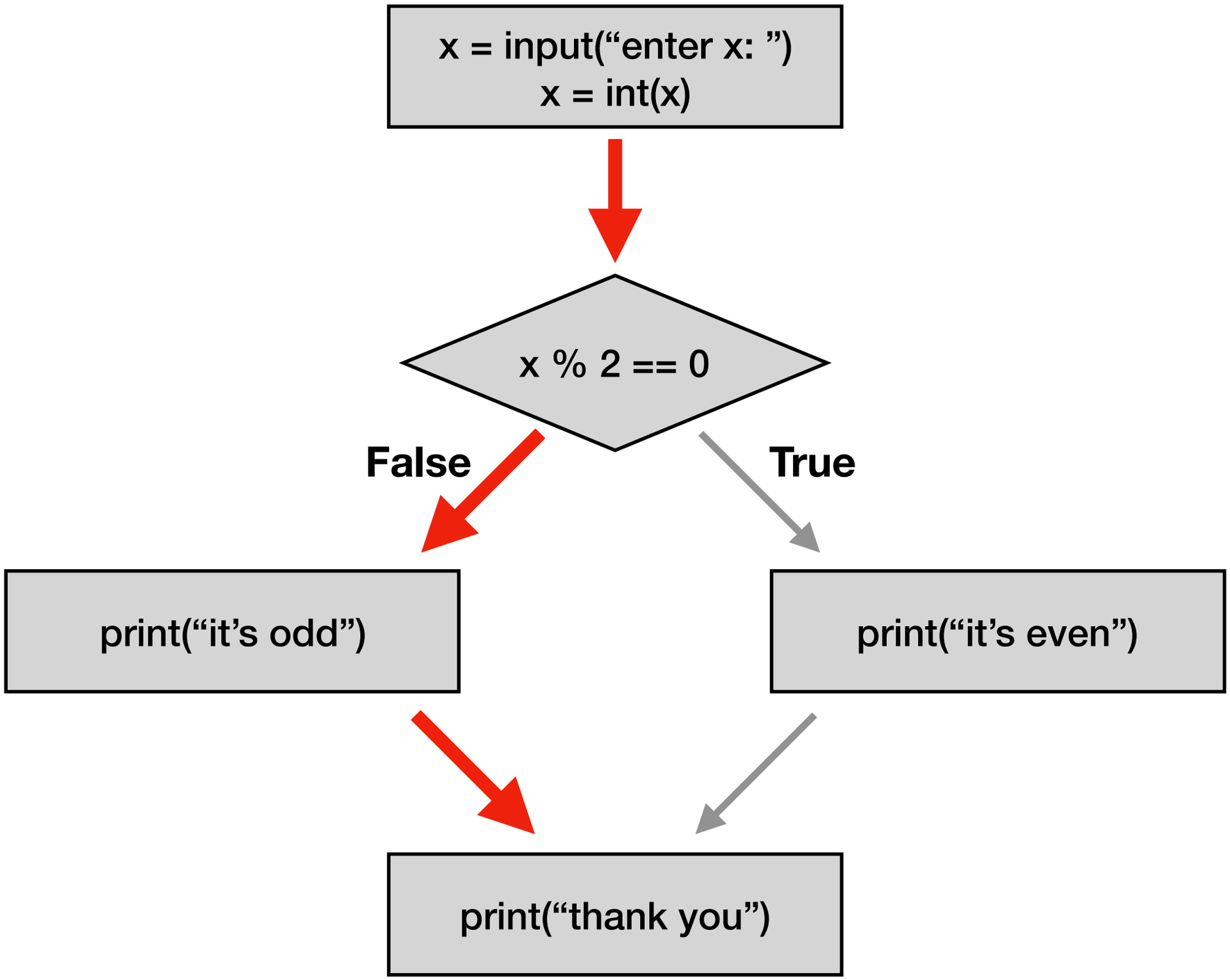
Control Flow Diagrams (Flowcharts for Code)



Branches (aka "Paths of Execution")

Input/Output:

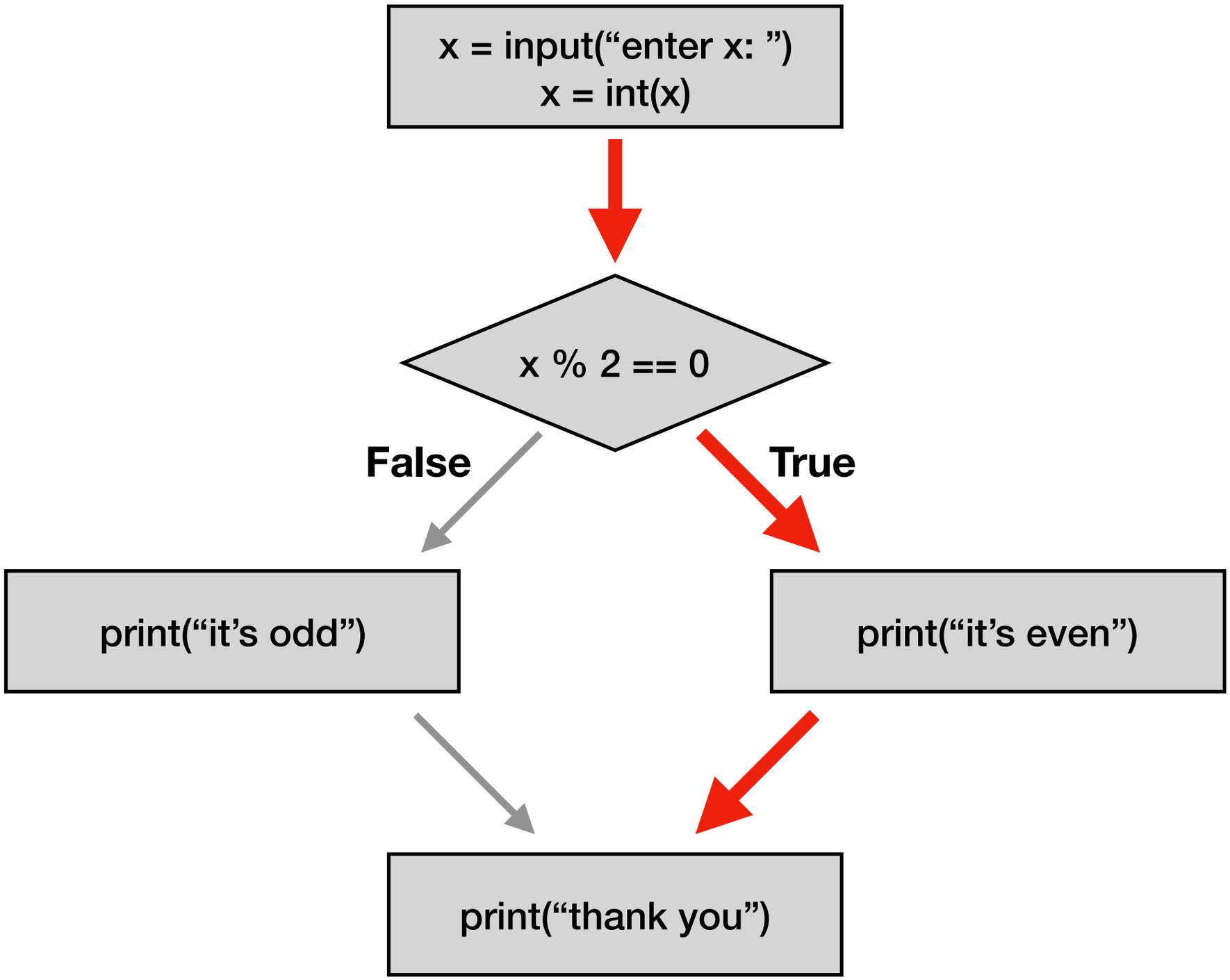
```
enter x: 7
it's odd
thank you
```



Branches (aka "Paths of Execution")

Input/Output:

```
enter x: 8
it's even
thank you
```

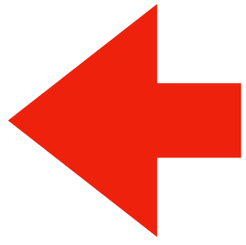


Today's Outline

Review

Control Flow Diagrams

Basic syntax for “if”

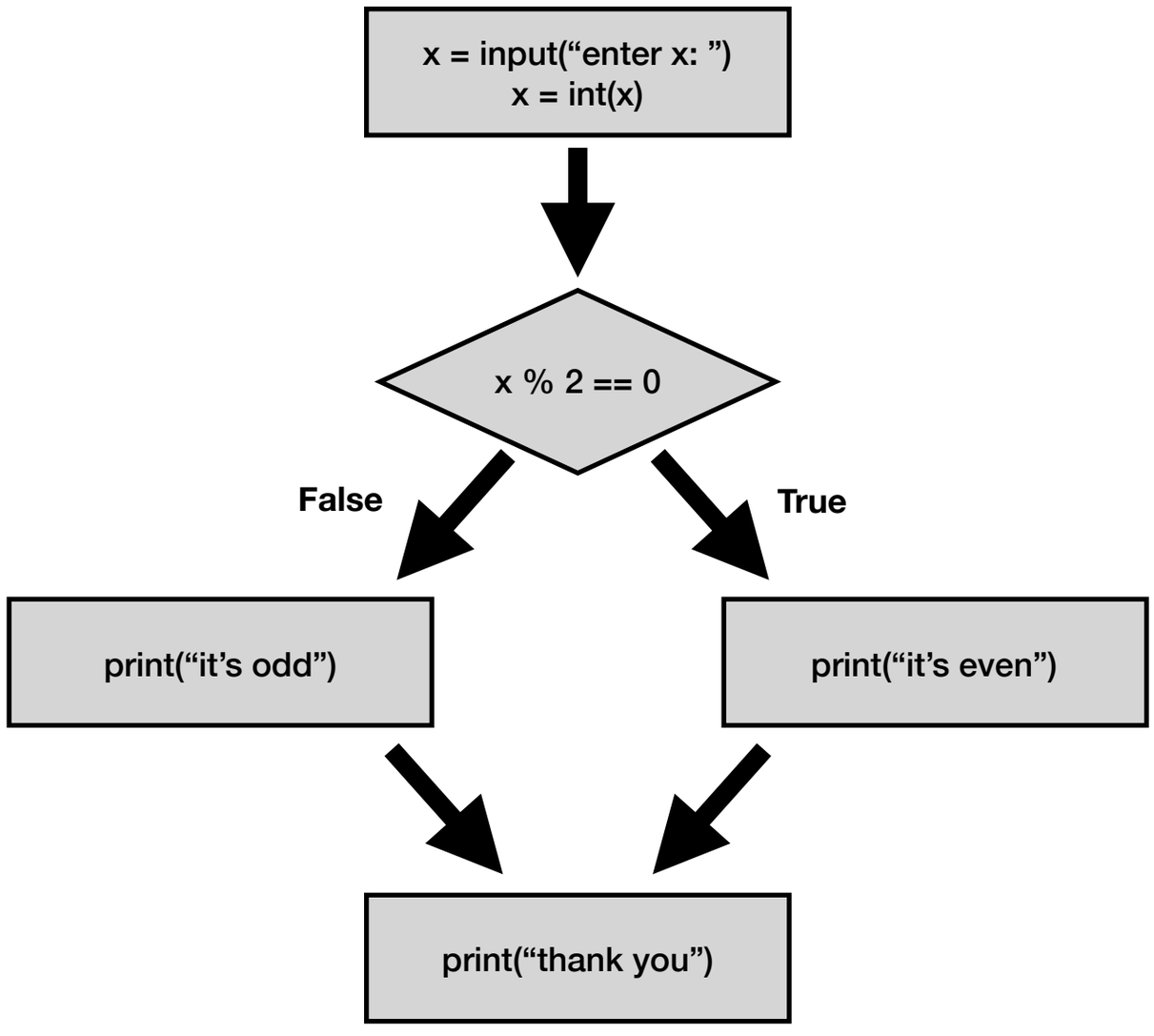


Identifying code blocks

Demos

Writing conditions in Python

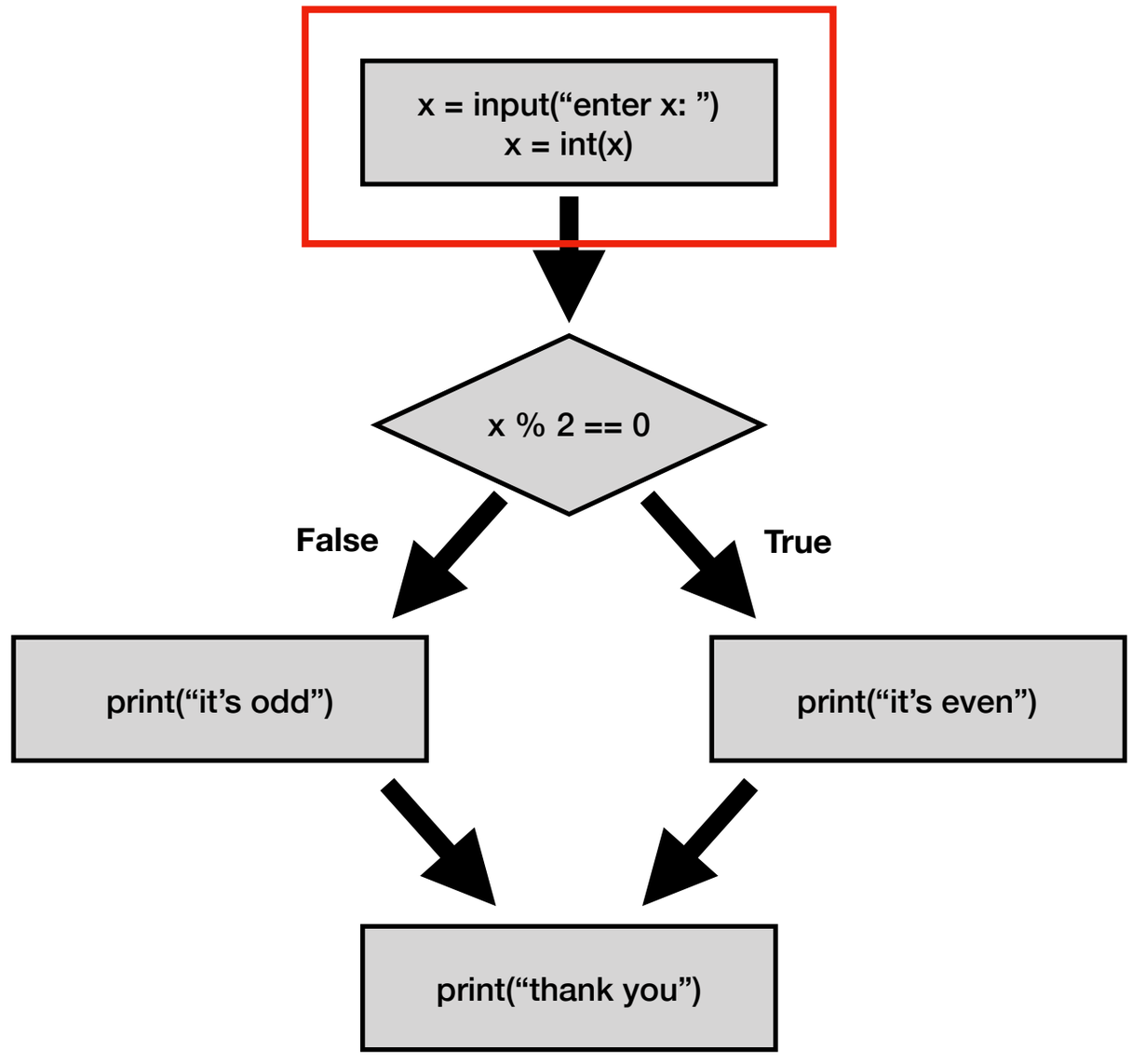
Code:



Writing conditions in Python

Code:

```
x = input(“enter x: ”)
x = int(x)
```

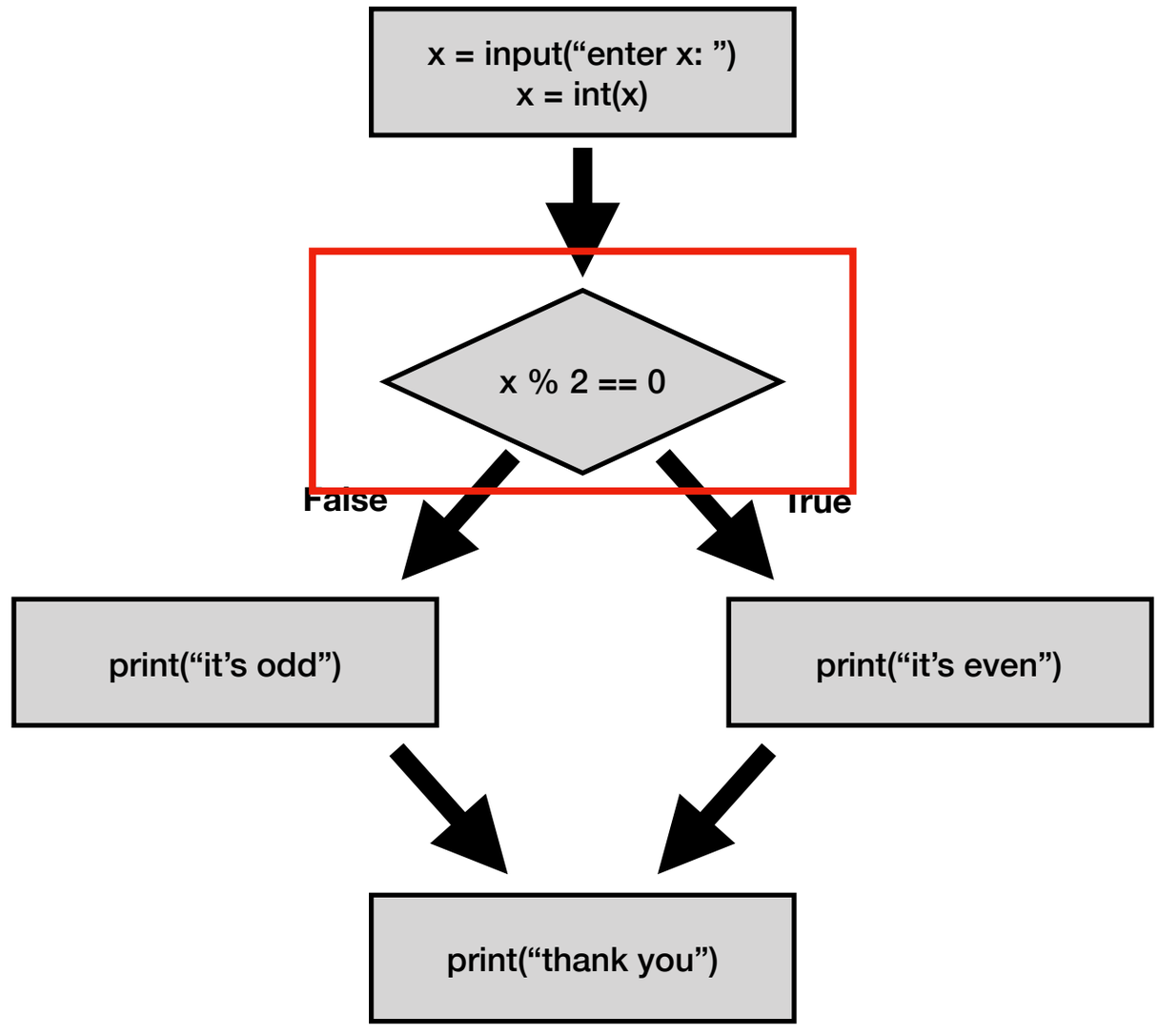


Writing conditions in Python

Code:

```
x = input(“enter x: ”)
x = int(x)

if x % 2 == 0:
```



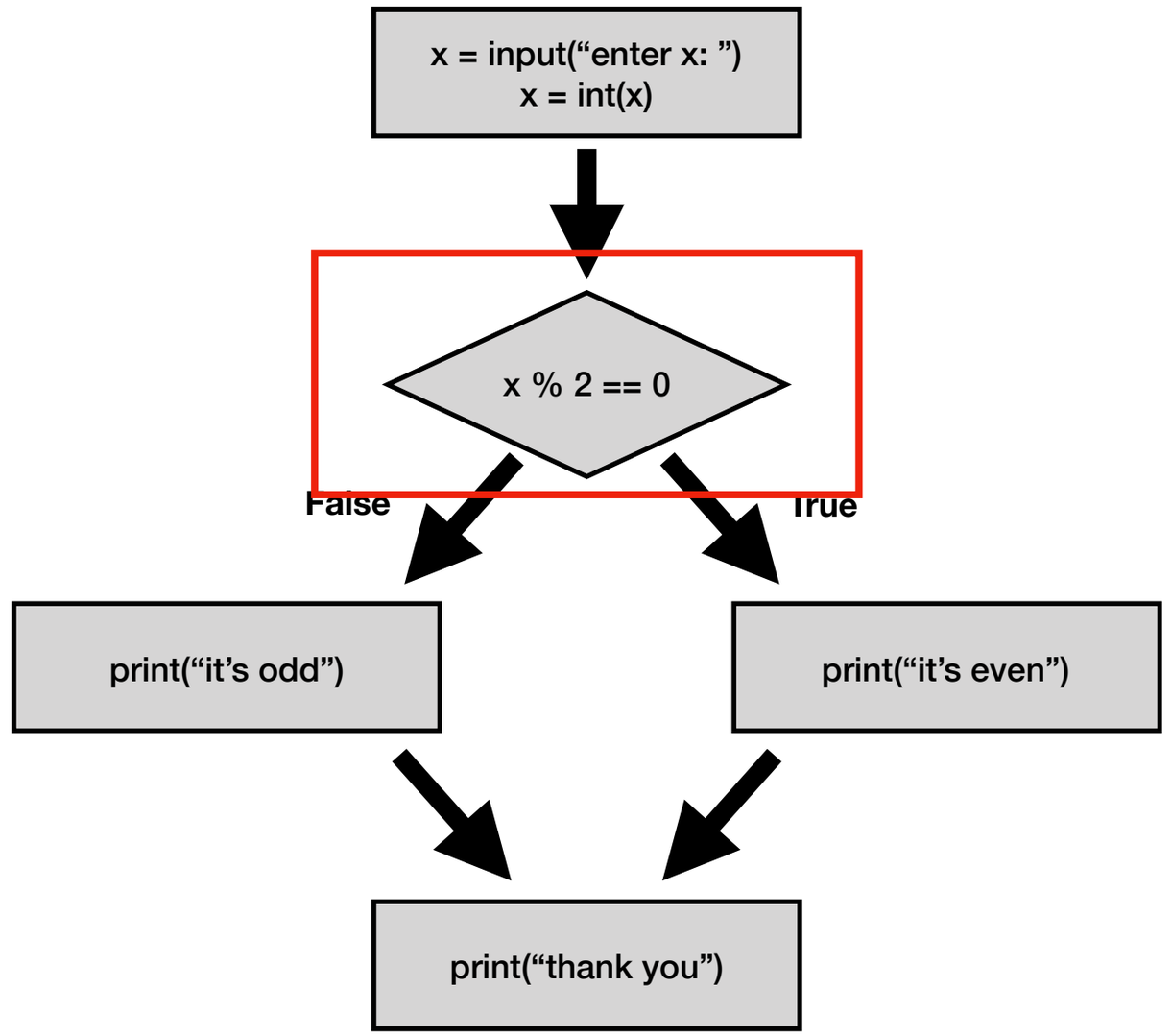
Writing conditions in Python

Code:

```
x = input(“enter x: ”)
x = int(x)

if x % 2 == 0:
```

boolean expression

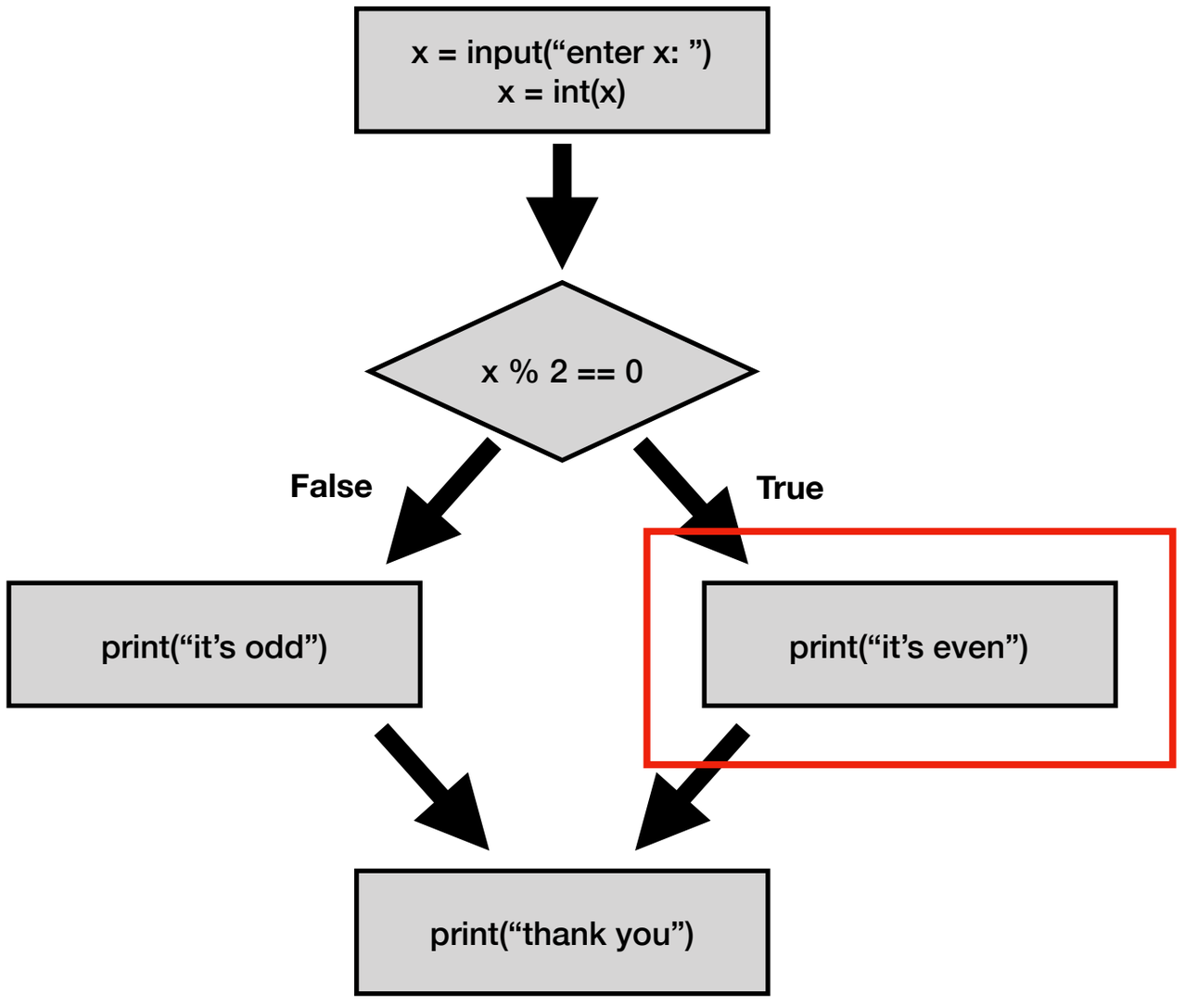


Writing conditions in Python

Code:

```
x = input(“enter x: ”)
x = int(x)

if x % 2 == 0:
    print(“it’s even”)
```

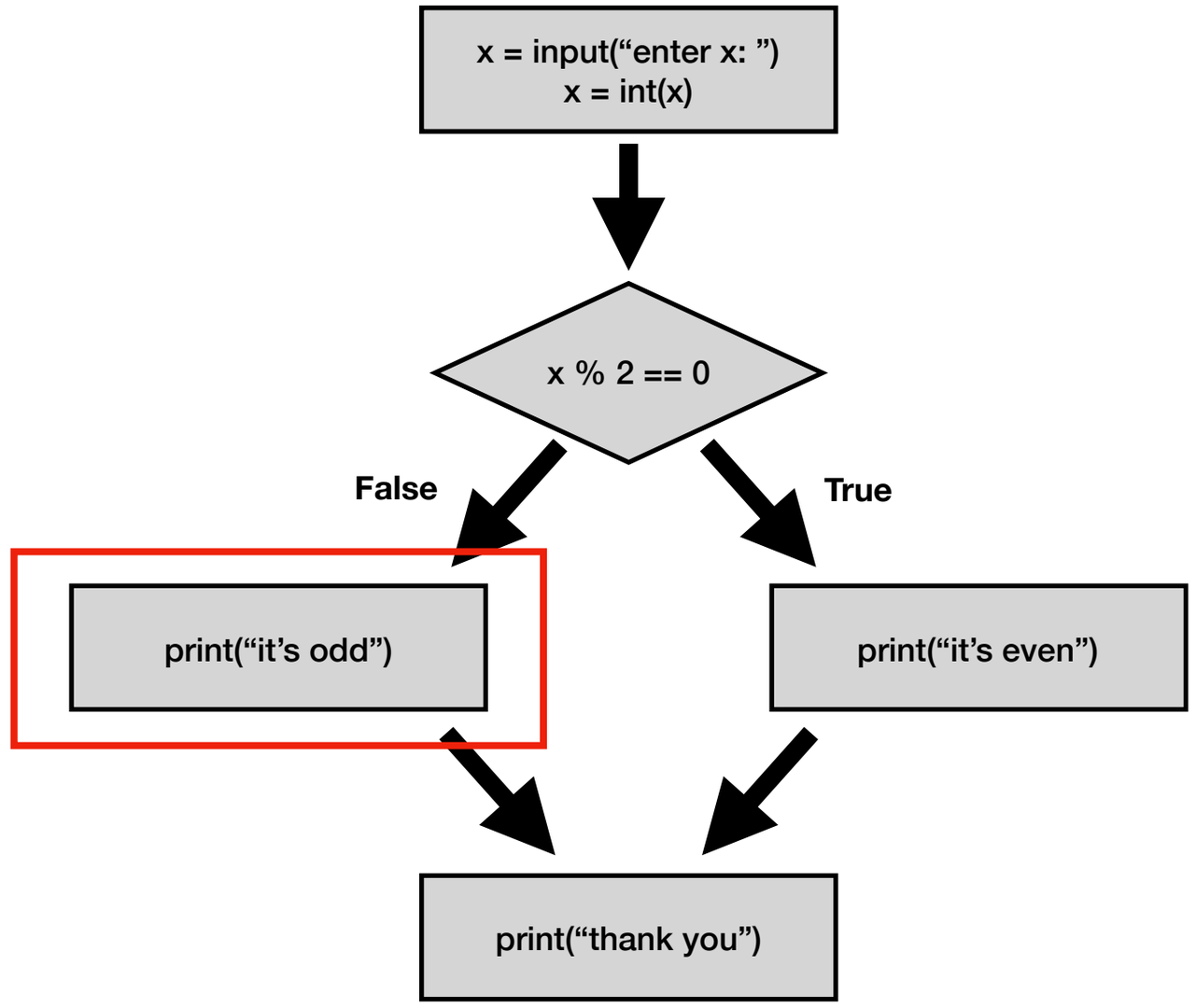


Writing conditions in Python

Code:

```
x = input(“enter x: ”)
x = int(x)

if x % 2 == 0:
    print(“it’s even”)
else:
    print(“it’s odd”)
```



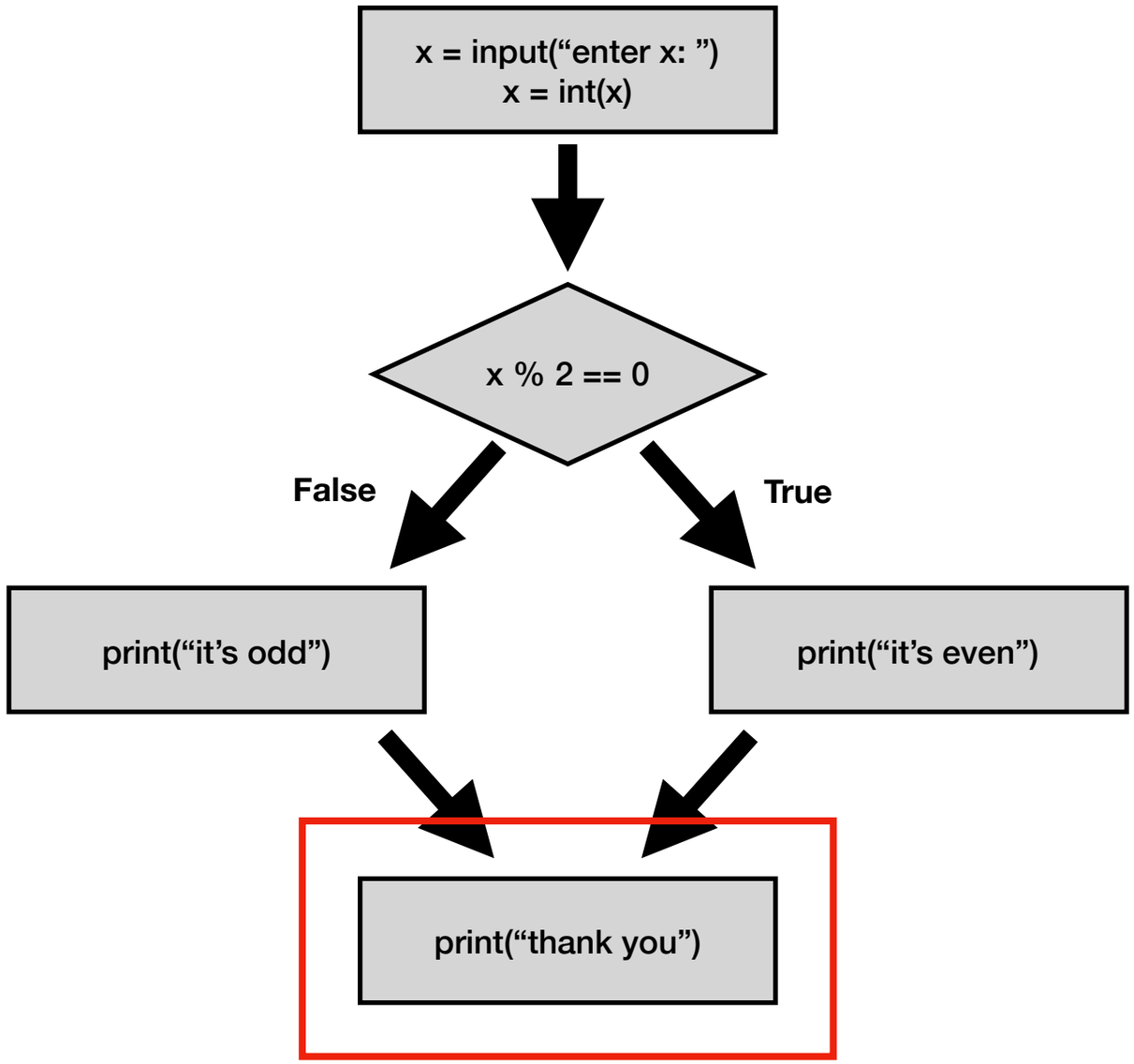
Writing conditions in Python

Code:

```
x = input(“enter x: ”)
x = int(x)

if x % 2 == 0:
    print(“it’s even”)
else:
    print(“it’s odd”)

print(“thank you”)
```



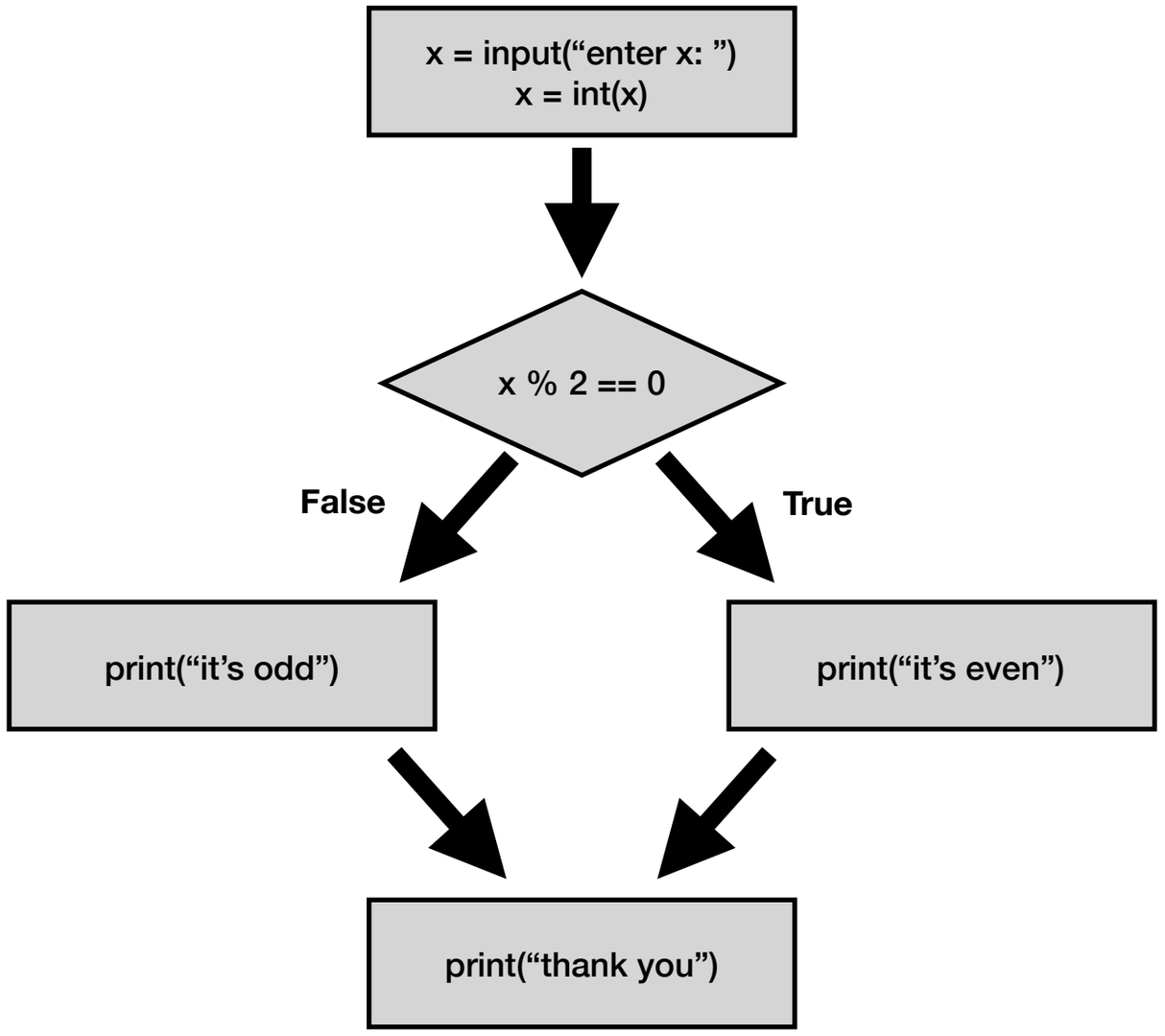
Writing conditions in Python

Code:

```
x = input(“enter x: ”)
x = int(x)

if x % 2 == 0:
    print(“it’s even”)
else:
    print(“it’s odd”)

print(“thank you”)
```



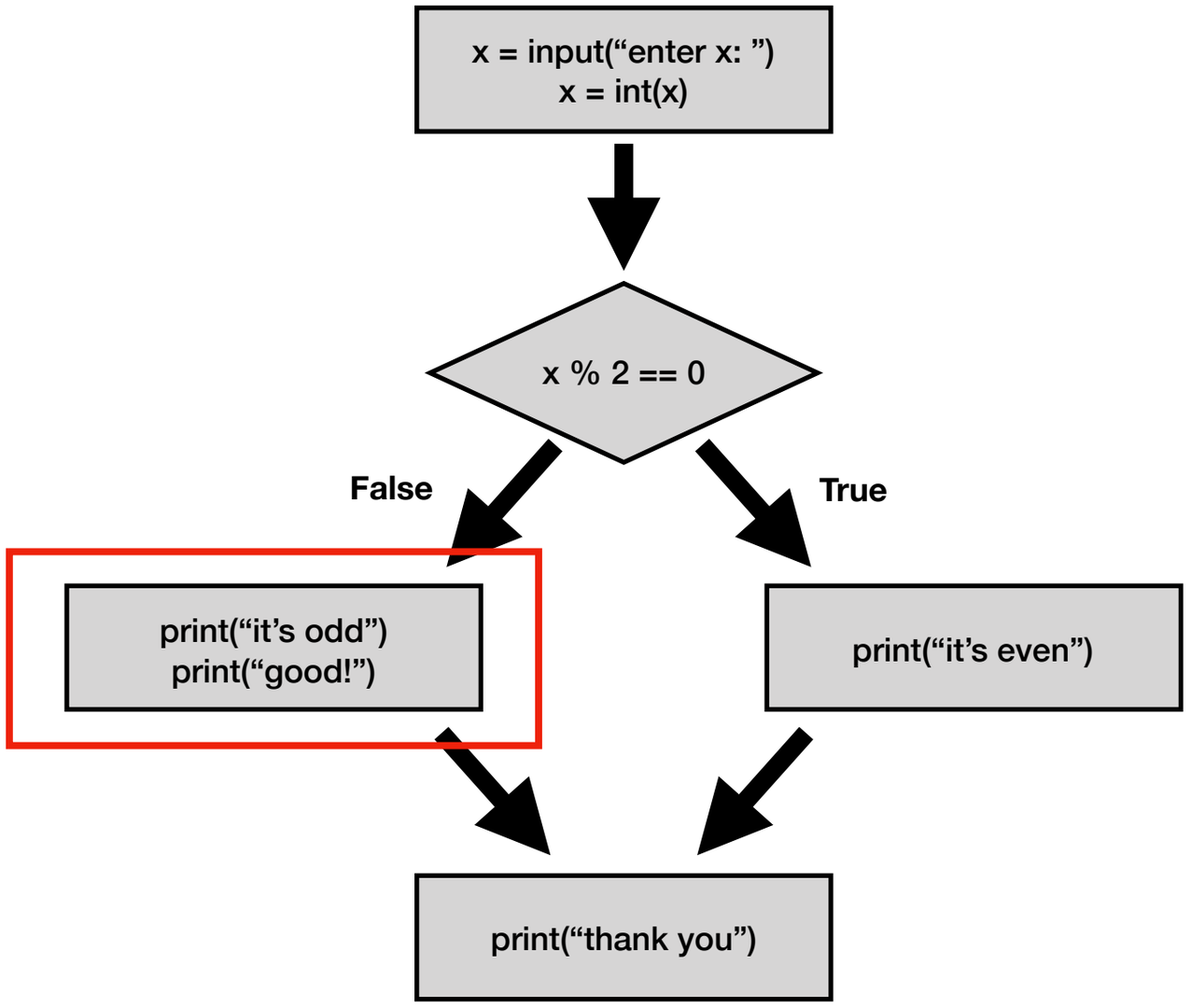
Writing conditions in Python

Code:

```
x = input(“enter x: ”)
x = int(x)

if x % 2 == 0:
    print(“it’s even”)
else:
    print(“it’s odd”)
    print(“good!”)

print(“thank you”)
```



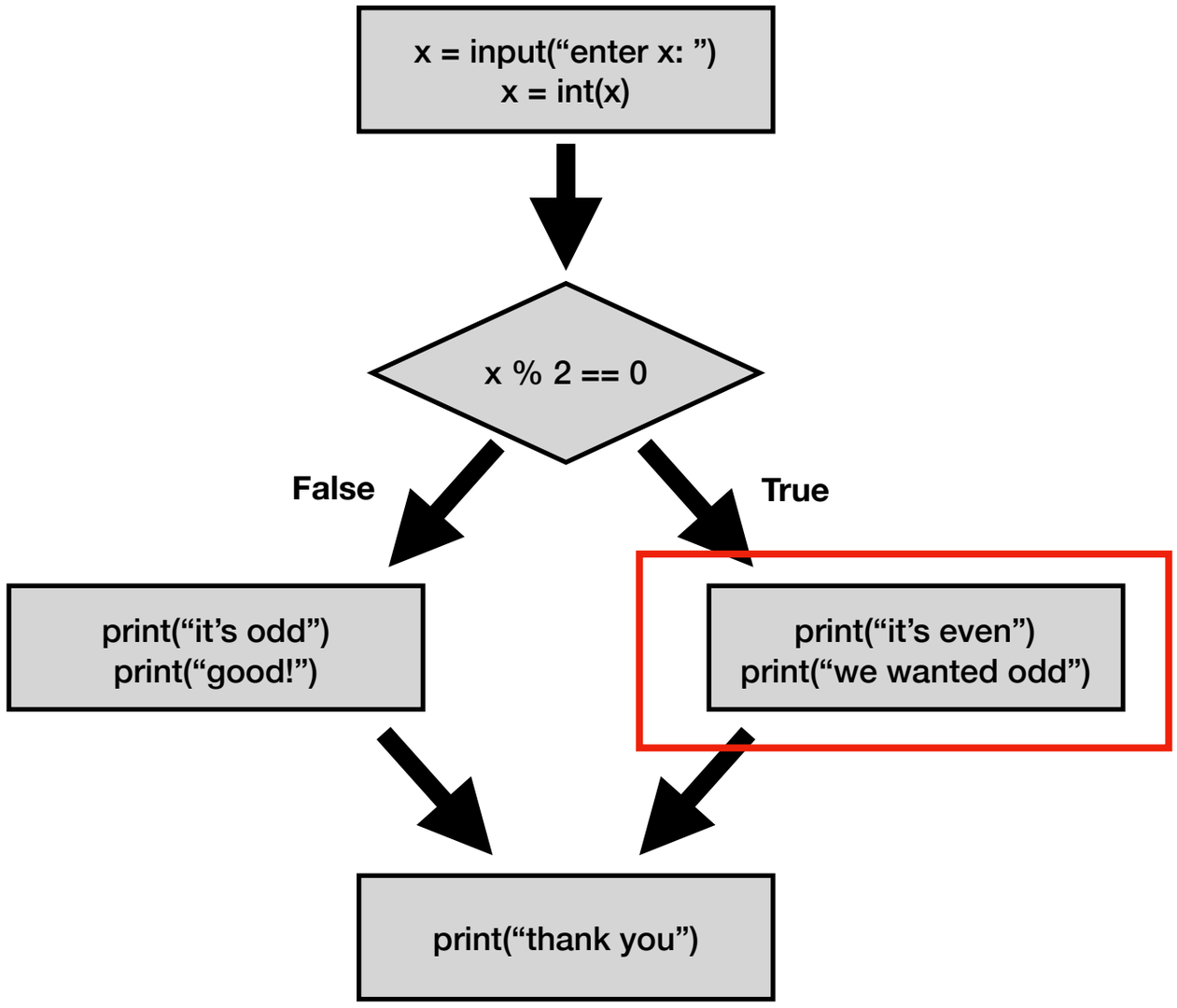
Writing conditions in Python

Code:

```
x = input(“enter x: ”)
x = int(x)

if x % 2 == 0:
    print(“it’s even”)
    print(“we wanted odd”)
else:
    print(“it’s odd”)
    print(“good!”)

print(“thank you”)
```



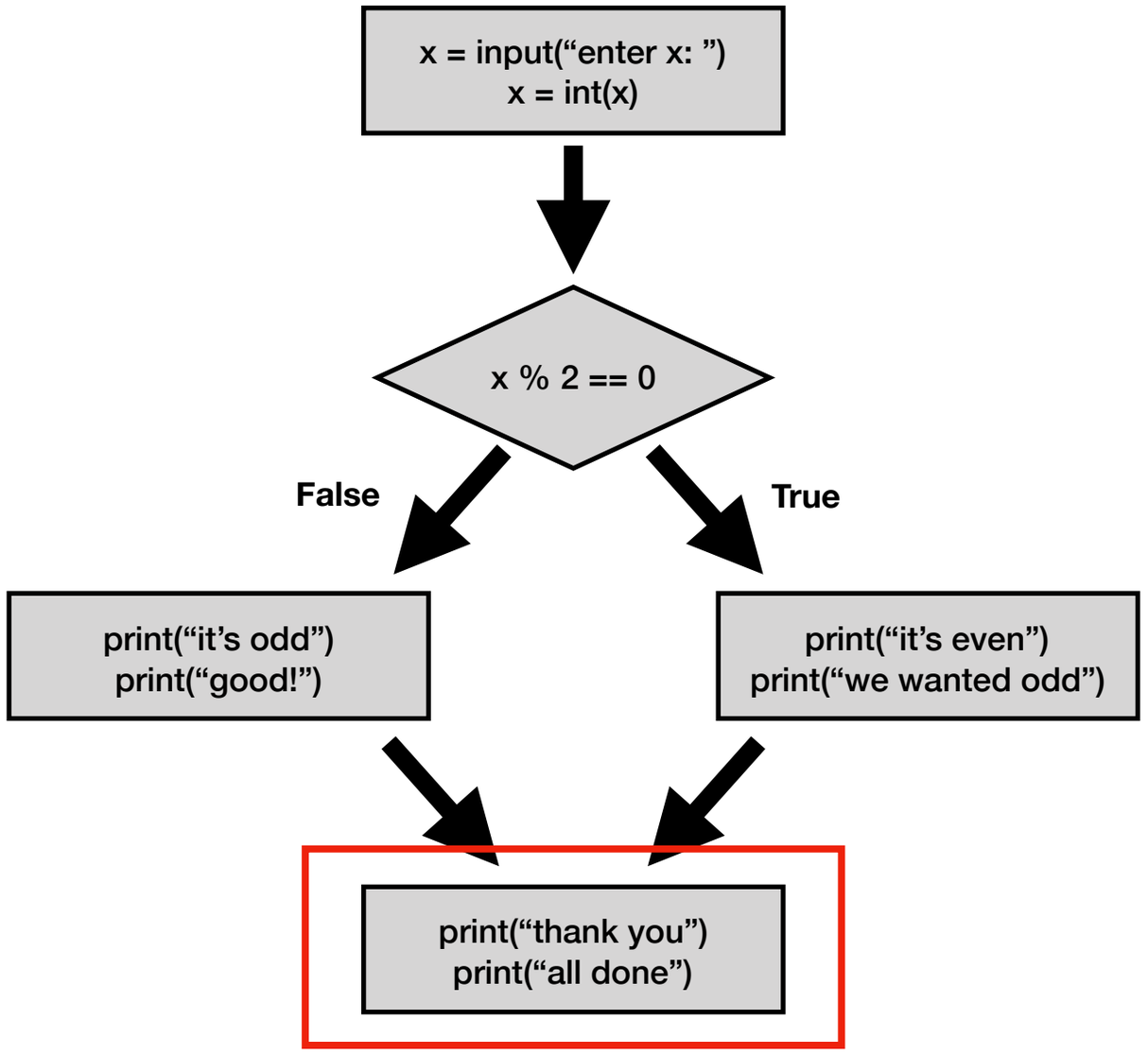
Writing conditions in Python

Code:

```
x = input(“enter x: ”)
x = int(x)

if x % 2 == 0:
    print(“it’s even”)
    print(“we wanted odd”)
else:
    print(“it’s odd”)
    print(“good!”)

print(“thank you”)
print(“all done”)
```



Today's Outline

Review

Control Flow Diagrams

Basic syntax for “if”

Identifying code blocks

Demos

Code Blocks

Code:

```
x = input(“enter x: ”)
x = int(x)

if x % 2 == 0:
    print(“it’s even”)
    print(“we wanted odd”)
else:
    print(“it’s odd”)
    print(“good!”)

print(“thank you”)
print(“all done”)
```

Code Blocks

Code:

```
x = input("enter x: ")
x = int(x)

if x % 2 == 0:
    print("it's even")
    print("we wanted odd")
else:
    print("it's odd")
    print("good!")

print("thank you")
print("all done")
```

block of code inside "if"

Code Blocks

Code:

```
x = input("enter x: ")
x = int(x)

if x % 2 == 0:
    print("it's even")
    print("we wanted odd")
else:
    print("it's odd")
    print("good!")

print("thank you")
print("all done")
```

block of code inside "if"

block of code inside "else"

Code Blocks

Code:

```
x = input("enter x: ")
x = int(x)

if x % 2 == 0:
    print("it's even")
    print("we wanted odd")
else:
    print("it's odd")
    print("good!")

print("thank you")
print("all done")
```

block of code inside "if"

block of code inside "else"

What if all this were inside a function?

Code Blocks

Code:

```
def check_oddness():
    x = input(“enter x: ”)
    x = int(x)

    if x % 2 == 0:
        print(“it’s even”)
        print(“we wanted odd”)
    else:
        print(“it’s odd”)
        print(“good!”)

    print(“thank you”)
    print(“all done”)

check_oddness()
```

block of code inside “if”

block of code inside “else”

Code Blocks

Code:

```
def check_oddness():
    x = input("enter x: ")
    x = int(x)

    if x % 2 == 0:
        print("it's even")
        print("we wanted odd")
    else:
        print("it's odd")
        print("good!")

    print("thank you")
    print("all done")

check_oddness()
```

block of code inside "if"

block of code inside "else"

block of code in check_oddness

Code Blocks

You need to get good at “seeing” code blocks in Python code.

Code:

```
def check_oddness():
    x = input(“enter x: ”)
    x = int(x)

    if x % 2 == 0:
        print(“it’s even”)
        print(“we wanted odd”)
    else:
        print(“it’s odd”)
        print(“good!”)

    print(“thank you”)
    print(“all done”)

check_oddness()
```

block of code inside “if”

block of code inside “else”

block of code in check_oddness

Code Blocks

You need to get good at “seeing” code blocks in Python code.
Even blocks inside blocks inside blocks…

Code:

```
def check_oddness():
    x = input(“enter x: ”)
    x = int(x)

    if x % 2 == 0:
        print(“it’s even”)
        print(“we wanted odd”)
    else:
        print(“it’s odd”)
        print(“good!”)

    print(“thank you”)
    print(“all done”)

check_oddness()
```

block of code inside “if”

block of code inside “else”

block of code in check_oddness

Identifying Code Blocks

Code:

```
def check_oddness():
    x = input(“enter x: ”)
    x = int(x)

    if x % 2 == 0:
        print(“it’s even”)
        print(“we wanted odd”)
    else:
        print(“it’s odd”)
        print(“good!”)

    print(“thank you”)
    print(“all done”)

check_oddness()
```

Identifying Code Blocks

Code:

```
def check_oddness():
    x = input("enter x: ")
    x = int(x)

    if x % 2 == 0:
        print("it's even")
        print("we wanted odd")
    else:
        print("it's odd")
        print("good!")

    print("thank you")
    print("all done")

check_oddness()
```

Step 1: look for a colon at end of a line

Identifying Code Blocks

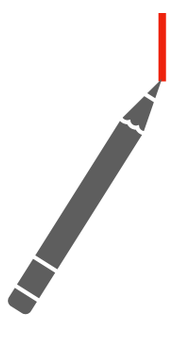
Code:

```
def check_oddness():
    x = input(“enter x: ”)
    x = int(x)

    if x % 2 == 0:
        print(“it’s even”)
        print(“we wanted odd”)
    else:
        print(“it’s odd”)
        print(“good!”)

    print(“thank you”)
    print(“all done”)

check_oddness()
```



Step 2: start drawing a line on next code line, indented in

Identifying Code Blocks

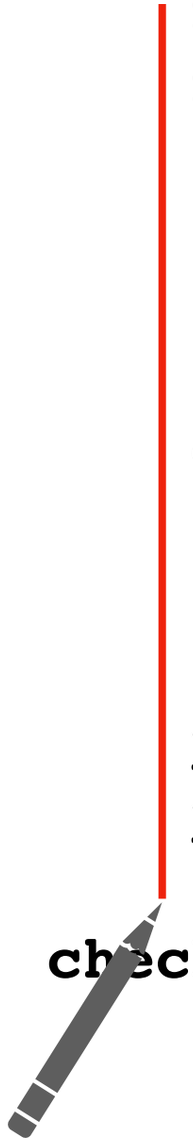
Code:

```
def check_oddness():
    x = input(“enter x: ”)
    x = int(x)

    if x % 2 == 0:
        print(“it’s even”)
        print(“we wanted odd”)
    else:
        print(“it’s odd”)
        print(“good!”)

    print(“thank you”)
    print(“all done”)

check_oddness()
```



Step 3: continue down until you hit code that is less indented

Identifying Code Blocks

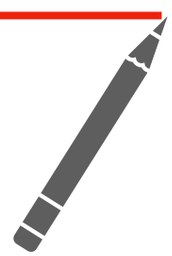
Code:

```
def check_oddness():
    x = input(“enter x: ”)
    x = int(x)

    if x % 2 == 0:
        print(“it’s even”)
        print(“we wanted odd”)
    else:
        print(“it’s odd”)
        print(“good!”)

    print(“thank you”)
    print(“all done”)

check_oddness()
```



Step 4: box off the code

Identifying Code Blocks

Code:

```
def check_oddness():
    x = input(“enter x: ”)
    x = int(x)

    if x % 2 == 0:
        print(“it’s even”)
        print(“we wanted odd”)
    else:
        print(“it’s odd”)
        print(“good!”)

    print(“thank you”)
    print(“all done”)

check_oddness()
```

Step 4: box off the code

Identifying Code Blocks

Code:

```
def check_oddness():
    x = input("enter x: ")
    x = int(x)

    if x % 2 == 0:
        print("it's even")
        print("we wanted odd")
    else:
        print("it's odd")
        print("good!")

    print("thank you")
    print("all done")

check_oddness()
```

to find more boxes,
look for the next colon
and repeat

Identifying Code Blocks

Code:

```
def check_oddness():
    x = input(“enter x: ”)
    x = int(x)

    if x % 2 == 0:
        print(“it’s even”)
        print(“we wanted odd”)
    else:
        print(“it’s odd”)
        print(“good!”)

    print(“thank you”)
    print(“all done”)

check_oddness()
```

to find more boxes,
look for the next colon
and repeat

Identifying Code Blocks

Code:

```
def check_oddness():
    x = input(“enter x: ”)
    x = int(x)

    if x % 2 == 0:
        print(“it’s even”)
        print(“we wanted odd”)
    else:
        print(“it’s odd”)
        print(“good!”)

    print(“thank you”)
    print(“all done”)

check_oddness()
```

to find more boxes,
look for the next colon
and repeat

Identifying Code Blocks

Code:

```
def check_oddness():
    x = input(“enter x: ”)
    x = int(x)

    if x % 2 == 0:
        print(“it’s even”)
        print(“we wanted odd”)
    else:
        print(“it’s odd”)
        print(“good!”)

    print(“thank you”)
    print(“all done”)

check_oddness()
```

to find more boxes, look for the next colon and repeat

Identifying Code Blocks

Do problem 1 on worksheet

Code:

```
def check_oddness():
    x = input("enter x: ")
    x = int(x)

    if x % 2 == 0:
        print("it's even")
        print("we wanted odd")
    else:
        print("it's odd")
        print("good!")

    print("thank you")
    print("all done")

check_oddness()
```

to find more boxes, look for the next colon and repeat

Today's Outline

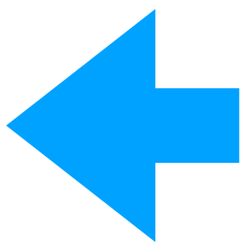
Review

Control Flow Diagrams

Basic syntax for “if”

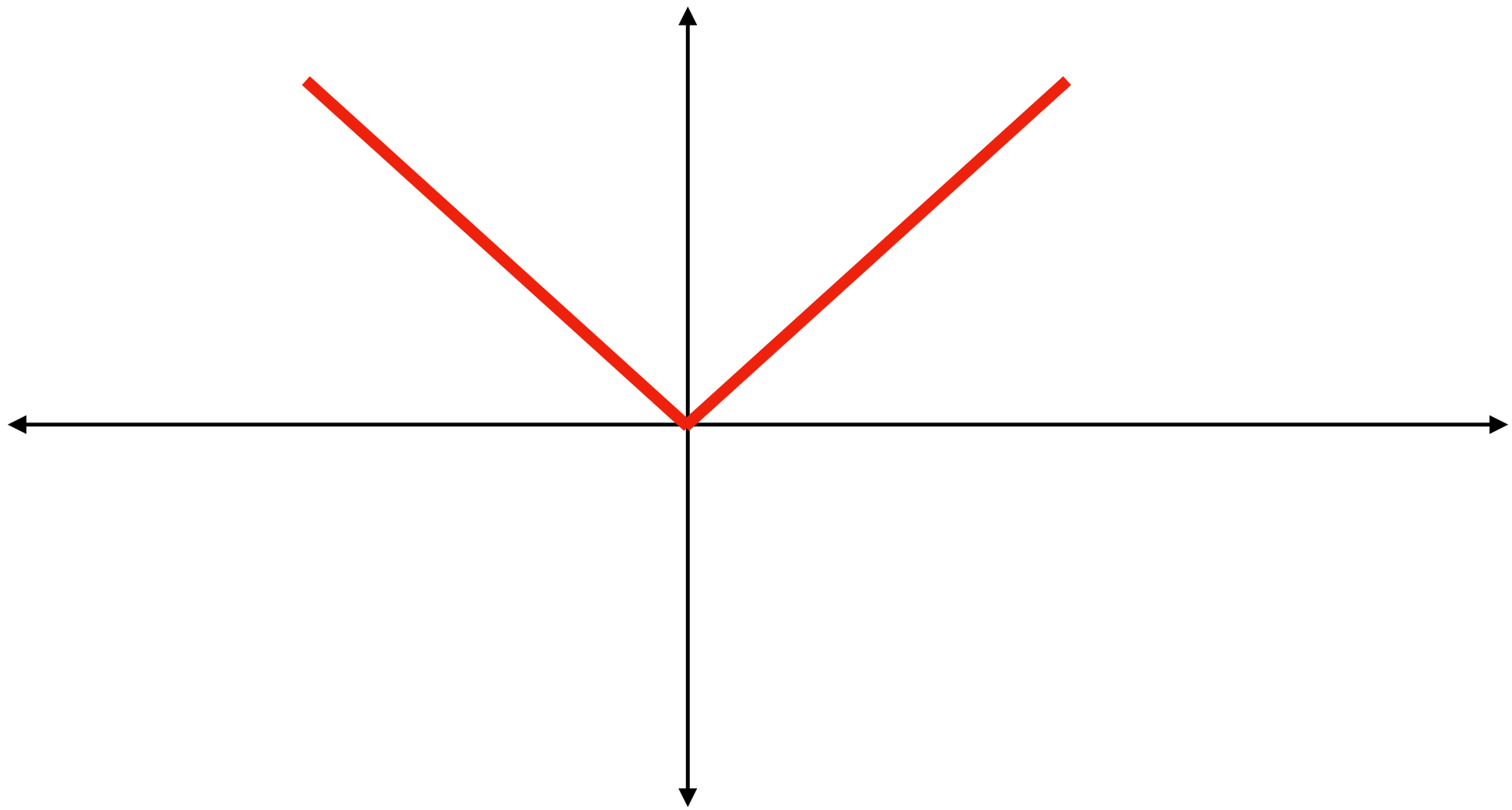
Identifying code blocks

Demos

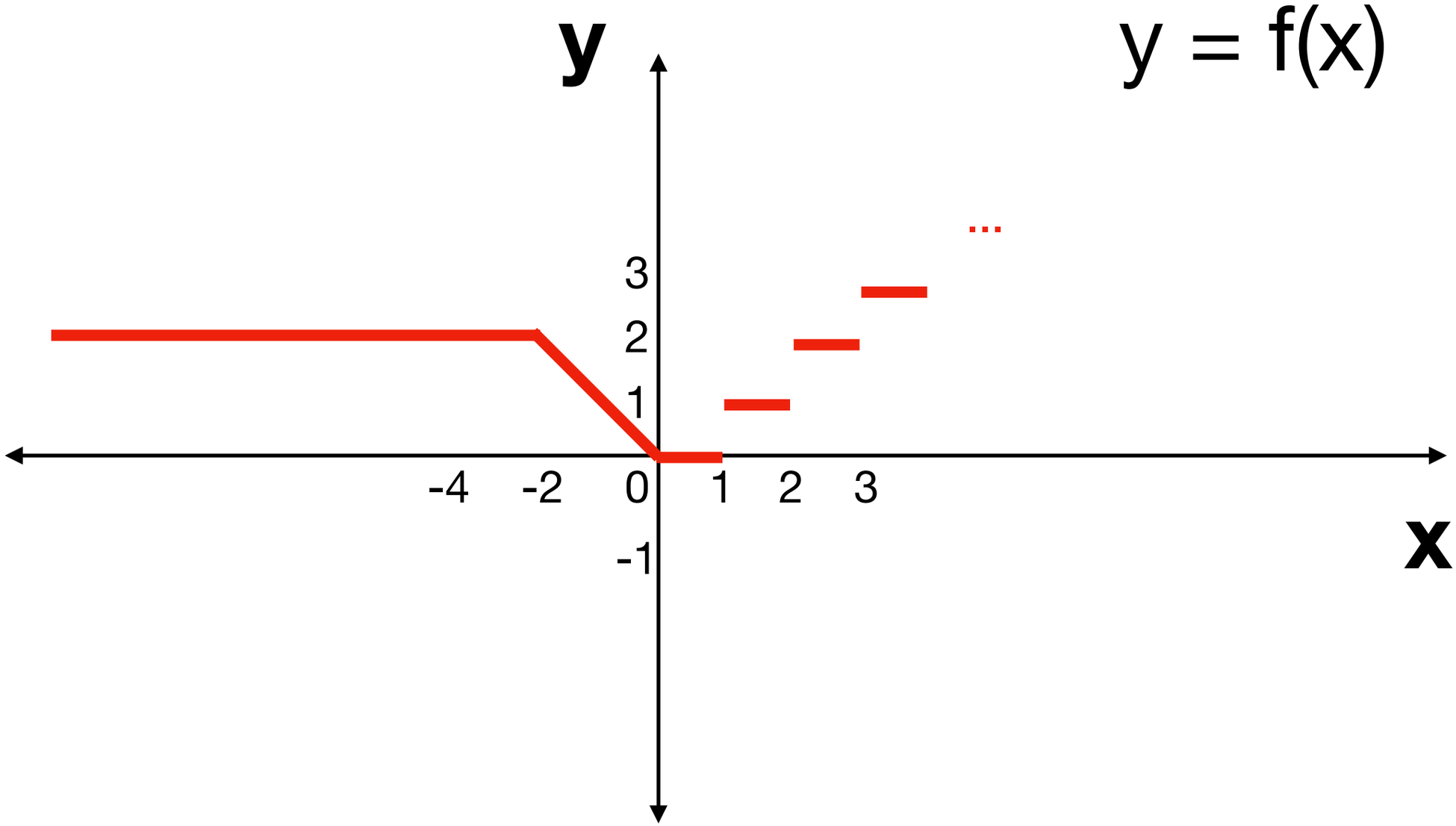


Demo: Absolute

compare 4 ways to compute the absolute of a number
(step through in Interactive Exercises)

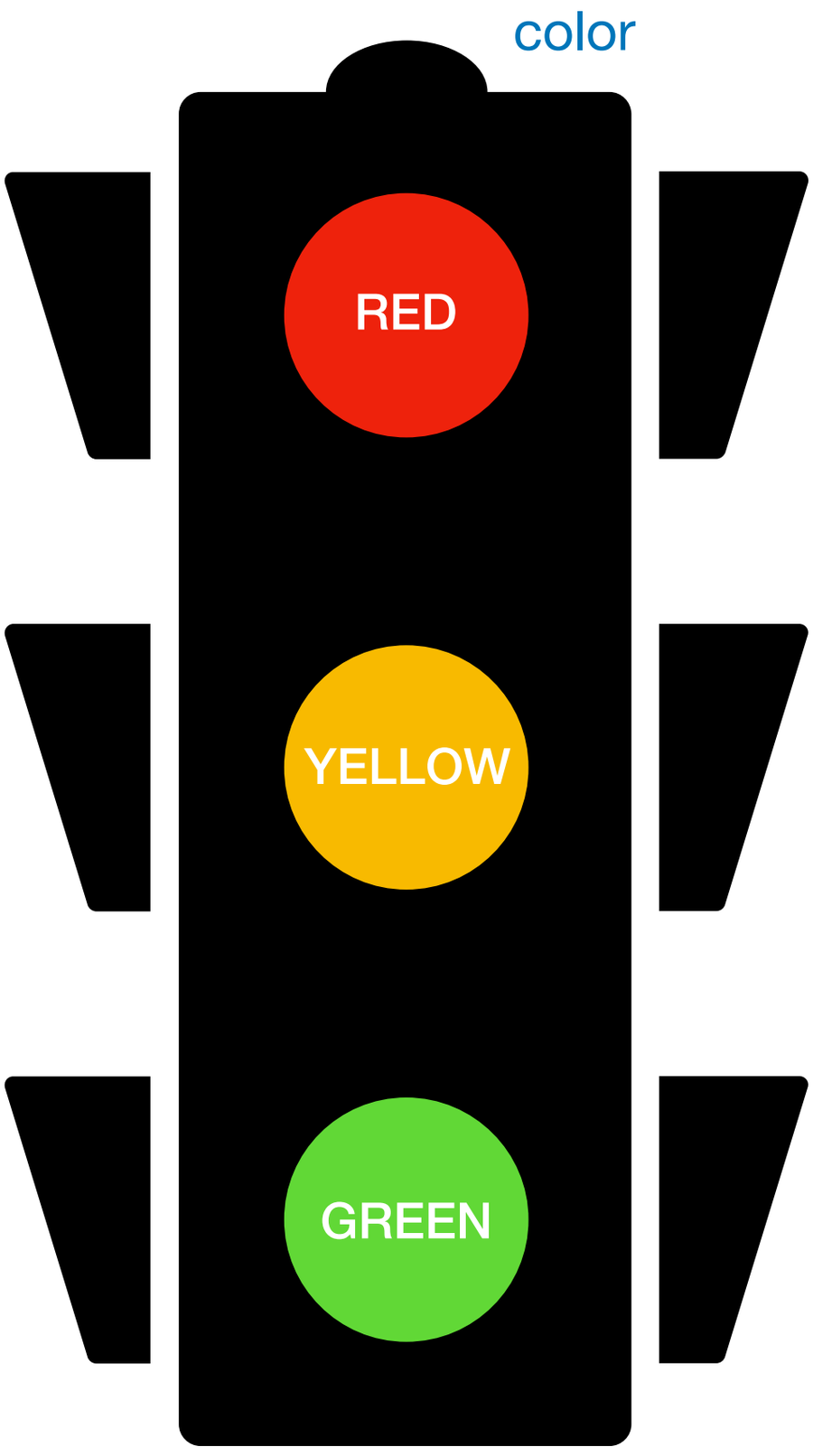


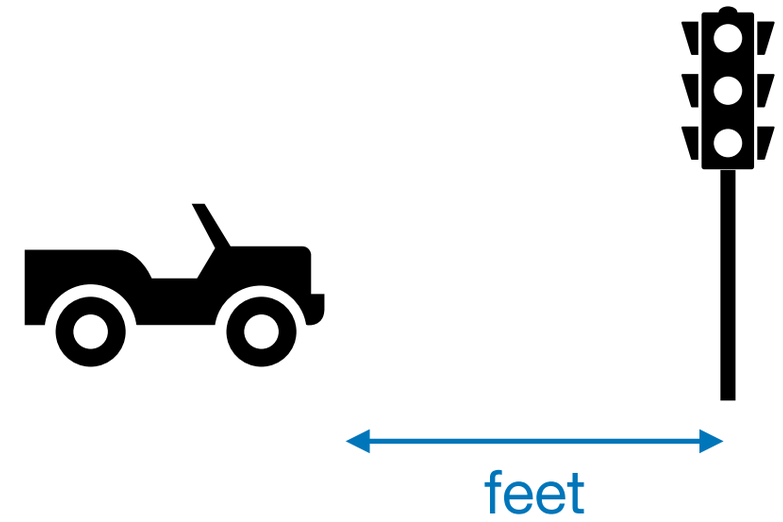
Demo: Piecewise Function



Implement the f function in Python

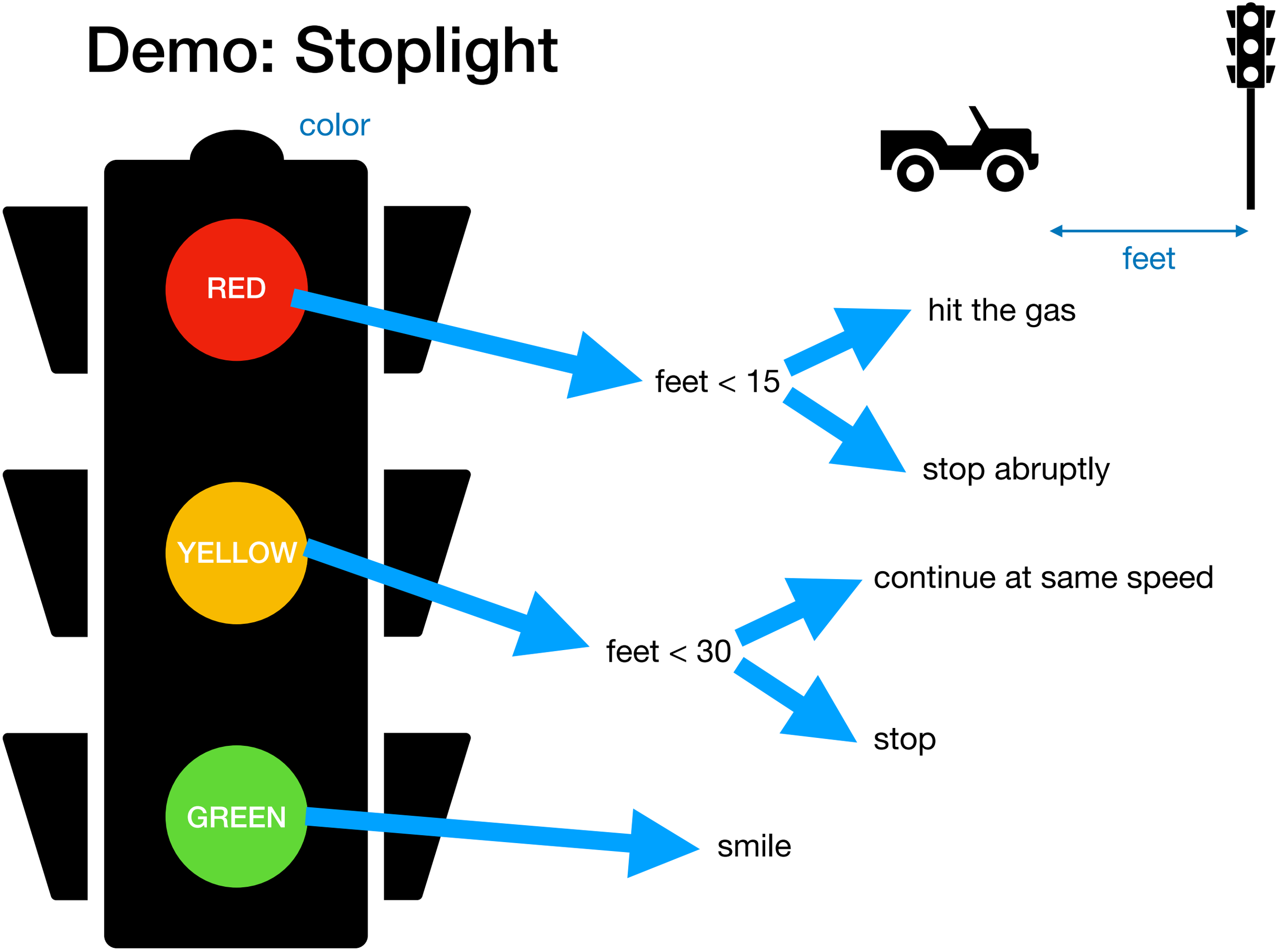
Demo: Stoplight





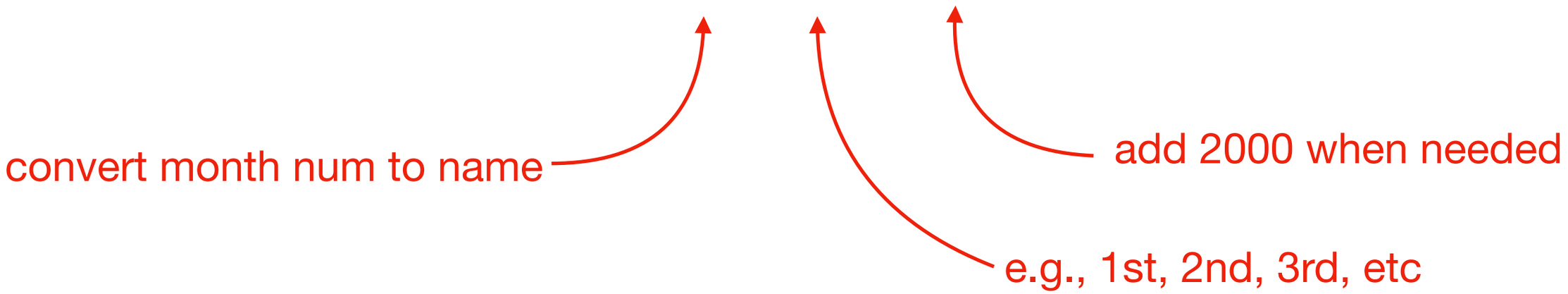
what should the driver do?

Demo: Stoplight

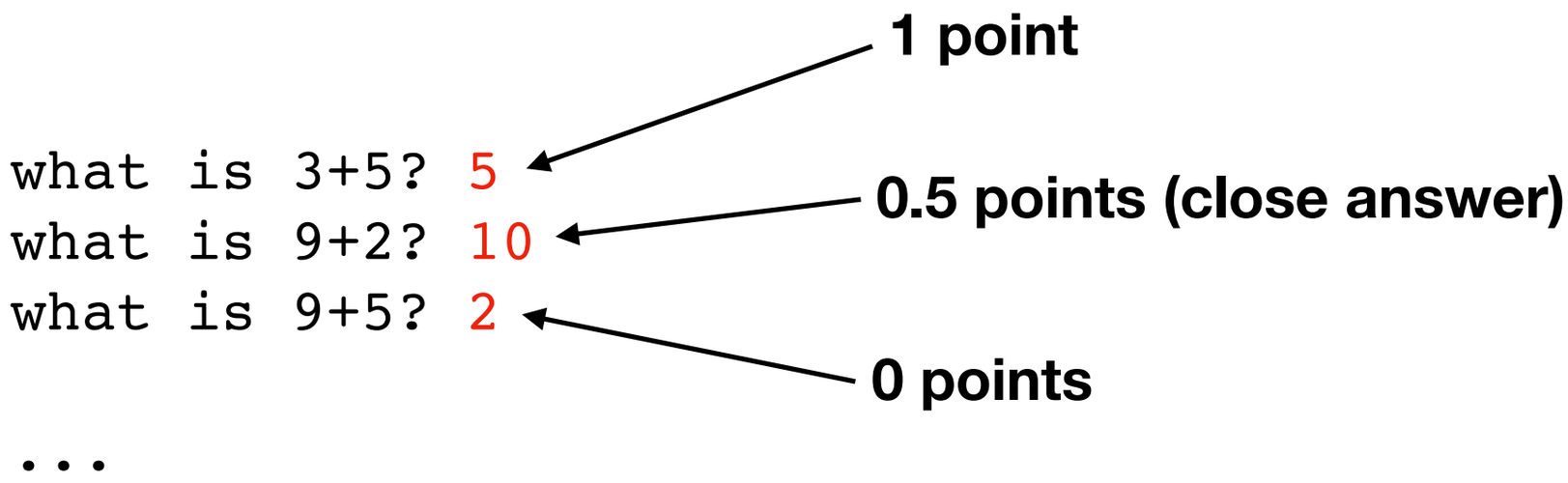


Demo: Date Printer

```
please enter a year: (YYYY): 18
please enter a month (1-12): 2
please enter a day (1-31): 3
the date is: Feb 3rd, 2018
```



Demo: Addition Tester

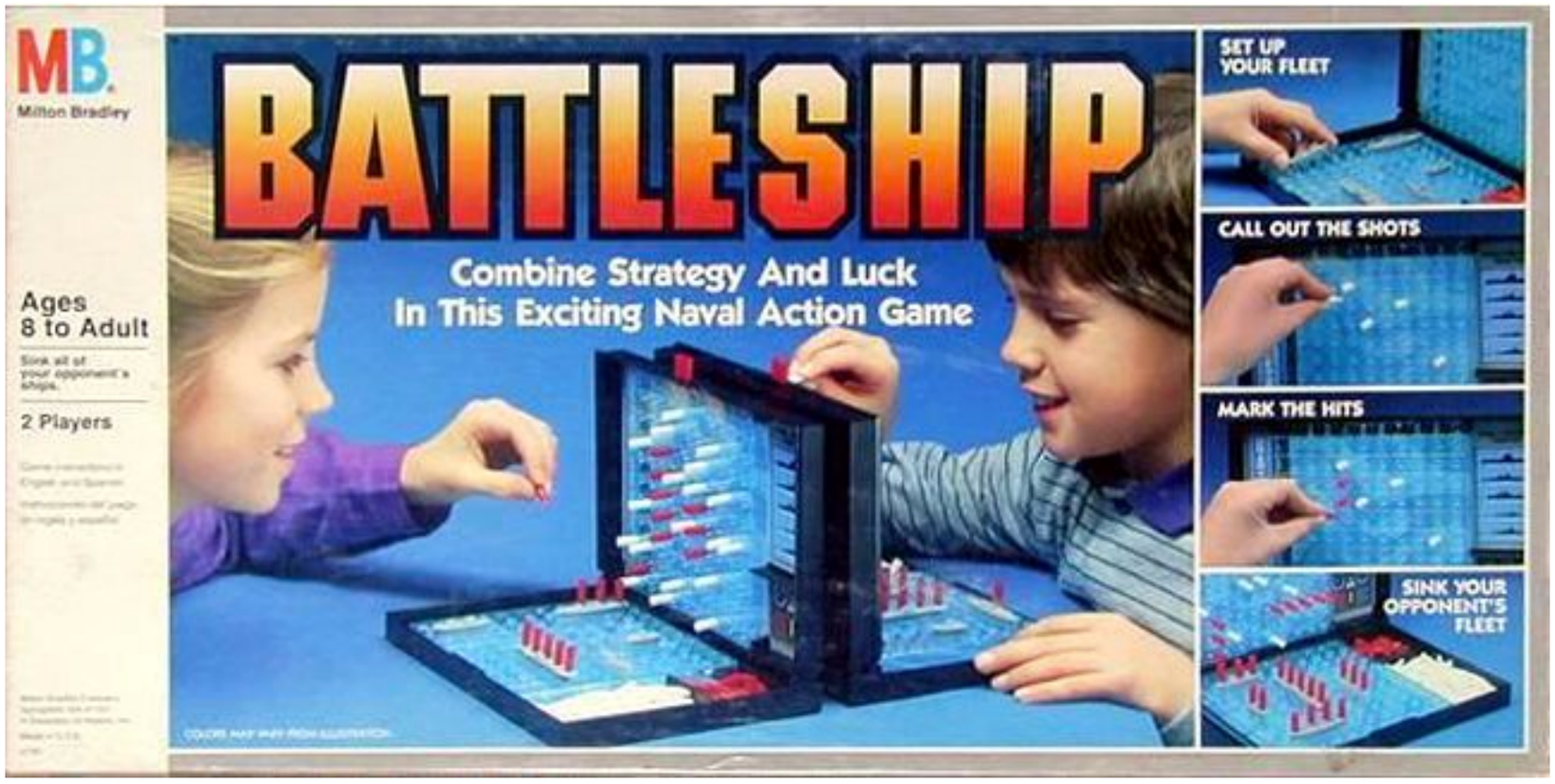



```
Your score is 6.5 of 10
```

We can get random number by using the random module:

```
random.randint(1, 10)
```

Demo: Better Battleship



Improvements

- give more meaningful feedback (not "True" or "False")
- check that user guessed in a reasonable range
- choose random placement for two ships, not overlapping
- show different symbols depending on which ship was hit
- give user up to 3 guesses (or until they get a hit)