[220 / 319] Conditionals

Department of Computer Sciences
University of Wisconsin-Madison

Readings:
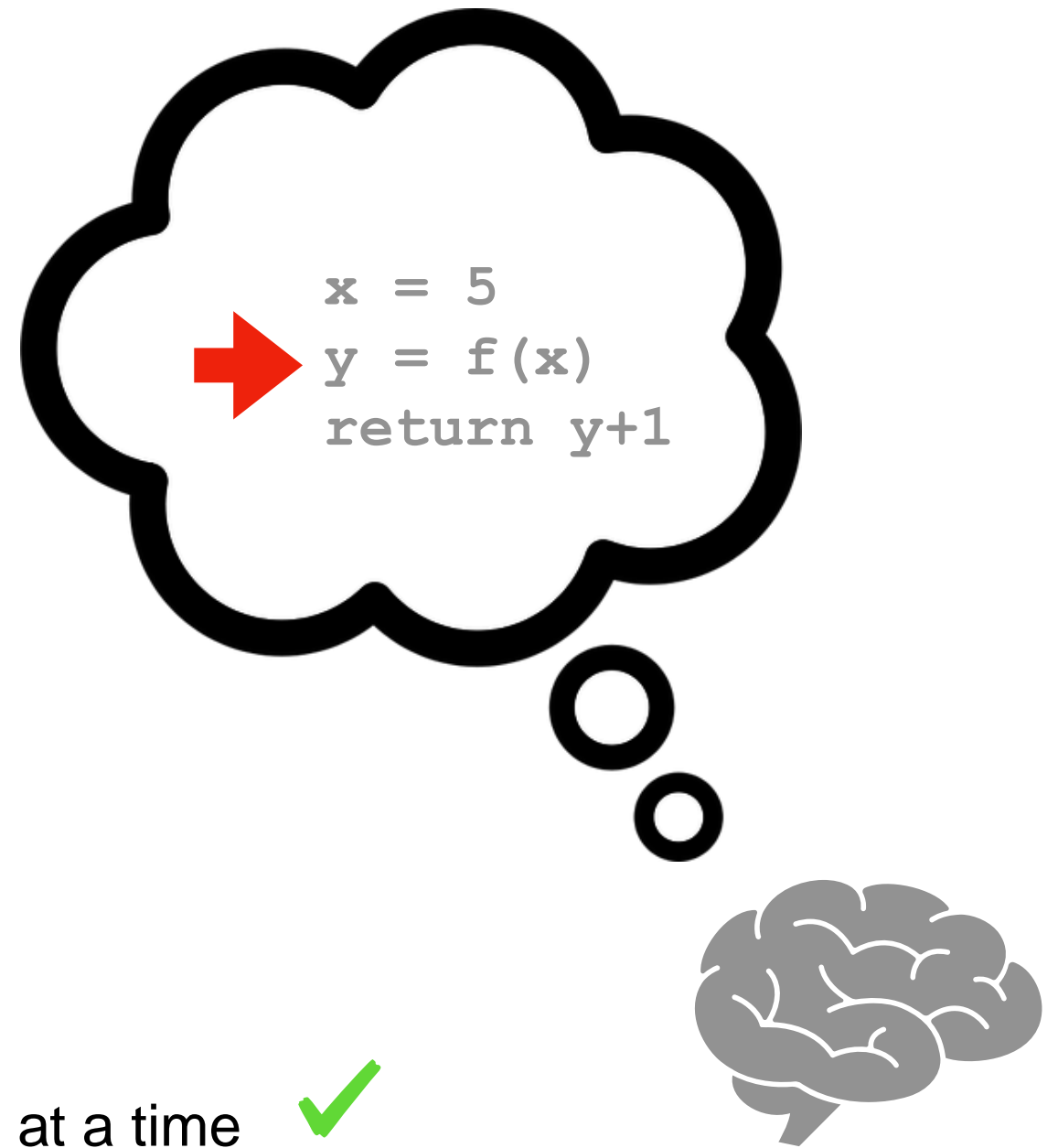Parts of Chapter 5 of Think Python
Chapter 5.5 to 5.8 of Python for Everybody

Due: Quiz2

Mental Model of Control Flow

Code:

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


x = 5
➡ y = f(x)
return y+1

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

three exceptions

Learning Objectives Today

Write conditional statements
- Conditional execution (if)
- Alternate execution (else)
- Chained conditionals (elif)

Chapter 5 of Think Python
(skip "Recursion" sections)
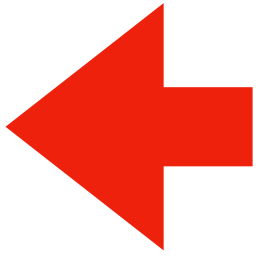
Do PythonTutor Practice!
(posted on schedule)

Determine the output of conditional statements

Identify nested code blocks
- Count the number of blocks in a segment of code

Today's Outline

Review

Control Flow Diagrams

Basic syntax for "if"

Identifying code blocks

Demos

Review 1: Indentation Example

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

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


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


print_letters()
```

Review 1: Indentation Example

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

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

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

print_letters()
```

what does it print?

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")

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

```
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")

print_letters()
```

what does it print?

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")
```

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()
```

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

blank lines are **irrelevant**

```
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()
```

we'll often call the lines of code **inside** something a **"block"** of code

A
B
E
F
C
D

Review 1: Indentation Example

```
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

```python
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)
```
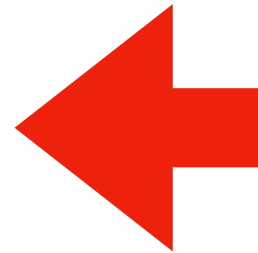
Today's Outline

Review

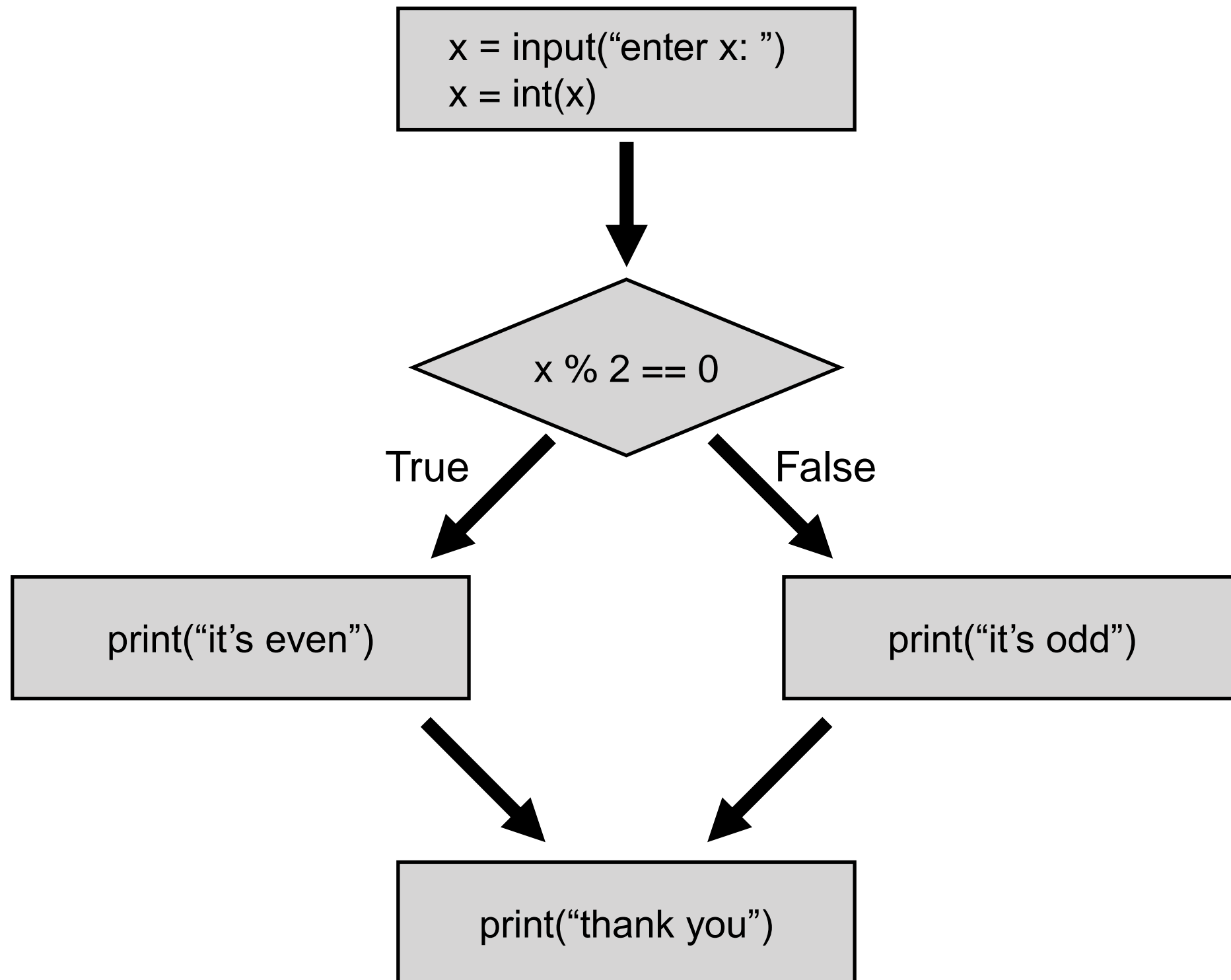Control Flow Diagrams ⬅
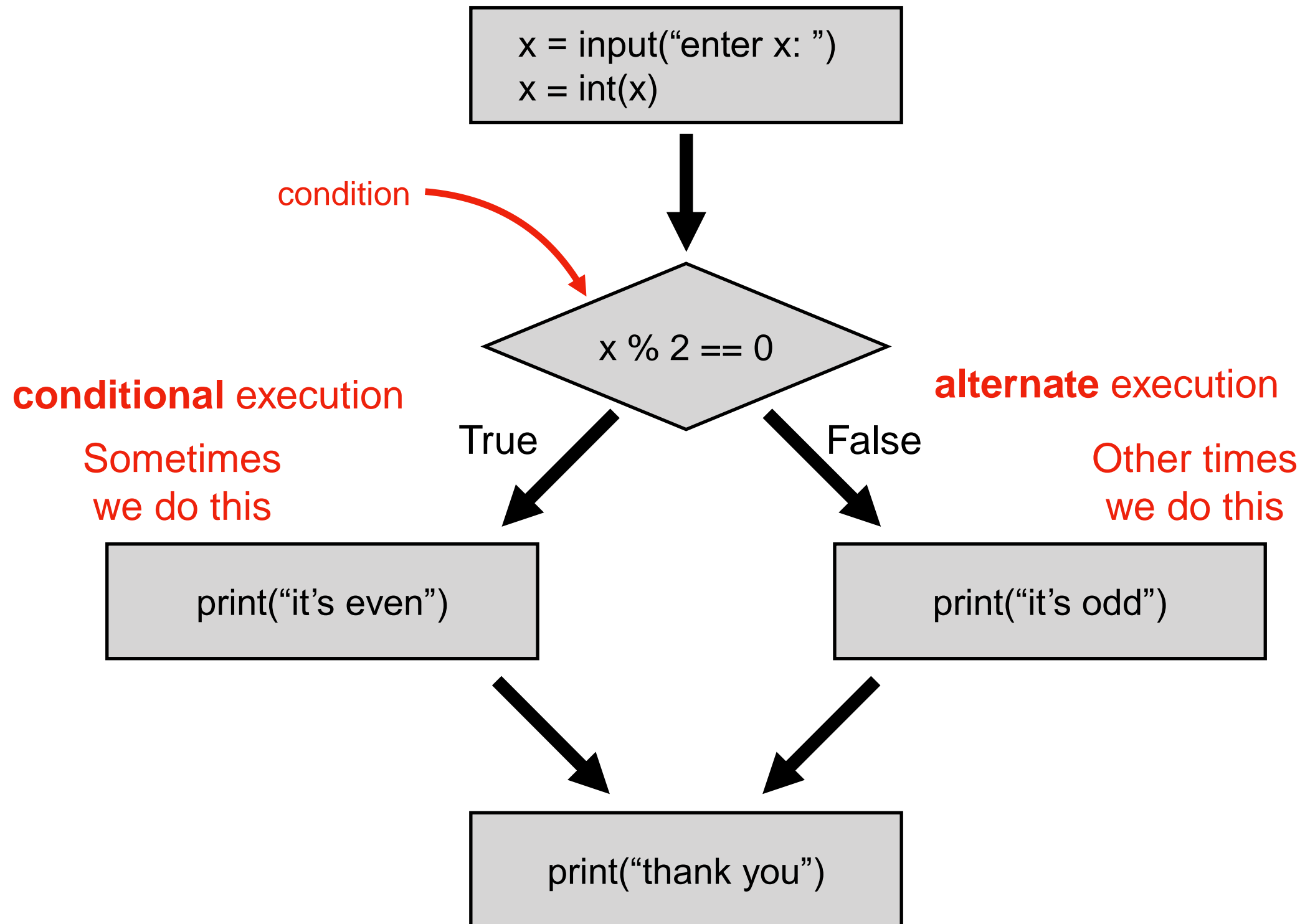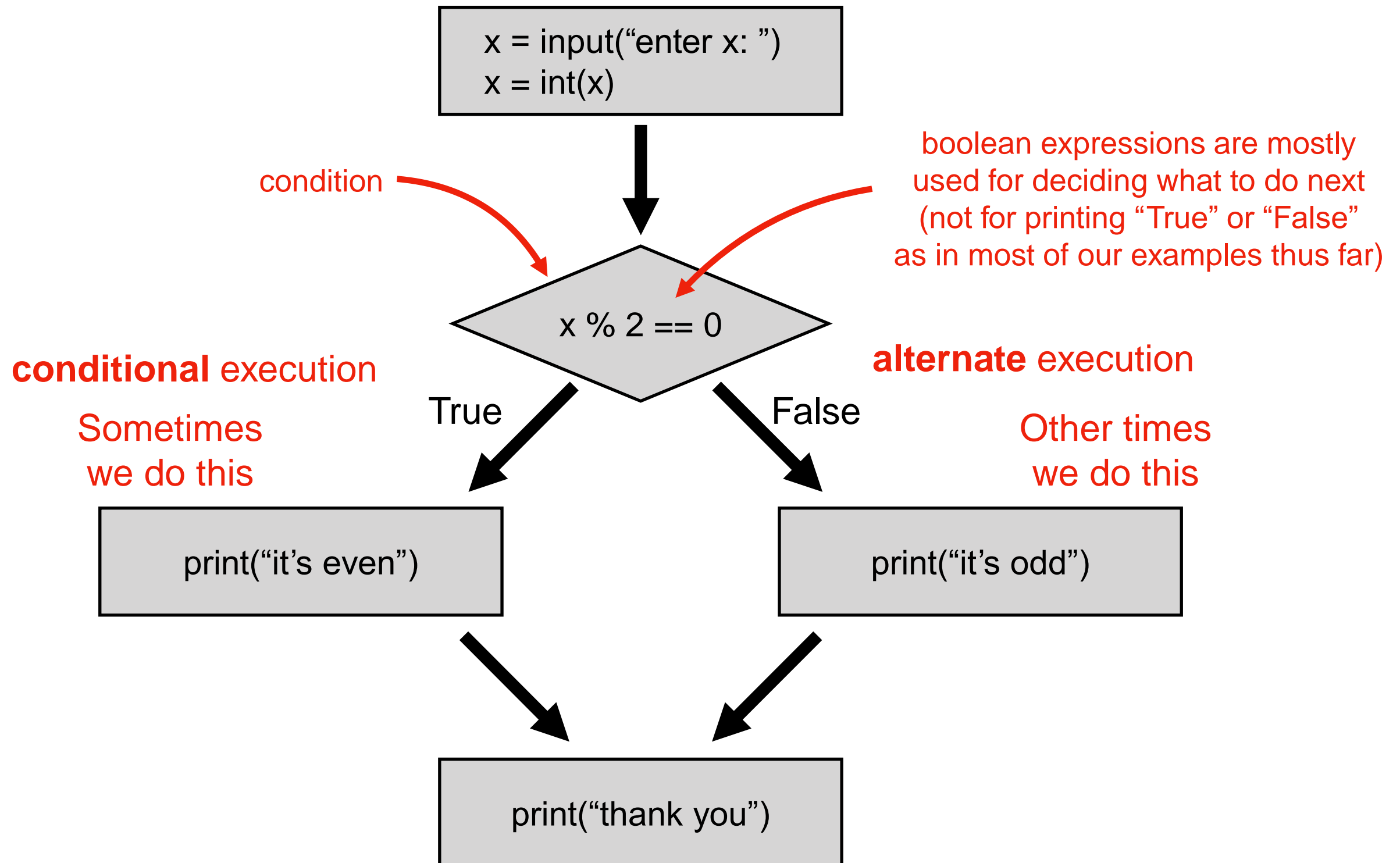
Basic syntax for "if"

Identifying code blocks

Demos

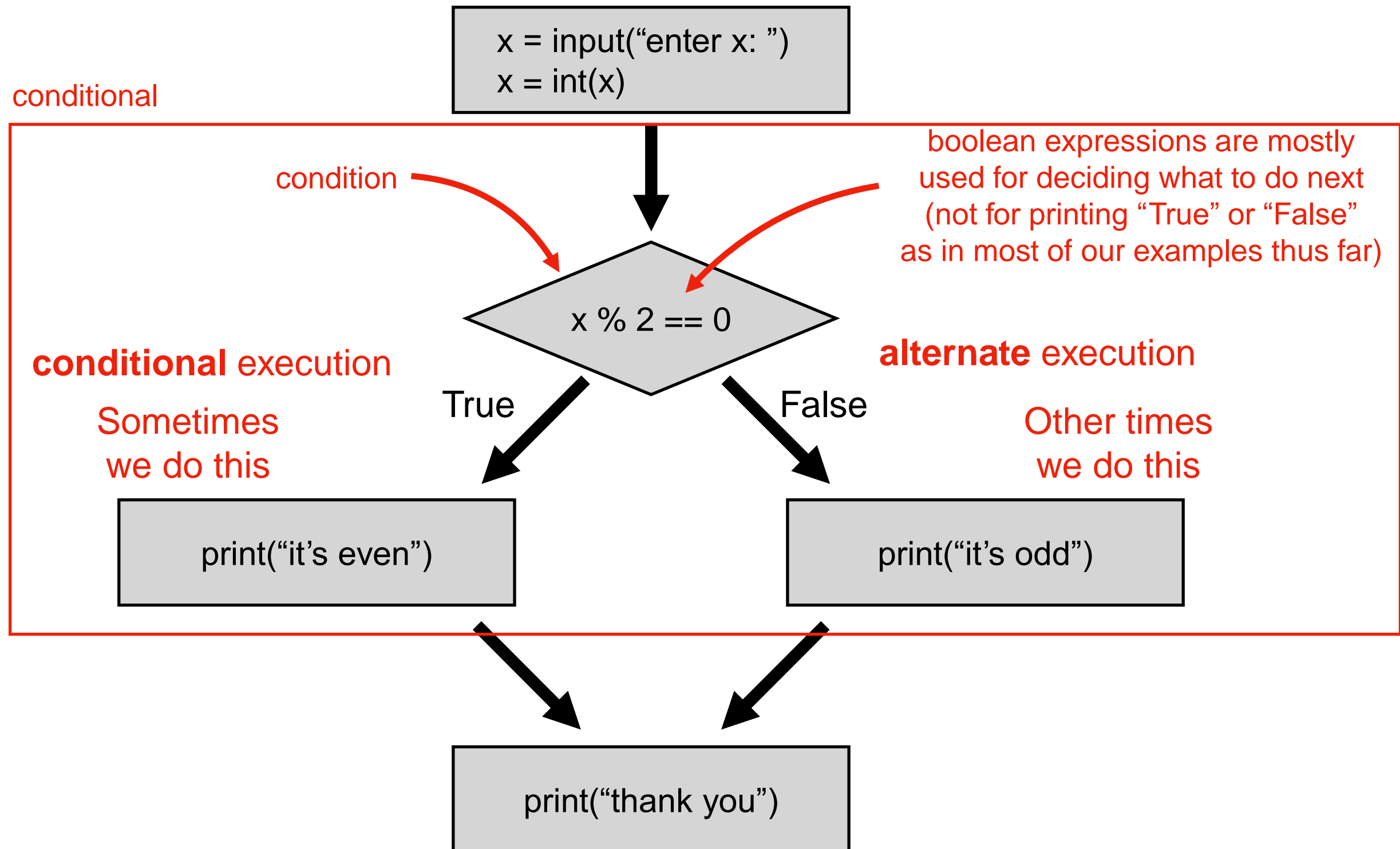# Control Flow Diagrams (Flowcharts for Code)

Control Flow Diagrams (Flowcharts for Code)
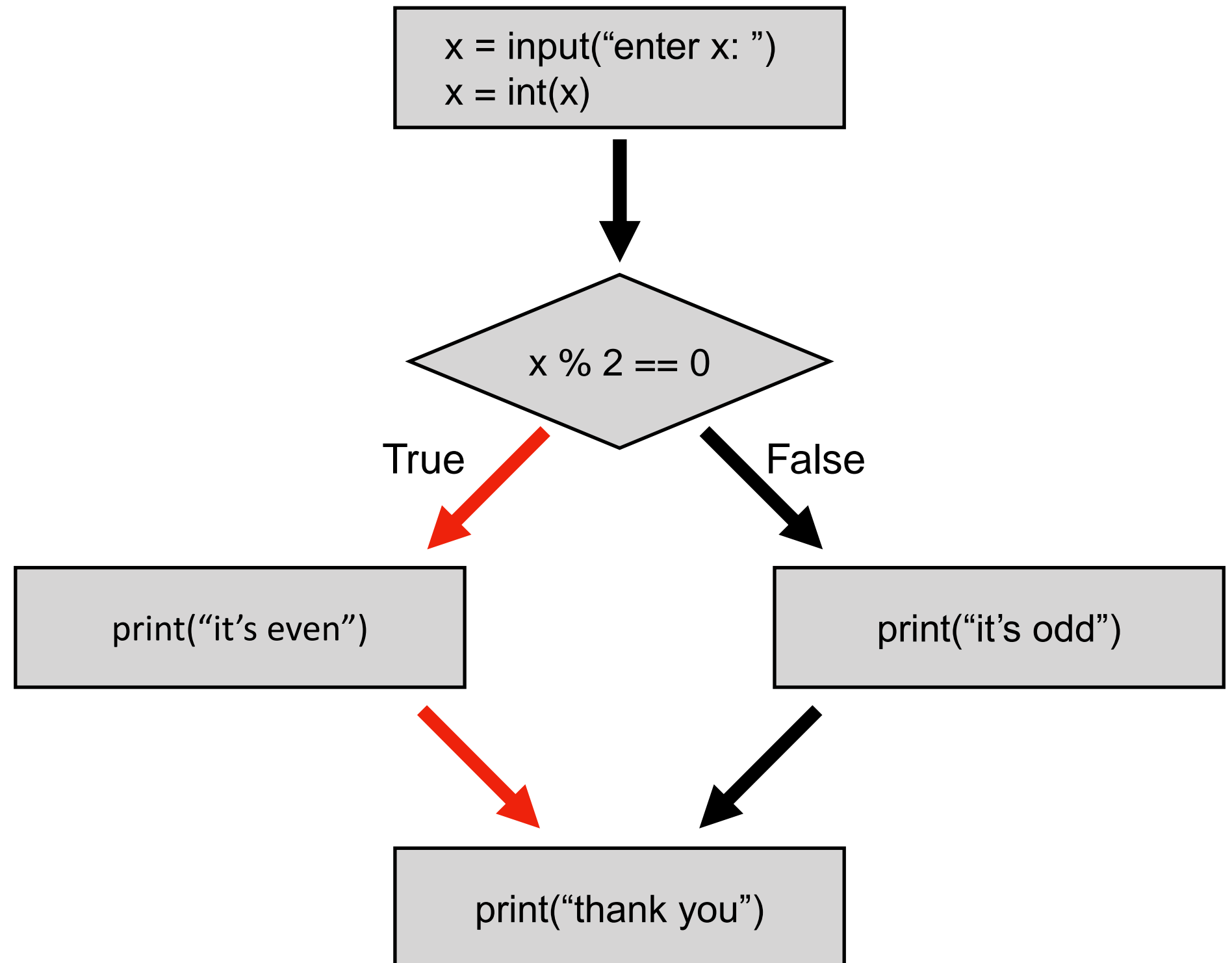
Control Flow Diagrams (Flowcharts for Code)

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

condition

boolean expressions are mostly used for deciding what to do next (not for printing "True" or "False" as in most of our examples thus far)

x % 2 == 0

conditional execution

alternate execution

True

False

Sometimes we do this

Other times we do this

```
print("it's even")
```

```
print("it's odd")
```

```
print("thank you")
```

Control Flow Diagrams (Flowcharts for Code)

conditional

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

boolean expressions are mostly used for deciding what to do next (not for printing "True" or "False" as in most of our examples thus far)

condition

x % 2 == 0

conditional execution

Sometimes we do this

True

False

alternate execution

Other times we do this

print("it's even")

print("it's odd")

print("thank you")

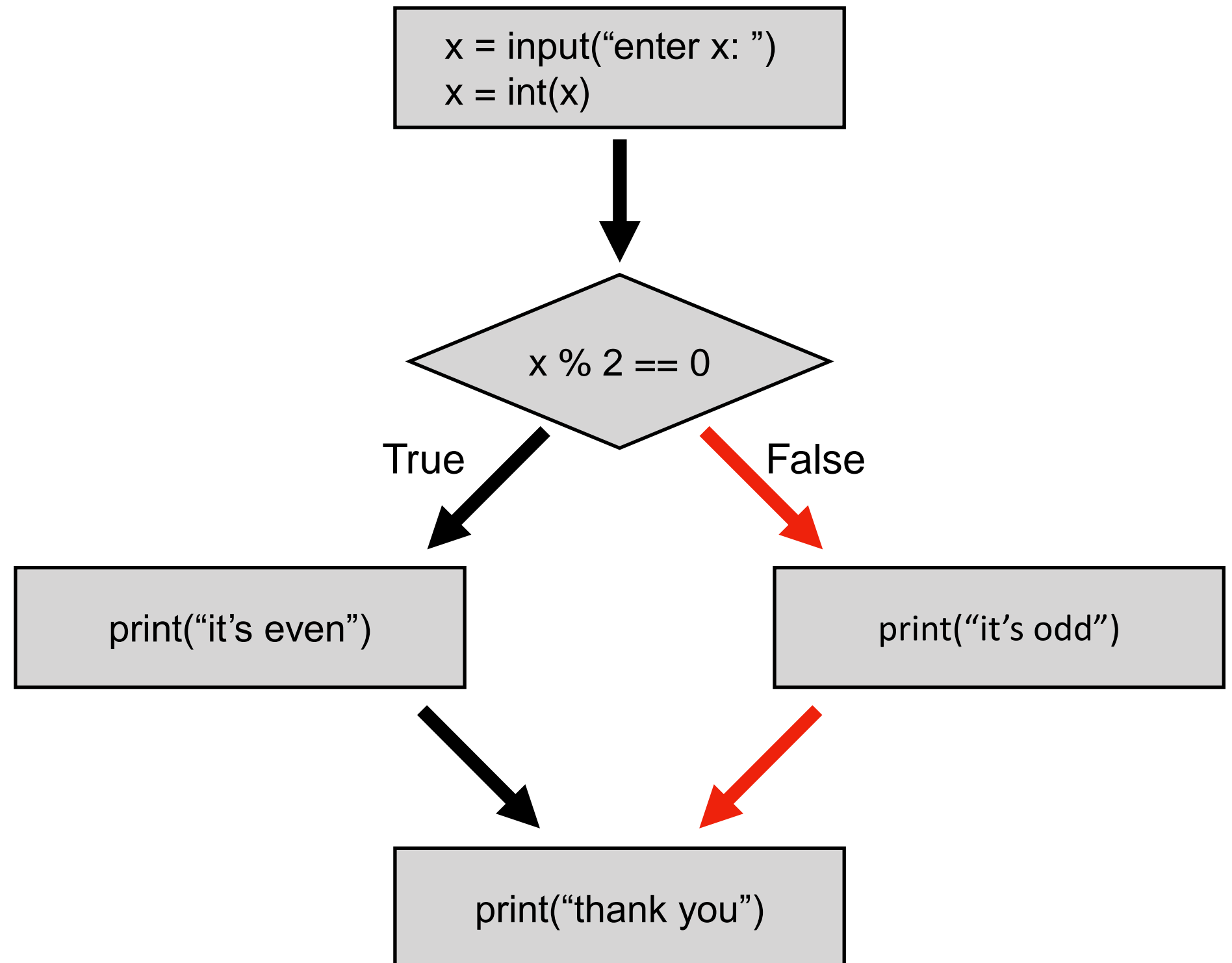# Branches (aka "Paths of Execution")

Input/Output:

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

Branches (aka "Paths of Execution")

Input/Output:

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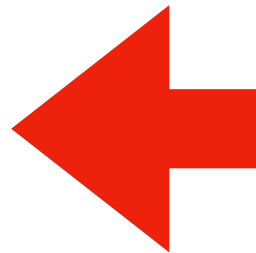
Today's Outline

Review

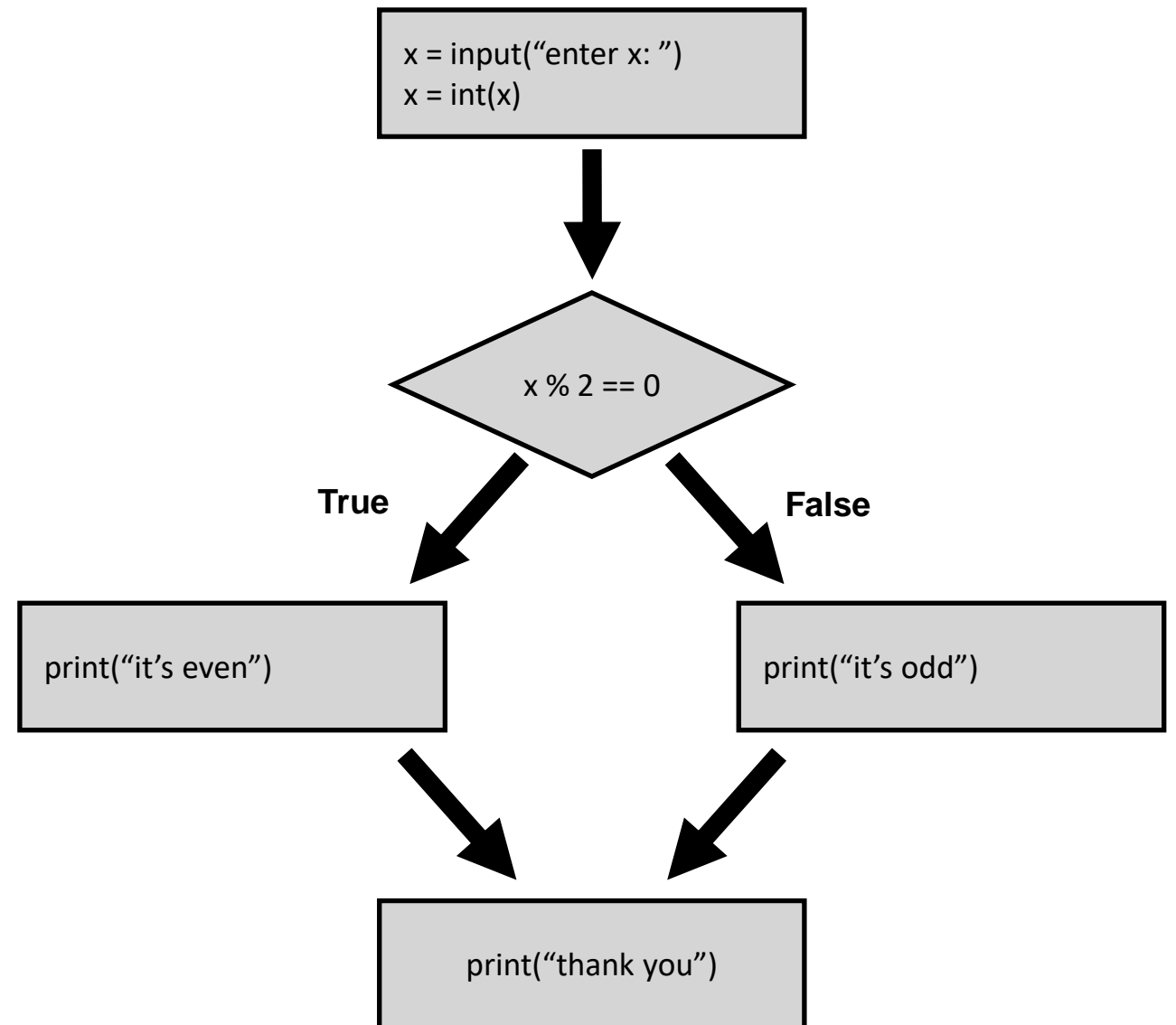Control Flow Diagrams

Basic syntax for "if"

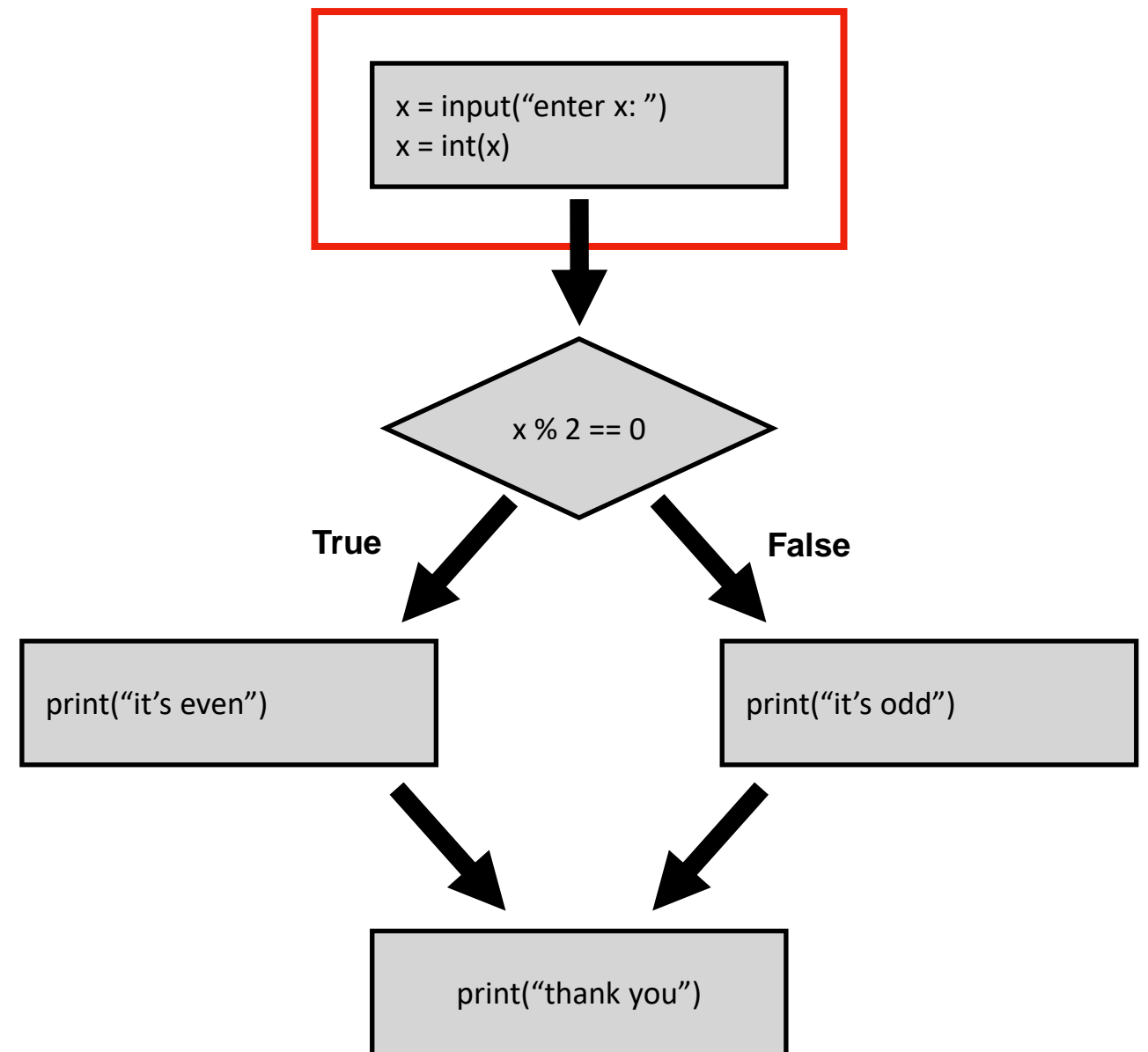Identifying code blocks

Demos

Writing conditions in Python

Code:

Writing conditions in Python

Code:

```python
x = input("enter x: ")
x = int(x)
```
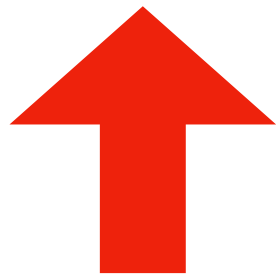
Writing conditions in Python

Code:

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

if x % 2 == 0:
```
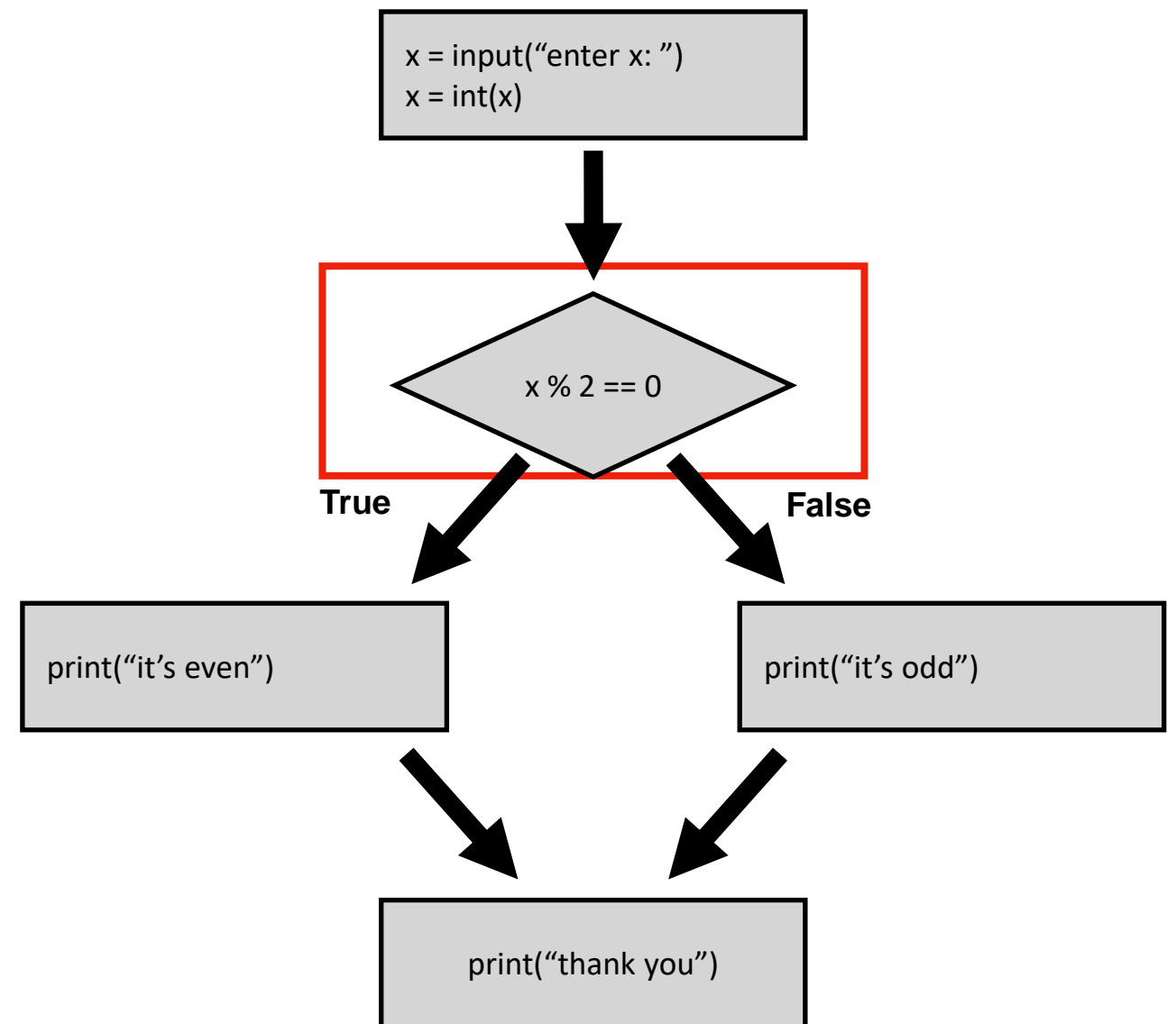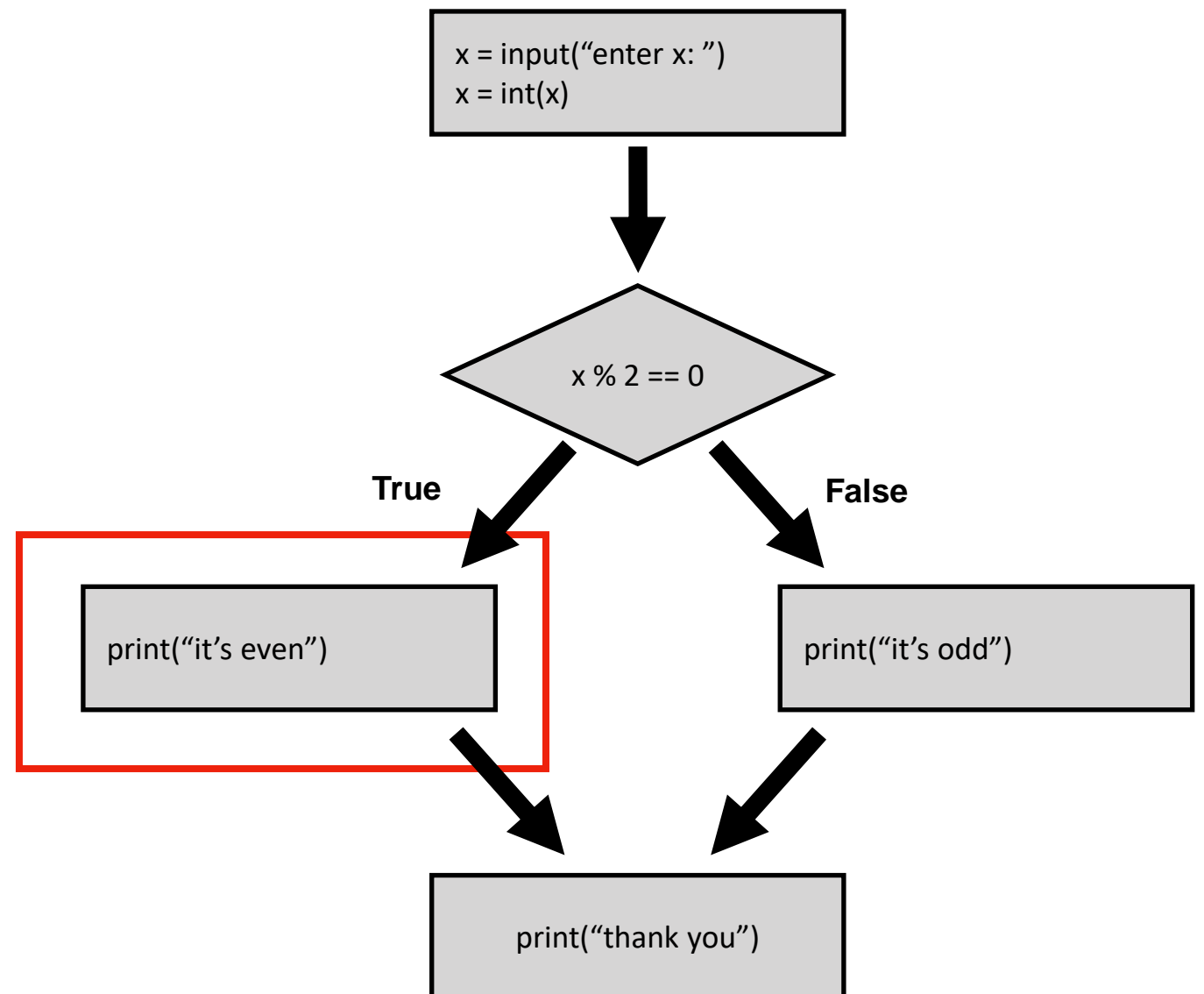
boolean expression

Writing conditions in Python

Code:

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

if x % 2 == 0:
    print("it's even")
```
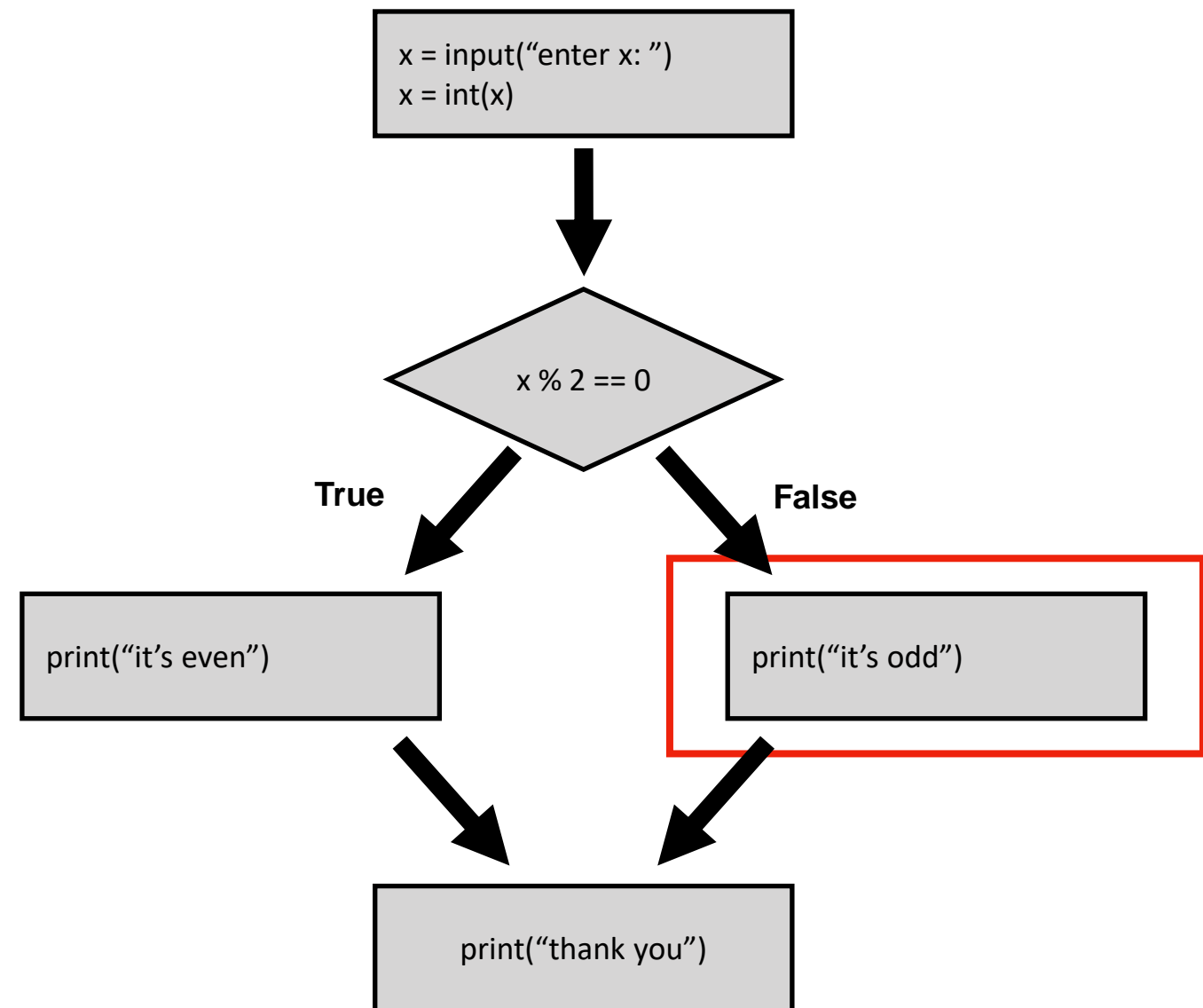
Writing conditions in Python

Code:

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

if x % 2 == 0:
    print("it's even")
else:
    print("it's odd")
```

colons will *almost* always be followed by a tabbed new line
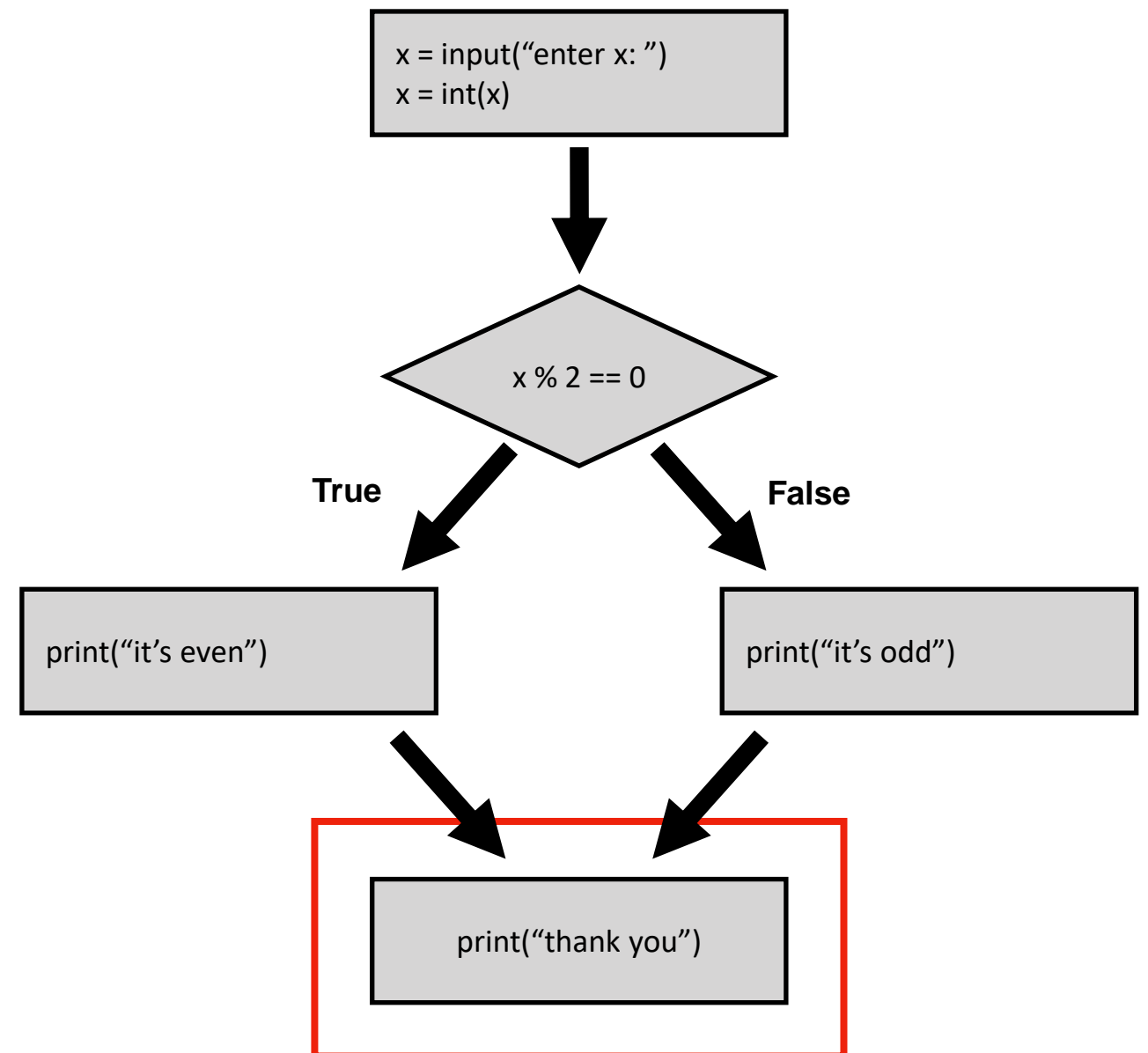
Writing conditions in Python

Code:

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

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

print("thank you")
```

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

```
x % 2 == 0
```

True   **False**

```
print("it's even")
```

```
print("it's odd")
```

```
print("thank you")
```

Writing conditions in Python

Code:

```python
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")
```
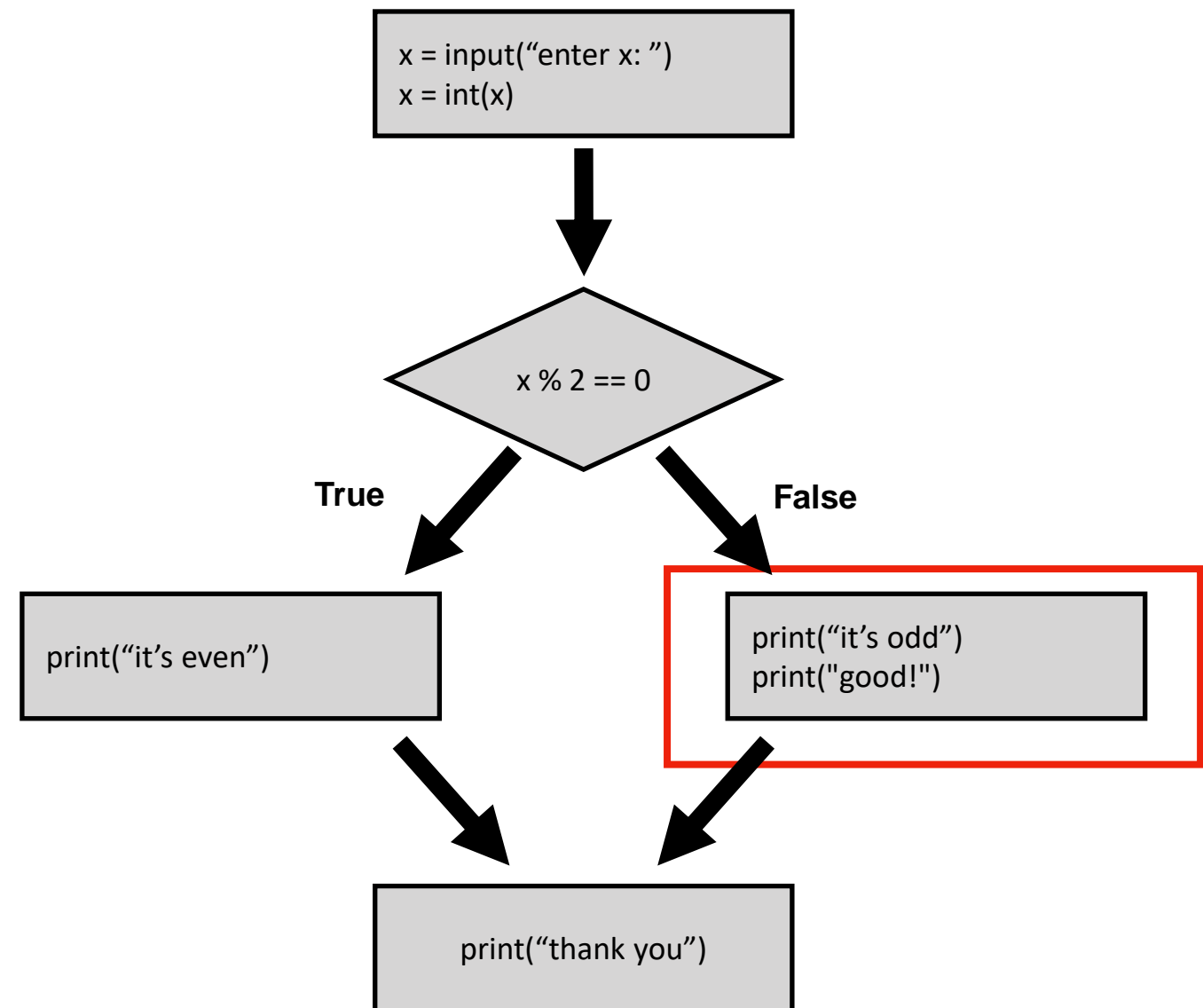
Writing conditions in Python

Code:

```python
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")
```

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

```
x % 2 == 0
```

True   **False**

```
print("it's even")
print("we wanted odd")
```

```
print("it's odd")
print("good!")
```
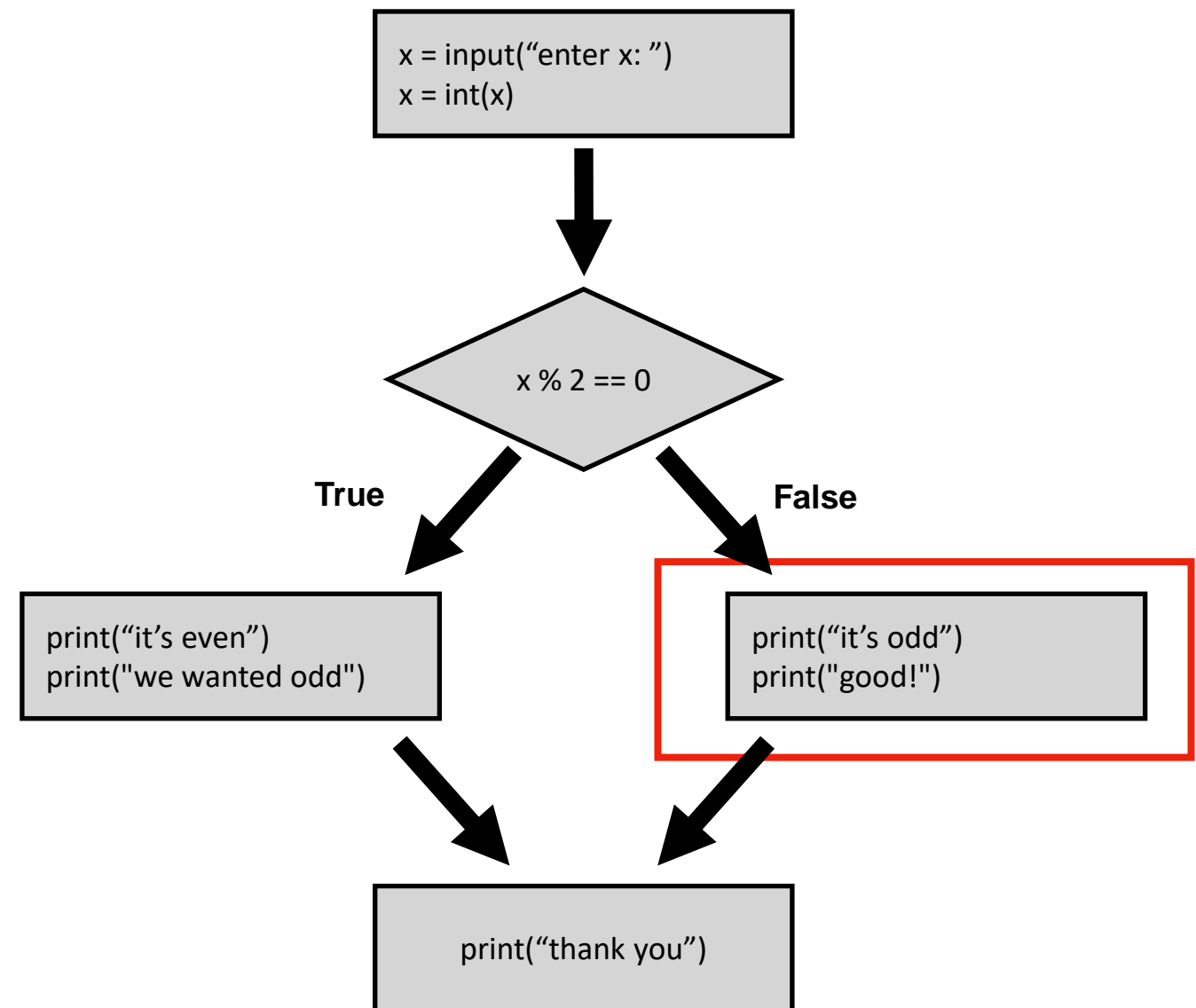
```
print("thank you")
```

Writing conditions in Python

Code:

```python
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")
```
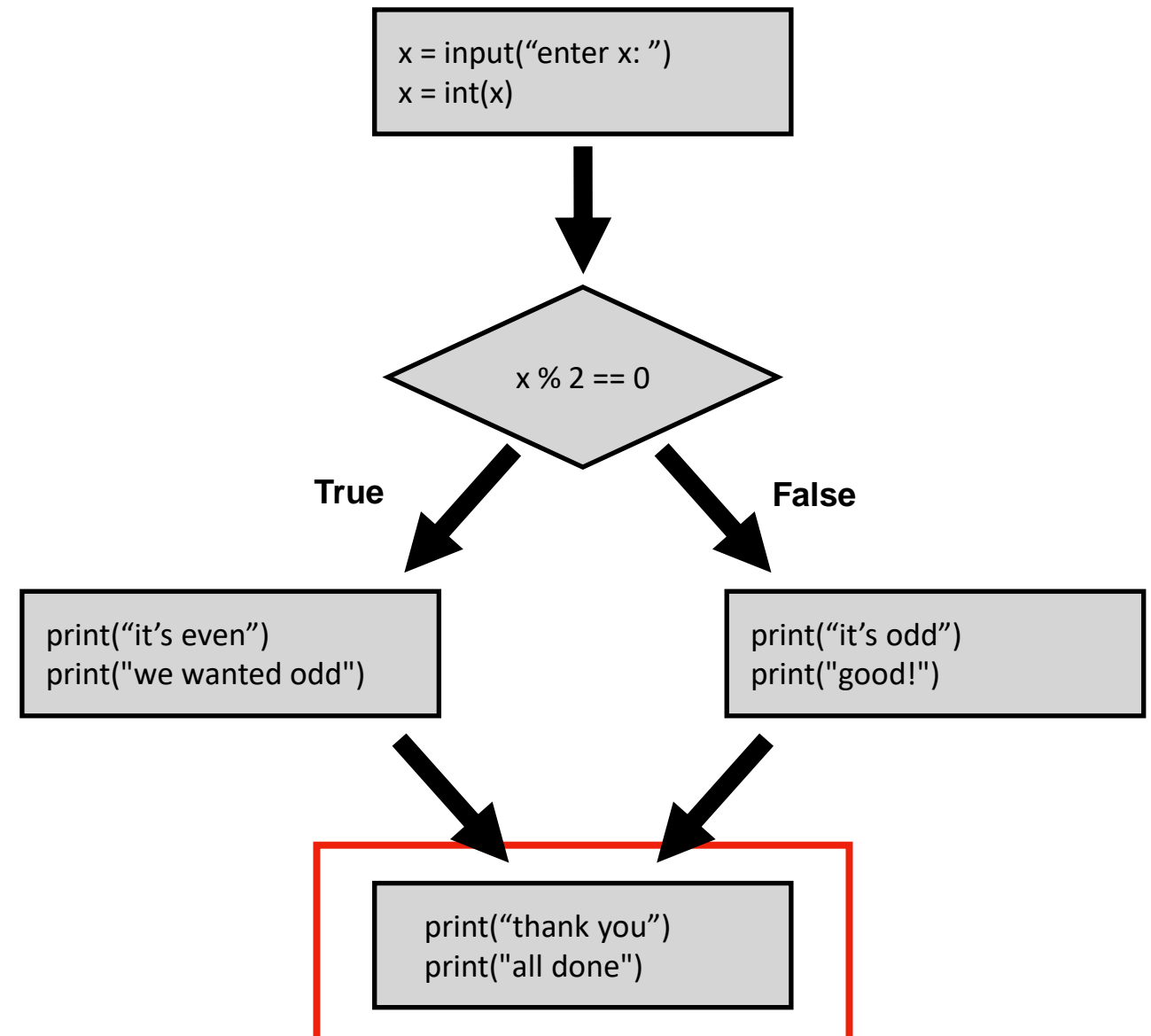
block of code inside "if"

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")
```
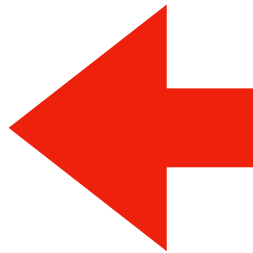
block of code inside "if"

block of code inside "else"

What if all this were inside a function?

Code Blocks

You need to get good at "seeing" code blocks in Python code.
Even blocks inside blocks inside blocks...

Code:

```python
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()
```

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

Identifying Code Blocks
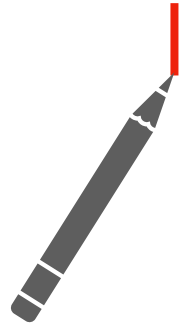
Code:

```
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:

```python
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()
```
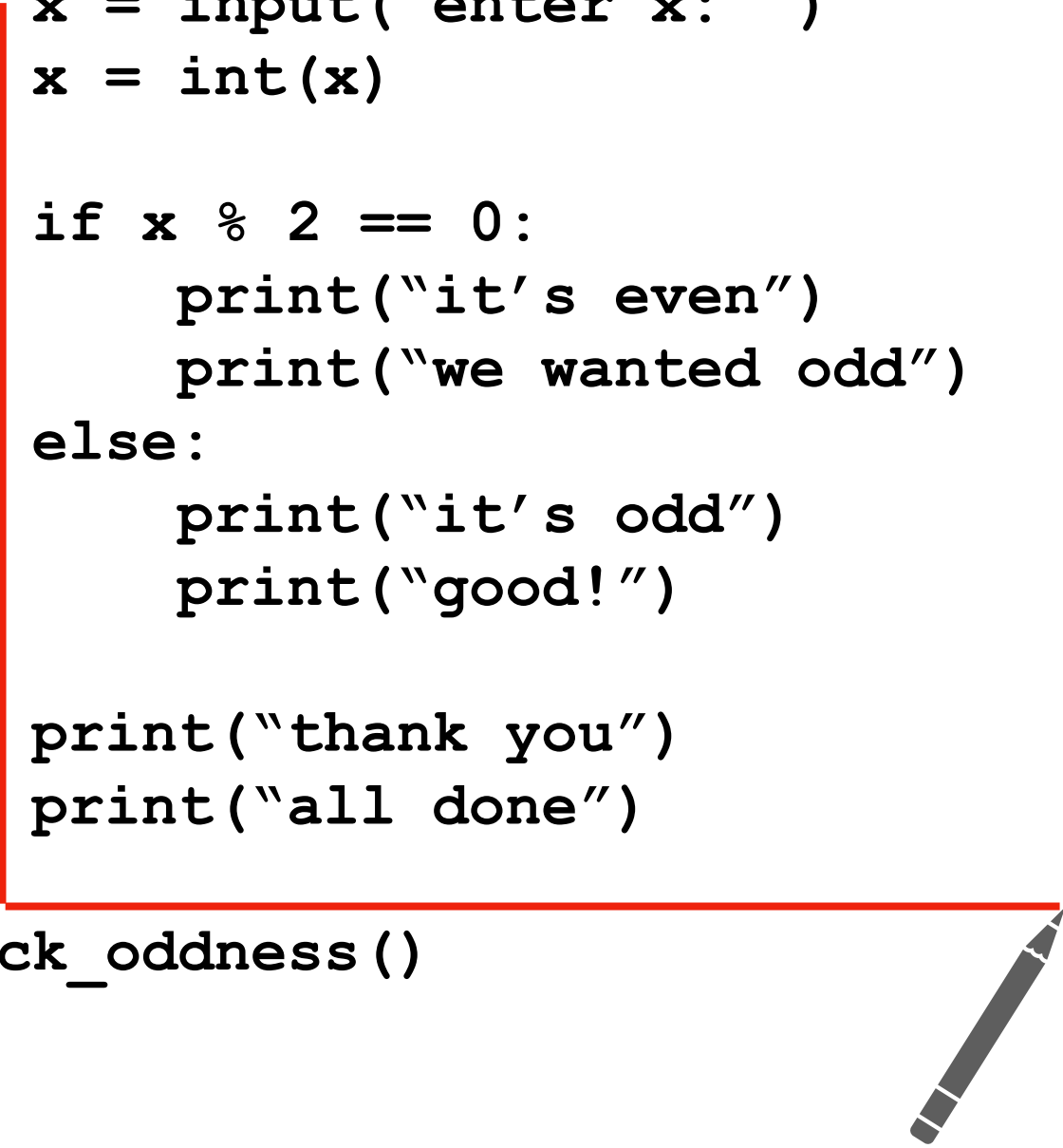
Step 4: box off the code

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:

```python
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,
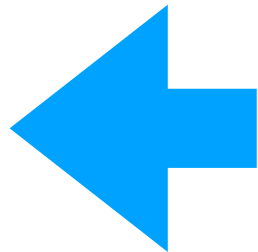look for the next colon
and repeat

Today's Outline

Review

Control Flow Diagrams

Basic syntax for "if"

Identifying code blocks

Demos

Example: Classifying Children by Age

What are all the different ways to classify children?
If you are 3 years old you are a ……
If you are 15 years old you are a ……

Write a function that is given an int and returns a string

```
def categorize_age(age):
    if age <= …..
        return 'baby'
```

Example: Date Printer

```
please enter a year: (YYYY): 2022
please enter a month (1-12): 2
please enter a day (1-31): 11
the date is: Feb 11th of '22
```

convert month num to name

'2-digit year

e.g., 1st, 2nd, 3rd, etc