[220 / 319] Iteration

Meena Syamkumar
Andy Kuemmel

- **Exam I next Friday**
- **Exam Conflict Form**

Learning Objectives Today

Reason about loops

- Motivation: need for repetition
- Condition and body of loop
- “while” syntax
- Hand-trace looping algorithms

Chapter 7 of Think Python

Understand common use cases

- Taking input from a user
- Computing over ranges of numbers

Recognize and avoid pitfalls

- Infinite loops (when unintentional)
- Off-by-one mistakes

Worksheet

State:

6

N	4		
total	0	answer	0

Code:

1. Put 1 in the “total” box
2. If “N” equals 1, skip to step 6, otherwise continue to step 3
3. Multiply the value in “total” by the value in “N”, and put the result back in “total”
4. Decrease the value in “N” by 1
5. Go to step 2
6. Copy the value in total to the answer box

Combination of conditionally skipping forward (2) with going back is (5) is called a “while loop”

Worksheet

State:

6

N	4
total	0
answer	0

loop condition

Code:

skip past loop body

continue to loop body

1. Put 1 in the "total" box
2. If "N" equals 1, skip to step 6, otherwise continue to step 3
3. Multiply the value in "total" by the value in "N", and put the result back in "total"
4. Decrease the value in "N" by 1
5. Go to step 2
6. Copy the value in total to the answer box

loop body

going back will be implicit in Python, and will happen right after loop body.
you can identify the loop body because it will be indented

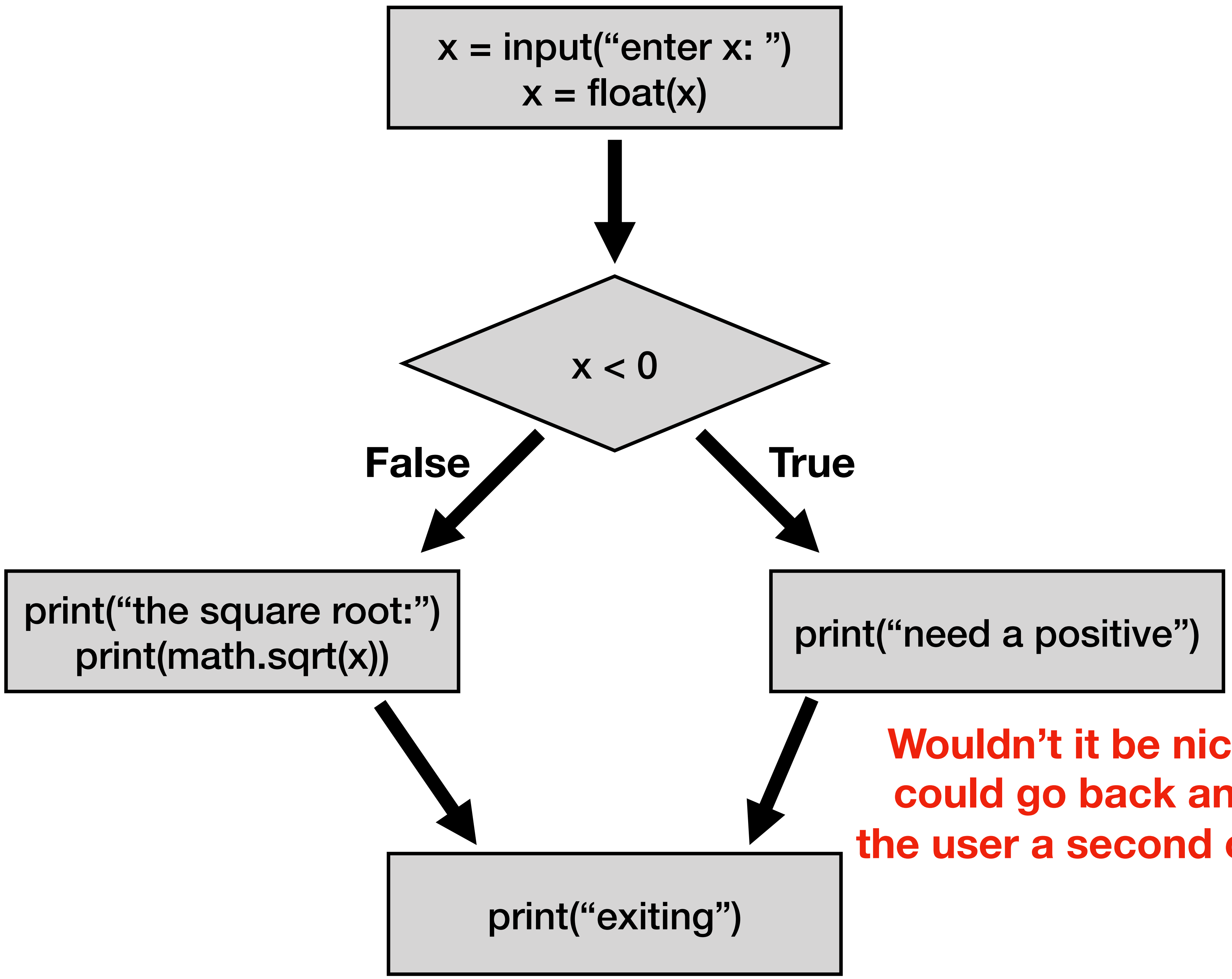
Today's Outline

Control Flow Diagrams

Basic syntax for “while”

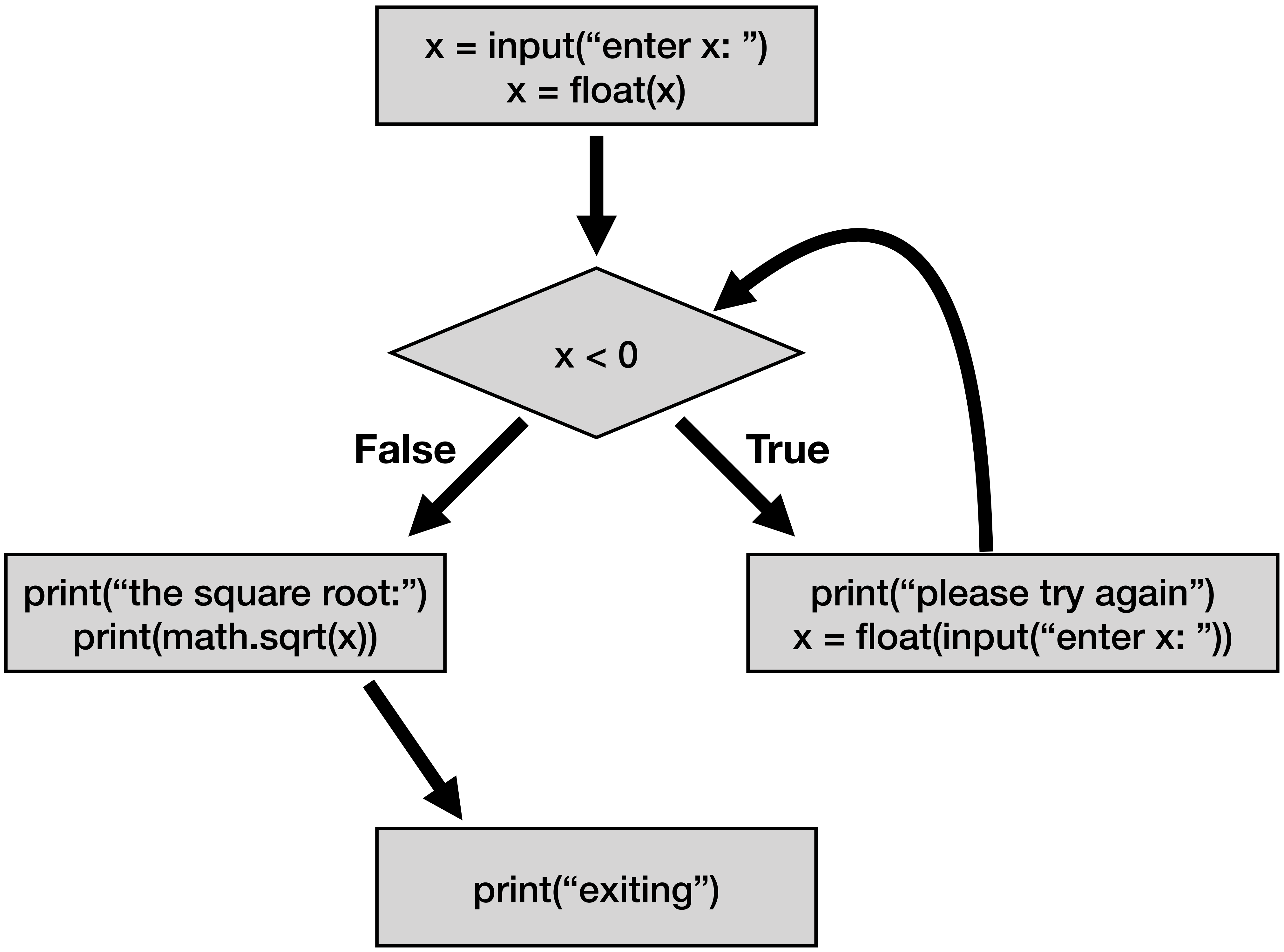
Demos

Control Flow Diagrams: “if”

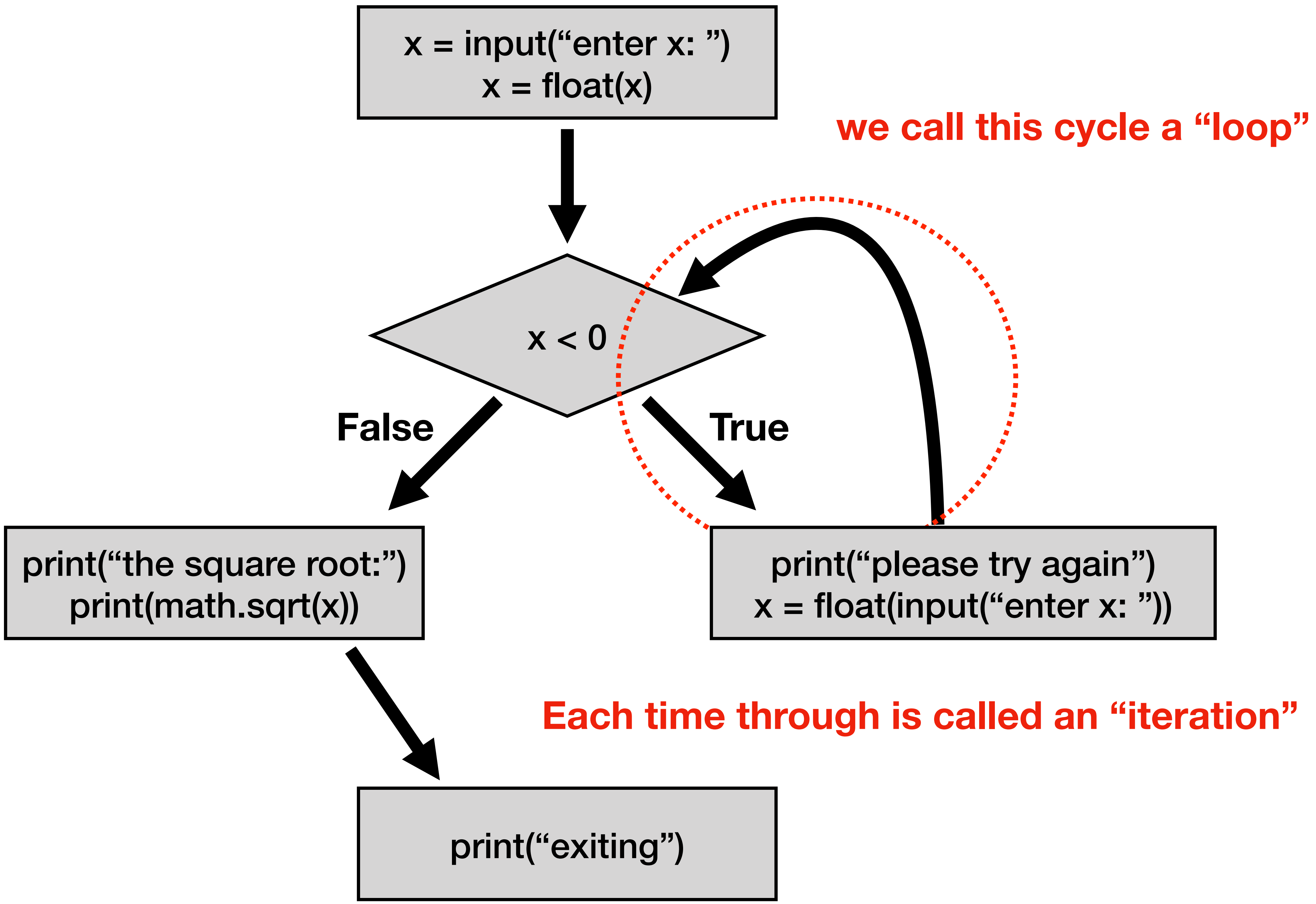


Wouldn’t it be nice if we could go back and give the user a second chance?

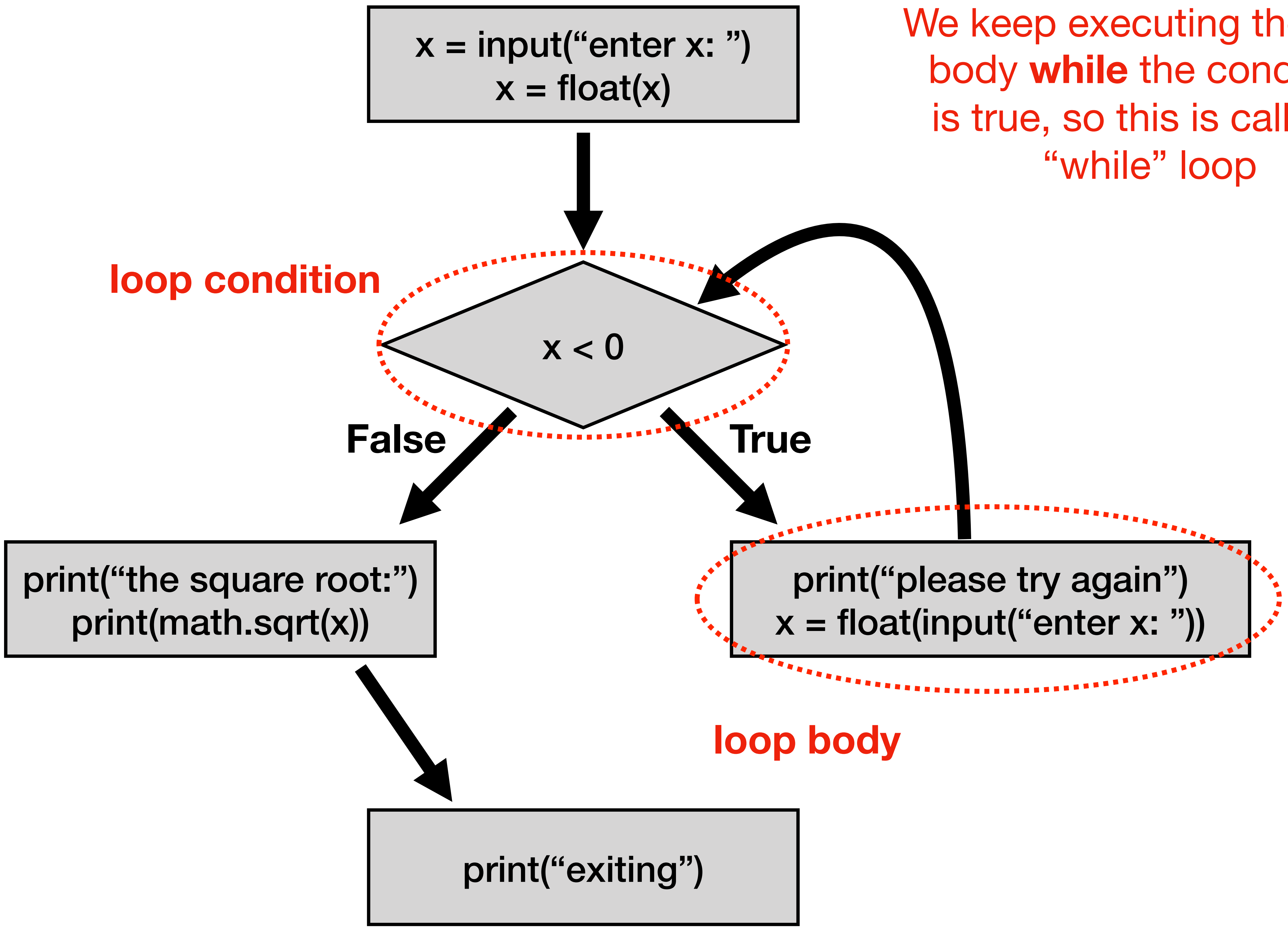
Control Flow Diagrams: “while”



Control Flow Diagrams: “while”

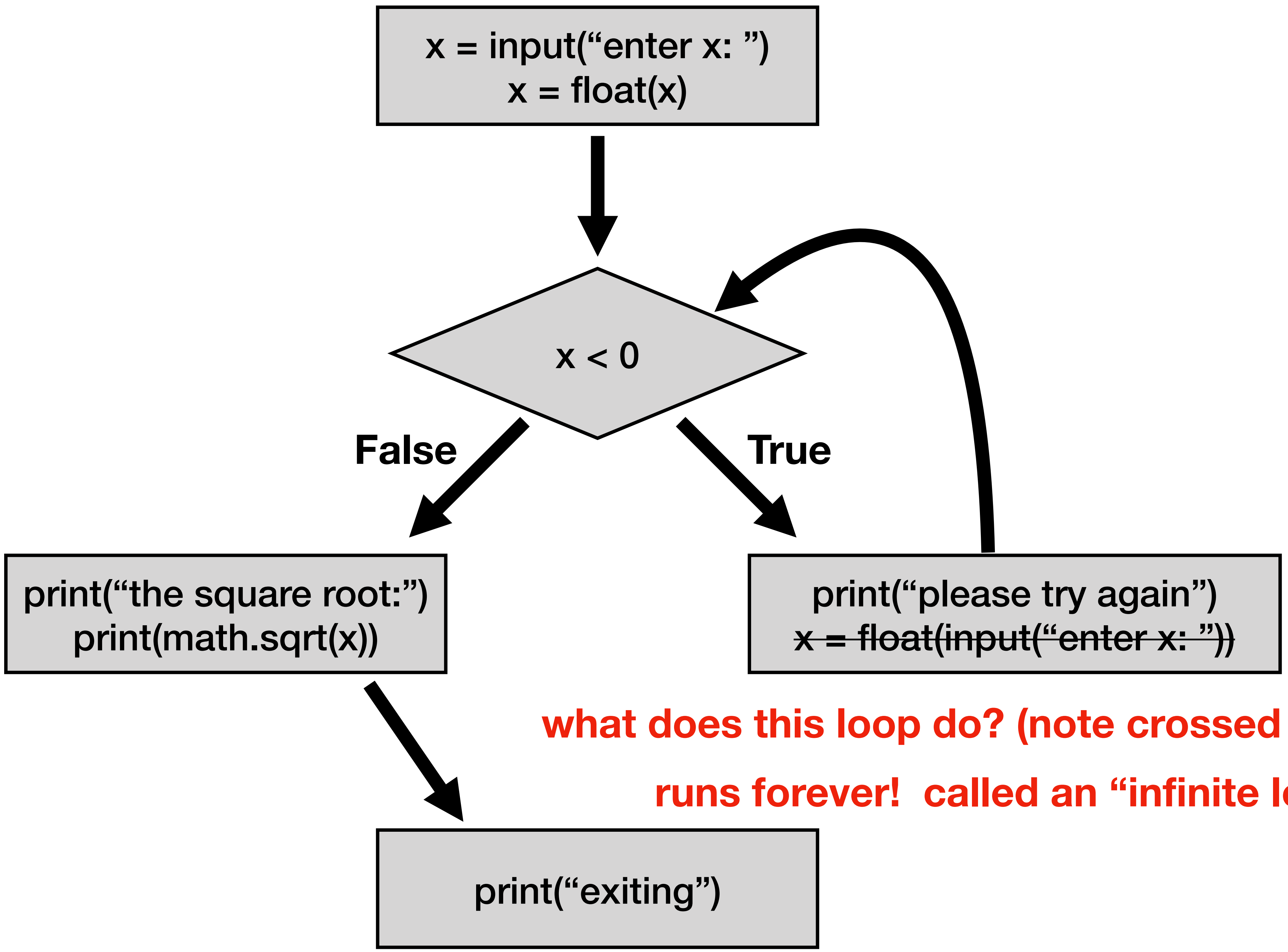

x = input(“enter x: ”)
x = float(x)
x < 0
False
True
print(“the square root:”)
print(math.sqrt(x))
print(“please try again”)
x = float(input(“enter x: ”))
print(“exiting”)
we call this cycle a “loop”
Each time through is called an “iteration”

Control Flow Diagrams: “while”



We keep executing the loop body **while** the condition is true, so this is called a “while” loop

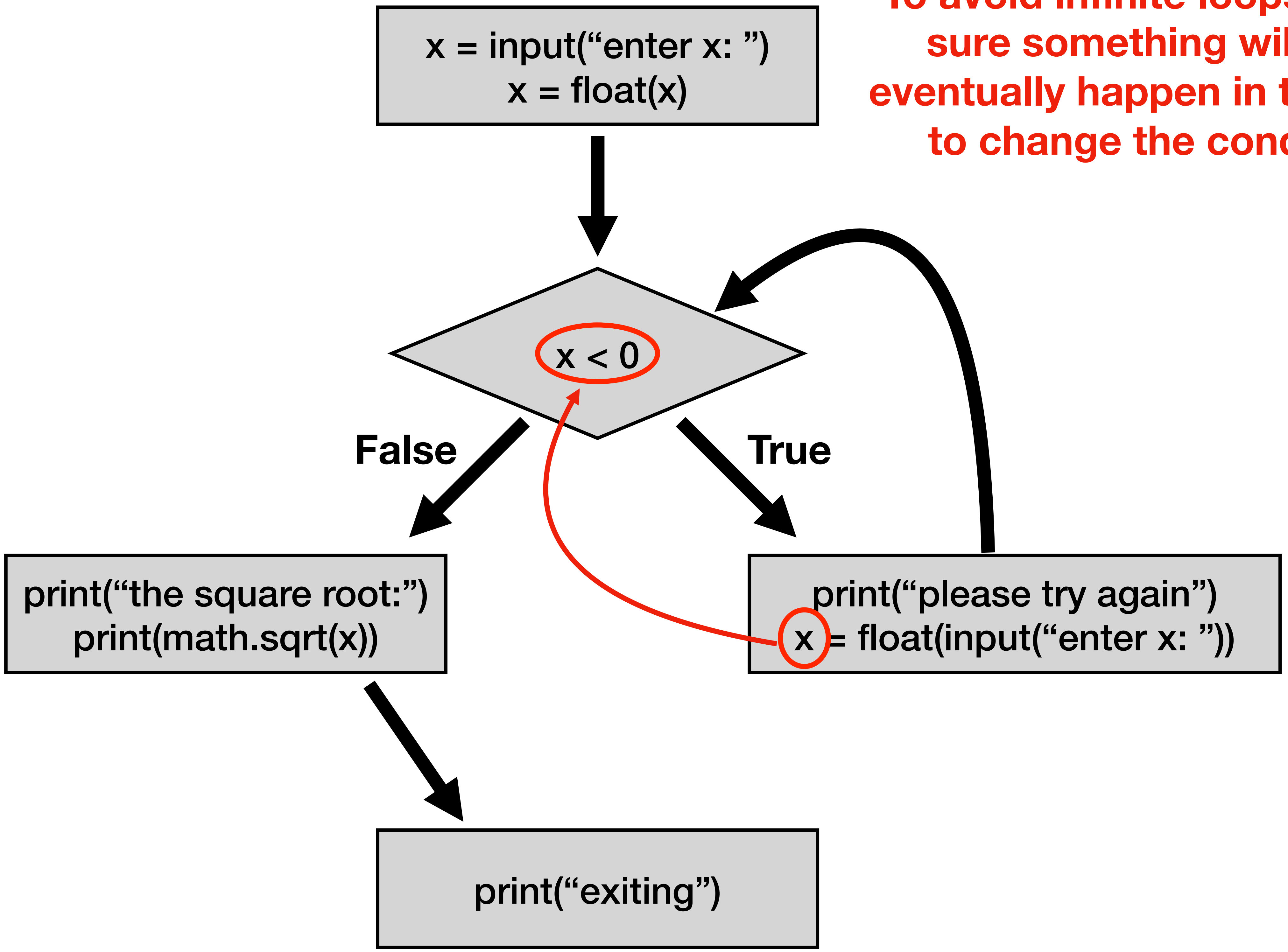
Control Flow Diagrams: “while”


x = input(“enter x: ”)
x = float(x)
x < 0
False
True
print(“the square root:”)
print(math.sqrt(x))
print(“please try again”)
~~x = float(input(“enter x: ”))~~
print(“exiting”)

what does this loop do? (note crossed out line)

runs forever! called an “infinite loop”

Control Flow Diagrams: “while”



To avoid infinite loops, make sure something will/can eventually happen in the body to change the condition

Today's Outline

Control Flow Diagrams

Basic syntax for “while”

Demos

Syntax

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

if x < 0:
    x = int(input("please try again: "))
```

Syntax for "if"

Syntax

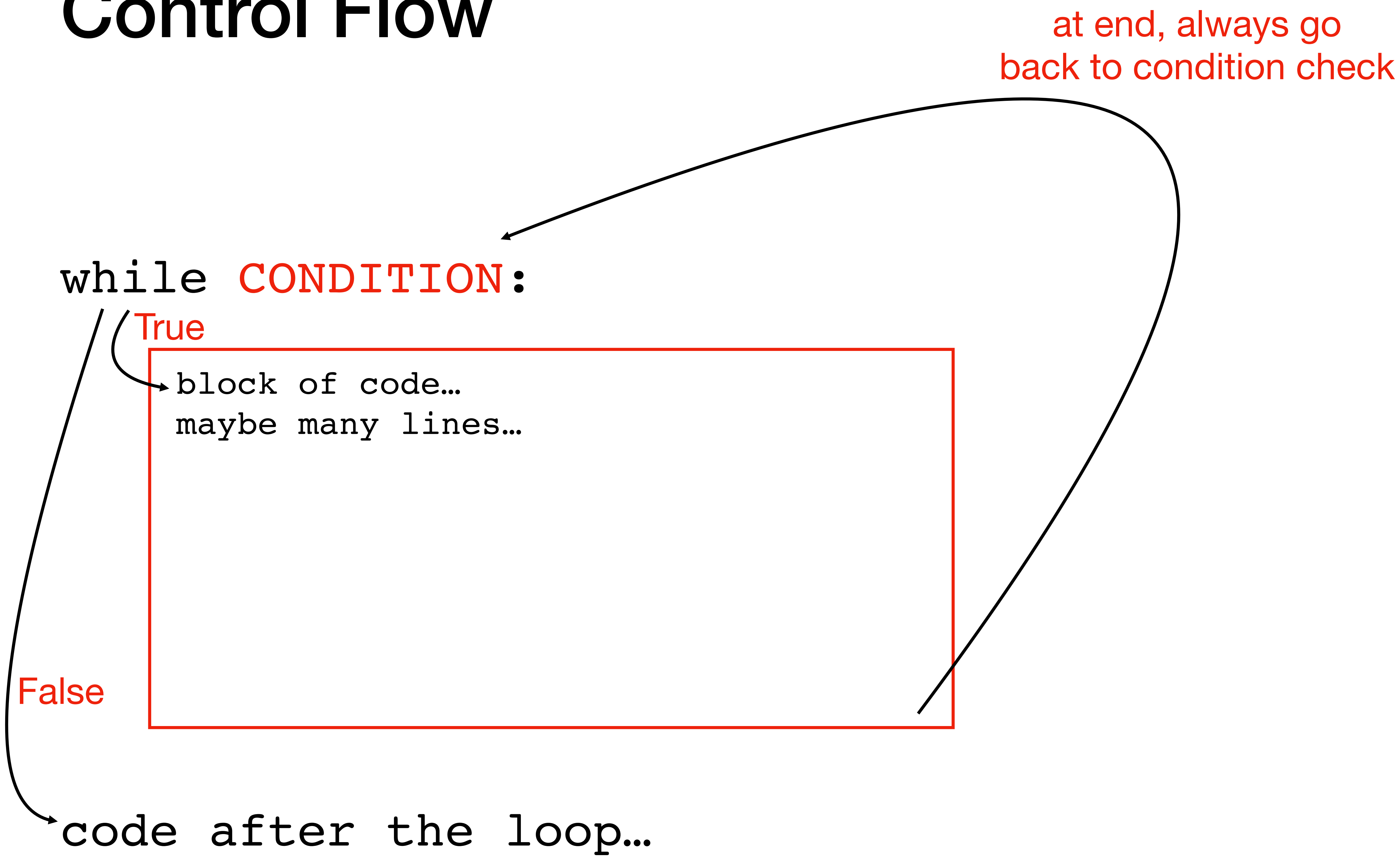
```
x = int(input(“enter x: “))

while x < 0:
    x = int(input(“please try again: ”))
```

Syntax for “while loop” is just like for “if”, just replace “if” with “while”

This example gives user an arbitrary number of tries until they get it right

Control Flow



```
while CONDITION:
    block of code…
    maybe many lines…

code after the loop…
```

Steps to follow

Whenever you write a while loop, keep these in mind:

1. **Initialize** your loop condition variable

2. a) **Update** your loop condition variable in loop body

   b) Make **progress towards** eventually turning your loop condition to **False**

Congrats!

You now understand the 4 key **Flow of Execution** ideas,
in the context of Python.

1. **generally, proceed forward, one step at a time**

2. sometimes go run a “mini program” somewhere else before continuing to the next line
   - This is a function call

3. sometimes skip forward over some lines of code
   - Conditional or while loop, when the condition is false

4. sometimes go back to a previous line of code
   - while loop. When at the end of body, always go back to condition

three primary exceptions to the general case (1)

Today's Outline

Control Flow Diagrams

Basic syntax for “while”

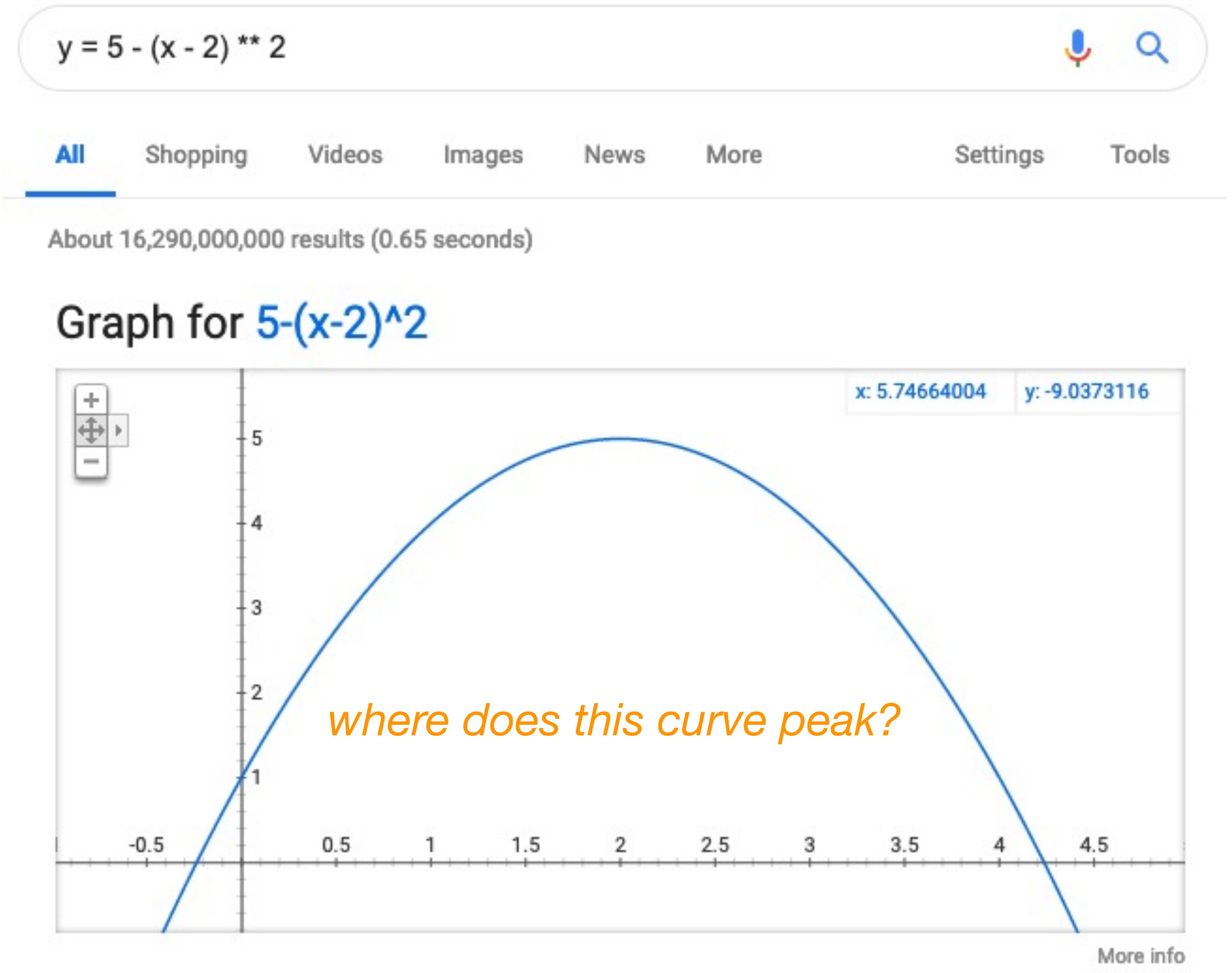
Demos

Example: Countdown Timer

use `time.sleep(1)` $\longrightarrow$

```
how many seconds? 5
5
4
3
2
1
DING DING DING DING DING!
```

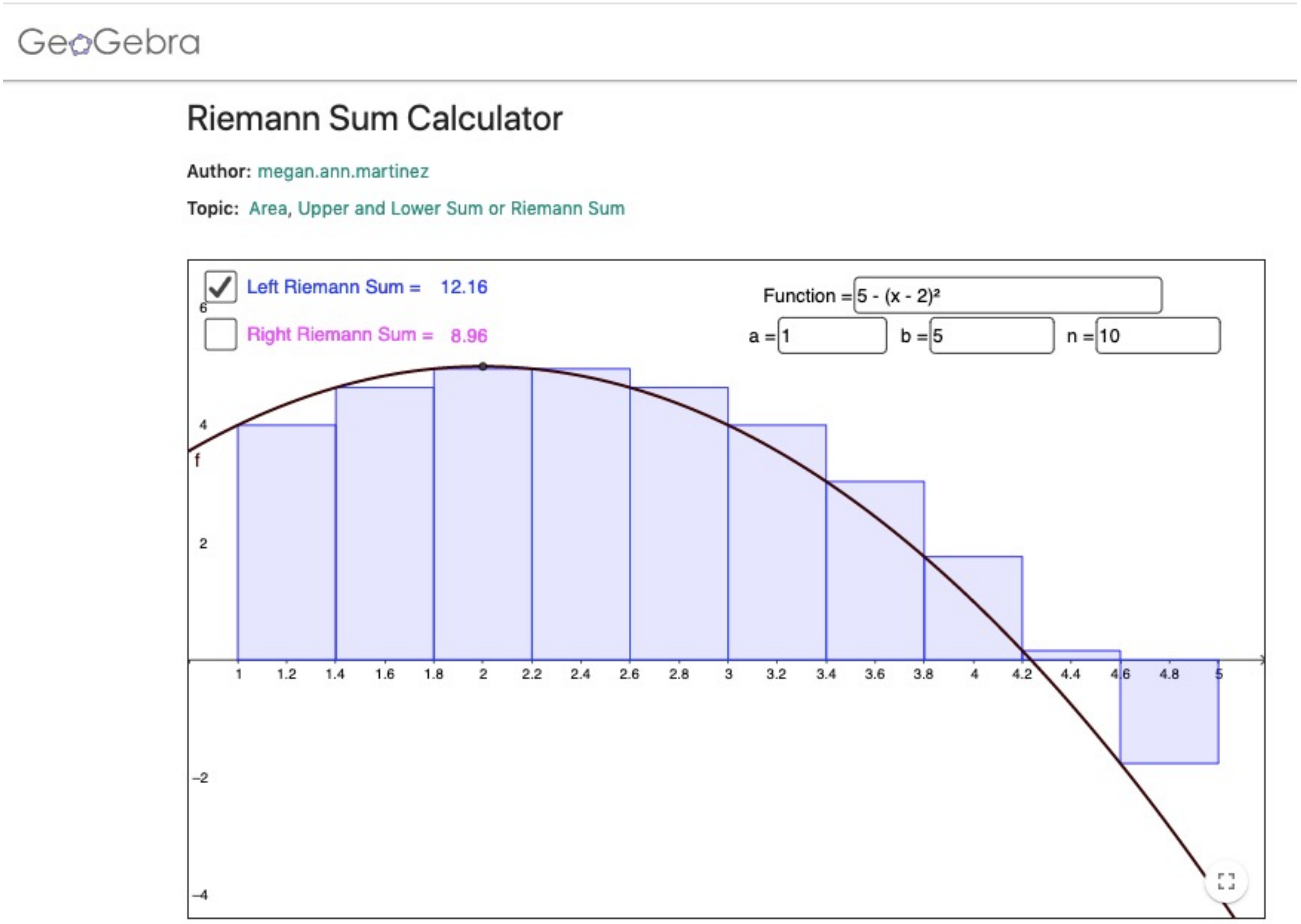
Example: Maximum (Finding the Peak)


y = 5 - (x - 2) ** 2

About 16,290,000,000 results (0.65 seconds)

Graph for 5-(x-2)^2

Example: Integration (Riemann Sum)


GeoGebra

Riemann Sum Calculator

Author: megan.ann.martinez

Topic: Area, Upper and Lower Sum or Riemann Sum

✓ Left Riemann Sum = 12.16

Right Riemann Sum = 8.96

Function = 5 - (x - 2)²

a = 1 b = 5 n = 10

Example: Prime Finder

```
Prime numbers:
2 is prime
3 is prime
4 is not prime
5 is prime
6 is not prime
7 is prime
8 is not prime
9 is not prime
…
```

Challenge: Countdown Timer

use `time.sleep(1)`

```
how many seconds? 5
5
4
3
2
1
DING DING DING DING DING!
how many seconds? 2
2
1
0
DING DING DING DING DING!
how many seconds? q
good bye!
```

exit program

this program should involve a nested loop!!!

Challenge: Battleship

columns

rows

```
..........
..........
..*+***...
..........
........*.
........*.
..........
....*.....
....*.....
....*.....
```

show where ship(s) are after guess

```
guess and ship: +
     just ship: *
guess and miss: -
    blank spot: .
```