

# [220] Iteration, part 1

Department of Computer Sciences  
University of Wisconsin-Madison

## Readings:

Chapter 7 of Think Python

Chapter 6.1 to 6.3 of Python for Everybody

Due: Project 2

Assigned: Quiz 3

# Learning Objectives

Implement an iterative algorithm using a while loop

- printing /counting
- validating user input
- performing iterative calculation
- printing grids / character art

Trace iterative algorithms and determine their output

Recognize common while loop errors:

- infinite loops (when unintentional)
- off-by-one mistakes in the loop control variable increment / decrement

# Worksheet

**State:**

N

4

total

0

answer

0

6

**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

# Worksheet

State:



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:



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

loop condition

loop body

# Worksheet

State:



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

loop condition

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

# Worksheet

State:

N

4

total

0

answer

0

6

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

loop condition

skip past loop body

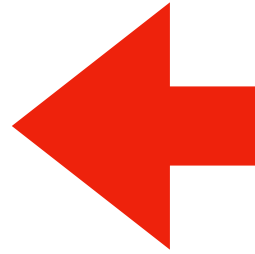
continue to loop body

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”

Examples

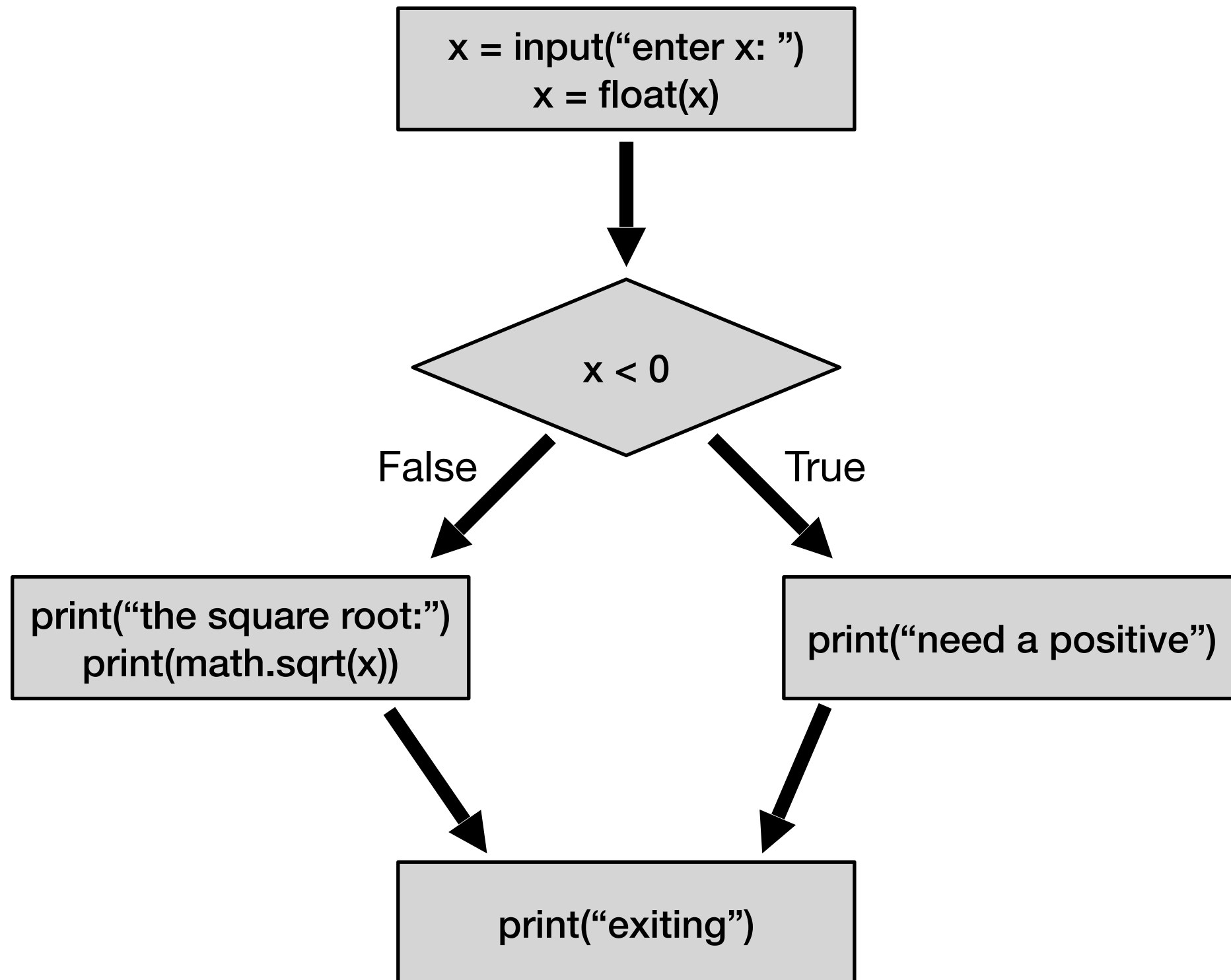
Basic syntax for “for”

Examples

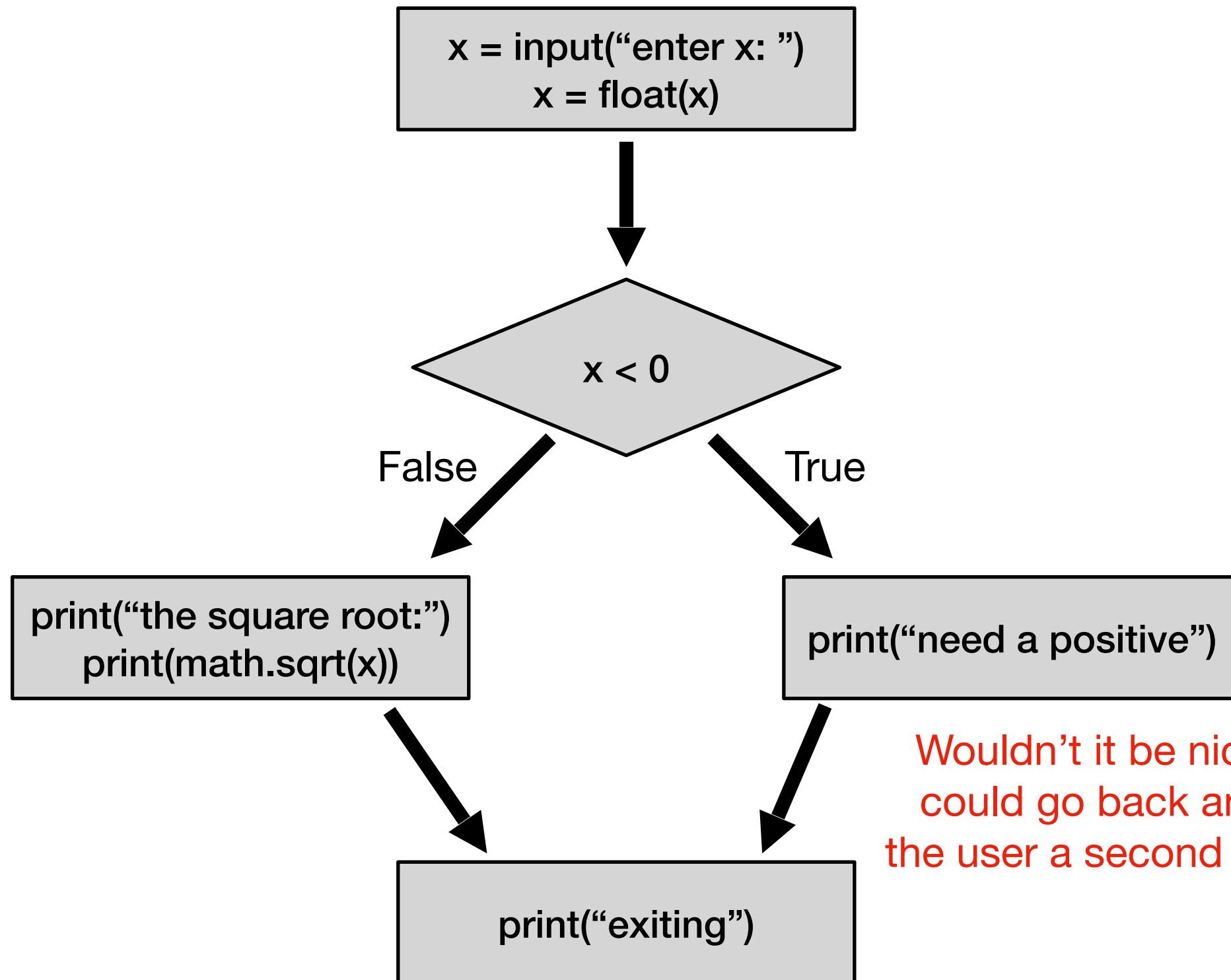
“break” and “continue”



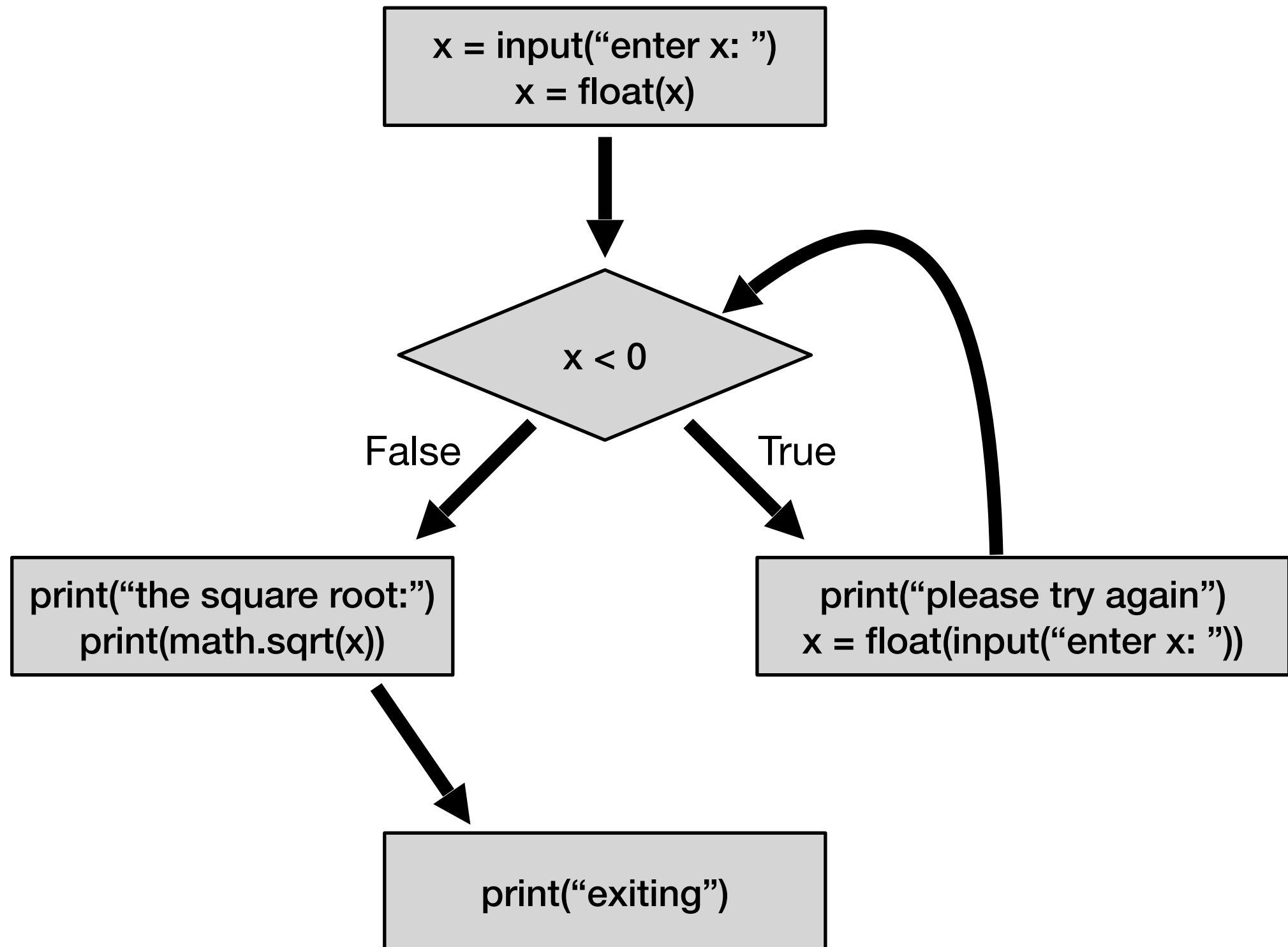
# Control Flow Diagrams: “if”



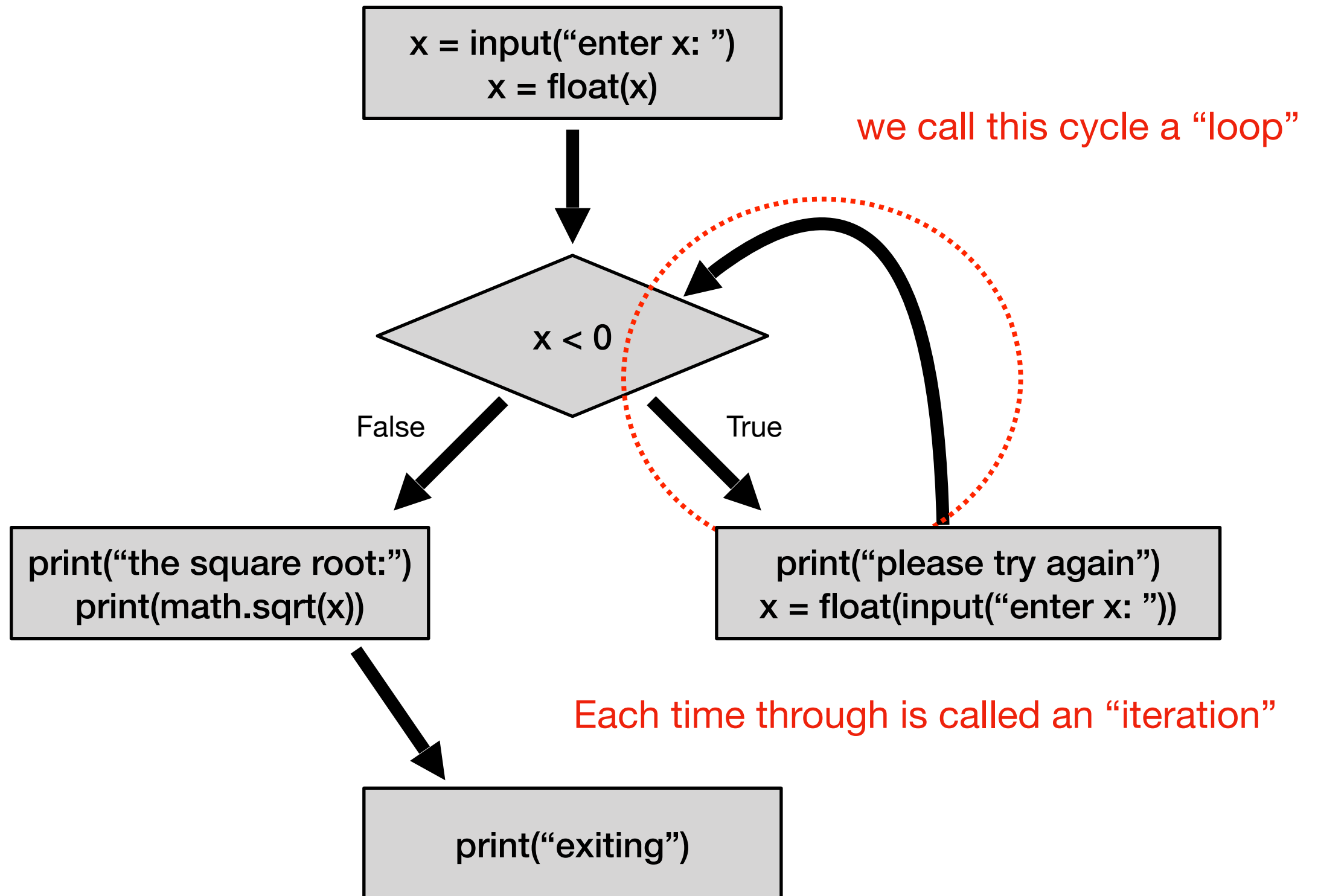
# Control Flow Diagrams: “if”



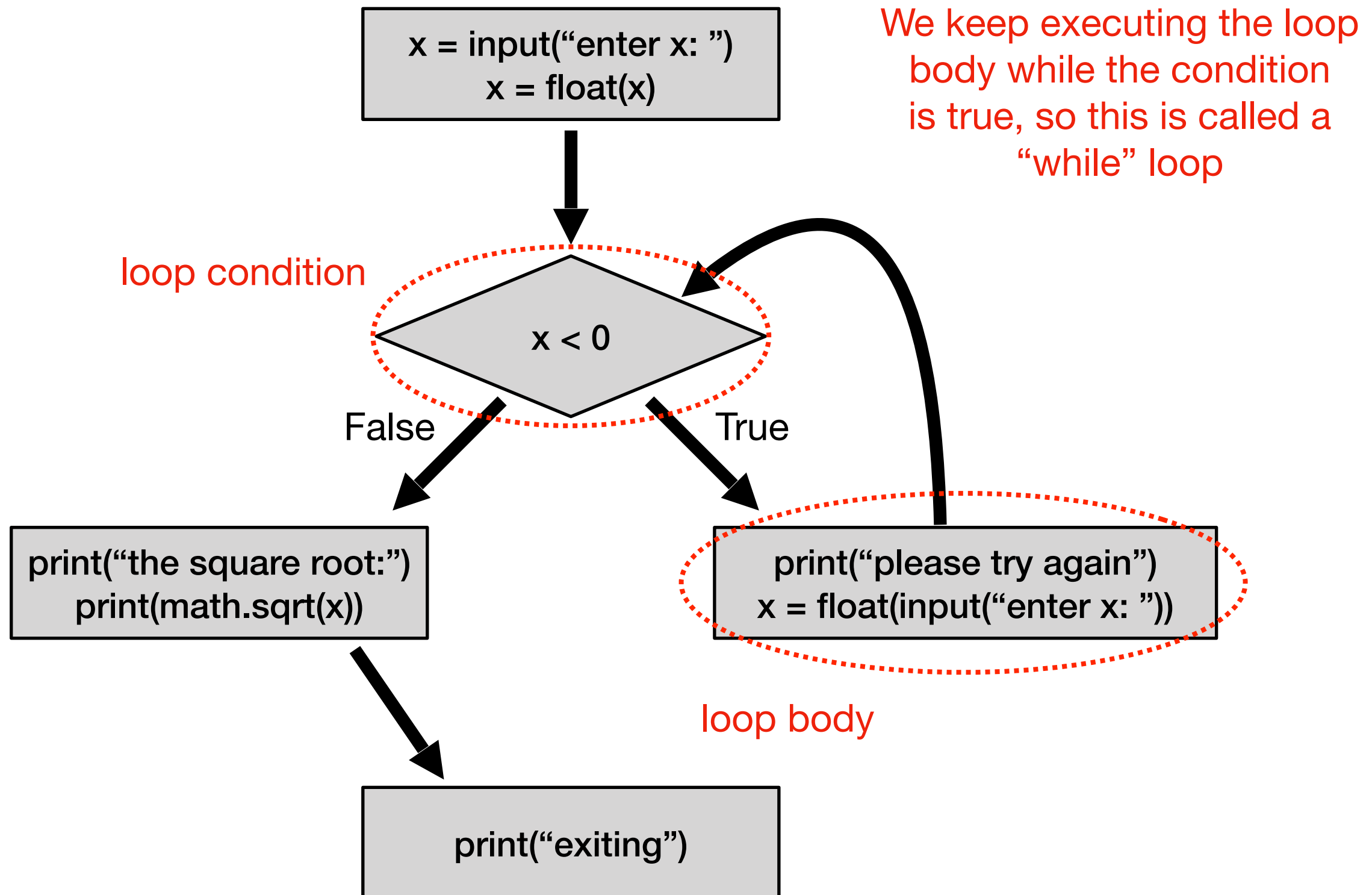
# Control Flow Diagrams: “while”



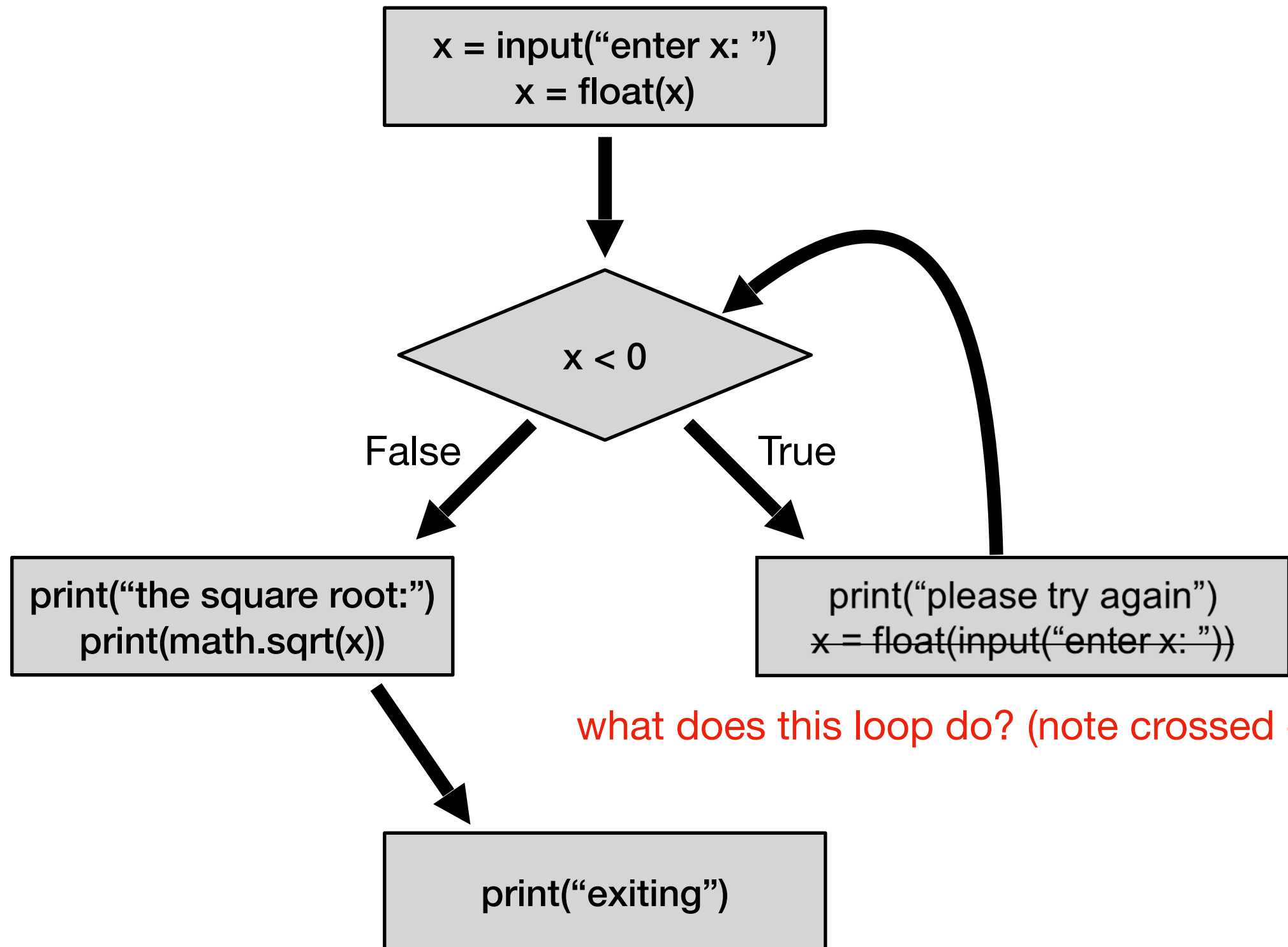
# Control Flow Diagrams: “while”



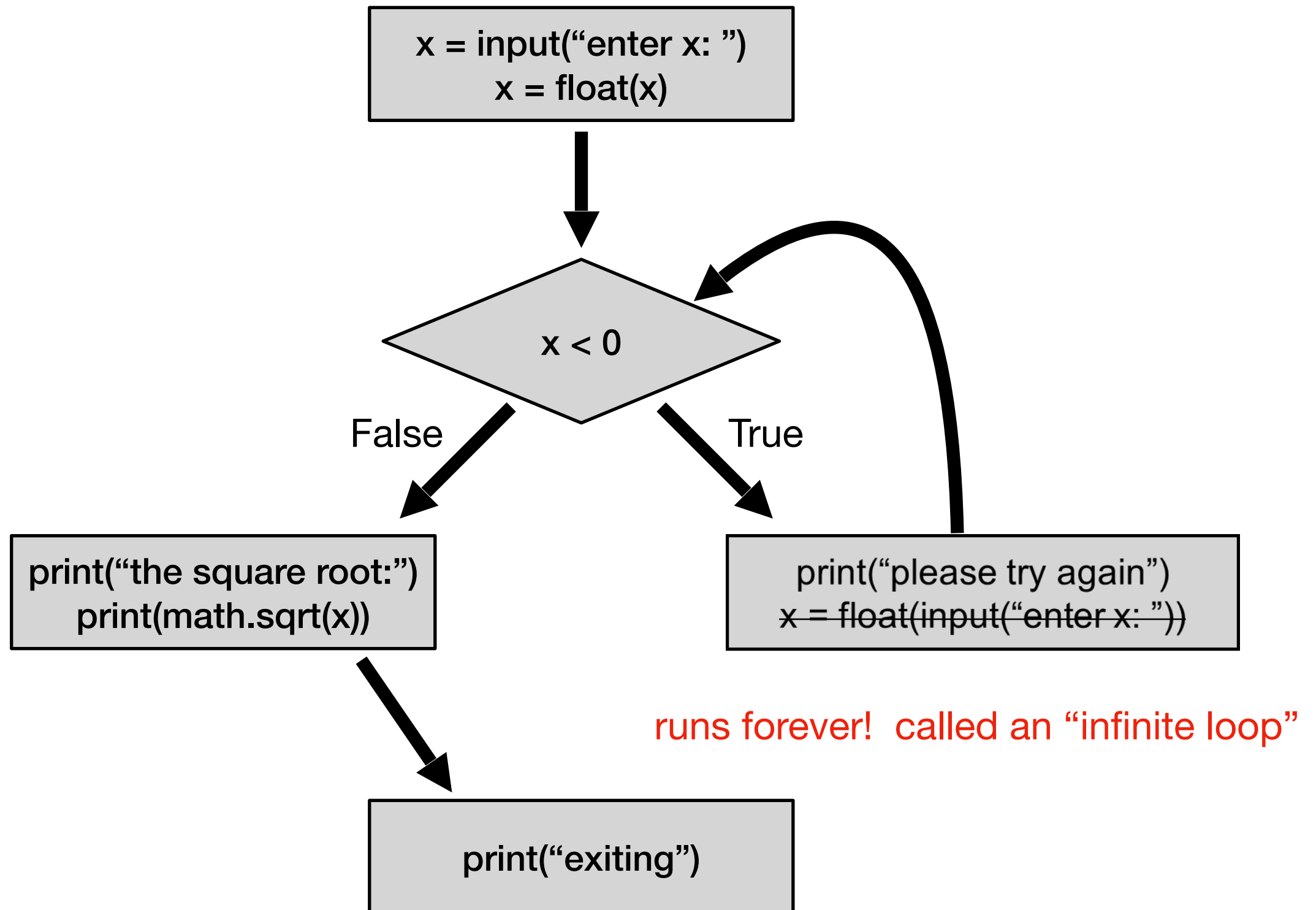
# Control Flow Diagrams: “while”



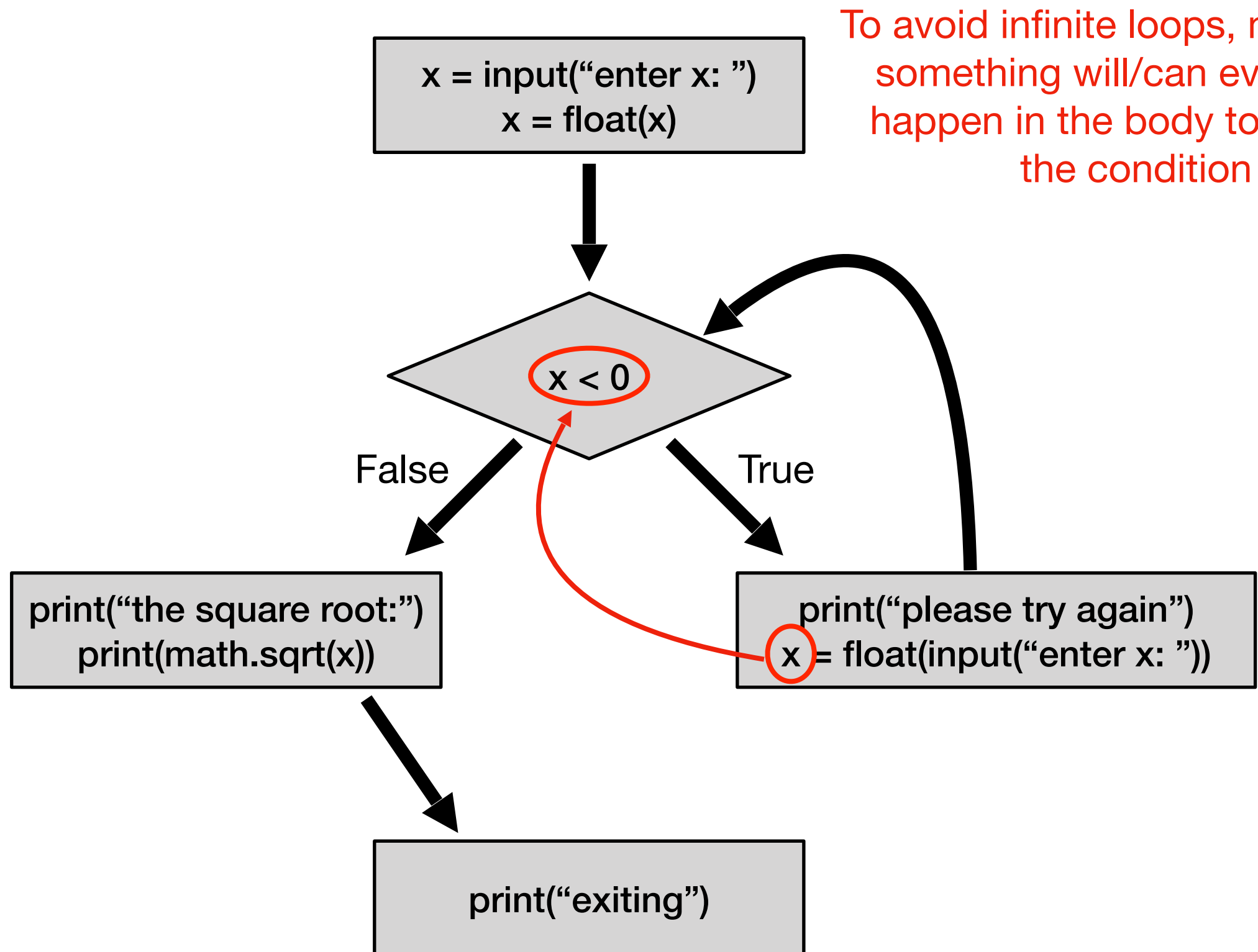
# Control Flow Diagrams: “while”



# Control Flow Diagrams: “while”



# Control Flow Diagrams: “while”

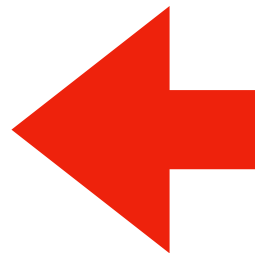




# Today's Outline

Control Flow Diagrams

Basic syntax for “while”



Examples

Basic syntax for “for”

Examples

“break” and “continue”

# 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

# Terminology

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

```
while x < 0: ← loop condition
```

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

loop control variable

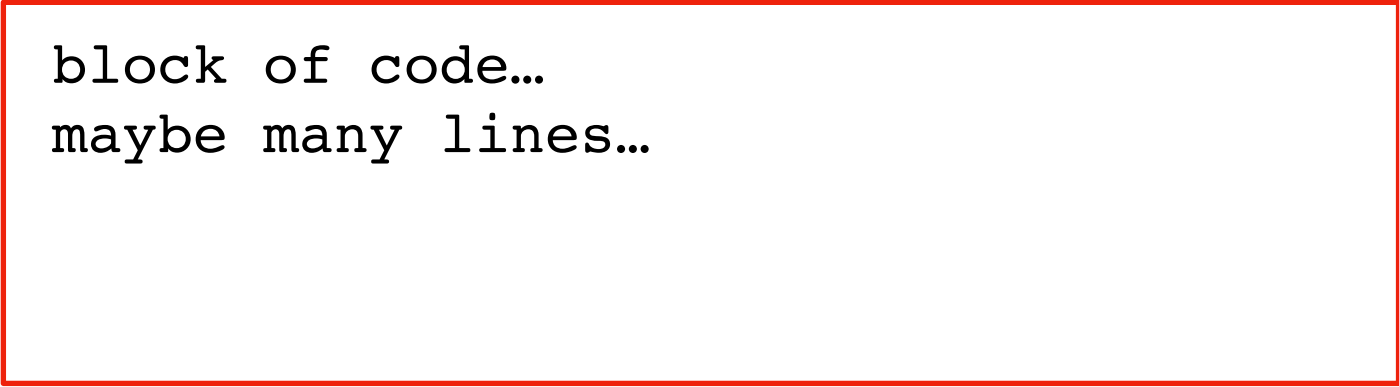
loop body

# Control Flow

```
while CONDITION:  
    # your code
```

# Control Flow

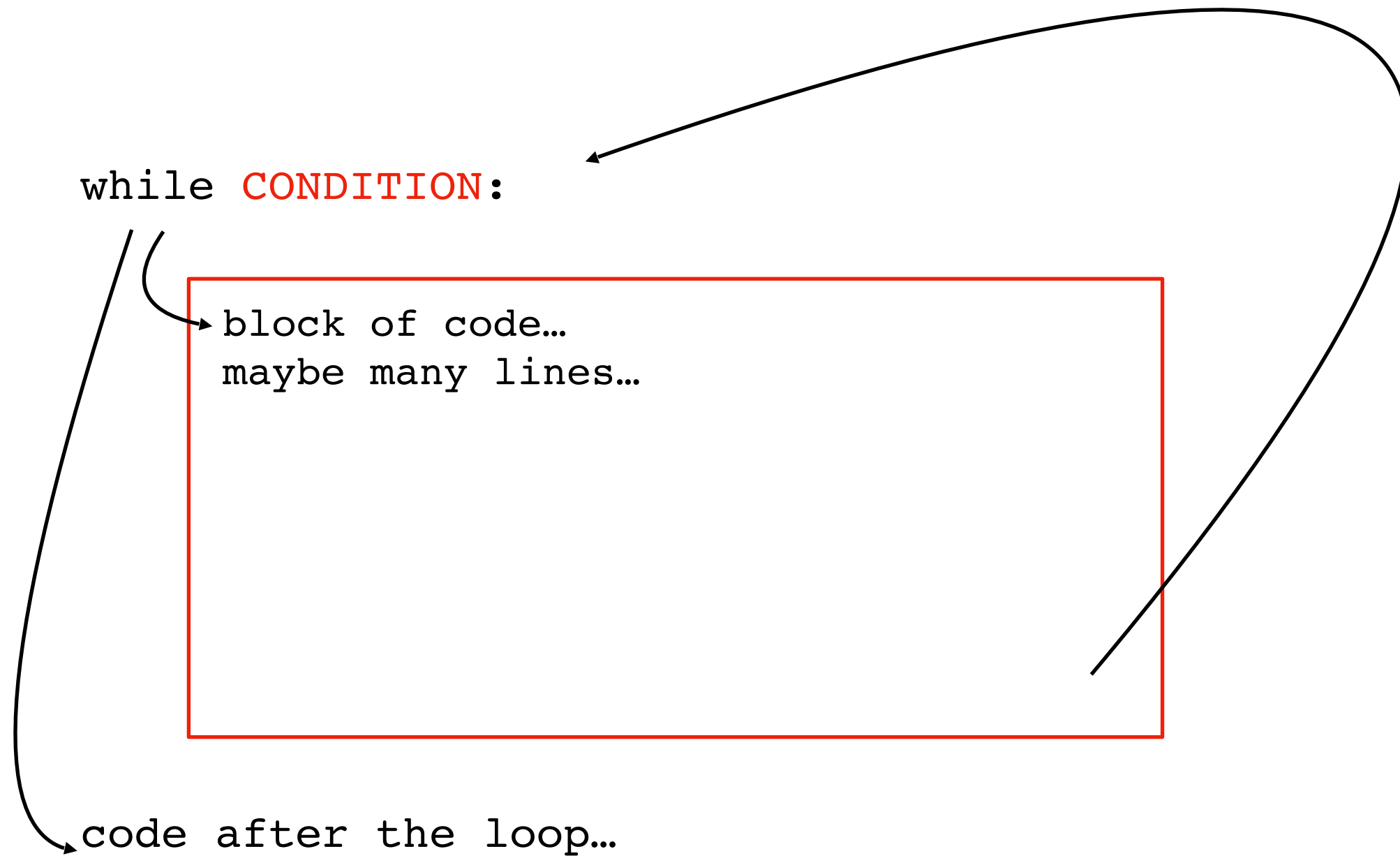
`while` **CONDITION**:



block of code...  
maybe many lines...

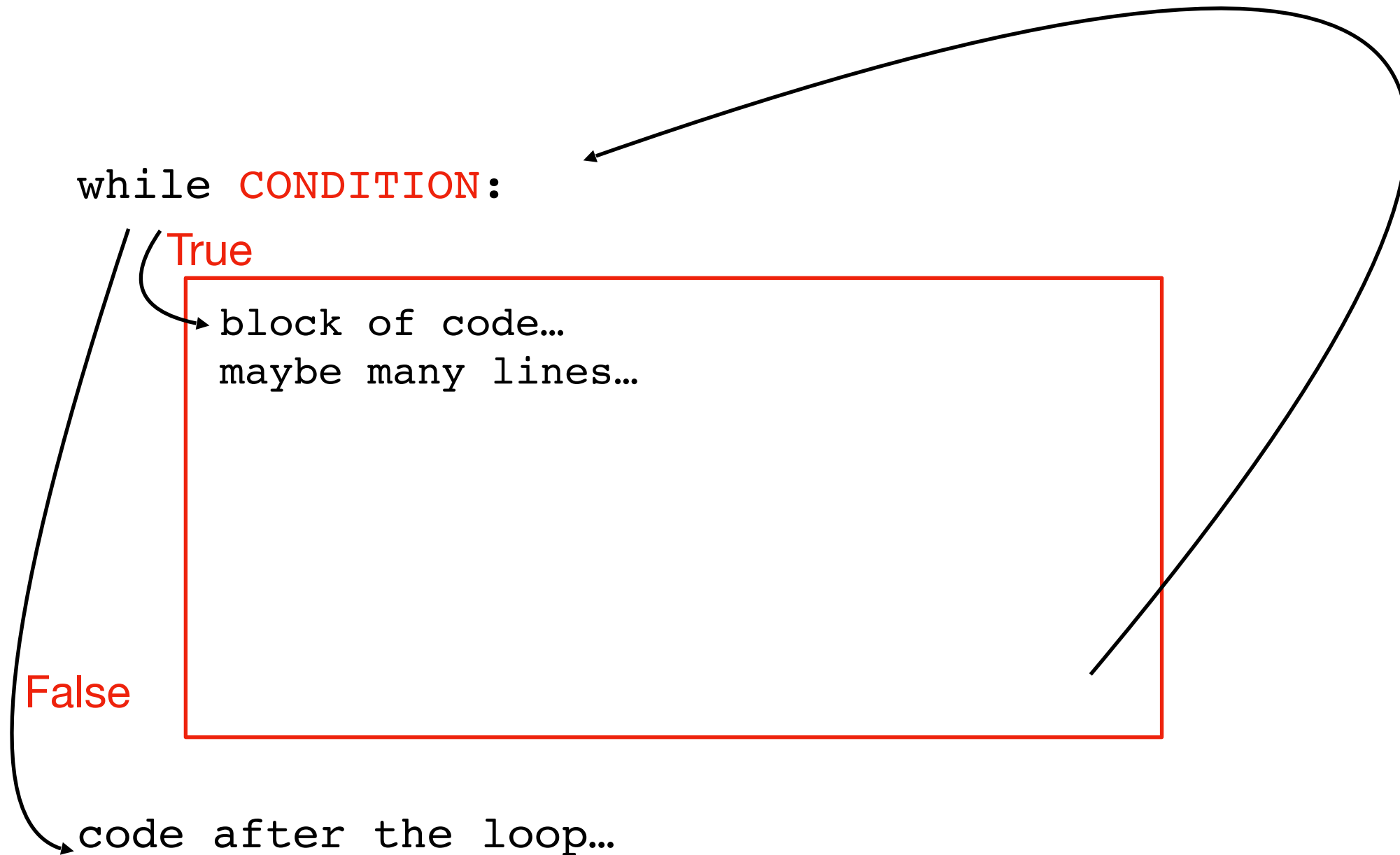
code after the loop...

# Control Flow



# Control Flow

at end, always go  
back to condition check





# 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

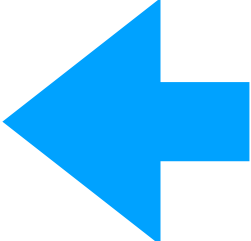
- **while loop**. When at the end of body, always go back to condition

three primary exceptions to the general case (1)

# Today's Outline

Control Flow Diagrams

Basic syntax for “while”

Examples 

Basic syntax for “for”

Examples

“break” and “continue”

# Example: Countdown Timer

use `time.sleep(1)` 

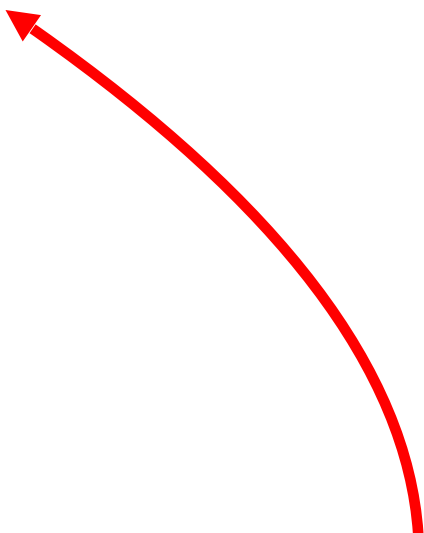
```
How many seconds? 5
5 seconds left
4 seconds left
3 seconds left
2 seconds left
1 seconds left
DING DING DING DING DING!
```

# for with range

Output:

0  
3  
6  
9  
12

```
for item in range(5):  
    print(item * 3)
```



using range(N) with a for loop will  
iterate with these values for item:  
0, 1, 2, ..., N-2, N-1

# Example: Maximum (Finding the Peak)

$y = 5 - (x - 2)^{**} 2$

All

Shopping

Videos

Images

News

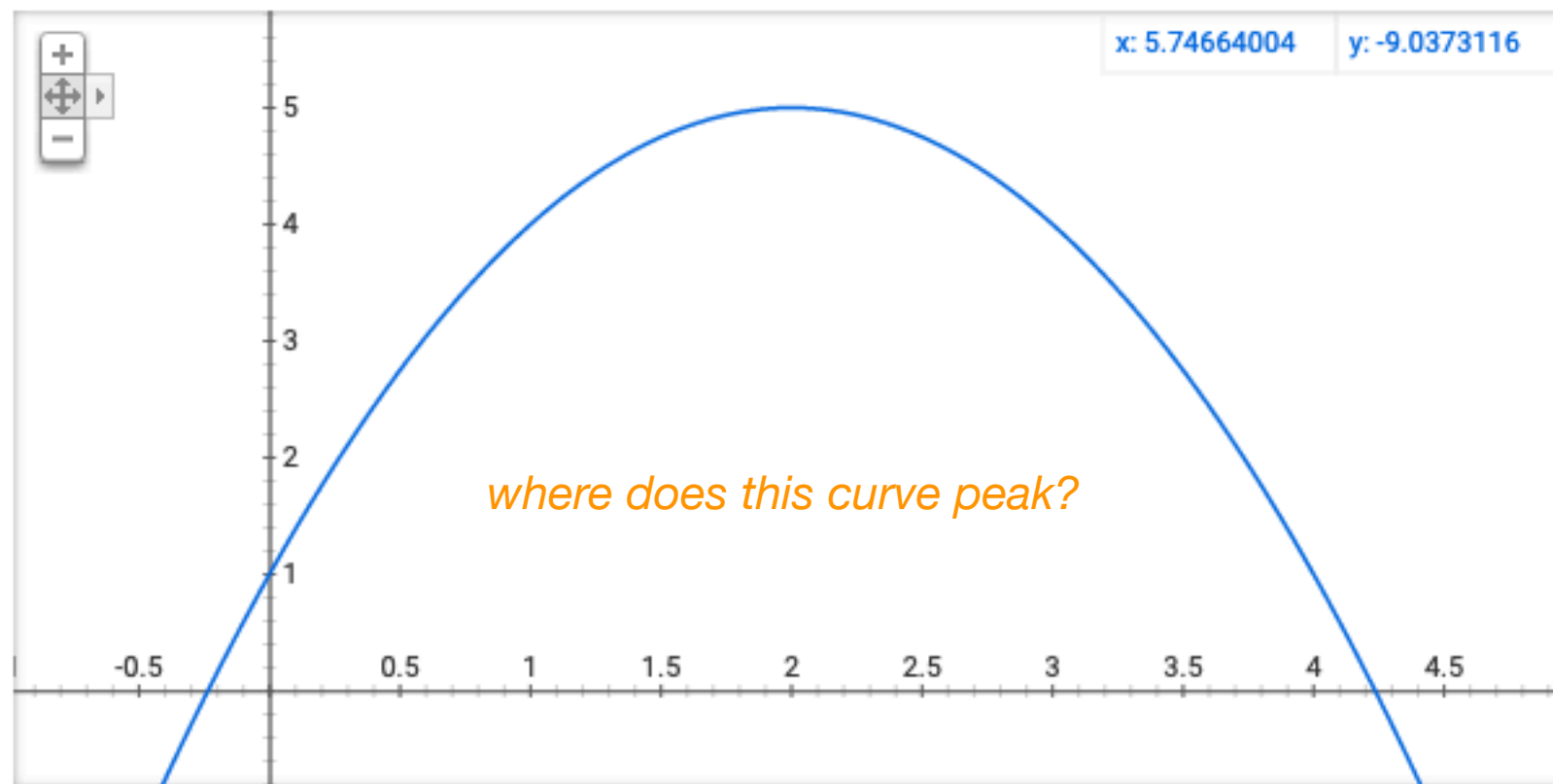
More

Settings

Tools

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

Graph for  $5-(x-2)^2$



More info

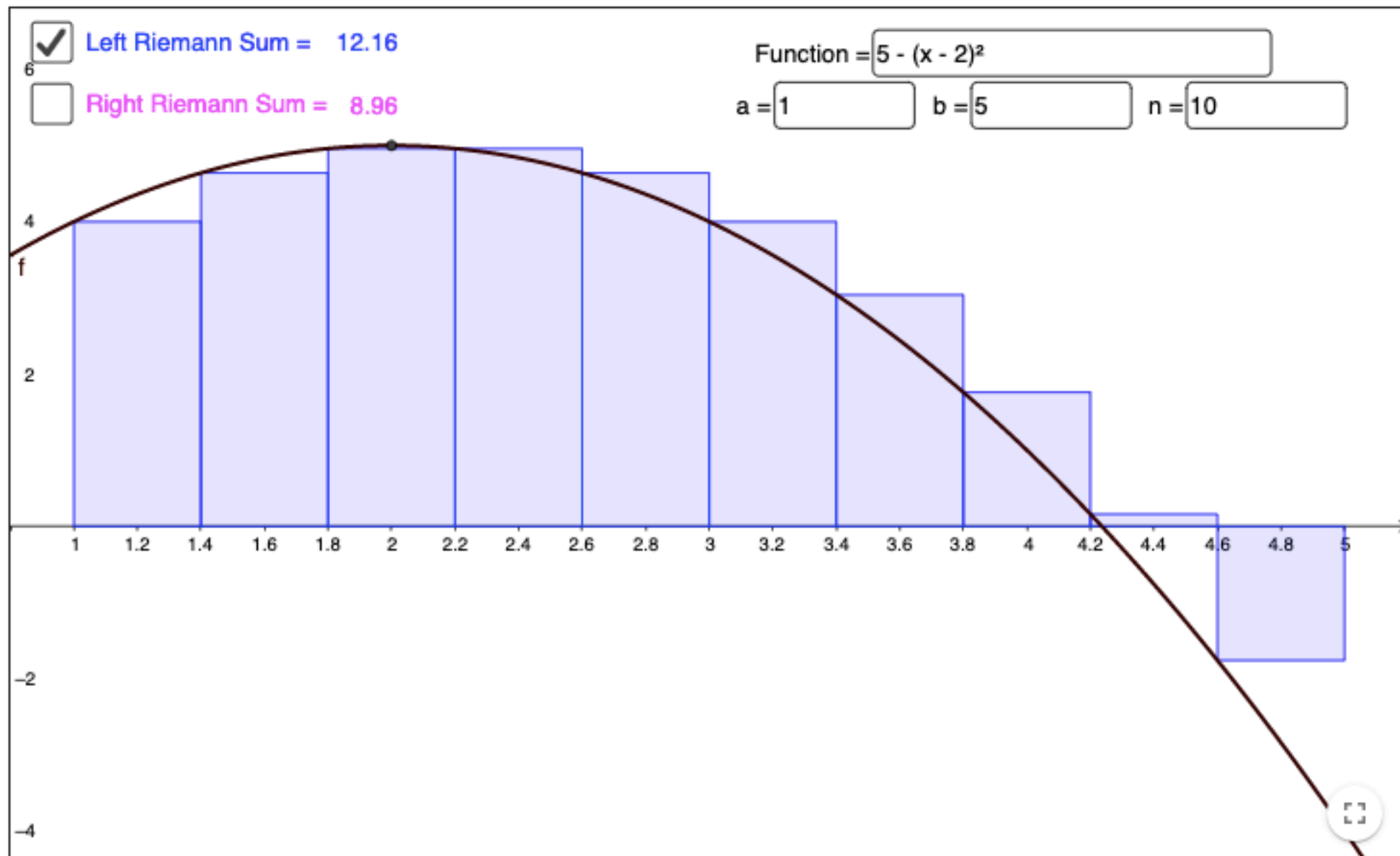
# Example: Integration (Riemann Sum)

GeoGebra

## Riemann Sum Calculator

Author: [megan.ann.martinez](#)

Topic: [Area](#), [Upper and Lower Sum](#) or [Riemann Sum](#)



# 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

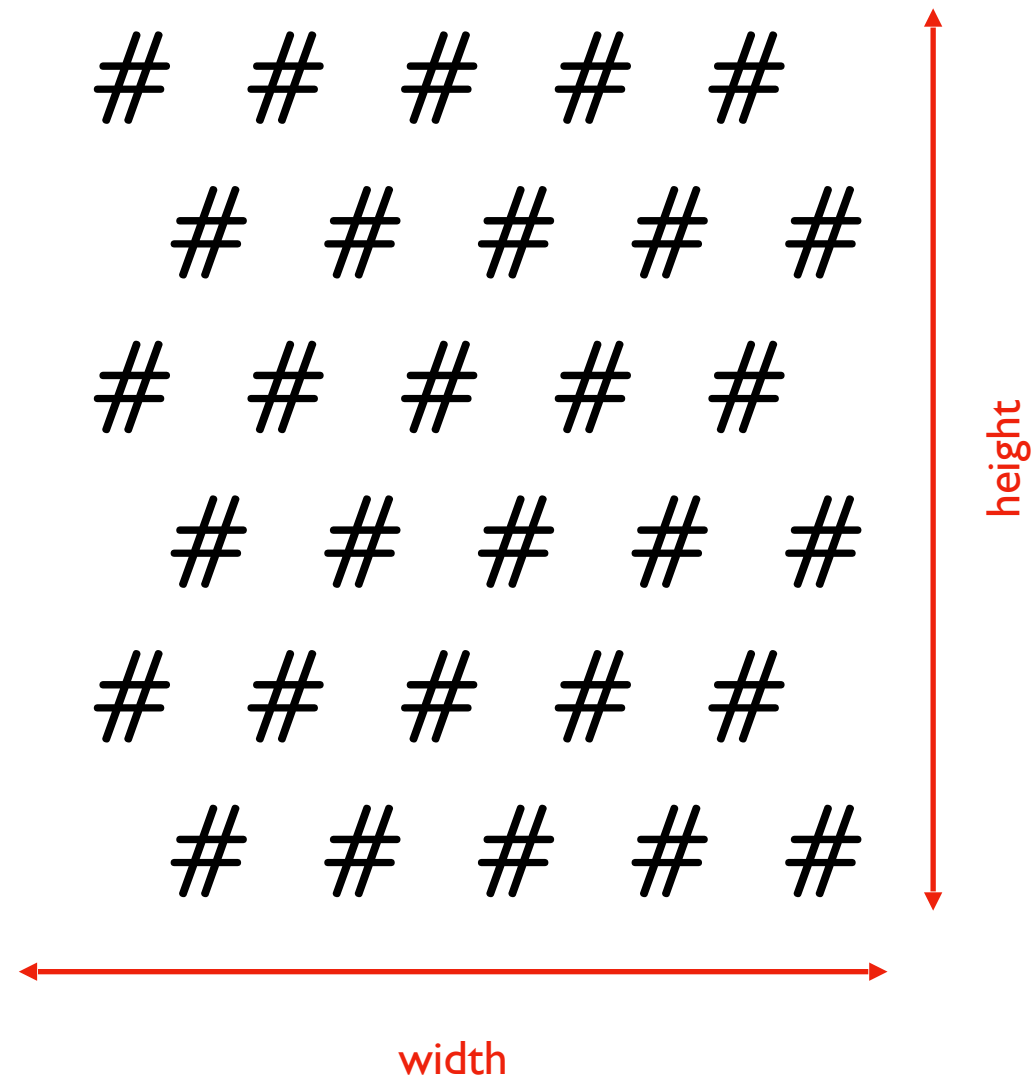
9 is not prime

...



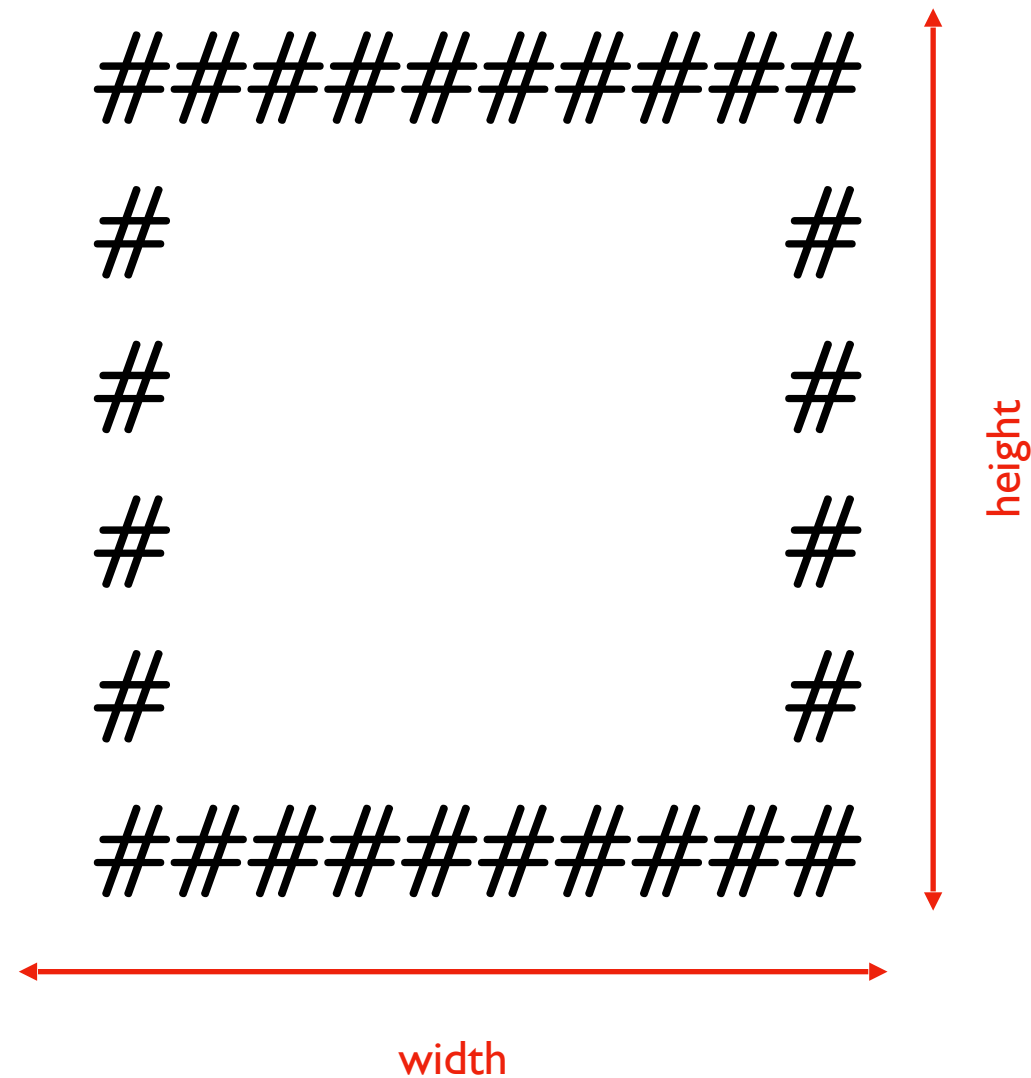
# Practice: Checkers

write a loop to draw the following:



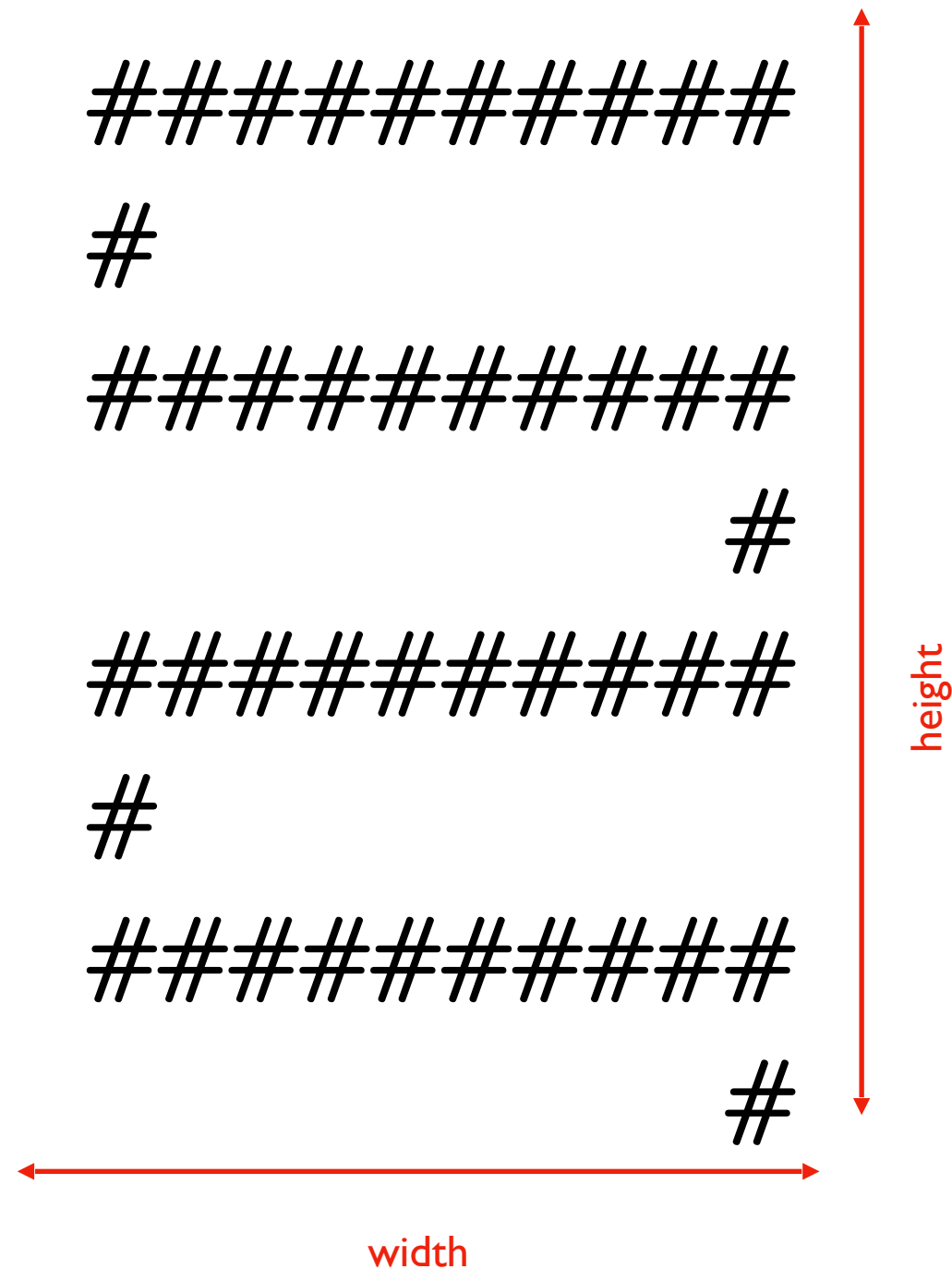
# Practice: Border

write a loop to draw the following:



# Practice: Snake

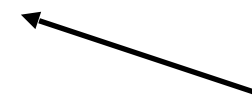
write a loop to draw the following:



# 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!!!

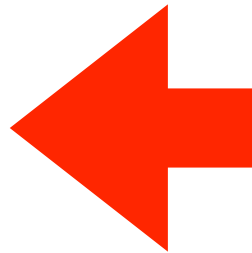
# Today's Outline

Control Flow Diagrams

Basic syntax for “while”

Examples

Basic syntax for “for”

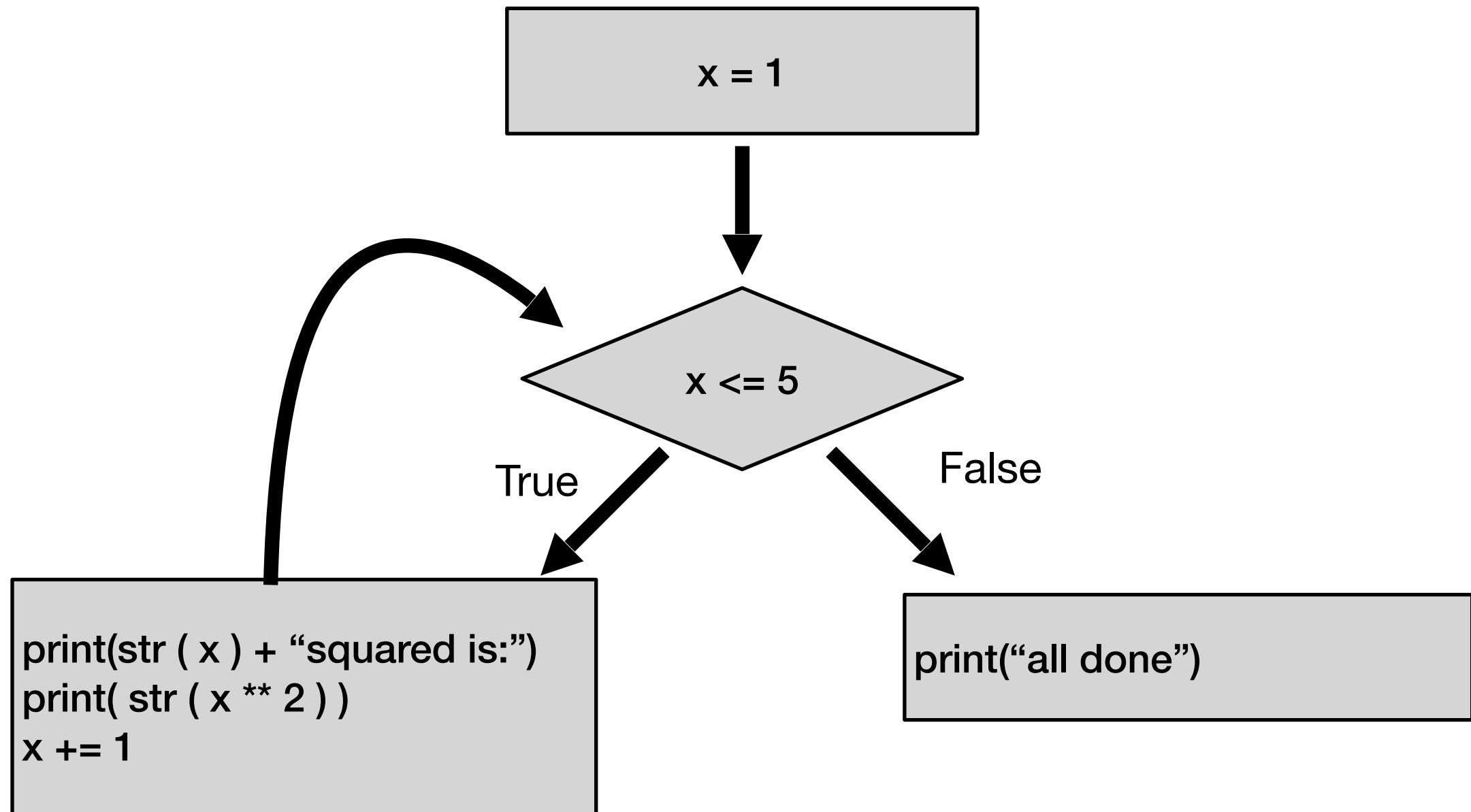


Examples

“break” and “continue”

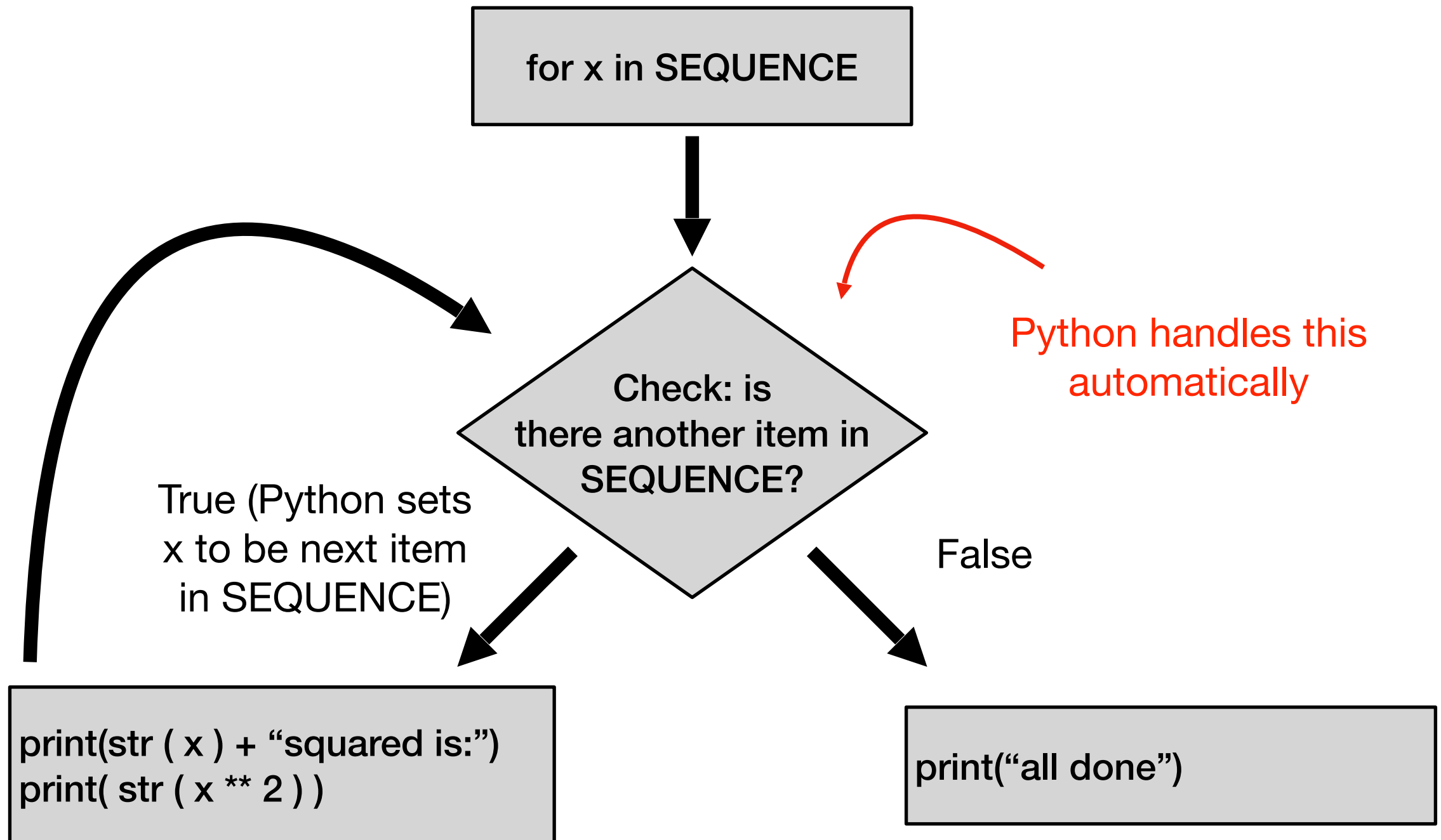
# Control Flow Diagrams: “while”

Print the square of all positive numbers  $\leq 5$



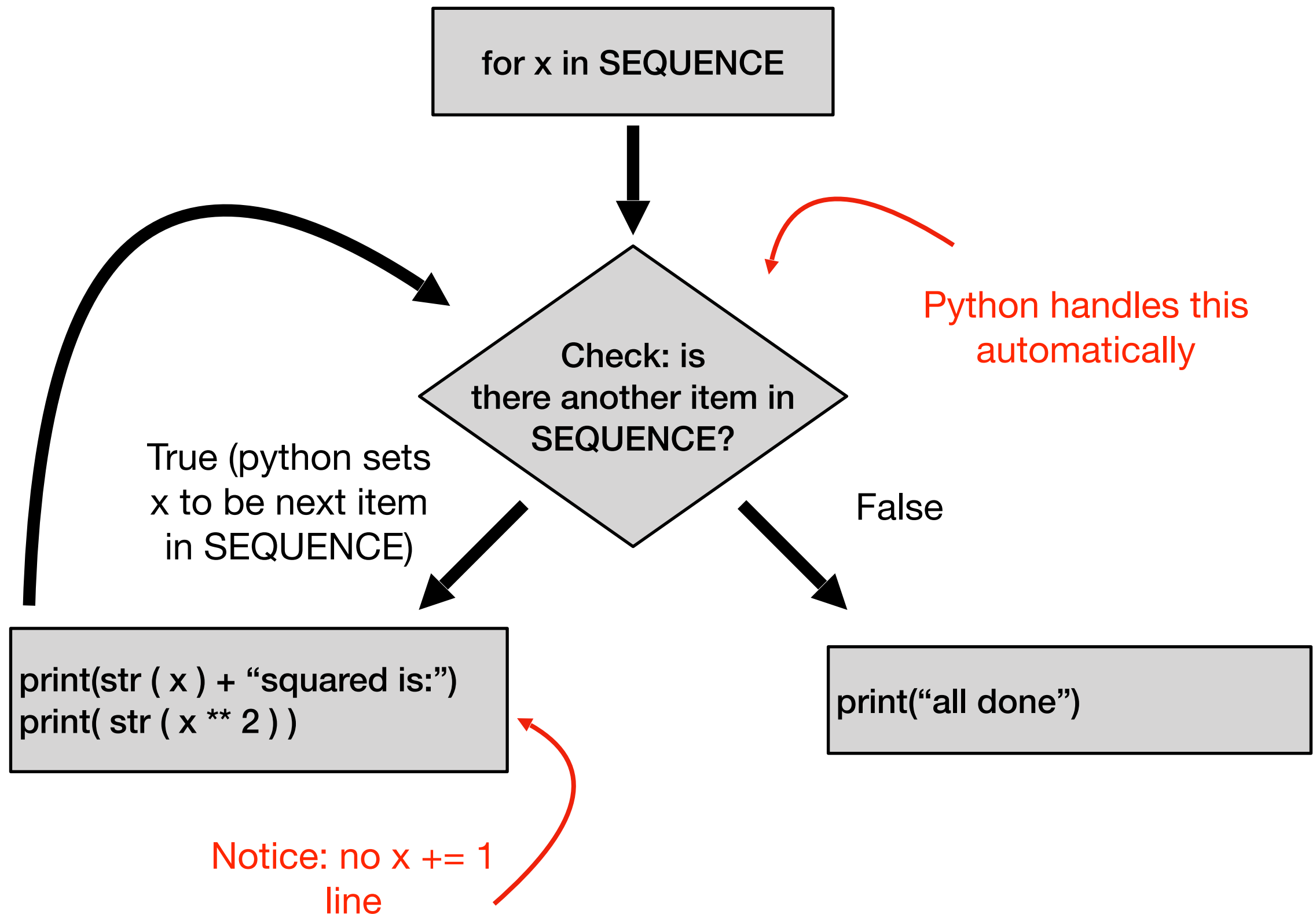
# Control Flow Diagrams: “for”

## Print the square of all positive numbers $\leq 5$



# Control Flow Diagrams: “for”

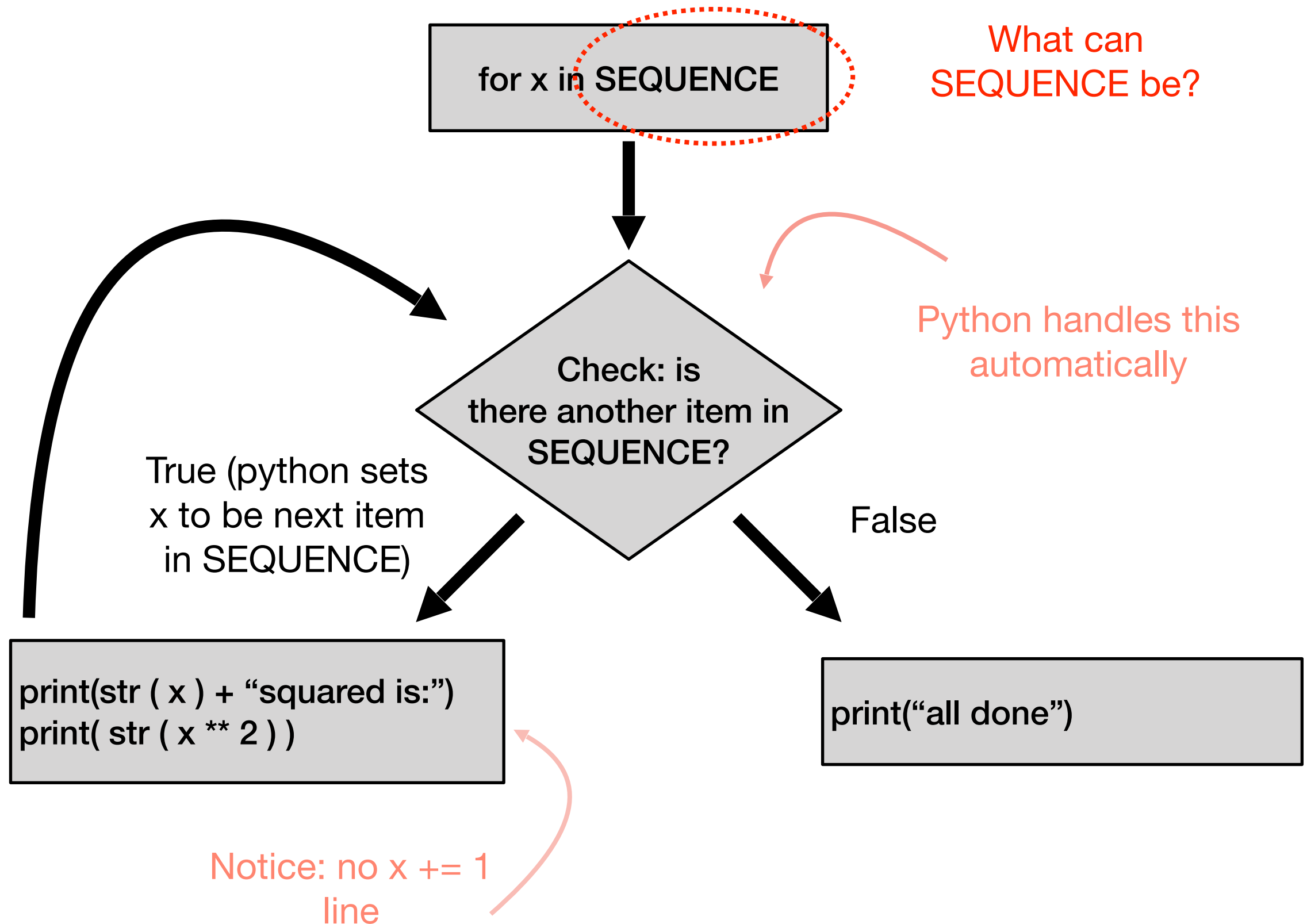
## Print the square of all positive numbers $\leq 5$





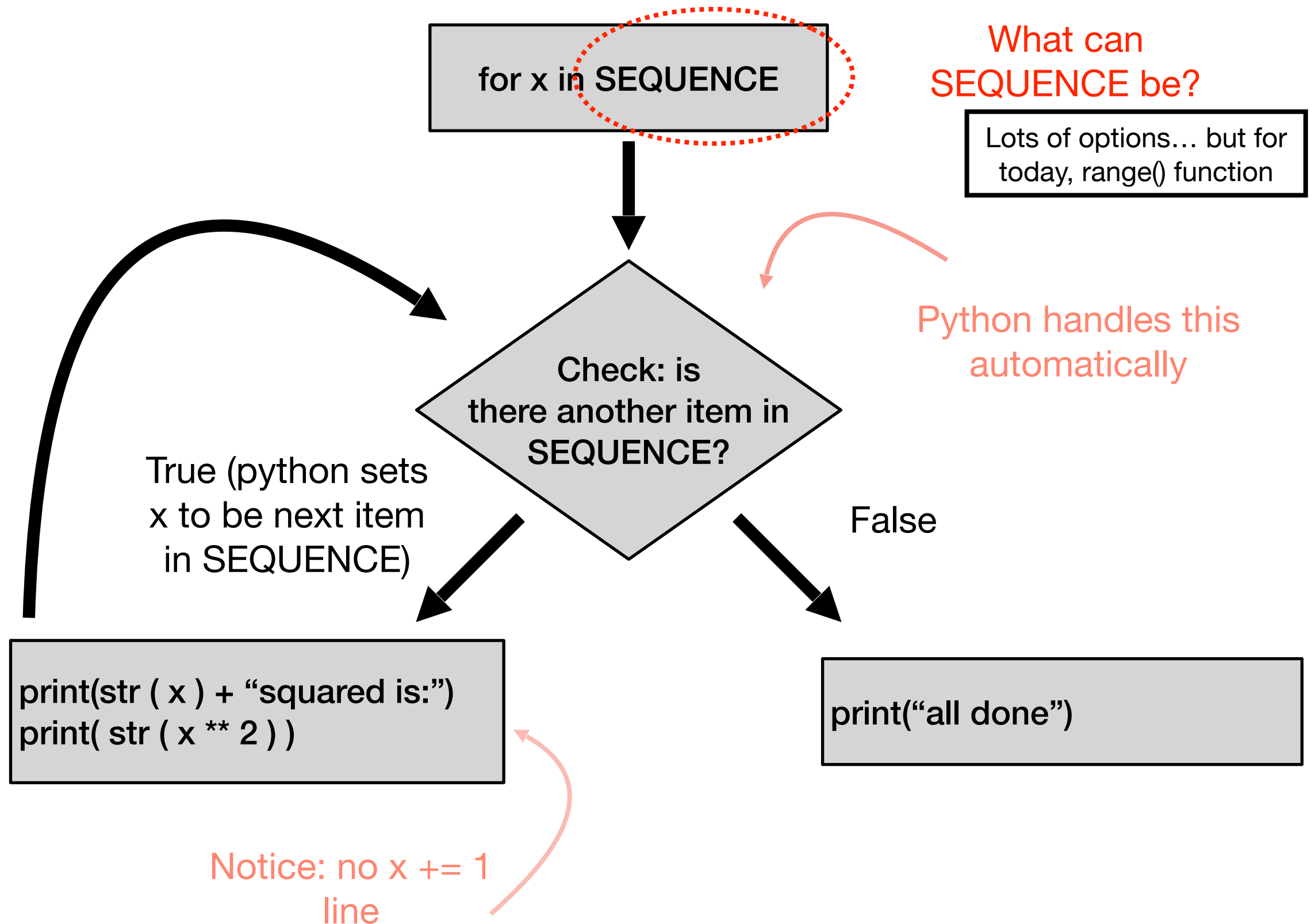
# Control Flow Diagrams: “for”

## Print the square of all positive numbers $\leq 5$



# Control Flow Diagrams: “for”

## Print the square of all positive numbers $\leq 5$



**range()** is a special built-in python function that is used with for loops

```
for x in range(1, 5):  
    print(x)
```

tells python we  
want x to have  
values 1, 2, 3, 4

The output will be

1  
2  
3  
4

**range()** is a special built-in python function that is used with for loops

Can also provide just one  
argument to **range()**

```
for x in range(5):
```

```
    print(x)
```

tells python we  
want x to have  
values 0, 1, 2, 3, 4

The output will be

```
0
1
2
3
4
```

# Today's Outline

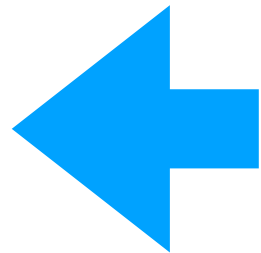
Control Flow Diagrams

Basic syntax for “while”

Examples

Basic syntax for “for”

Examples



“break” and “continue”

# Example: Countdown Timer

use `time.sleep(1)` 

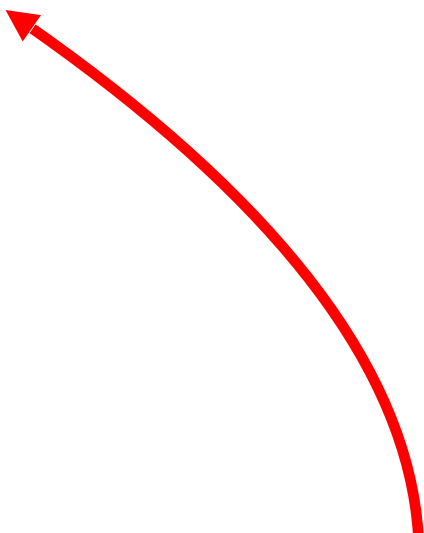
```
How many seconds? 5
5 seconds left
4 seconds left
3 seconds left
2 seconds left
1 seconds left
DING DING DING DING DING!
```

# for with range

Output:

0  
3  
6  
9  
12

```
for item in range(5):  
    print(item * 3)
```



using range(N) with a for loop will  
iterate with these values for item:  
0, 1, 2, ..., N-2, N-1

# Example: Maximum (Finding the Peak)

$y = 5 - (x - 2)^{**} 2$

All

Shopping

Videos

Images

News

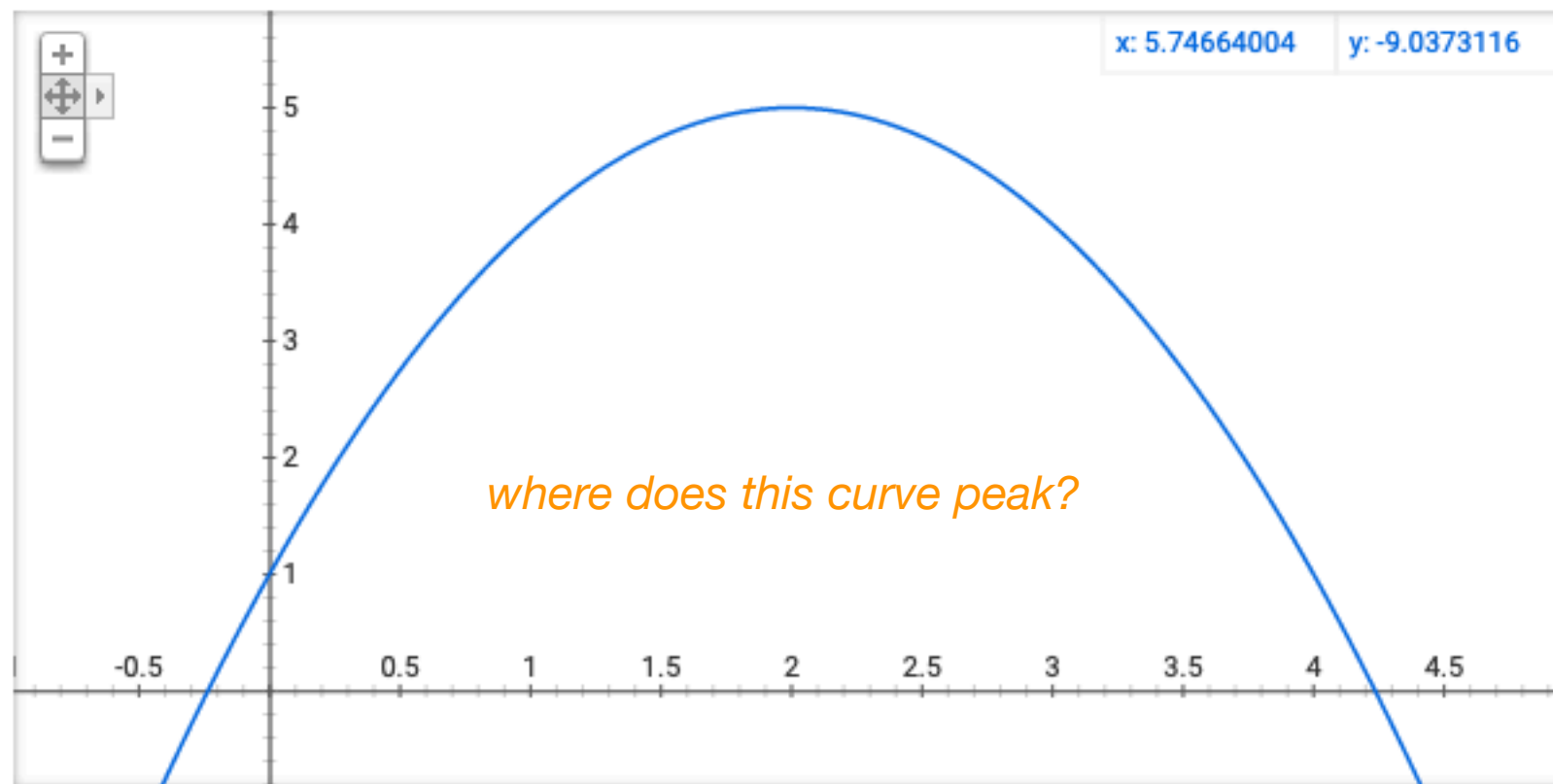
More

Settings

Tools

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

Graph for  $5-(x-2)^2$



More info



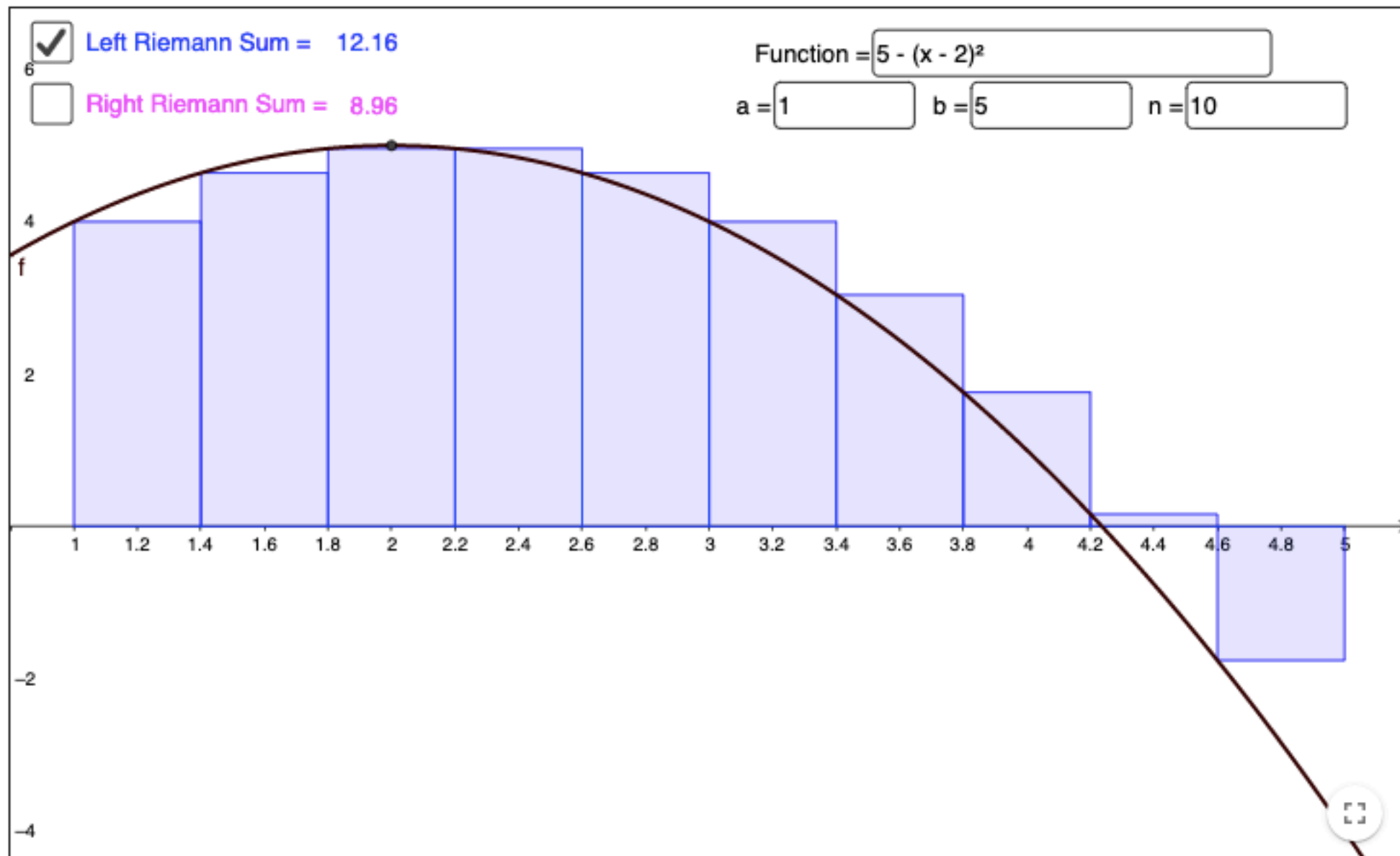
# Example: Integration (Riemann Sum)

GeoGebra

## Riemann Sum Calculator

Author: [megan.ann.martinez](#)

Topic: [Area](#), [Upper and Lower Sum or Riemann Sum](#)



# 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

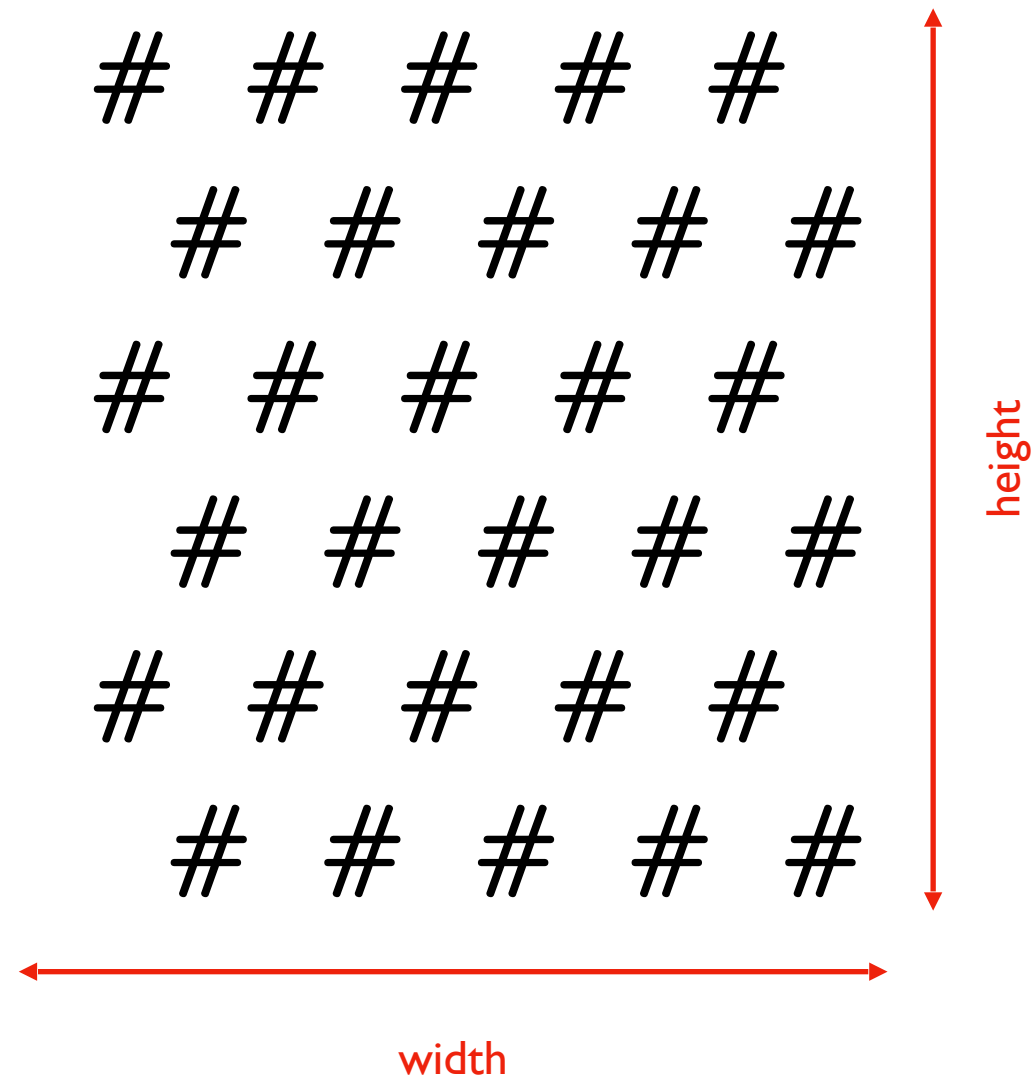
8 is not prime

9 is not prime

...

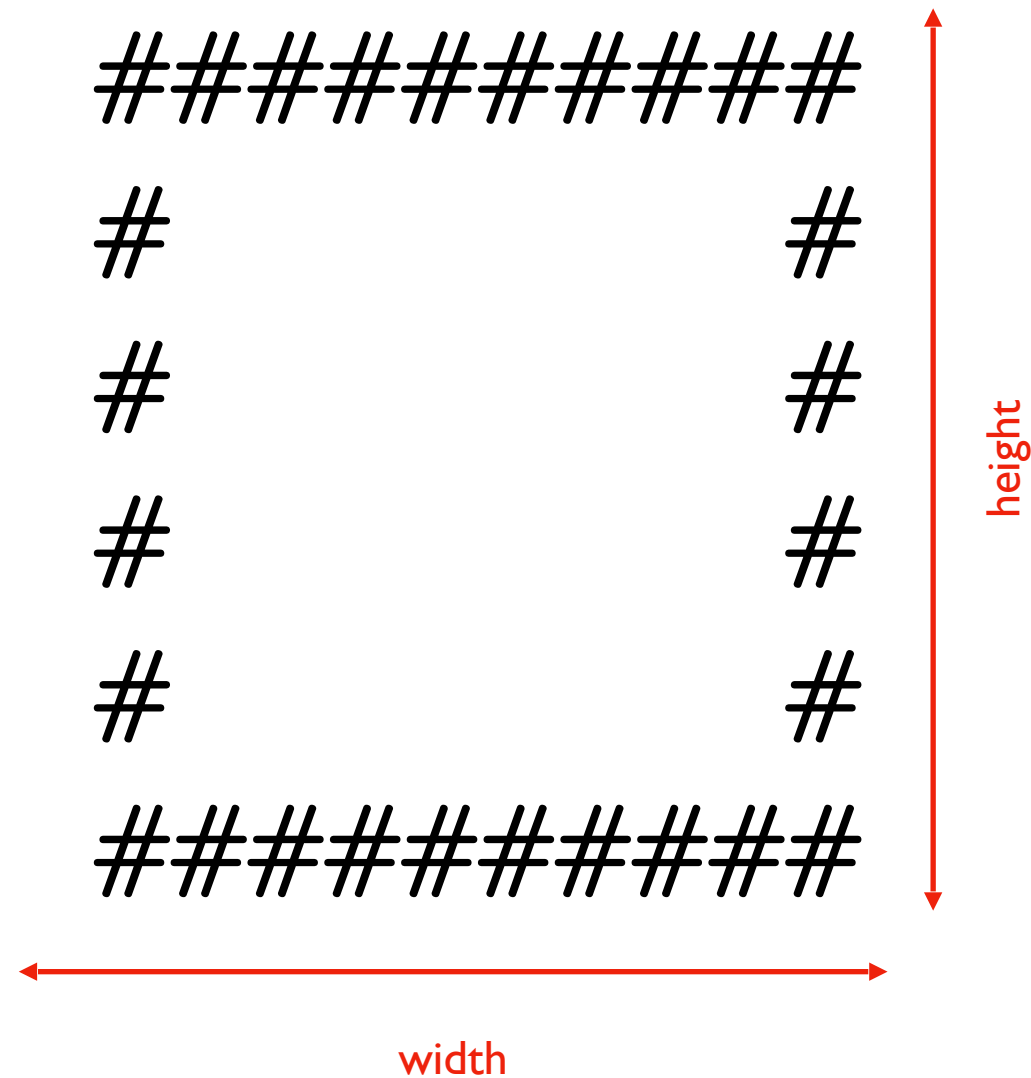
# Practice: Checkers

write a loop to draw the following:



# Practice: Border

write a loop to draw the following:



# Practice: Snake

write a loop to draw the following:

#####

#

#####

#

#####

#

#####

#

height

width

# 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!!!

# Today's Outline

Control Flow Diagrams

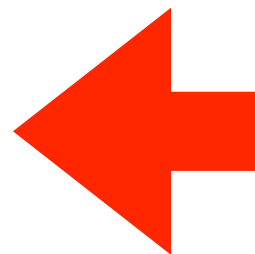
Basic syntax for “while”

Examples

Basic syntax for “for”

Examples

“break” and “continue”



**break lets us escape a loop early**

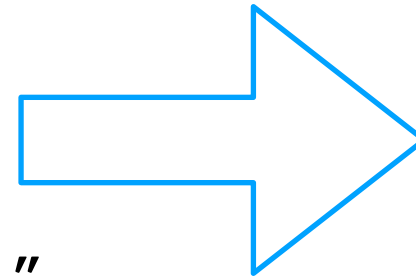
**e.g., suppose I want to find a prime number in the interval [a, b]**



**break lets us escape a loop early**

**e.g., suppose I want to find a prime number in the interval [a, b]**

```
for x in range(a, b+1):  
    is_prime_x = get_prime(x)  
    if is_prime_x:  
        print(str(x) + " is prime!")
```

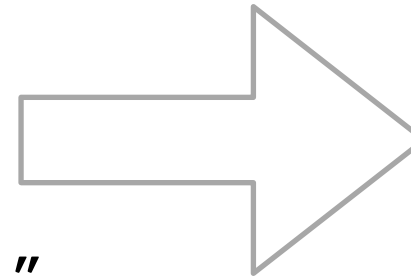


**Prints all prime  
numbers in [a,b]**

**break lets us escape a loop early**

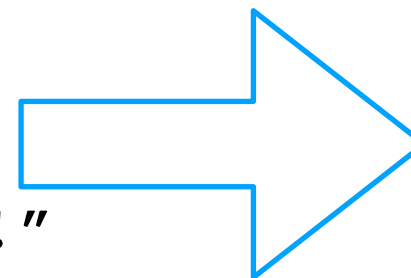
**e.g., suppose I want to find a prime number in the interval [a, b]**

```
for x in range(a, b+1):  
    is_prime_x = get_prime(x)  
    if is_prime_x:  
        print(str(x) + " is prime!")
```



Prints all prime  
numbers in [a,b]

```
for x in range(a, b+1):  
    is_prime_x = get_prime(x)  
    if is_prime_x:  
        print(str(x) + " is prime!")  
        break
```



Prints the first  
prime number in  
[a,b]

**continue lets us jump to the next loop iteration early**

**e.g., suppose I want to do a bunch of stuff for each prime number in the interval [a, b]**

```
for x in range(a, b+1):  
    is_prime_x = get_prime(x)  
    if not is_prime_x:  
        continue  
    print(str(x) + " is prime!")
```