The **for** Loop

When we drew the square, it was quite tedious. We had to move then turn, move then turn, etc. etc. four times. If we were drawing a hexagon, or an octogon, or a polygon with 42 sides, it would have been a nightmare to duplicate all that code.

A basic building block of all programs is to be able to repeat some code over and over again. In computer science, we refer to this repetitive idea as **iteration**. In this section, we will explore some mechanisms for basic iteration.

In Python, the **for** statement allows us to write programs that implement iteration. As a simple example, let's say we have some friends, and we'd like to send them each an email inviting them to our party. We don't quite know how to send email yet, so for the moment we'll just print a message for each friend.

```
1 for name in ["Joe", "Amy", "Brad", "Angelina", "Zuki", "Thandi", "Paris"]:
2     print("Hi", name, "Please come to my party on Saturday!")
3
```

ActiveCode: 1 (ch03_4)

Run | Start Audio Tour

```
Hi Joe Please come to my party on Saturday!
Hi Amy Please come to my party on Saturday!
Hi Brad Please come to my party on Saturday!
Hi Angelina Please come to my party on Saturday!
Hi Zuki Please come to my party on Saturday!
Hi Thandi Please come to my party on Saturday!
Hi Paris Please come to my party on Saturday!
```

Take a look at the output produced when you press the `run` button. There is one line printed for each friend. Here's how it works:

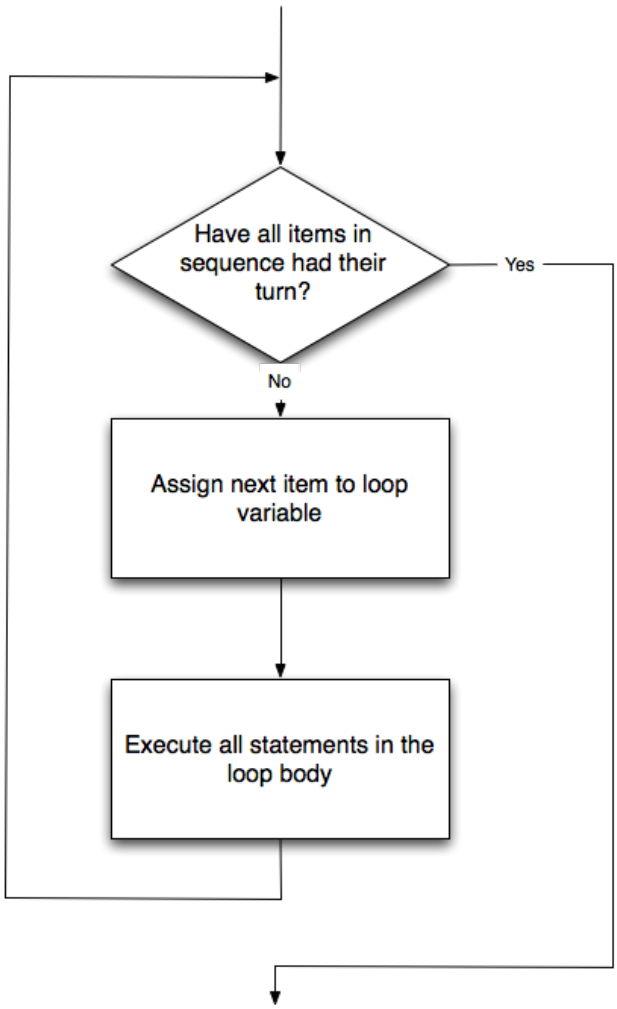
- **name** in this `for` statement is called the **loop variable**.
- The list of names in the square brackets is called a Python **list**. Lists are very useful. We will have much more to say about them later.
- Line 2 is the **loop body**. The loop body is always indented. The indentation determines exactly what statements are "in the loop". The loop body is performed one time for each name in the list.
- On each *iteration* or *pass* of the loop, first a check is done to see if there are still more items to be processed. If there are none left (this is called the **terminating condition** of the loop), the loop has finished. Program execution continues at the next statement after the loop body.
- If there are items still to be processed, the loop variable is updated to refer to the next item in the list. This means, in this case, that the loop body is executed here 7 times, and each time friendName will refer to a different friend.
- At the end of each execution of the body of the loop, Python returns to the `for` statement, to see if there are more items to be handled.

Flow of Execution of the for Loop

As a program executes, the interpreter always keeps track of which statement is about to be executed. We call this the **control flow**, or the **flow of execution** of the program. When humans execute programs, they often use their finger to point to each statement in turn. So you could think of control flow as "Python's moving finger".

Control flow until now has been strictly top to bottom, one statement at a time. We call this type of control **sequential**. Sequential flow of control is always assumed to be the default behavior for a computer program. The `for` statement changes this.

Flow of control is often easy to visualize and understand if we draw a flowchart. This flowchart shows the exact steps and logic of how the `for` statement executes.



A codelens demonstration is a good way to help you visualize exactly how the flow of control works with the for loop. Try stepping forward and backward through the program by pressing the buttons. You can see the value of `name` change as the loop iterates thru the list of friends.

```
1  for name in ["Joe", "Amy", "Brad", "Angelina", "Zuk
2      print("Hi ", name, "  Please come to my party o
```

<< First | < Back | Program terminated | Forward > | Last >>

line that has just executed

next line to execute

Program output:

```
('Hi ', 'Joe', '  Please come to my party on Saturday!')
('Hi ', 'Amy', '  Please come to my party on Saturday!')
('Hi ', 'Brad', '  Please come to my party on Saturday!')
('Hi ', 'Angelina', '  Please come to my party on Saturda
('Hi ', 'Zuki', '  Please come to my party on Saturday!')
('Hi ', 'Thandi', '  Please come to my party on Saturday
('Hi ', 'Paris', '  Please come to my party on Saturday!
```

Frames Objects

Global variables

name "Paris"

CodeLens: 1 (vtest)

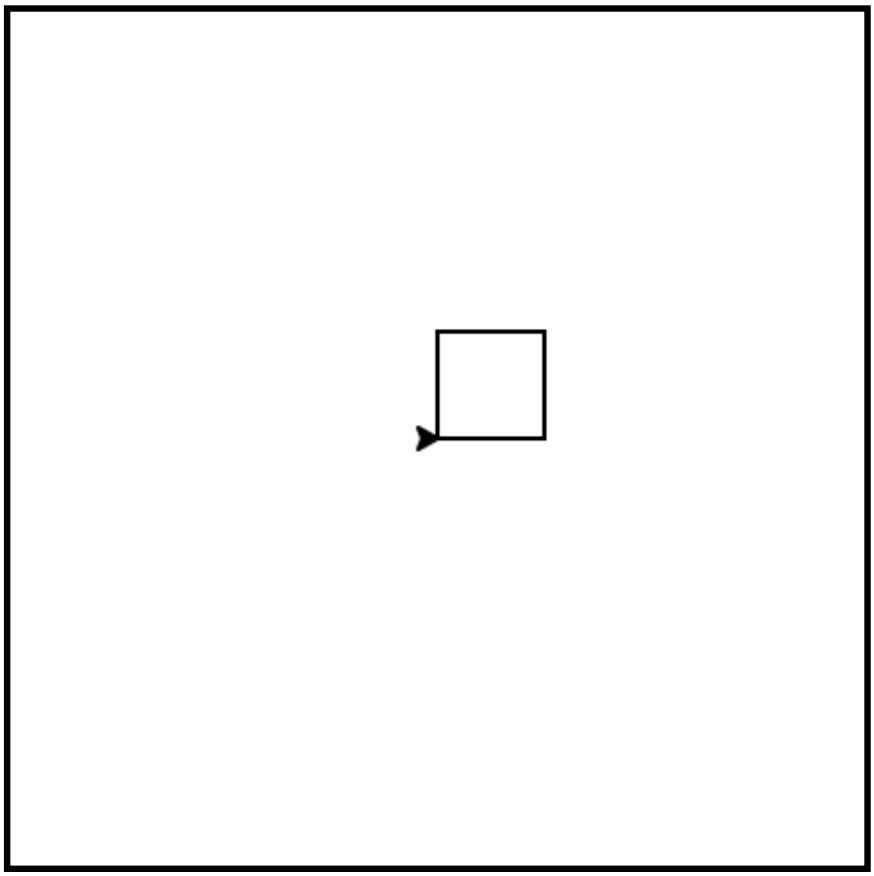
Iteration Simplifies our Turtle Program

To draw a square we'd like to do the same thing four times — move the turtle forward some distance and turn 90 degrees. We previously used 8 lines of Python code to have alex draw the four sides of a square. This next program does exactly the same thing but, with the help of the for statement, uses just three lines (not including the setup code). Remember that the for statement will repeat the forward and left four times, one time for each value in the list.

```
1  import turtle            #set up alex
2  wn = turtle.Screen()
3  alex = turtle.Turtle()
4
5  for i in [0,1,2,3]:      #repeat four times
6      alex.forward(50)
7      alex.left(90)
8
9  wn.exitonclick()
10
```

ActiveCode: 2 (ch03_for1)

Run



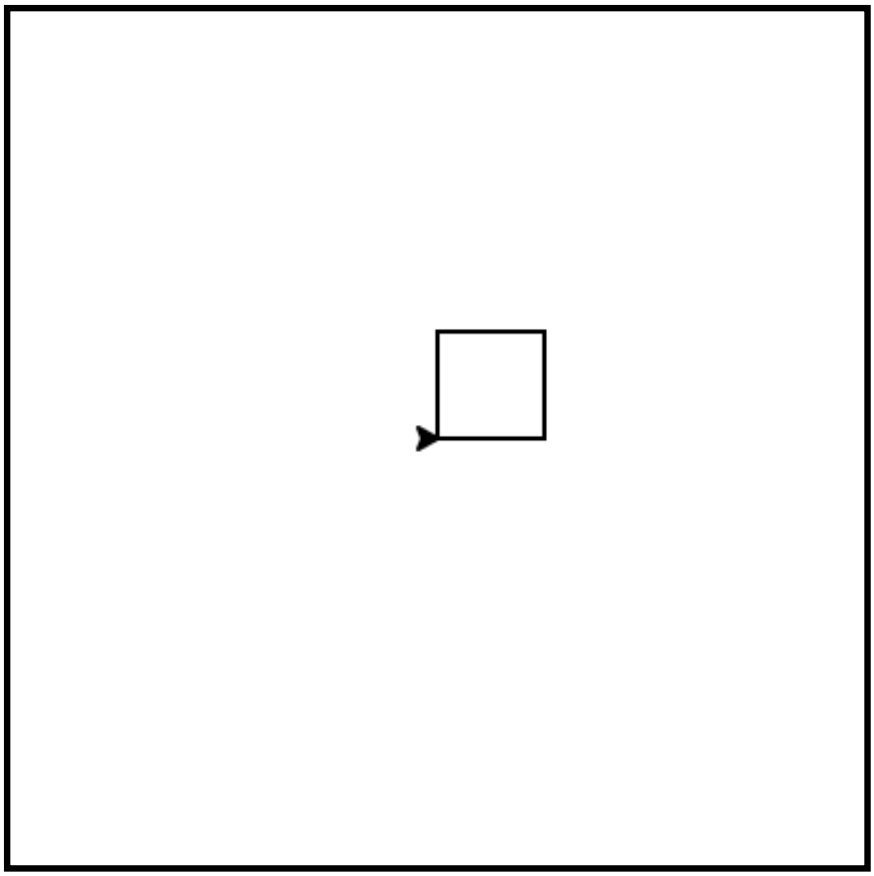
While “saving some lines of code” might be convenient, it is not the big deal here. What is much more important is that we’ve found a “repeating pattern” of statements, and we reorganized our program to repeat the pattern. Finding the chunks and somehow getting our programs arranged around those chunks is a vital skill when learning *How to think like a computer scientist*.

The values [0,1,2,3] were provided to make the loop body execute 4 times. We could have used any four values. For example, consider the following program.

```
1  import turtle            #set up alex
2  wn = turtle.Screen()
3  alex = turtle.Turtle()
4
5  for aColor in ["yellow", "red", "purple", "blue"]:      #repeat four times
6      alex.forward(50)
7      alex.left(90)
8
9  wn.exitonclick()
10
```

ActiveCode: 3 (ch03_forcolor)

Run

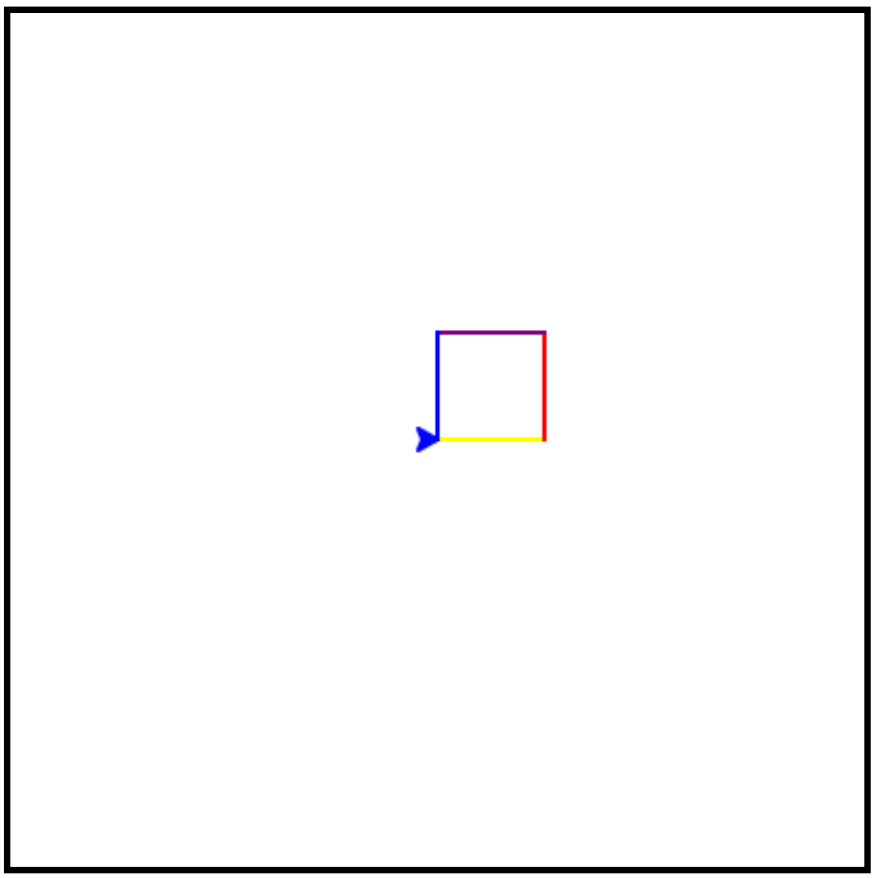


In the previous example, there were four integers in the list. This time there are four strings. Since there are four items in the list, the iteration will still occur four times. `aColor` will take on each color in the list. We can even take this one step further and use the value of `aColor` as part of the computation.

```
1  import turtle            #set up alex
2  wn = turtle.Screen()
3  alex = turtle.Turtle()
4
5  for aColor in ["yellow", "red", "purple", "blue"]:
6     alex.color(aColor)
7     alex.forward(50)
8     alex.left(90)
9
10 wn.exitonclick()
11
```

ActiveCode: 4 (colorlist)

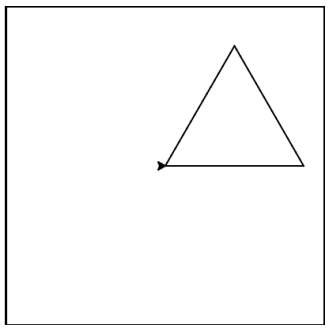
Run



In this case, the value of `aColor` is used to change the color attribute of `alex` . Each iteration causes `aColor` to change to the next value in the list.

Mixed up program

trl-14: The following program uses a turtle to draw a triangle as shown to the left, but the lines are mixed up. The program should do all necessary set-up and create the turtle. After that, iterate (loop) 3 times, and each time through the loop the turtle should go forward 175 pixels, and then turn left 120 degrees. After the loop, set the window to close when the user clicks in it.



Drag the blocks of statements from the left column to the right column and put them in the right order with the correct indention. Click on *Check Me* to see if you are right. You will be told if any of the lines are in the wrong order or are incorrectly indented.

Drag from here

Drop blocks here

```
import turtle
```

```
wn = turtle.Screen()
marie = turtle.Turtle()
```

```
# repeat 3 times
for i in [0,1,2]:
```

```
    marie.forward(175)
```

```
    marie.left(120)
```

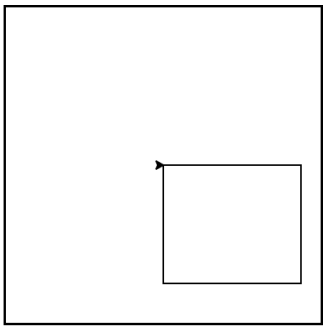
```
wn.exitonclick()
```

Check Me Reset

Perfect!

Mixed up program

trl-15: The following program uses a turtle to draw a rectangle as shown to the left, but the lines are mixed up. The program should do all necessary set-up and create the turtle. After that, iterate (loop) 2 times, and each time through the loop the turtle should go forward 175 pixels, turn right 90 degrees, go forward 150 pixels, and turn right 90 degrees. After the loop, set the window to close when the user clicks in it.



Drag the blocks of statements from the left column to the right column and put them in the right order with the correct indention. Click on *Check Me* to see if you are right. You will be told if any of the lines are in the wrong order or are incorrectly indented.

Drag from here

Drop blocks here

```
import turtle
wn = turtle.Screen()
carlos = turtle.Turtle()
```

```
# repeat 2 times
for i in [1,2]:
```

```
    carlos.forward(175)
```

```
    carlos.right(90)
```

```
    carlos.forward(150)
    carlos.right(90)
```

```
wn.exitonclick()
```

Check Me Reset

Perfect!

Check your understanding

trl-16: In the following code, how many lines does this code print?

```
for number in [5, 4, 3, 2, 1, 0]:
    print("I have", number, "cookies.  Iím going to eat one.")
```

- a) 1
- b) 5
- c) 6
- d) 10

Check Me | Compare Me

Correct!! The loop body will execute (and print one line) for each of the 6 elements in the list [5, 4, 3, 2, 1, 0].

trl-17: How does python know what lines are contained in the loop body?

- a) They are indented to the same degree from the loop header.
- b) There is always exactly one line in the loop body.
- c) The loop body ends with a semi-colon (;) which is not shown in the code above.

Check Me | Compare Me

Correct!! The loop body can have any number of lines, all indented from the loop header.

trl-18: In the following code, what is the value of number the second time Python executes the loop?

```
for number in [5, 4, 3, 2, 1, 0]:
    print("I have", number, "cookies.  Iím going to eat one.")
```

- a) 2

- (●) b) 4
- ( ) c) 5
- ( ) d) 1

Check Me | Compare Me

Correct!! Yes, Python will process the items from left to right so the first time the value of number is 5 and the second time it is 4.

trl-19: Consider the following code:

```
for aColor in ["yellow", "red", "green", "blue"]:
   alex.forward(50)
   alex.left(90)
```

What does each iteration through the loop do?

- ( ) a) Draw a square using the same color for each side.
- ( ) b) Draw a square using a different color for each side.
- (●) c) Draw one side of a square.

Check Me | Compare Me

Correct!! The body of the loop only draws one side of the square. It will be repeated once for each item in the list. However, the color of the turtle never changes.