Python Coding Schools

8th lesson: While, For Loop

Seed Academy

Agenda

- wk1. Installing Python, HelloWorld
- wk2. Arithmetic Operators
- wk3. Data Types : Integer, Floating point, Boolean, String
- wk4. Data Structures: List
- wk5. Data Structures: Set, Tuples
- wk6. Data Structures: Dictionary

Agenda

- wk7. Control flows : IF statement
- wk8. Loops: While, For
- wk9. Function
- wk10. Class
- wk11. Data Visualization

Class materials

https://github.com/TaeheeJeong/seedacademy

https://github.com/TaeheeJeong/SummerCoding2023

Today’s topic: Loop

- Conditional: IF statement
- While Loop
- For Loop

Repeated Steps

- Loops (repeated steps) have iteration variables that change each time through a loop.
- Often these iteration variables go through a sequence of numbers.

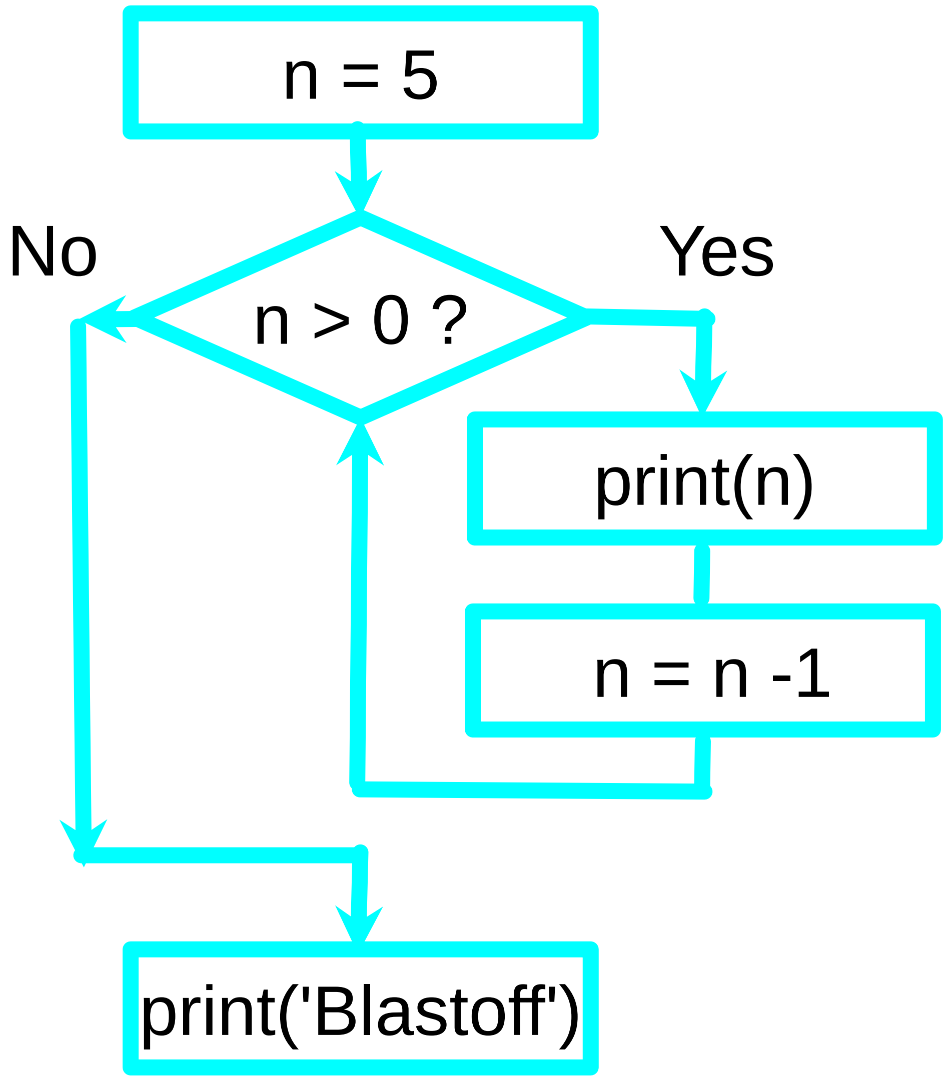
Program:

```
n = 5
while n > 0 :
    print(n)
    n = n – 1
print('Blastoff!')
print(n)
```

Output:

```
5
4
3
2
1
Blastoff!
0
```

Repeated Steps



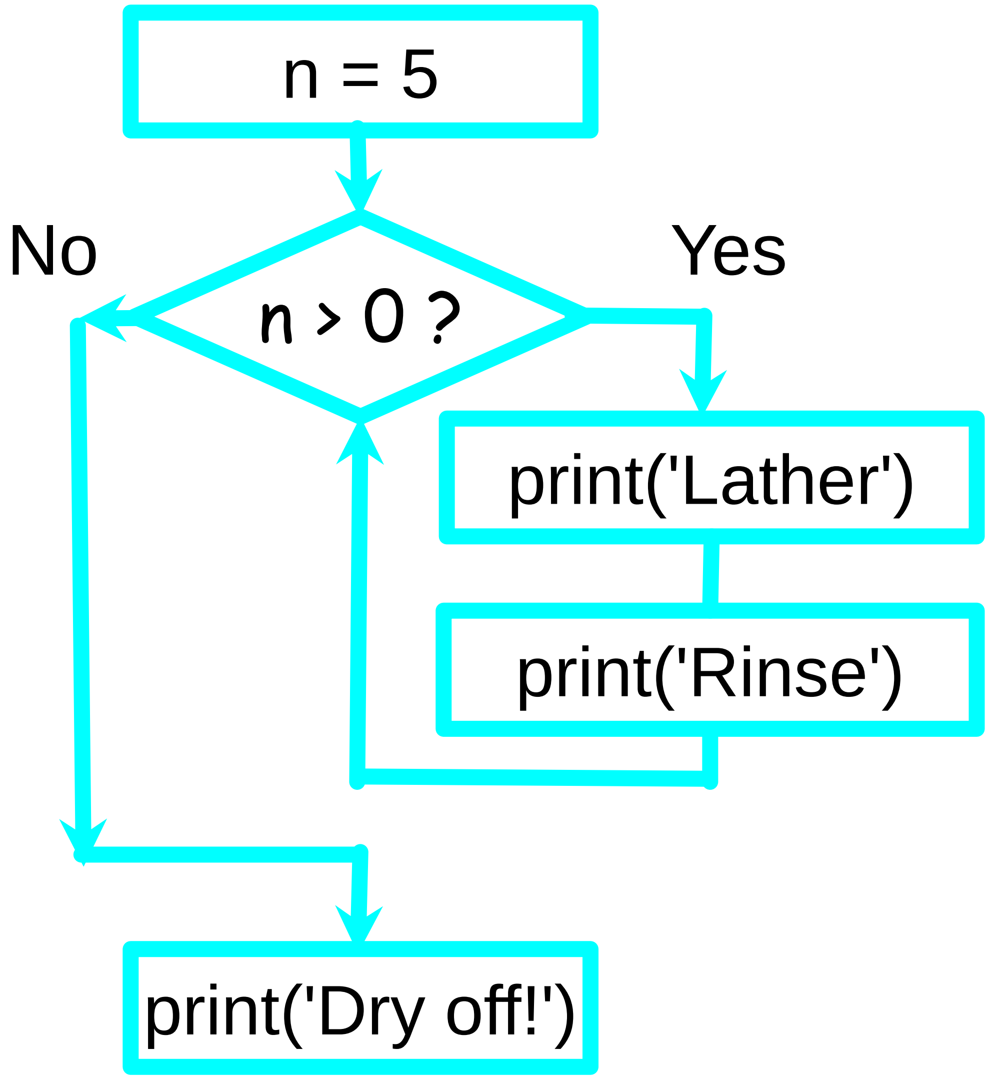
Program:

```
n = 5
while n > 0 :
    print(n)
    n = n – 1
print('Blastoff!')
print(n)
```

Output:

```
5
4
3
2
1
Blastoff!
0
```

Source: Python for Everybody, Charles Severance, University of Michigan School of Information

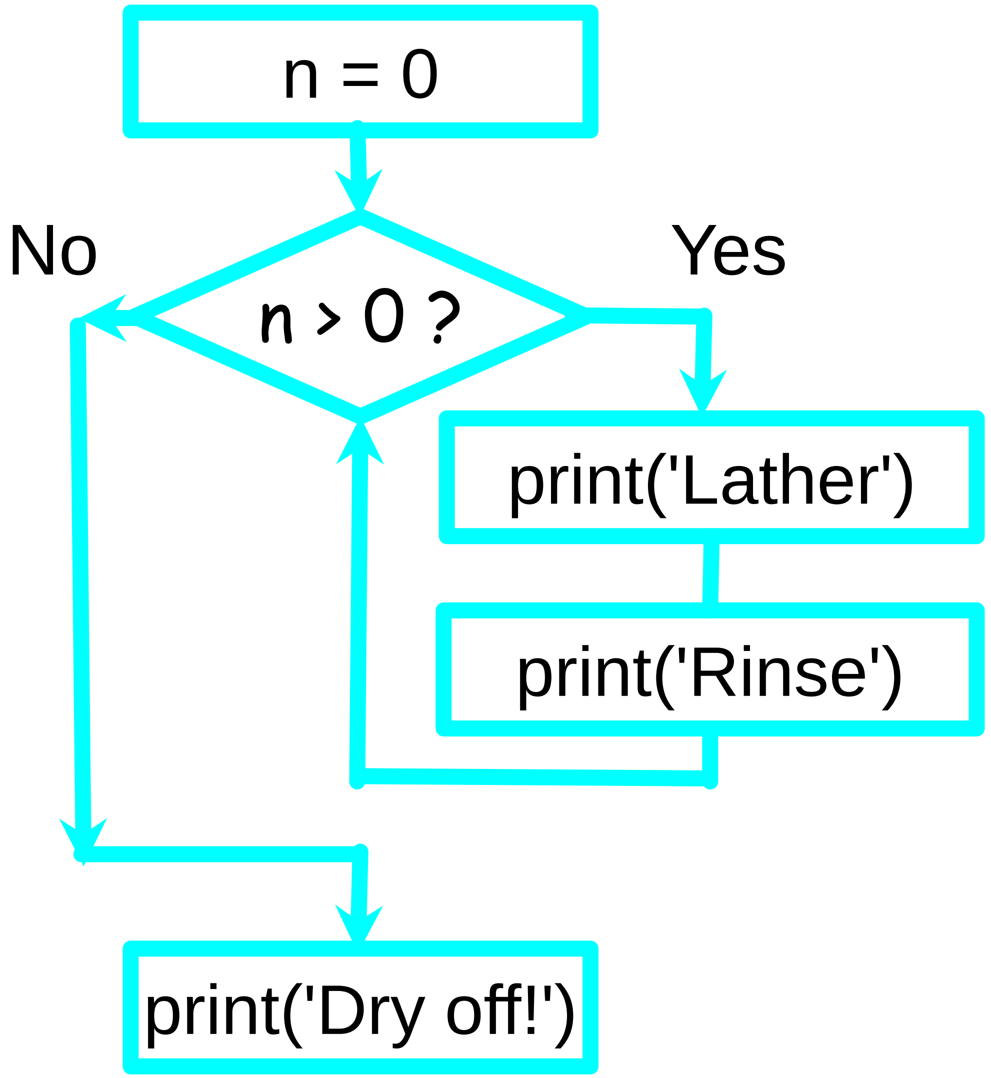


An Infinite Loop

```
n = 5
while n > 0 :
    print('Lather')
    print('Rinse')
print('Dry off!')
```

What is wrong with this loop?

Source: Python for Everybody, Charles Severance, University of Michigan School of Information



Another Loop

```
n = 0
while n > 0 :
    print('Lather')
    print('Rinse')
print('Dry off!')
```

What is this loop doing?

Source: Python for Everybody, Charles Severance, University of Michigan School of Information

Indefinite Loops

- “While loops” are called “indefinite loops” because they keep going until a logical condition becomes False
- The loops we have seen so far are pretty easy to examine to see if they will terminate or if they will be “infinite loops”
- Sometimes it is a little harder to be sure if a loop will terminate

Breaking Out of a Loop

- The break statement ends the current loop and jumps to the statement immediately following the loop
- It is like a loop test that can happen anywhere in the body of the loop

```
while True:
    line = input('> ')
    if line == 'done' :
        break
    print(line)
print('Done!')
```

Finishing an Iteration with continue

The continue statement ends the current iteration and jumps to the top of the loop and starts the next iteration

```
while True:
    line = input('> ')
    if line[0] == '#' :
        continue
    if line == 'done' :
        break
    print(line)
print('Done!')
```

Definite Loops

Quite often we have a list of items of the lines in a file - effectively a finite set of things

We can write a loop to run the loop once for each of the items in a set using the Python for construct

These loops are called “definite loops” because they execute an exact number of times

We say that “definite loops iterate through the members of a set”

A Simple Definite Loop

Definite loops (for loops) have explicit iteration variables that change each time through a loop.
These iteration variables move through the sequence or set.

```
for i in [5, 4, 3, 2, 1] :
    print(i)
print('Blastoff!')
```

Output
```
5
4
3
2
1
Blastoff!
```

Looking at in...

The iteration variable "iterates" through the sequence (ordered set)

The block (body) of code is executed once for each value in the sequence

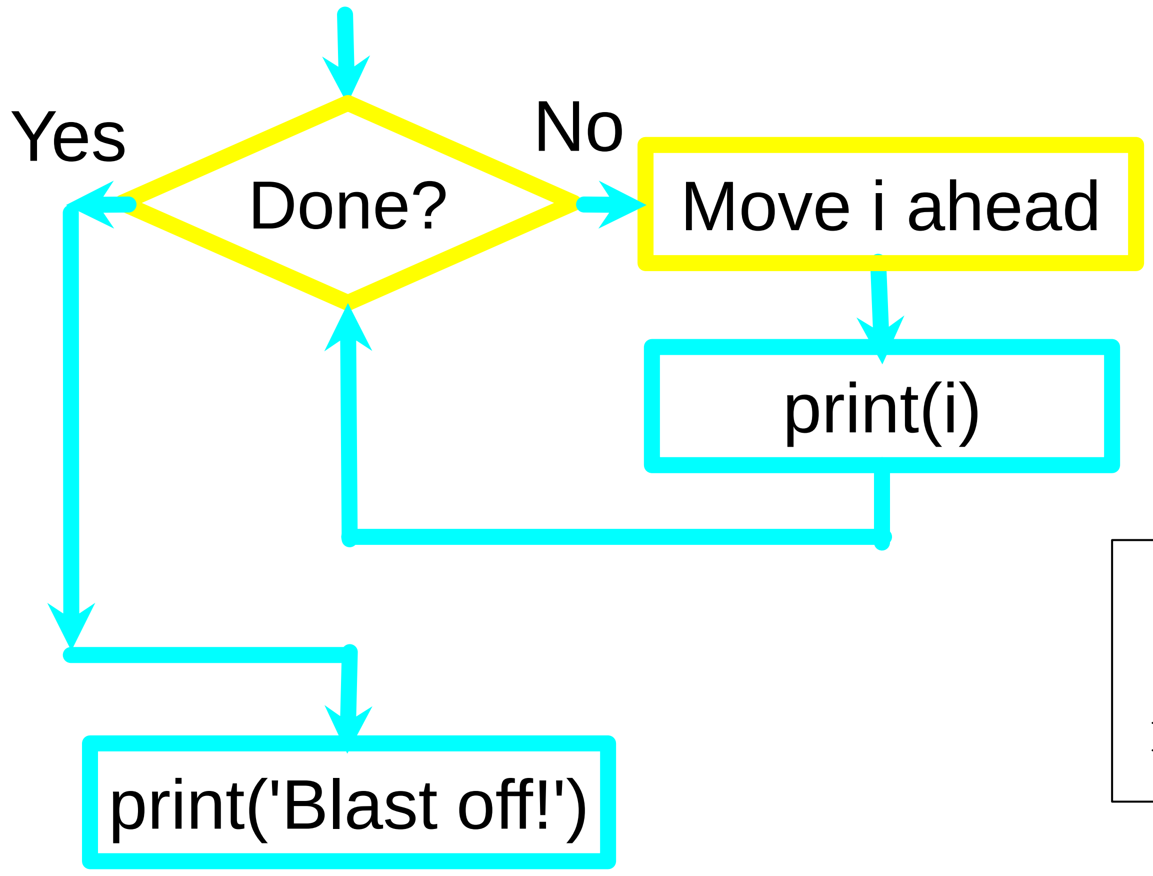
The iteration variable moves through all of the values in the sequence

Five-element sequence

Iteration variable

```
for i in [5, 4, 3, 2, 1] :
    print(i)
```

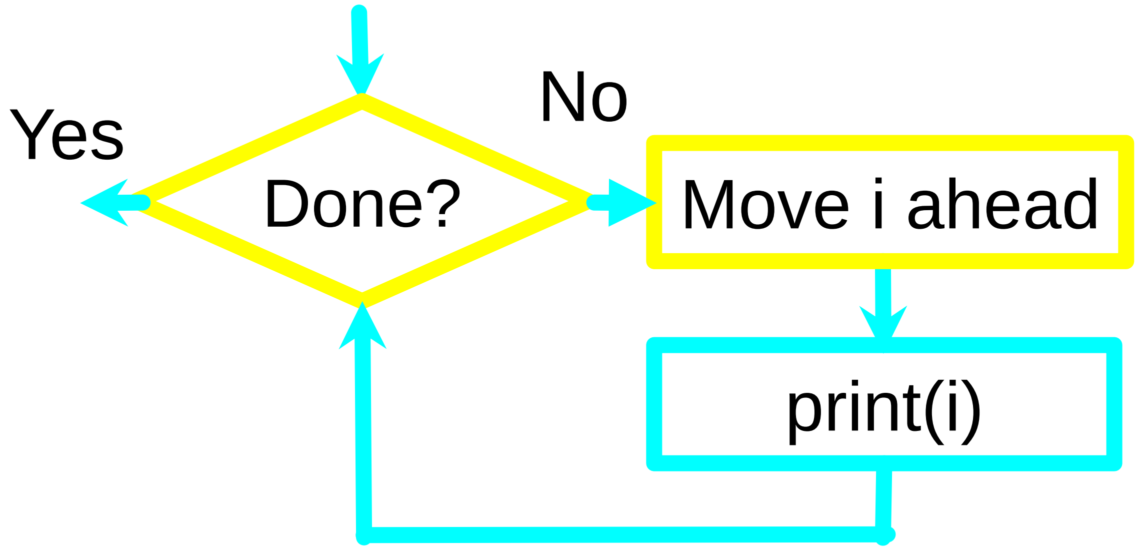
A Simple Definite Loop



```
for i in [5, 4, 3, 2, 1] :
    print(i)
print('Blastoff!')
```

Output
```
5
4
3
2
1
Blastoff!
```

```
for i in [5, 4, 3, 2, 1] :
    print(i)
```

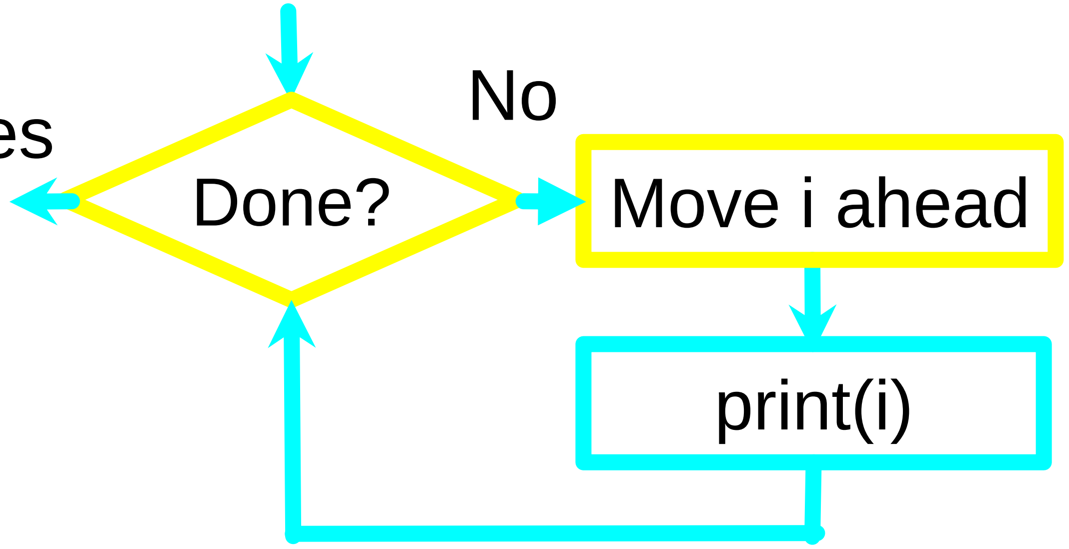


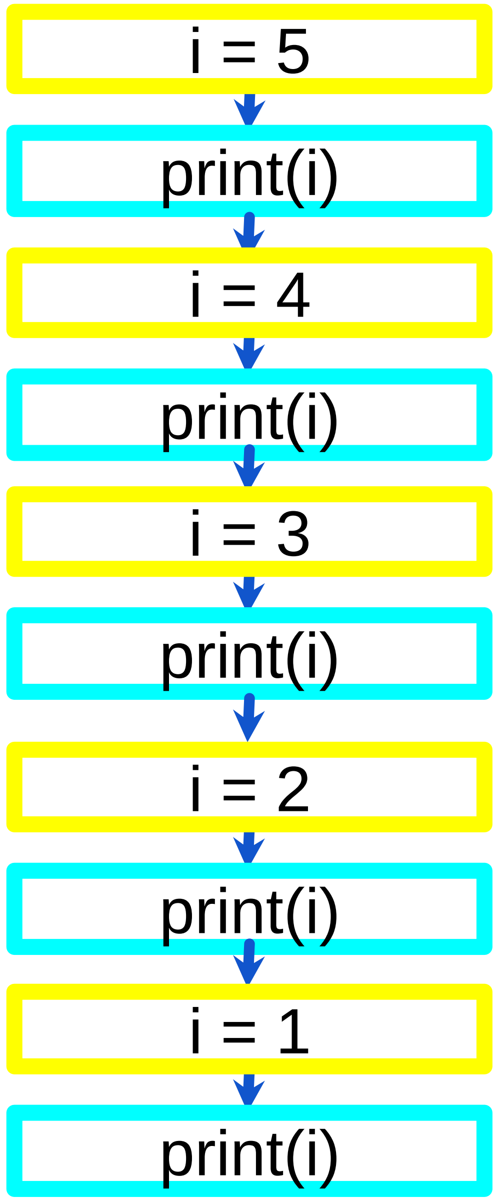
The iteration variable “iterates” through the sequence (ordered set)

The block (body) of code is executed once for each value in the sequence

The iteration variable moves through all of the values in the sequence

```
for i in [5, 4, 3, 2, 1] :
    print(i)
```





Source: Python for Everybody, Charles Severance, University of Michigan School of Information

A Simple Definite Loop

```
for i in [5, 4, 3, 2, 1] :
    print(i)
print('Blastoff!')
```

Output

```
5
4
3
2
1
Blastoff!
```

Looping Through a Set

```
print('Before')
for thing in [9, 41, 12, 3, 74, 15] :
    print(thing)
print('After')
```

Output

```
Before
9
41
12
3
74
15
After
```

For Loop with Strings

```
friends = ['Joseph', 'Glenn', 'Sally']
for friend in friends :
   print('Happy New Year:', friend)
print('Done!')
```

Output

```
Happy New Year: Joseph
Happy New Year: Glenn
Happy New Year: Sally

Done!
```

begin/end Blocks

```
for i in range(5) :
    print(i)
    if i > 2 :
        print('Bigger than 2')
    print('Done with i', i)
print('All Done')
```

Finding the Largest value

We make a variable that contains the largest value we have seen so far.
If the current number we are looking at is larger,
it is the new largest value we have seen so far.

```
largest_so_far = -1
print('Before', largest_so_far)
for the num in [9, 41, 12, 3, 74, 15] :
   if the_num > largest_so_far :
      largest_so_far = the_num
   print(largest_so_far, the_num)

print('After', largest_so_far)
```

Output

```
Before -1
9  9
41  41
41 12
41  3
74  74
74  15
After 74
```

Finding the smallest value?

Quiz: How would we change the code to make it find the smallest value in the list?

```
largest_so_far = -1
print('Before', largest_so_far)
for the_num in [9, 41, 12, 3, 74, 15] :
    if the_num > largest_so_far :
        largest_so_far = the_num
    print(largest_so_far, the_num)

print('After', largest_so_far)
```

Finding the smallest value

```
smallest_so_far = ?
print('Before’, smallest_so_far)
for the_num in [9, 41, 12, 3, 74, 15] :
   if the_num < smallest_so_far :
      smallest_so_far = the_num
   print(smallest_so_far, the_num)

print('After’, smallest_so_far)
```

Counting in a Loop

To count how many times we execute a loop, we introduce a counter variable that starts at 0 and we add one to it each time through the loop.

```
count = 0
print('Before’, count)
for value in [9, 41, 12, 3, 74, 15] :
    count = count + 1
    print(count, value)
print('After’, count)
```

Output

```
Before 0
1 9
2 41
3 12
4 3
5 74
6 15
After 6
```

Summing in a Loop

To add up a value we encounter in a loop, we introduce a sum variable that starts at 0 and we add the value to the sum each time through the loop.

```
sum = 0
print('Before’, sum)
for value in [9, 41, 12, 3, 74, 15] :
    sum = sum + value
    print(sum, value)
print('After’, sum)
```

Output

```
Before 0
9 9
50 41
62 12
65 3
139 74
154 15
After 154
```

Source: Python for Everybody, Charles Severance, University of Michigan School of Information

Finding the Average in a Loop

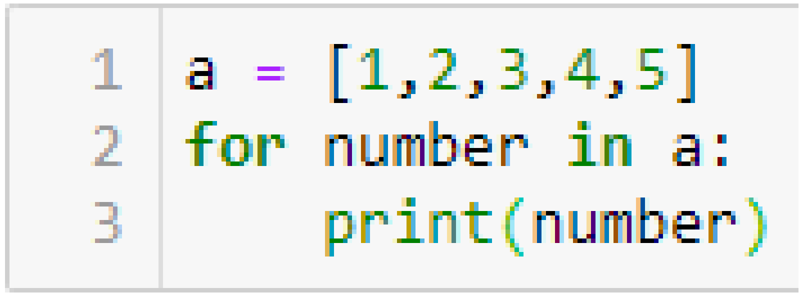
An average just combines the counting and sum patterns and divides when the loop is done.

```
count = 0
sum = 0
print('Before', count, sum)
for value in [9, 41, 12, 3, 74, 15] :
    count = count + 1
    sum = sum + value
    print(count, sum, value)
print('After', count, sum, sum / count)
```

Output

```
Before 0 0
1 9 9
2 50 41
3 62 12
4 65 3
5 139 74
6 154 15
After 6 154 25.666
```

Recap: For loop

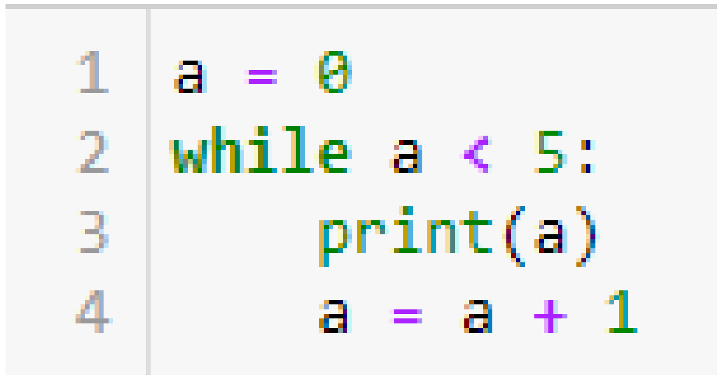


```python
a = [1,2,3,4,5]
for number in a:
    print(number)
```

Output

```
1
2
3
4
5
```

Recap: While loop



```python
a = 0
while a < 5:
    print(a)
    a = a + 1
```

Output

```
0
1
2
3
4
```

Acknowledgements / Contributions

Initial Development: Charles Severance, University of Michigan School of Information

Modification: Taehee Jeong, San Jose State University