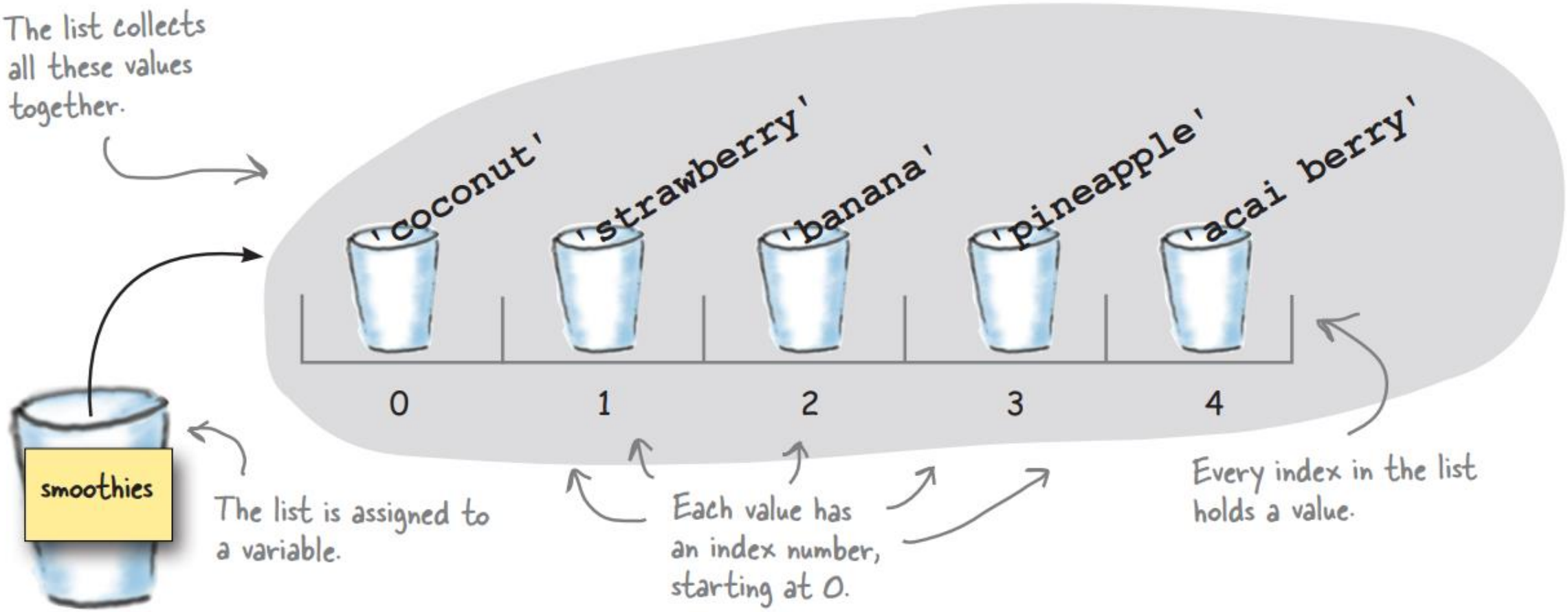
Python: Lists & Iterations


The list collects all these values together.
'coconut'
'strawberry'
'banana'
'pineapple'
'acai berry'
0
1
2
3
4
smoothies
The list is assigned to a variable.
Each value has an index number, starting at 0.
Every index in the list holds a value.

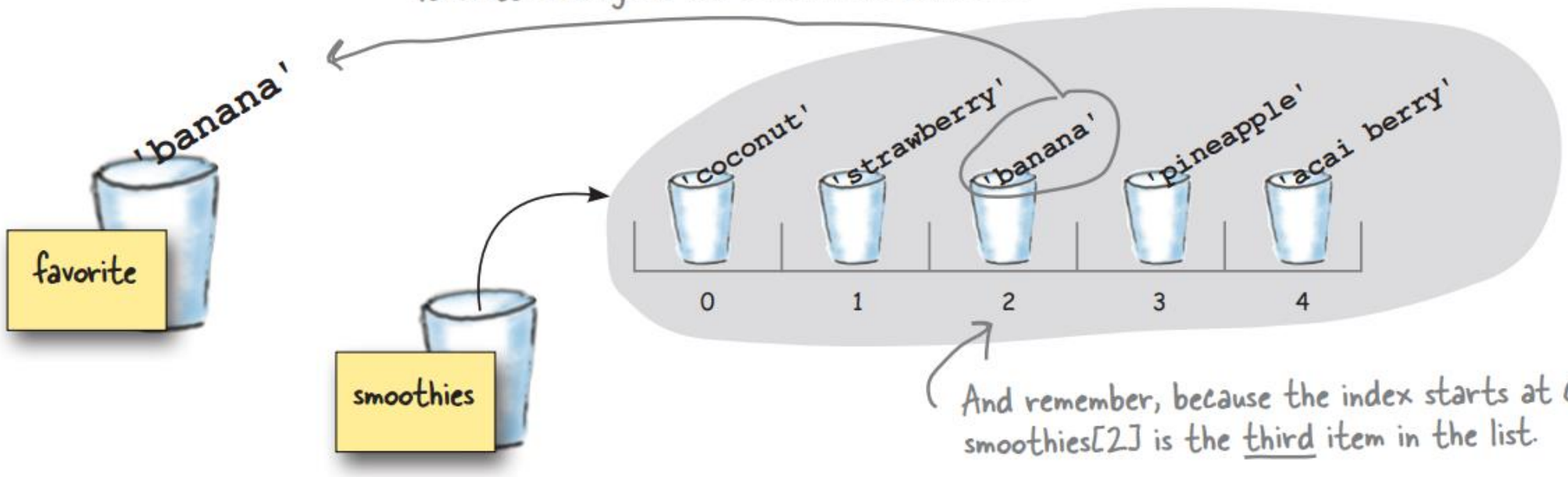
How to access a list item

```
favorite = smoothies[2]
```

This evaluates to the value of the smoothies list at index 2 (which is 'banana'), which is then assigned to the variable favorite.

To get an element from a list, you need both the variable name of the list, and the index of the value you want.

favorite is assigned the value in smoothies[2].



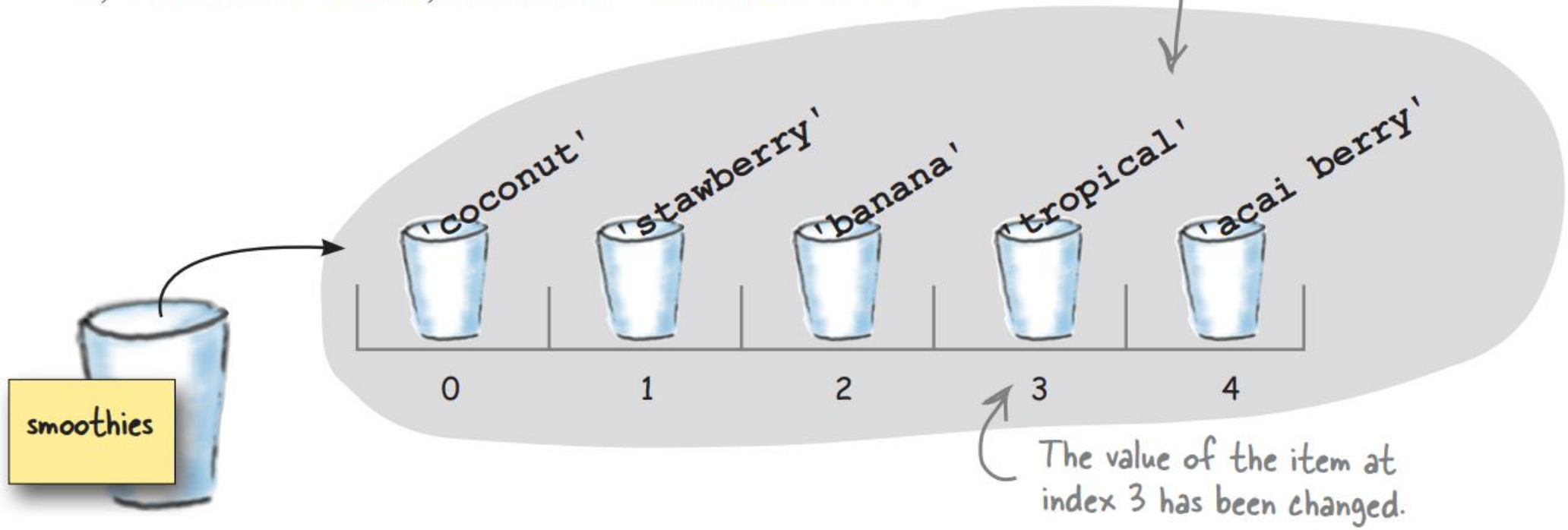
And remember, because the index starts at 0, smoothies[2] is the third item in the list.

Updating a value in the list

```
smoothies[3] = 'tropical'
```

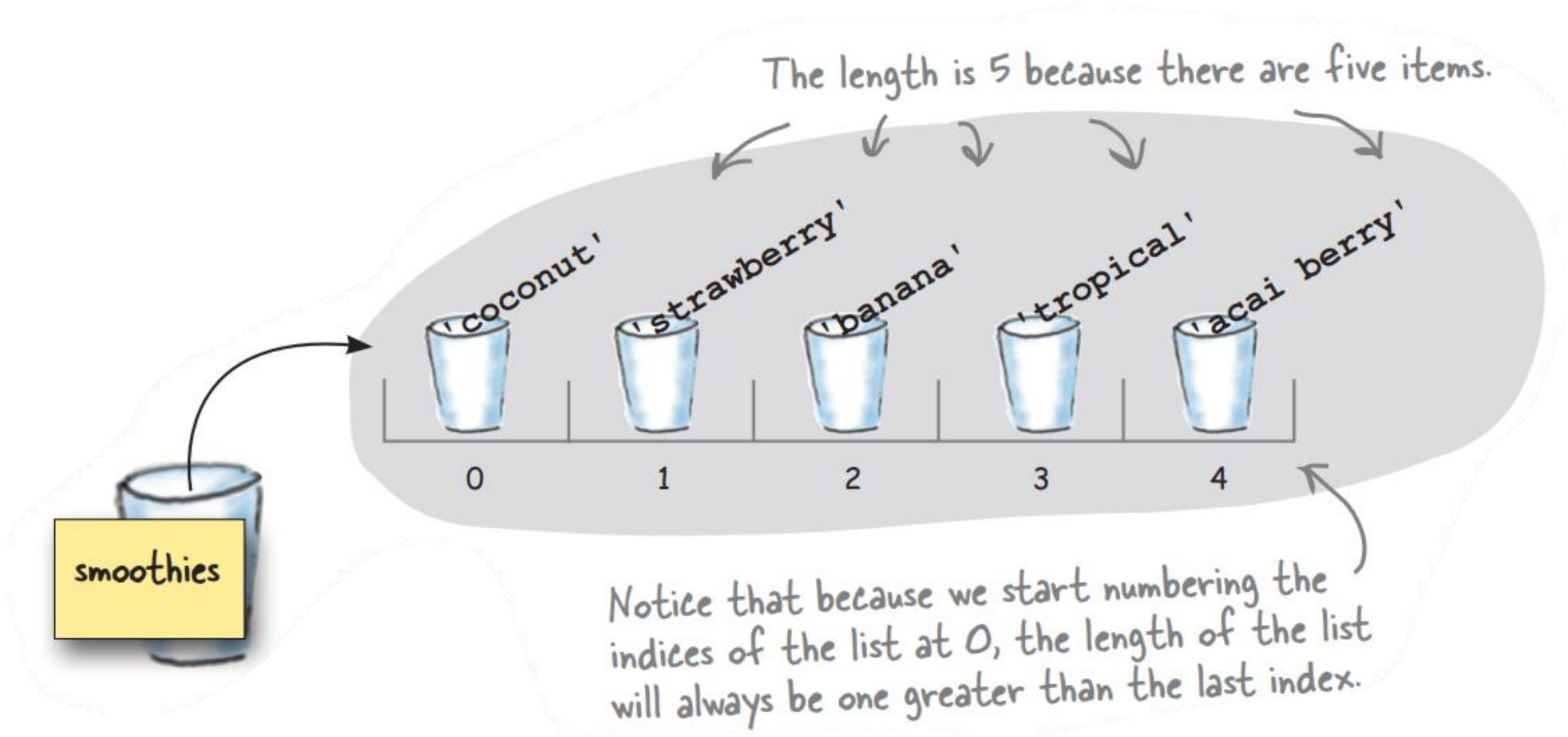
This sets the value of the item at index 3 (previously 'pineapple') to a new value, 'tropical'.

So, after this line of code, the smoothie list will look like this:



How big is that list, anyway?

```
length = len(smoothies)
```

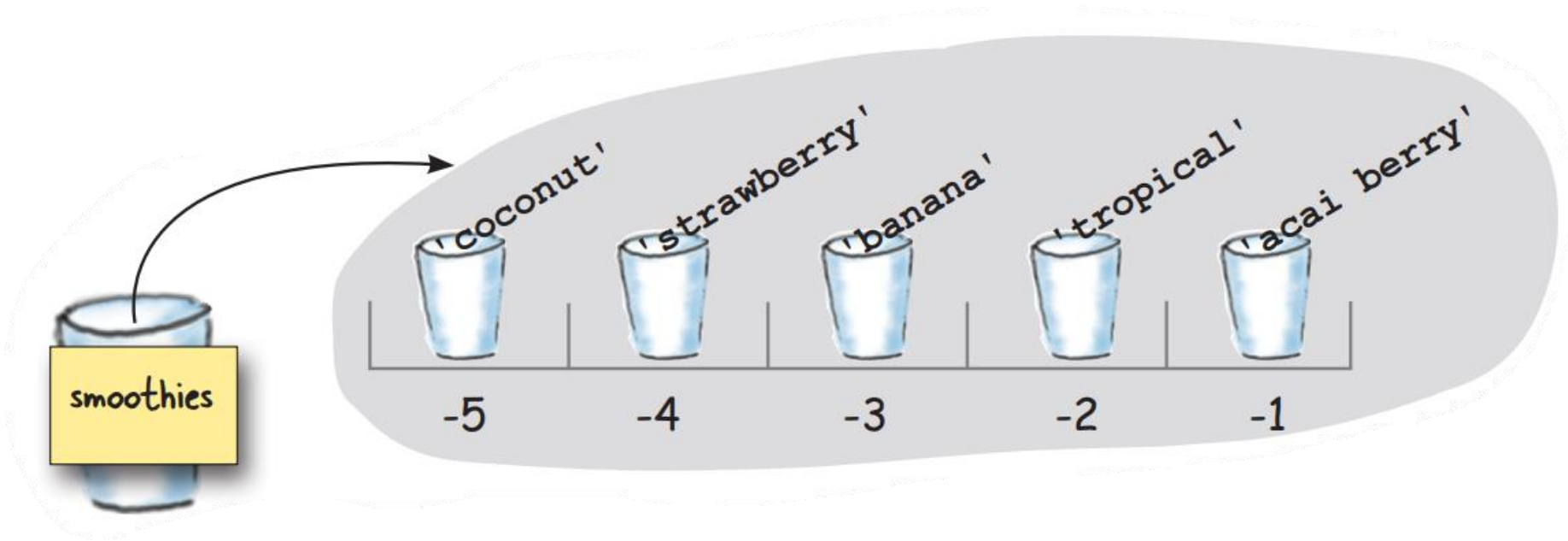


Accessing the last item in the list

```
length = len(smoothies)
last = smoothies[length-1]
print(last)
```

This is a common technique in most languages: figure out the length of the list and then subtract one to get the index of the last item.

Python makes this even easier



Using Python's negative indices

```
last = smoothies[-1]
second_last = smoothies[-2]
third_last = smoothies[-3]
print(last)
print(second_last)
print(third_last)
```

With an index of –1 we get the last item.

And –2 gets us the second to the last item.

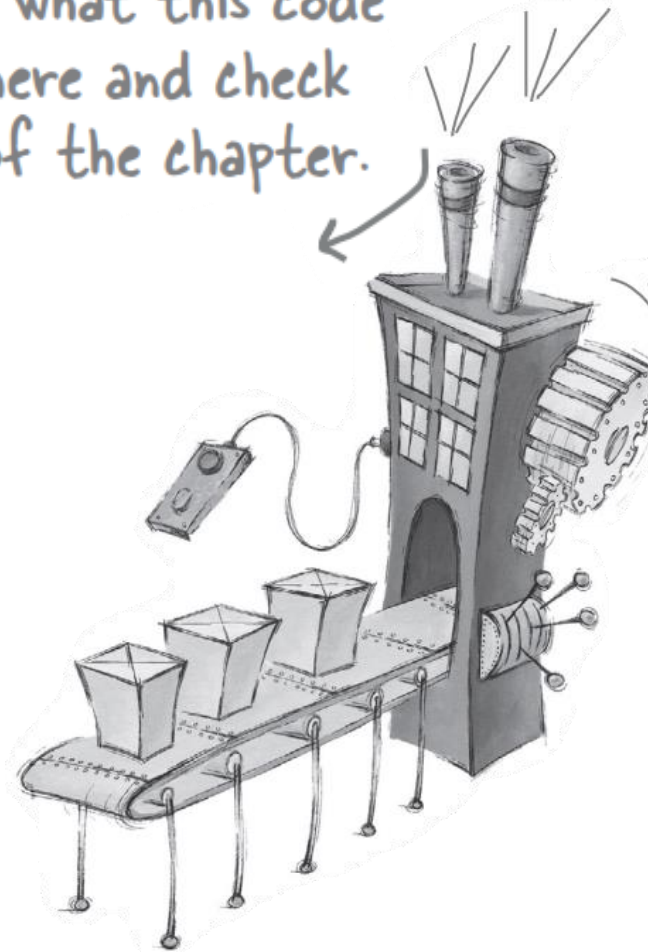
Likewise –3 gets us the third to the last item.

The Thing-A-Ma-Jig

View this as a character-building exercise—spend some real time on this and make your brain work; it will thank you later.

```
characters = ['t', 'a', 'c', 'o']

output = ''
length = len(characters)
i = 0
while (i < length):
    output = output + characters[i]
    i = i + 1

length = length * -1
i = -2

while (i >= length):
    output = output + characters[i]
    i = i - 1

print(output)
```

When you think you know what this code does, write your answer here and check the solution at the end of the chapter.



Try these as an alternative for the characters list above:

```
characters = ['a', 'm', 'a', 'n', 'a', 'p', 'l', 'a', 'n', 'a', 'e']
```

or

```
characters = ['w', 'a', 's', 'i', 't', 'a', 'r']
```

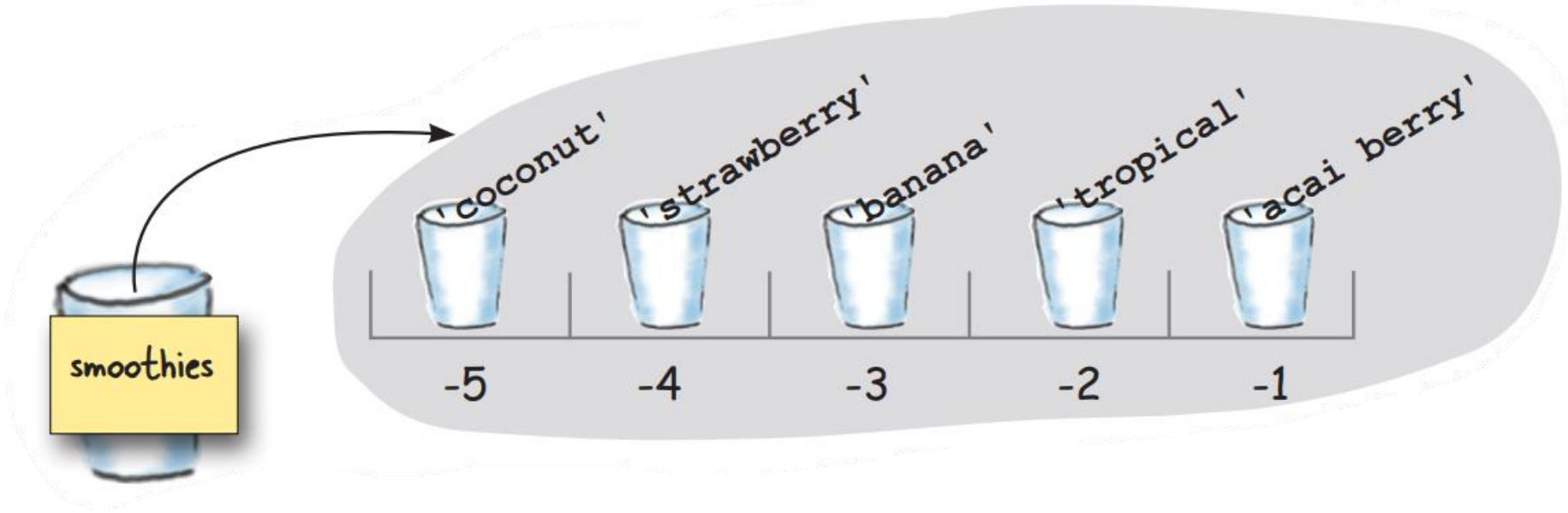
```
scores = [60, 50, 60, 58, 54, 54, 58, 50, 52, 54, 48, 69,
          34, 55, 51, 52, 44, 51, 69, 64, 66, 55, 52, 61,
          46, 31, 57, 52, 44, 18, 41, 53, 55, 61, 51, 44]

i = 0
length = len(scores)
while i < length:
  print('Bubble solution #' + str(i), 'score:', scores[i])
  i = i + 1
```

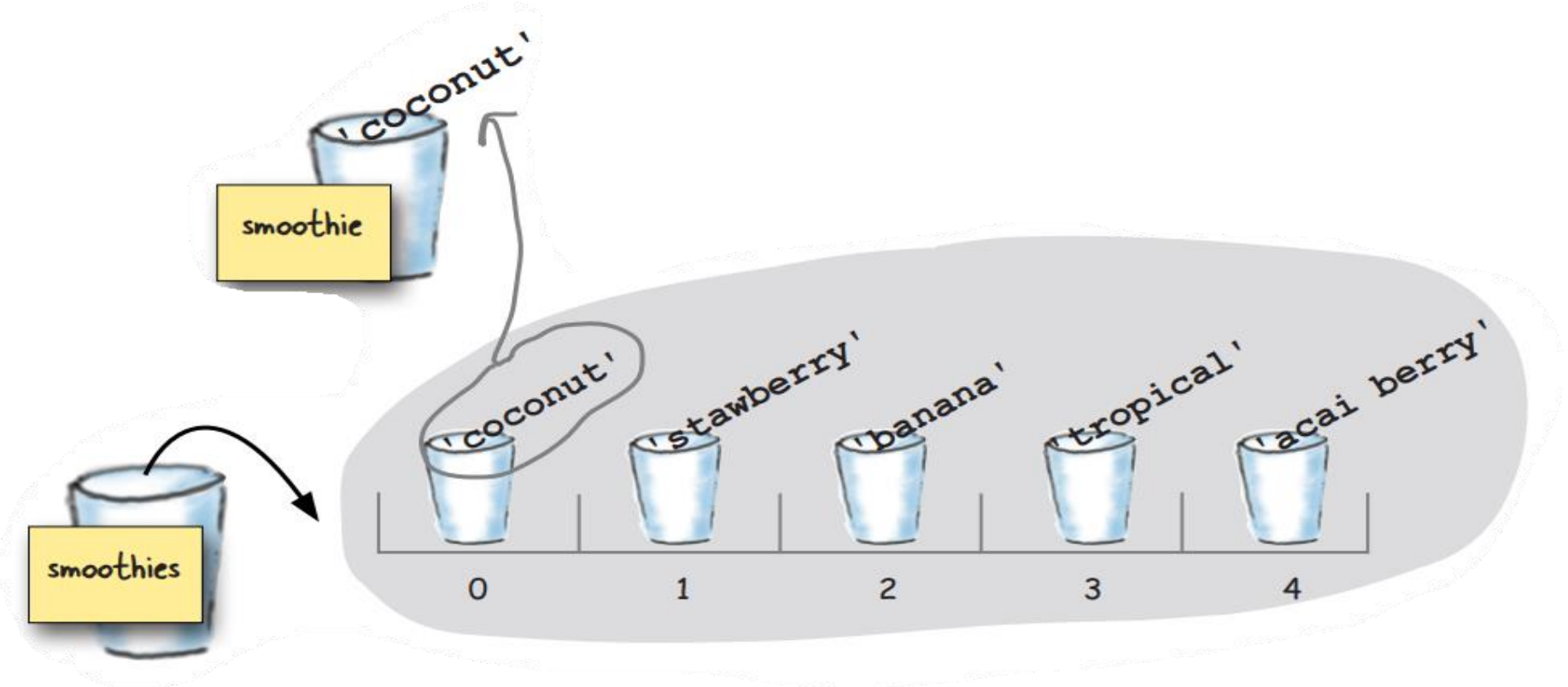
Here we just concatenate "Bubble solution #" with i before it is passed to print. And we make sure to use the str function so we have a string representation of i.

Python 3.6.0 Shell

```
Bubble solution #0 score: 60
Bubble solution #1 score: 50
Bubble solution #2 score: 60
 ...
Bubble solution #34 score: 51
Bubble solution #35 score: 44
>>>
```

Think of the `for` loop as the `while` loop's cousin—the two basically do about the same thing, except we typically use a `while` loop when we're looping over some *condition*, and a `for` loop when we're *iterating over* a sequence of values (like a list).

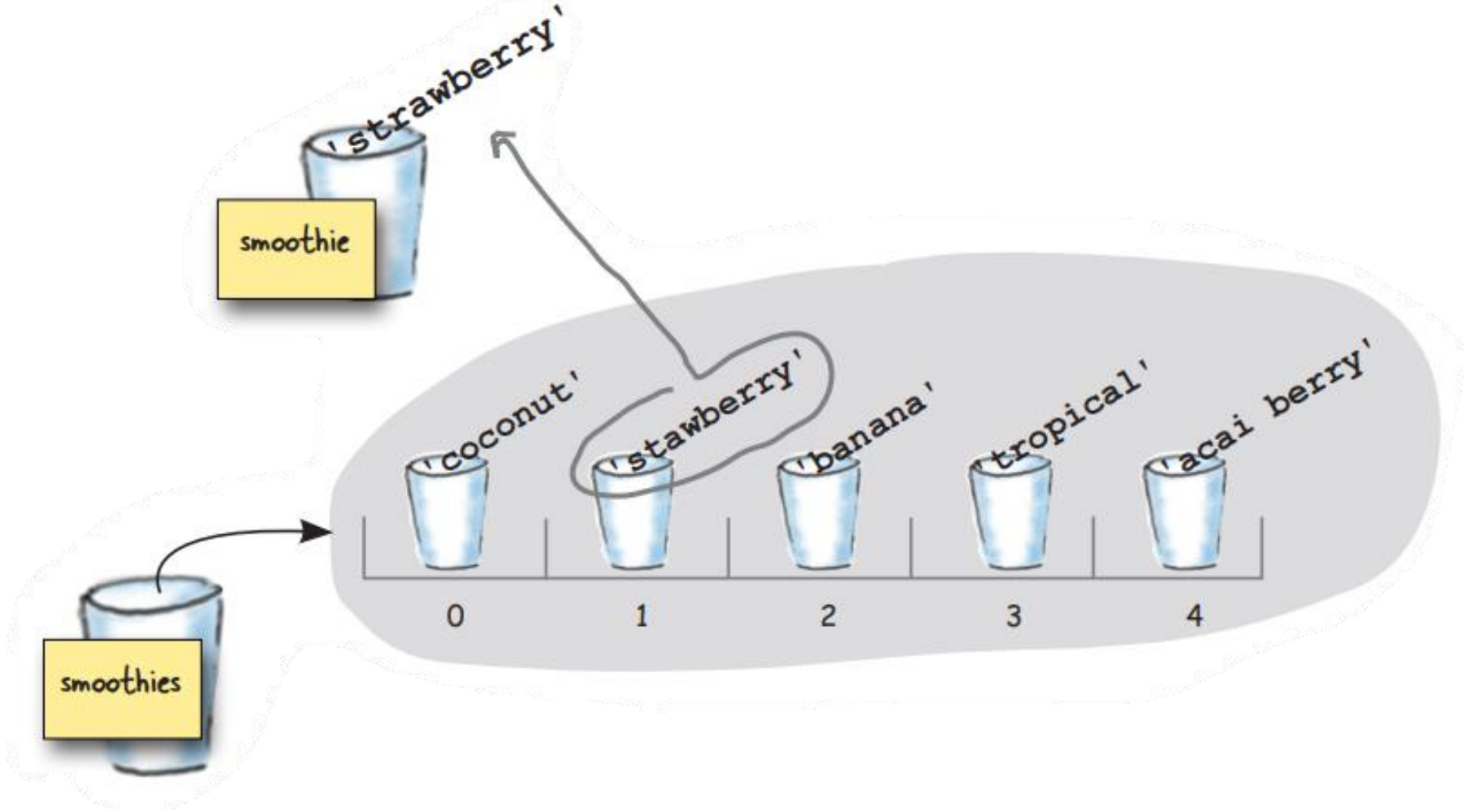

smoothies
'coconut'
'strawberry'
'banana'
'tropical'
'acai berry'
-5
-4
-3
-2
-1




```
for smoothie in smoothies:
    output = 'We serve ' + smoothie
    print(output)
```

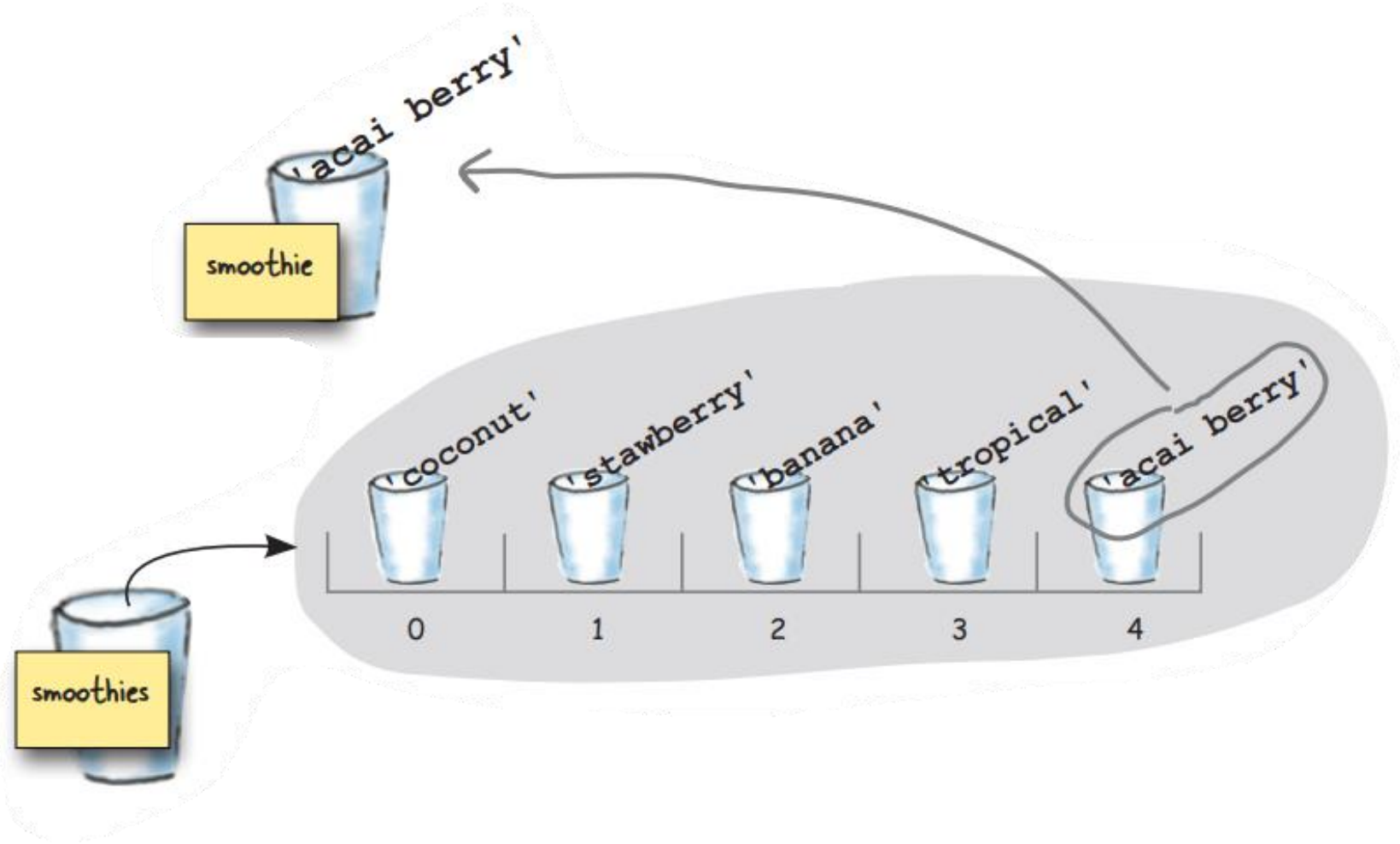
Python 3.6.0 Shell

```
We serve coconut
We serve strawberry
```



Python 3.6.0 Shell

```
We serve coconut
We serve strawberry
We serve banana
We serve tropical
We serve acai berry
```



How the for loop works on a range of numbers

```
range(5)
```

Creates the sequence 0, 1, 2, 3, 4.

The range starts at 0 and gives you a sequence of 5 numbers.

range makes the sequence 0, 1, 2, 3, 4.

```
for i in range(5):
    print('Iterating through', i)
```

The i variable is assigned to each item of the sequence before the body is executed.

Python 3.6.0 Shell

```
Iterating through 0
Iterating through 1
Iterating through 2
Iterating through 3
Iterating through 4
>>>
```

Create a range from zero to the length of smoothies.

```
length = len(smoothies)
for i in range(length):
    print('Smoothie #', i, smoothies[i])
```

Each time we iterate, we print "Smoothie#", the index, and the smoothie at that index.

Python 3.6.0 Shell

```
Smoothie # 0 coconut
Smoothie # 1 strawberry
Smoothie # 2 banana
Smoothie # 3 tropical
Smoothie # 4 acai
>>>
```

Try a starting and ending number

Start of range

Up to end of range (but not including this number)

```
range(5, 10)
```

Creates a sequence starting at 5 and going up to 10, so: 5, 6, 7, 8, 9

Add a step size

You can add a "step size" as well, which tells Python to count by increments.

```
range(3, 10, 2)
```

Creates a sequence starting at 3 and going up to 10, but counting by steps of 2, so: 3, 5, 7, 9

Count backward

We can even count backward by making the first argument larger than the second, and using a negative step size.

```
range(10, 0, -1)
```

Creates a sequence starting at 10 and going down to 0, but by steps of -1, so: 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

Or start from negative numbers

You can start at a negative number too.

```
range(-10, 2)
```

Creates a sequence starting at -10 counting to 2, so: -10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1

`range(8)`	0, 1, 2, 3, 4, 5, 6, 7
`range(0, 8)`	0, 1, 2, 3, 4, 5, 6, 7
`range(0, 8, 1)`	0, 1, 2, 3, 4, 5, 6, 7
`range(0, 8, 2)`	0, 2, 4, 6
`range(0, 7)`	0, 1, 2, 3, 4, 5, 6
`range(-3, 1)`	-3, -2, -1, 0
`range(3, 8)`	3, 4, 5, 6, 7
`range(3, 8, 3)`	3, 5, 7
`range(3, 8, 2)`	3, 6
	1, 2, 3, 4

Putting it all together

```
scores = [60, 50, 60, 58, 54, 54, 58, 50, 52, 54, 48, 69,
          34, 55, 51, 52, 44, 51, 69, 64, 66, 55, 52, 61,
          46, 31, 57, 52, 44, 18, 41, 53, 55, 61, 51, 44]
```

First, get length of the scores list, as before.

You can delete the while loop.

Then create a range from the scores length, and iterate over those values from zero to the length of scores minus one.

```
length = len(scores)
~~while i < length:~~
for i in range(length):
    print('Bubble solution #' + str(i), 'score:', scores[i])
```

Then we create our output. Notice this is exactly the same print statement we used with the while loop—nothing changed!

```
smoothies = ['coconut',
             'strawberry',
             'banana',
             'tropical',
             'acai berry']
```

```
has_coconut = [True,
               False,
               False,
               True,
               False]
```

Rewrite your fridge magnet code, so that it uses a `for` loop instead of a `while` loop.

```
i = 0
while i < len(has_coconut) :
    if has_coconut[i] :
        print(smoothies[i],
              'contains coconut')
    i = i + 1
```

```
smoothies = ['coconut',
             'strawberry',
             'banana',
             'tropical',
             'acai berry']
```

```
has_coconut = [True,
               False,
               False,
               True,
               False]
```

Rewrite your fridge magnet code, so that it uses a `for` loop instead of a `while` loop.

```
length = len(has_coconut)
for i in range(length):
if has_coconut[i]:
    print(smoothies[i], 'contains coconut')
```

Go ahead and implement the pseudocode to find the highest score.

```
scores = [60, 50, 60, 58, 54, 54, 58, 50, 52, 54, 48, 69,
34, 55, 51, 52, 44, 51, 69, 64, 66, 55, 52, 61, 46, 31,
57, 52, 44, 18, 41, 53, 55, 61, 51, 44]
```

Go ahead and implement the pseudocode to find the highest score.

```
scores = [60, 50, 60, 58, 54, 54, 58, 50, 52, 54, 48, 69, 34, 55, 51, 52, 44, 51, 69, 64, 66, 55,
52, 61, 46, 31, 57, 52, 44, 18, 41, 53, 55, 61, 51, 44]
```

```
DECLARE a variable high_score and set to 0.

FOR i in range(length)

    PRINT i and the bubble solution score[i]

    IF scores[i] > high_score:

        high_score = scores[i];

PRINT high_score
```

Go ahead and implement the pseudocode to find the highest score.

```
scores = [60, 50, 60, 58, 54, 54, 58, 50, 52, 54, 48, 69, 34, 55, 51, 52, 44, 51, 69, 64, 66, 55,
52, 61, 46, 31, 57, 52, 44, 18, 41, 53, 55, 61, 51, 44]
```

```
high_score = ______


length = len(scores)
for i in range(length):
    print('Bubble solution #' + str(i), 'score:', scores[i])
    if ____________ > high_score:
        ____________ = scores[i]


print('Bubbles tests:', ____________)
print('Highest bubble score:', ____________)
```

Go ahead and implement the pseudocode to find the highest score.

```
scores = [60, 50, 60, 58, 54, 54, 58, 50, 52, 54, 48, 69, 34, 55, 51, 52, 44, 51, 69, 64, 66, 55,
52, 61, 46, 31, 57, 52, 44, 18, 41, 53, 55, 61, 51, 44]

high_score = 0


length = len(scores)
for i in range(length):
    print('Bubble solution #' + str(i), 'score:', scores[i])
    if scores[i] > high_score:
        high_score = scores[i]


print('Bubbles tests:', length)
print('Highest bubble score:', high_score)
```

Building your own list, from scratch

```
menu = ['Pizza', 'Pasta', 'Soup', 'Salad']
```

```
menu = []   # length of zero
menu.append('Burger’)
menu.append('Sushi’)
print(menu)
```

Sharpen your pencil

```
mystery = ['secret'] * 5
```

Multiplication of a number and a list? What on earth does this do?

```
mystery = 'secret' * 5
```

How is it different from this?

Delete an item from a list

```
del menu[0]
print(menu)
```

Add one list to another

```
menu.extend(['BBQ', 'Tacos’])
print(menu)
```

Question??

```
menu = menu + ['BBQ', 'Tacos']
```

Or insert items into your list

```
menu.insert(1, 'Stir Fry’)
print(menu)
```

Can you help write the loop to find *all the scores* that match the high score?

```
scores = [60, 50, 60, 58, 54, 54, 58, 50, 52, 54, 48, 69, 34, 55, 51, 52, 44, 51,
69, 64, 66, 55, 52, 61, 46, 31, 57, 52, 44, 18, 41, 53, 55, 61, 51, 44]
high_score = 0
length = len(scores)
for i in range(length):
    print('Bubble solution #' + str(i), 'score:',
scores[i])
    if scores[i] > high_score:
        high_score = scores[i]

print('Bubbles tests:', length)
print('Highest bubble score:', high_score)
best_solutions = []

    ....................
    ....................
    ....................
    ....................
```

Can you help write the loop to find *all the scores* that match the high score?

```
scores = [60, 50, 60, 58, 54, 54, 58, 50, 52, 54, 48, 69, 34, 55, 51, 52, 44, 51,
69, 64, 66, 55, 52, 61, 46, 31, 57, 52, 44, 18, 41, 53, 55, 61, 51, 44]

high_score = 0
length = len(scores)
for i in range(length):
      print('Bubble solution #' + str(i), 'score:', scores[i])
      if scores[i] > high_score:
            high_score = scores[i]

print('Bubbles tests:', length)
print('Highest bubble score:', high_score)
```

```
best_solutions = []
for i in range(length):
    if high_score == scores[i]:
        best_solutions.append(i)

print('Solutions with the highest score:', best_solutions)
```

Scores and costs are parallel lists because for each score there is a corresponding cost at the same index.

```
costs = [.25, .27, .25, .25, .25, .25, .33, .31, .25, .29, .27, .22, ..., .29]
```

The cost at 0 is the cost of the bubble solution at 0...

And likewise for the other cost and score values in the lists.

```
scores = [60,  50,  60,  58,  54,  54,  58,  50,  52,  54,  48,  69,  ..., 44]
```

can you figure out the most cost-effective bubble solution?

DECLARE a ***variable*** cost and set to 100.0

DECLARE a ***variable*** most_effective

FOR i in ***range(length):***

    IF the bubble solution at scores[i] equals high_score **AND** bubble solution at costs[i] is less than cost:

        SET the value of most_effective to the value of i

        SET the value of cost to the cost of the bubble solution

At the end of the loop most_effective holds the index of the solution with the highest score and lowest cost. And the variable cost holds the cost of that solution. Note, if there is a tie between one or more solutions, this code will always pick the solution it sees first in the list.

```
scores = [60, 50, 60, 58, 54, 54, 58, 50, 52, 54, 48, 69, 34, 55, 51, 52, 44, 51, 69,
64, 66, 55, 52, 61, 46, 31, 57, 52, 44, 18, 41, 53, 55, 61, 51, 44]

costs = [.25, .27, .25, .25, .25, .25, .33, .31, .25, .29, .27, .22, .31, .25, .25,
.33, .21, .25, .25, .25, .28, .25, .24, .22, .20, .25, .30, .25, .24, .25, .25, .25,
.27, .25, .26, .29]

high_score = 0

length = len(scores)
for i in range(length):
      print('Bubble solution #' + str(i), 'score:', scores[i])
if scores[i] > high_score:
            high_score = scores[i]

print('Bubbles tests:', length)
print('Highest bubble score:', high_score)

best_solutions = []
for i in range(length):
      if high_score == scores[i]:
            best_solutions.append(i)

print('Solutions with the highest score:', best_solutions)
```

```
cost = 100.0
most_effective = 0

for i in range(length):
     if scores[i] == high_score and costs[i] < cost:
          most_effective = i
          cost = costs[i]

print('Solution', most_effective,
          'is the most effective with a cost of’,
          costs[most_effective])
```