Bash Scripting Assignment

Objective: Familiarize yourself with basic file system navigation and manipulation commands.

1- Navigate to your home directory
2- Create a new directory named bash_first_task
3- Navigate into the bash_first_task directory
4- Create a new files named hello.txt, test.py, data.csv, text.docx
5- List the contents of the bash_first_task directory
6- Remove the hello.txt file
7- Create a new file named hello_python.py
8- Clear the terminal screen

Bash Scripting Assignment

Objective: Familiarize yourself with basic file system navigation and manipulation commands.

1- Navigate to your home directory

```
cd ~
```

2- Create a new directory named bash_first_task

```
mkdir bash_first_task
```

3- Navigate into the bash_first_task directory

```
cd bash_first_task
```

4- Create a new files named hello.txt, test.py, data.csv, text.docx

```
touch hello.txt test.py data.csv text.docx
```

5- List the contents of the bash_first_task directory

```
ls
```

6- Remove the hello.txt file

```
rm hello.txt
```

7- Create a new file named hello_python.py

```
 touch hello_python.py
```

8- Clear the terminal screen

```
clear
```

Bash Scripting Task

Objective: Practice basic file and directory operations using bash commands.

1- Navigate to your home directory
2- Create a new directory named simple_task
3- Navigate into the simple_task directory
4- Create two new directories named docs and images
5- Navigate into the docs directory and create three new files named doc1.txt, doc2.txt and doc3.txt
6- Navigate back to the simple_task directory
7- List the contents of the simple_task directory and its subdirectories
8- Remove the doc3.txt file from the docs directory
9- Clear the terminal screen

Bash Scripting Task

Objective: Practice basic file and directory operations using bash commands.

1- Navigate to your home directory:

```
cd ~
```

2- Create a new directory named simple_task:

```
mkdir simple_task
```

3- Navigate into the simple_task directory:

```
cd simple_task
```

4- Create two new directories named docs and images:

```
mkdir docs images
```

5- Navigate into the docs directory and create three new files named doc1.txt, doc2.txt and doc3.txt

```
cd docs
touch doc1.txt doc2.txt doc3.txt
```

6- Navigate back to the simple_task directory:

```
cd ..
```

5- List the contents of the simple_task directory and its subdirectories

```
ls -R
```

6- Remove the doc3.txt file from the docs directory:

```
rm docs/doc3.txt
```

7- Clear the terminal screen:

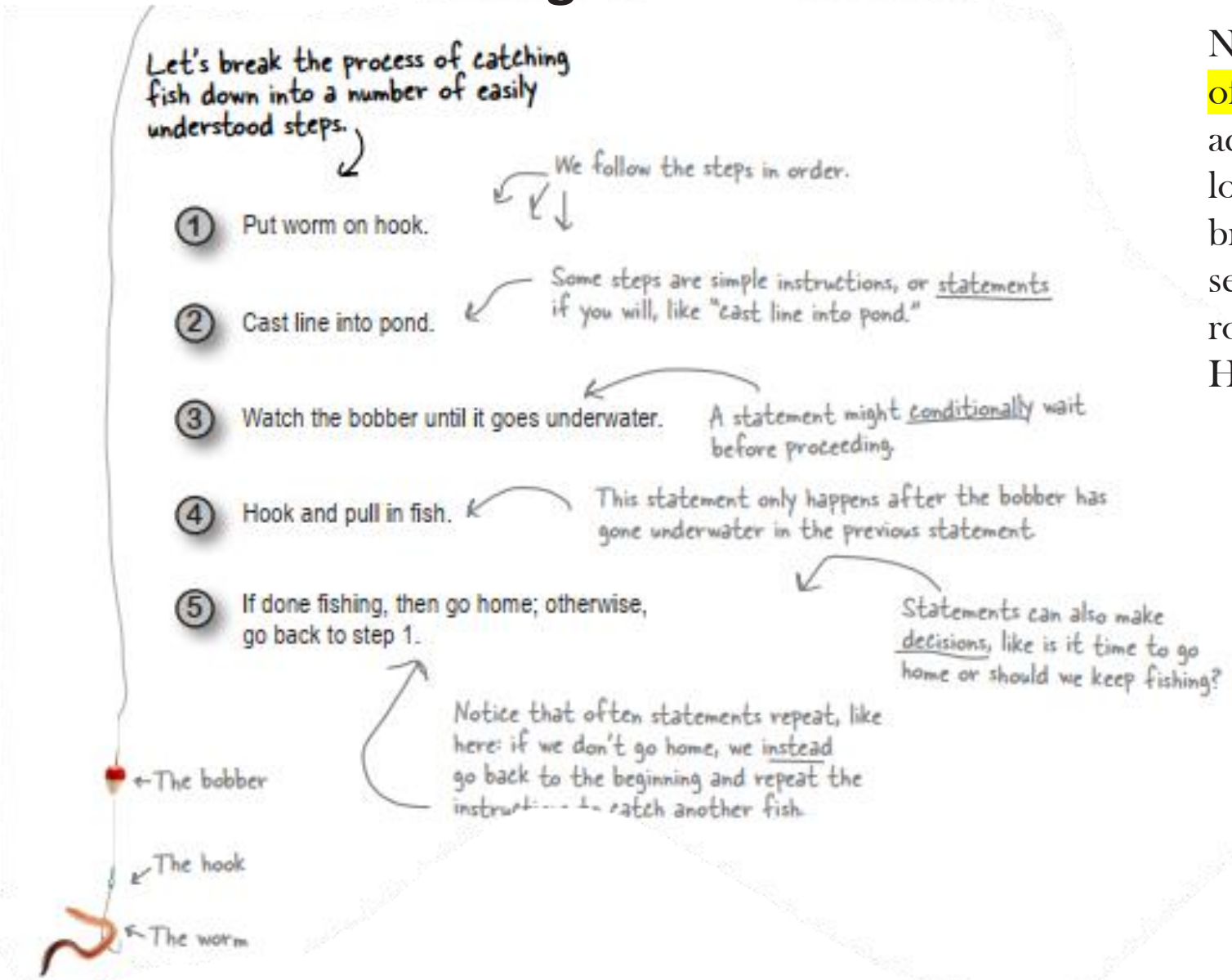
```
clear
```

Python Task

Objective: create new env named itsharks_24py310, install jupyter notebook, and print Hello.

Python Introduction

PROBLEM: Breaking it down



Now breaking problems down into a number of steps may sound like a new skill, but it's actually something you do every day. Let's look at a simple example: say you wanted to break the activity of fishing down into a simple set of instructions that you could hand to a robot, who would do your fishing for you. Here's our first attempt to do that:

You can think of these statements as a nice recipe for fishing.

Actually, a recipe is a perfectly good way to describe a set of instructions to a computer.

A computer scientist or serious software developer would commonly call a recipe an **algorithm.** What's an algorithm?

It's a sequence of instructions that solves some problem. Often you'll find algorithms are first written in an informal form of code called **pseudocode.**

Code Magnets

Let's get a little practice with ~~recipes~~ algorithms. We put the Head First Diner's algorithm for making an three-egg omelet on the fridge to remember it, but someone came along and messed it up. Can you put the magnets back in the right order to make our algorithm work? Note that the Head First Diner makes two kinds of omelets: plain and cheese. **Make sure you check your answer at the end of the chapter.**

Rearrange these magnets here to make the algorithm work.

If the customer ordered cheese:

Add cheese on top

while eggs aren't fully mixed:

Transfer eggs to plate

While eggs aren't fully cooked:

Remove pan from heat

Stir eggs

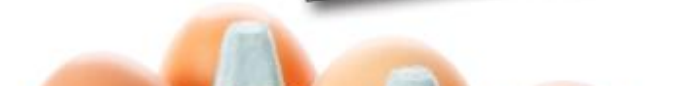
Serve

Heat saute pan

Whip eggs

Crack three eggs into bowl

Transfer eggs to pan



Here are the unscrambled magnets!

There are several correct variations you could come up with; just make sure you understand our solution, and that yours, if different, makes logical sense.

```
Heat saute pan
```

We start by setting everything up: heating the pan, cracking the eggs.

```
Crack three eggs into bowl
```

```
while eggs aren't fully mixed:
```

Then we whip the eggs until they are thoroughly mixed.

```
    Whip eggs
```

Notice we indented whipping the eggs to indicate this is done as long as the eggs aren't yet whipped. If you indicated this in another way, that is fine.

```
Transfer eggs to pan
```

Next we move the eggs to the preheated pan.

```
While eggs aren't fully cooked:
```

And cook them until they are done.

```
    Stir eggs
```

Notice we just indented stirring the eggs to indicate the eggs are stirred as long as they aren't done. If you indicated this in another way, that is fine.

```
If the customer ordered cheese:
```

If the customer wanted cheese, then add it.

```
    Add cheese on top
```

We also indented adding cheese, because that is only done if the customer ordered cheese.

```
Remove pan from heat
```

And in either case, remove the pan from the heat, and transfer it to a plate.

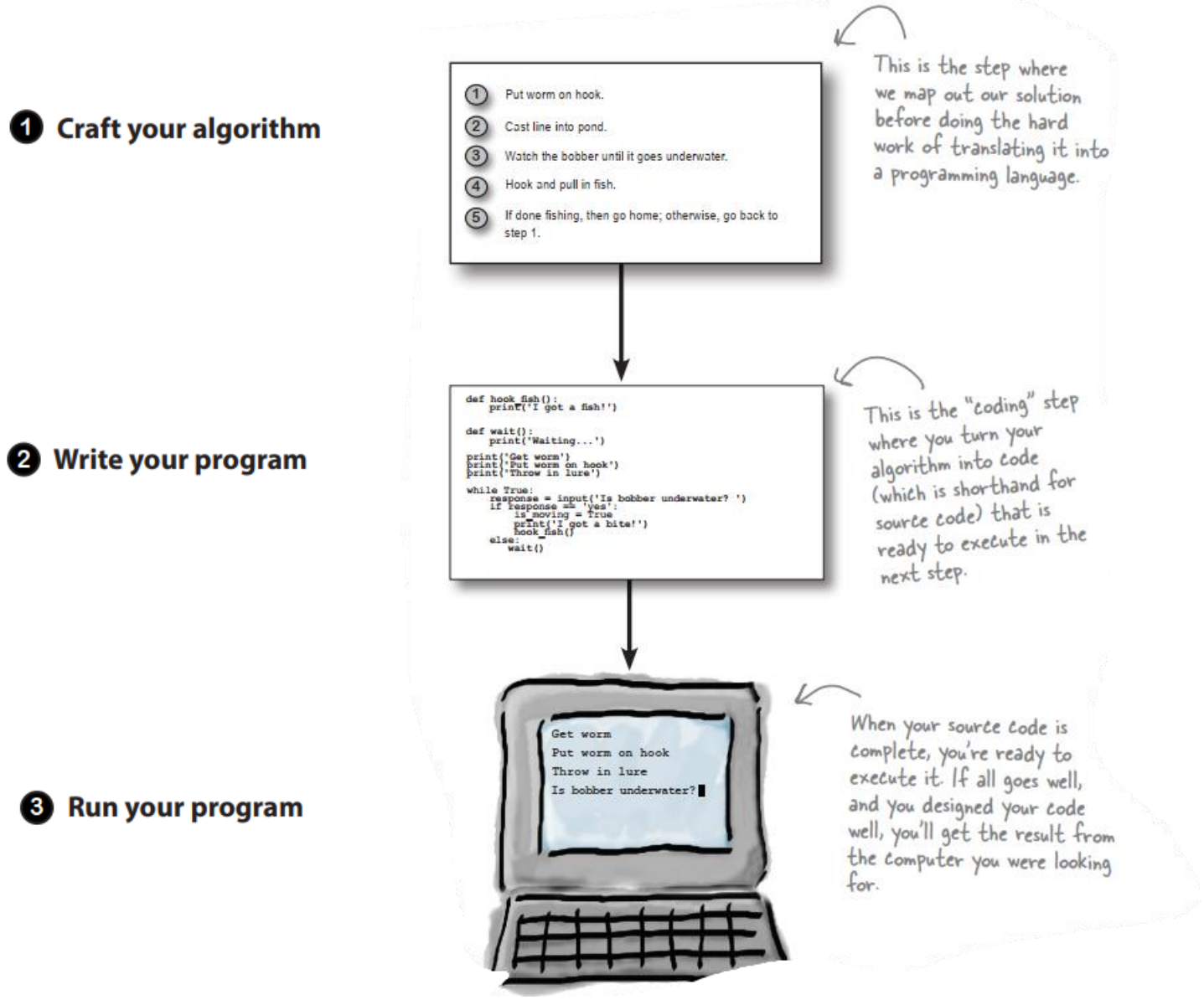
```
Transfer eggs to plate
```

```
Serve
```

Finally, we serve the omelet.

How coding works

So you've got a task you want the computer to do for you, and you know you'll need to break that task down into a number of instructions the computer can understand, but how are you going to *actually tell* the computer to do something? That's where a programming language comes in—with a programming language you can describe your task in terms that *you and the computer* both understand.



Are we even speaking the same language?

Programming languages give you a way to describe your recipes in a manner that is clear and precise enough that a computer can understand it.

what things can you say using the language, and what do those things mean? A computer scientist would call these the **syntax** and **semantics** of the language. Just stash those terms in the back of your brain for now;

YOU SAY TOMATO

On the left you'll find some statements written in English, and on the right you'll find statements written in a programming language. Draw a line from each English statement to its corresponding code translation. We did the first one for you. Make sure you check your work with the solution at the end of the chapter before proceeding.

Print "Hi there" on the screen.

If the temperature is more than 72, then print "Wear shorts" on the screen.

A grocery list with bread, milk, and eggs on it.

Pouring five drinks.

Ask the user, "What is your name?"

```
for num in range(0, 5):
        pour_drink()
```

```
name = input('What is your name? ')
```

```
if temperature > 72:
        print('Wear shorts')
```

```
grocery_list = ['bread', 'milk', 'eggs']
```

```
print('Hi there')
```

YOU SAY TOMATO SOLUTION

On the left you'll find some statements written in English, and on the right you'll find statements written in a programming language. Draw a line from each English statement to its corresponding code translation. We did the first one for you.

Print "Hi there" on the screen.

If the temperature is more than 72, then print "Wear shorts" on the screen.

A grocery list with bread, milk, and eggs on it.

Pouring five drinks.

Ask the user, "What's your name?"

```
for num in range(0,5):
        pour_drink();
```

```
name = input('What is your name? ')
```

```
if temperature > 72:
        print('Wear shorts')
```

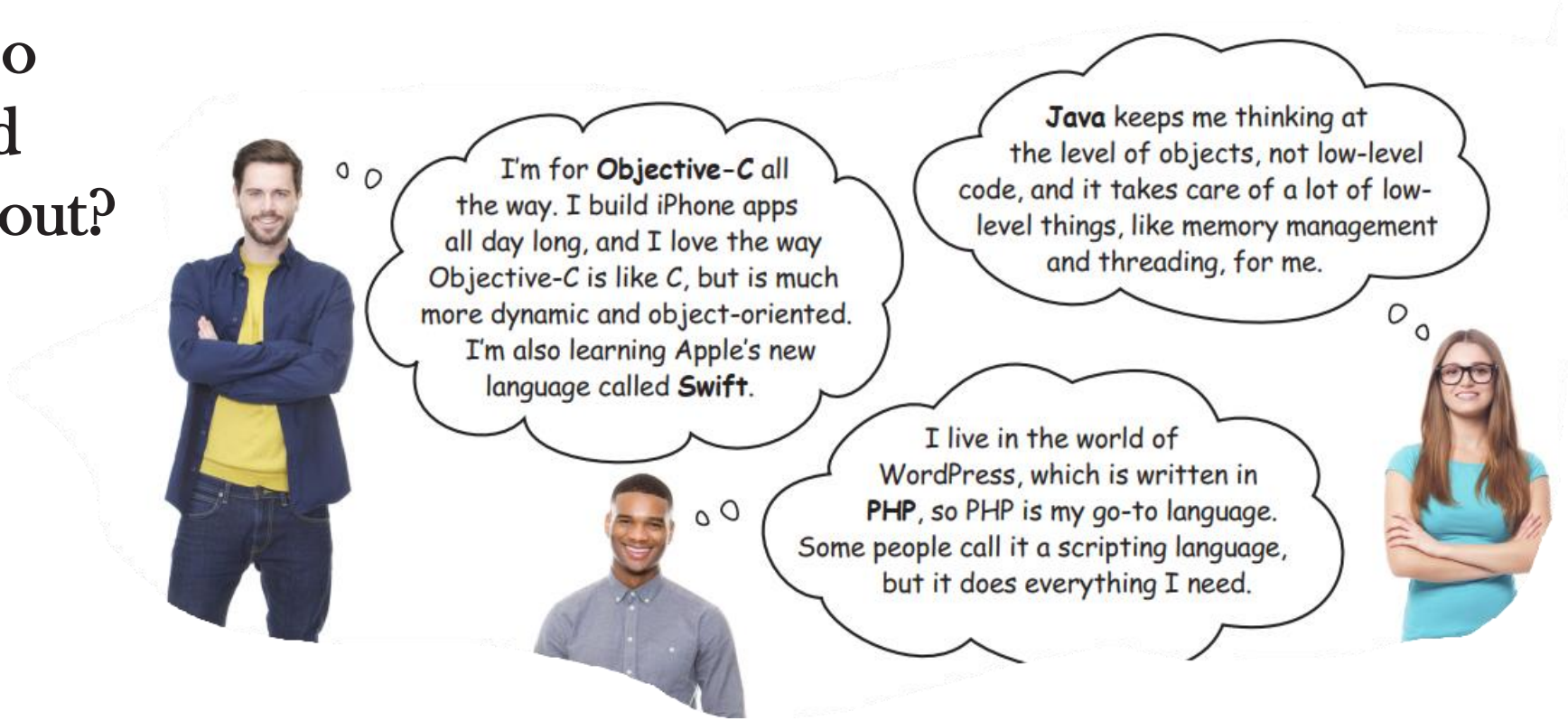
```
grocery_list = ['bread', 'milk', 'eggs']
```

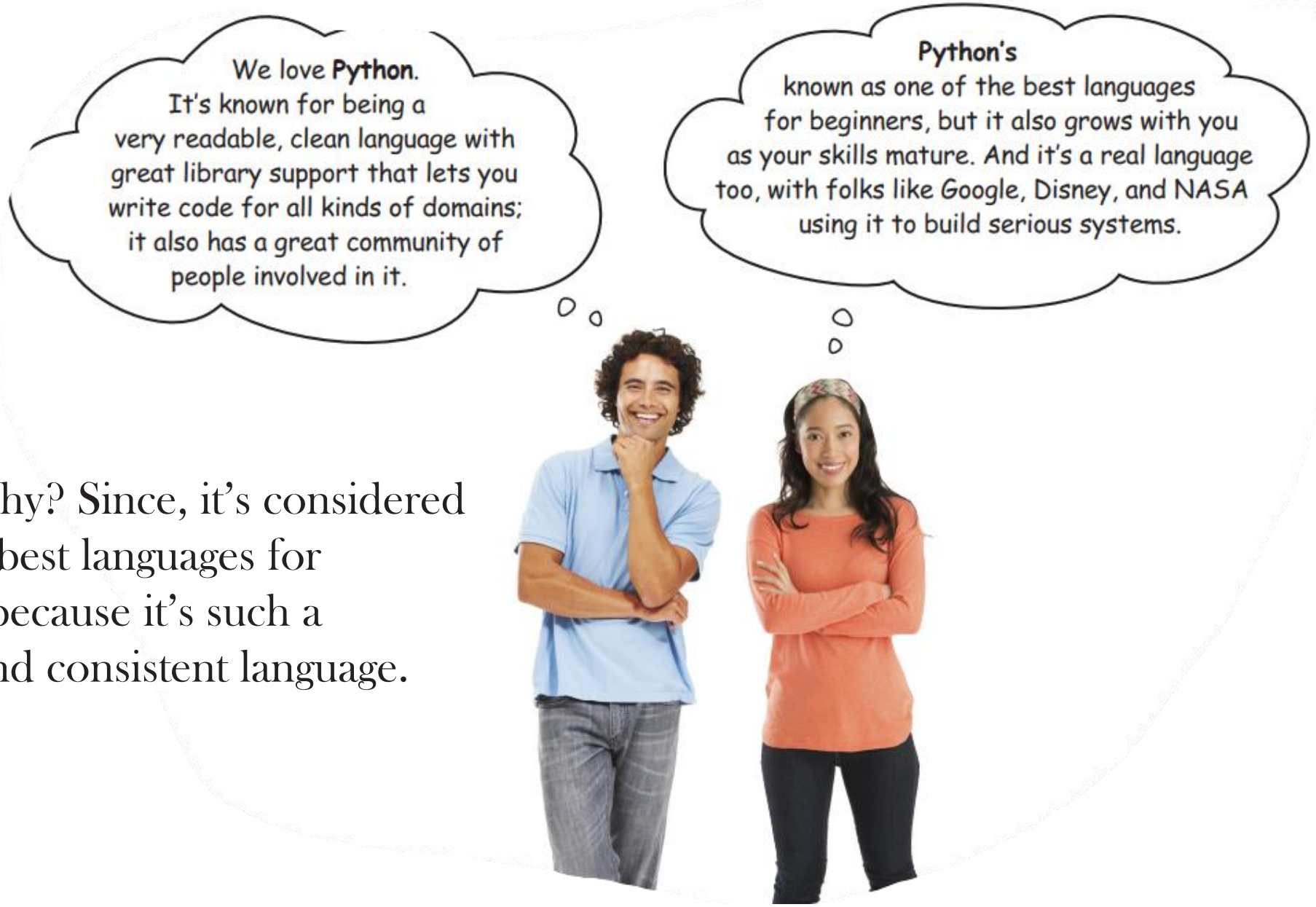
```
print('Hi there')
```

The world of programming languages

Java, C, C++, LISP, Scheme, Objective-C, Perl, PHP, Swift, Clojure, Haskell, COBOL, Ruby, Fortran, Smalltalk, BASIC, Algol, JavaScript, and of course Python

But why are there so many languages and what are they all about?





Python. Why? Since, it's considered one of the best languages for beginners because it's such a readable and consistent language.

Sharpen your pencil

Look how easy it is to write Python

You don't know Python yet, but we bet you can make some good guesses about how Python code works. Take a look at each line of code below and see if you can guess what it does. Write in your answers below. If you get stuck, the answers are on the next page. We did the first one for you.

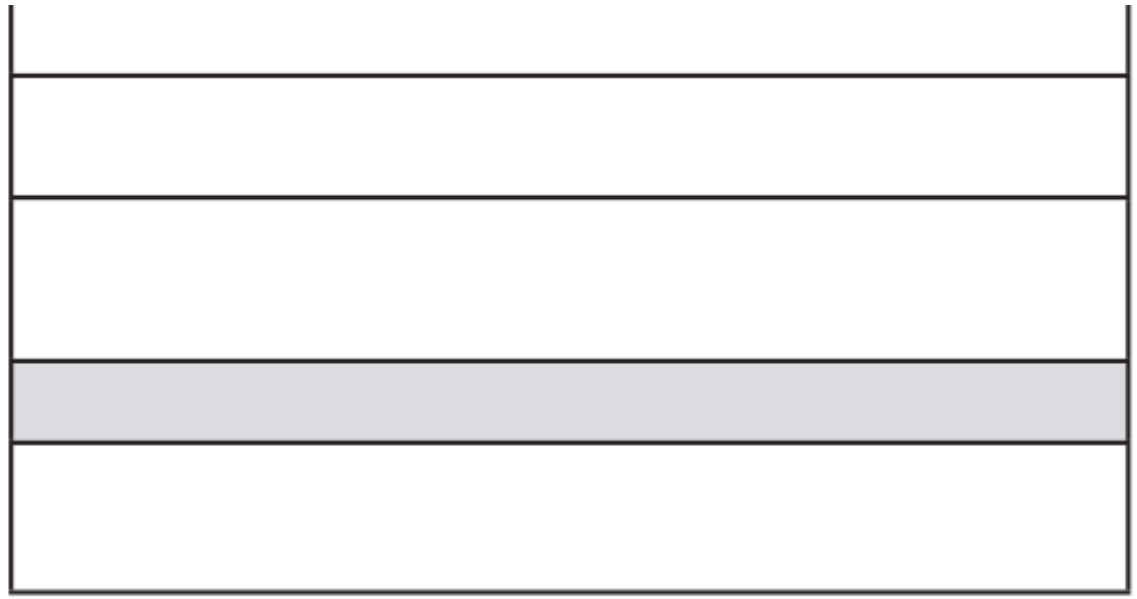
`customers = ['Jimmy', 'Kim', 'John', 'Stacie']`	Make a list of customers.
`winner = random.choice(customers)`	
`flavor = 'vanilla'`	
`print('Congratulations ' + winner +` `  ' you have won an ice cream sundae!')`	
`prompt = 'Would you like a cherry on top? '`	
`wants_cherry = input(prompt)`	

```
order = flavor + ' sundae '

if (wants_cherry == 'yes'):

    order = order + ' with a cherry on top'


print('One ' + order + ' for ' + winner +
      ' coming right up...')
```



Python Output

```
Congratulations Stacie you have won an ice cream sundae!
Would you like a cherry on top? yes
One vanilla sundae with a cherry on top for Stacie coming
right up...
```

This should help; it's the output of this code. Do you think this code has the same output every time you run it?

```
customers = ['Jimmy', 'Kim', 'John', 'Stacie']

winner = random.choice(customers)

flavor = 'vanilla'

print('Congratulations ' + winner +
    ' you have won an ice cream sundae!')

prompt = 'Would you like a cherry on top? '

wants_cherry = input(prompt)

order = flavor + ' sundae '

if (wants_cherry == 'yes'):
    order = order + ' with a cherry on top'

print('One ' + order + ' for ' + winner +
      ' coming right up...')
```

Make a list of customers.

Randomly choose one of those customers.

Set the name or variable called flavor to the text 'vanilla'.

Print out a congratulations message to the screen that includes the winning customer's name. For instance, if Kim is the winner this code prints "Congratulations Kim you have won an ice cream sundae!"

Set the name or variable called prompt to the text "Would you like a cherry on top? "

Ask the user to type in some text, and assign it to wants_cherry. Notice that when the user is asked for input, the prompt is first displayed (as seen in the Python output).

Set order to the text 'vanilla' followed by 'sundae'.

If the user answered yes to 'Would you like a cherry on top?', then add the text " with a cherry on top" to the order.

Print out that the winner's order is coming right up.

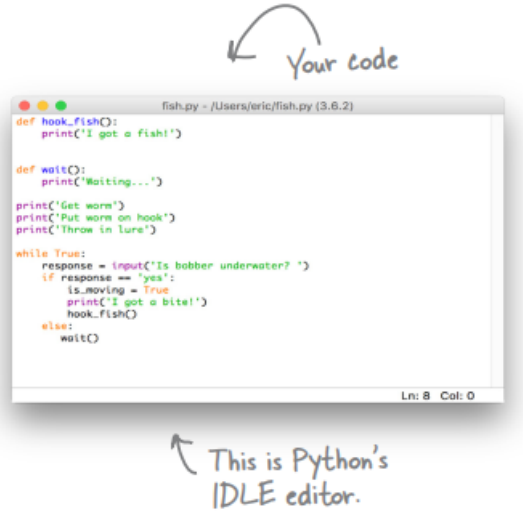
Python Output

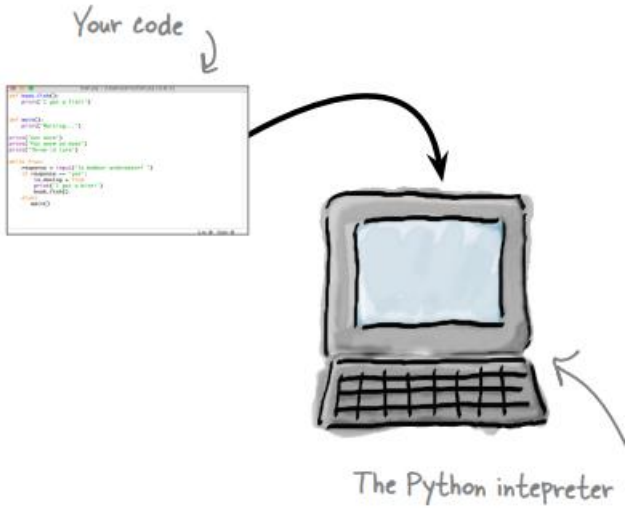
```
Congratulations Stacie you have won an ice cream sundae!
Would you like a cherry on top? yes
One vanilla sundae with a cherry on top for Stacie coming
right up...
```

P.S. If you can't help yourself and you have to type this code in, add `import random` at the very top of the file before running it. We'll get to what that does later, but note that running the code at this point is not required or all that useful. That said, we just know someone's going to have to try it. You know who you are!

How you'll write and run code with Python

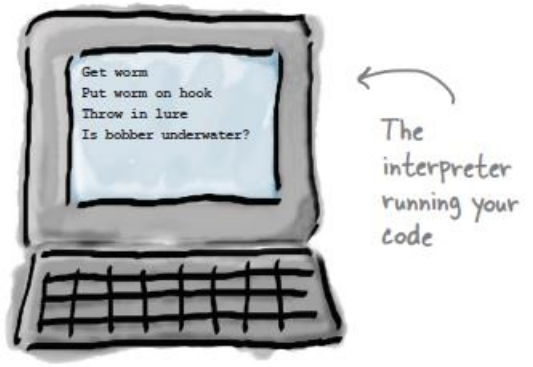
1- Writing your code





Interpreter actually translates your code behind the scenes into a lower-level machine code that can be directly executed by your computer hardware.

2- Running your code



3- How your code is interpreted

Python 1.0

Python 2.0

Python 3.0

Note from editor: umm, "the rest" is actually in the next few paragraphs.

We fully expect our flying car is going to be Python enabled.

1994 2000 2008 The Future!

Okay, it's time to get serious and to write a real-world business application using Python. Check out this Phrase-O-Matic code—you're going to be impressed.

Don't worry if you don't understand every aspect of this program; The point here is just to start to get some familiarity with code.

```
# let python know we'll be using some random
# functionality by importing the random module
import random

# make three lists, one of verbs, one of adjectives,
# and one of nouns
verbs = ['Leverage', 'Sync', 'Target',
         'Gamify', 'Offline', 'Crowd-sourced',
         '24/7', 'Lean-in', '30,000 foot']

adjectives = ['A/B Tested', 'Freemium',
              'Hyperlocal', 'Siloed', 'B-to-B',
              'Oriented', 'Cloud-based',
              'API-based']

nouns = ['Early Adopter', 'Low-hanging Fruit',
         'Pipeline', 'Splash Page', 'Productivity',
         'Process', 'Tipping Point', 'Paradigm']

# choose one verb, adjective, and noun from each list
verb = random.choice(verbs)
adjective = random.choice(adjectives)
noun = random.choice(nouns)

# now build the phrase by "adding" the words together
phrase = verb + ' ' + adjective + ' ' + noun

# output the phrase
print(phrase)
```

Know Your Value

Computers really only do two things well: store values and perform operations on those values.

Everything computers do can be broken down into **simple operations** that are performed on **simple values**. Now, part of **computational thinking** is learning to use these operations and values to build something that is much more sophisticated, complex, and meaningful.

Coding the Dog Age Calculator

Up next, the **Dog Age Calculator**. You already know what the calculator does: you enter a dog's chronological age and the calculator tells you the dog's age in relative *human years*. To perform that calcuation you simply multiply the dog's chronological age by the number 7.

But where do we even start? Do we just start trying to write some code? Well, remember the concept of pseudocode

So what exactly is pseudocode? Think of it as nothing more than your algorithm written in human-readable form. With pseudocode, you typically spell out, step by step, everything your solution needs to do to solve your problem

Write your pseudocode here.

Here is an example of how the Dog Age Calculator will work.

```
Python 3.6.0 Shell
What is your dog's name? Codie
What is your dog's age? 12
Your dog Codie is 84 years old in human years
>>>
```

Ask the user for the dog's name.

Ask the user for the dog's age (in dog years).

Multiply the dog's age by the number 7 to get the dog age in human years

Output to the user:

"Your dog"

then the dog's name

"is"

then the dog's age in human years

"years old in human years"

First we need to get some information from the user.

Next we'll need to calculate the dog's age in human terms.

Finally, we need to output our results in a user-friendly manner.


```
Python 3.6.0 Shell
What is your dog's name? Codie
What is your dog's age? 12
Your dog Codie is 84 years old in human years
>>>
```

Going from pseudocode to code

Of course the pseudocode doesn't provide every detail, but it will provide us with a nice guide to follow as we *implement* each step in code.

When we translate our ideas, algorithms, or pseudocode to real code, we often say we're implementing them.

Dog Age Calculator Pseudocode

1. **Ask the user for the dog's name.**
2. **Ask the user for the dog's age.**
3. **Multiply the dog's age by the number 7 to get the dog's age in human years.**
4. **Output to the user:**

   "Your dog"
   the dog's name
   "is"
   the dog's age in human years
   "years old in human years"

1. Prompt the user to get the dog's name and then have the user type it in. We'll presumably need to save the name somewhere so we can use it in step 4.

2.

3.

4.

Put your notes for each step here.

Dog Age Calculator Pseudocode

1. **Ask the user for the dog's name.**

2. **Ask the user for the dog's age.**

3. **Multiply the dog's age by the number 7 to get the dog's age in human years.**

4. **Output to the user:**

   "Your dog"
   the dog's name
   "is"
   the dog's age in human years
   "years old in human years"

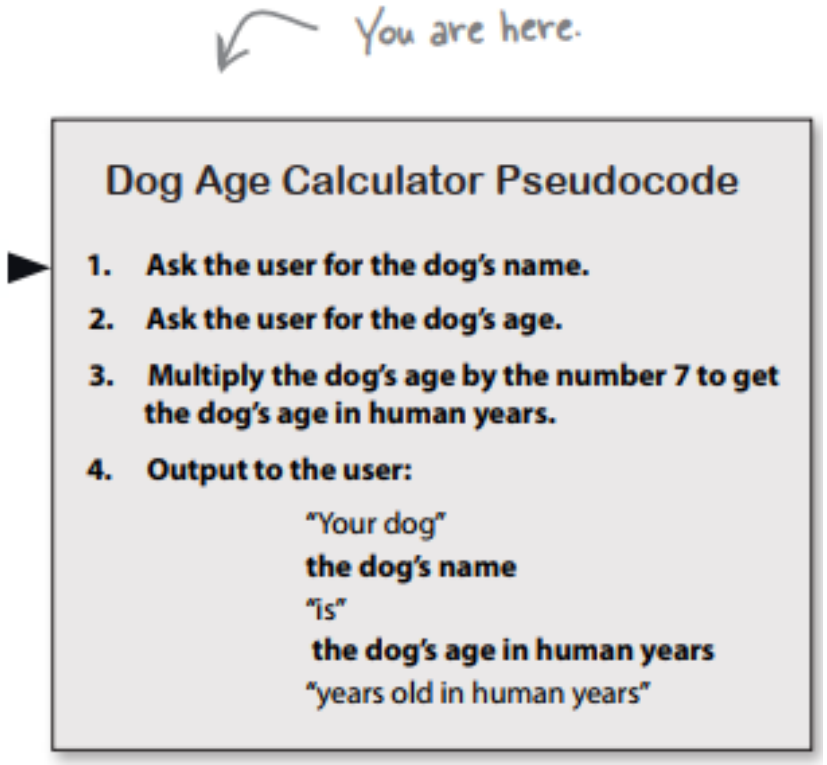
1. Prompt the user to get the dog's name and then have the user type it in. We'll presumably need to save the name somewhere so we can use it in step 4.

2. Prompt the user to get the dog's age and then have the user type it in. We'll also need to save this somewhere so we can use it in step 4.

3. Take the age from step 2 and multiply it by the number 7. We'll also need to store this somewhere for use in step 4.

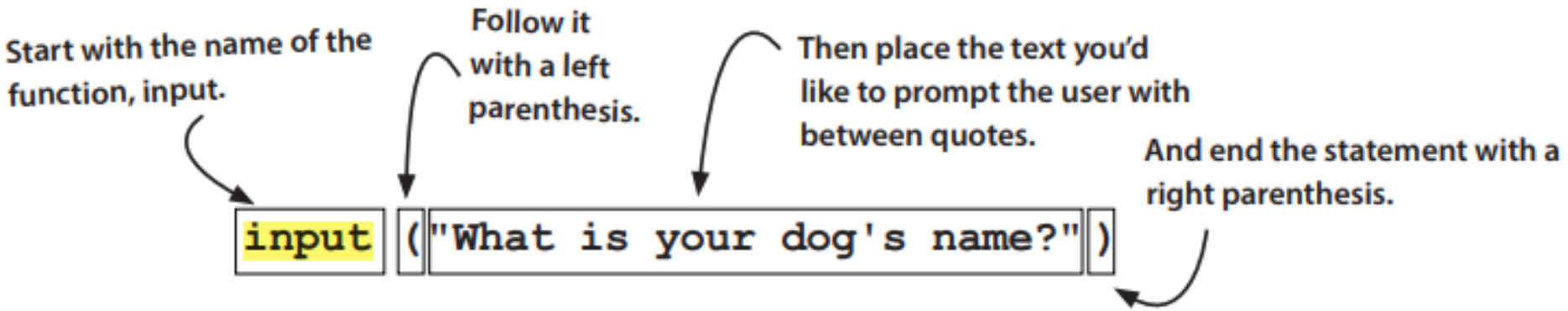
4. First print to the console "Your dog", then print the value from step 1, then print "is", then print the value from step 3, then print "years old in human years".

Step 1: Getting some input



So there's really two things we need to do here: prompt the user to get the dog's name, and then store that name for later use.

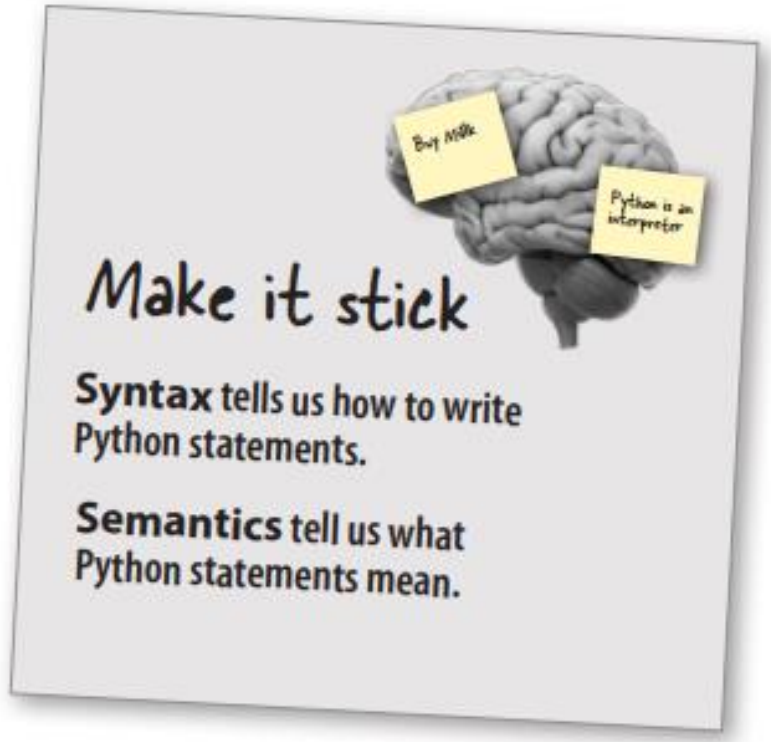
Let's look at the syntax for *calling* the `input` function and then we'll look at how it works:



How the input function works

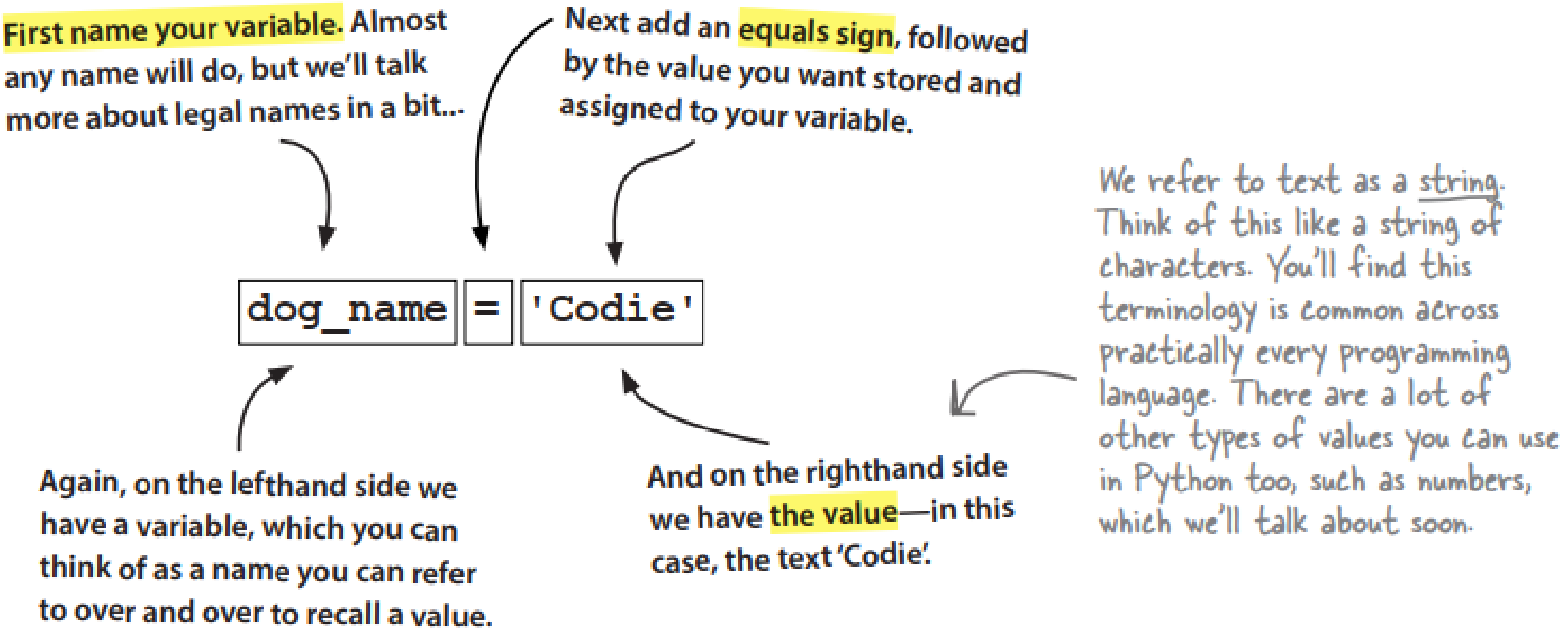
Alright, we now know how to type in the `input` function (in other words we know the syntax), but how does it actually work? Like this:

1. When the interpreter sees your call to the `input` function, it takes your prompt text and displays it for the user in the Python Shell.
2. The interpreter then waits for the user to type in a response, which the user completes by pressing the Return key.
3. Finally, the text the user typed in is passed back to your code.

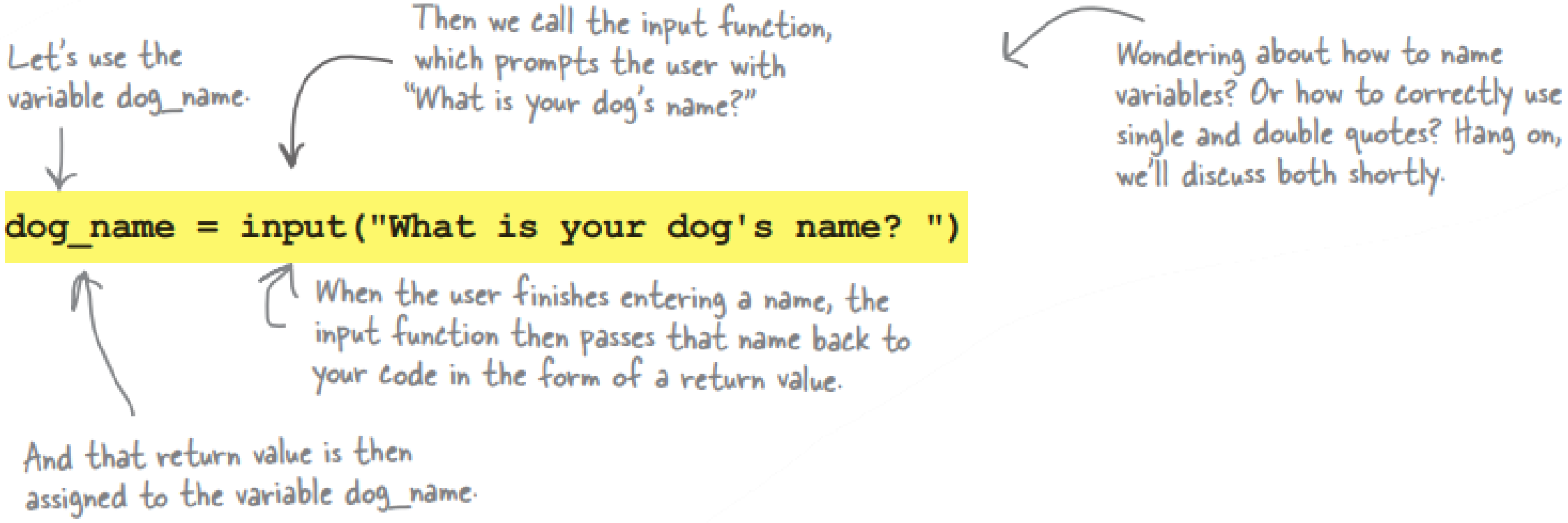


Now, that text isn't going to be too useful if we can't remember it for later, because we'll need it in step 4 when we print out our userfriendly output. So, how do we remember things with Python?

Using variables to remember and store values



Assigning the user's input to a variable



Step 2: Getting more input

We're moving on to step 2.

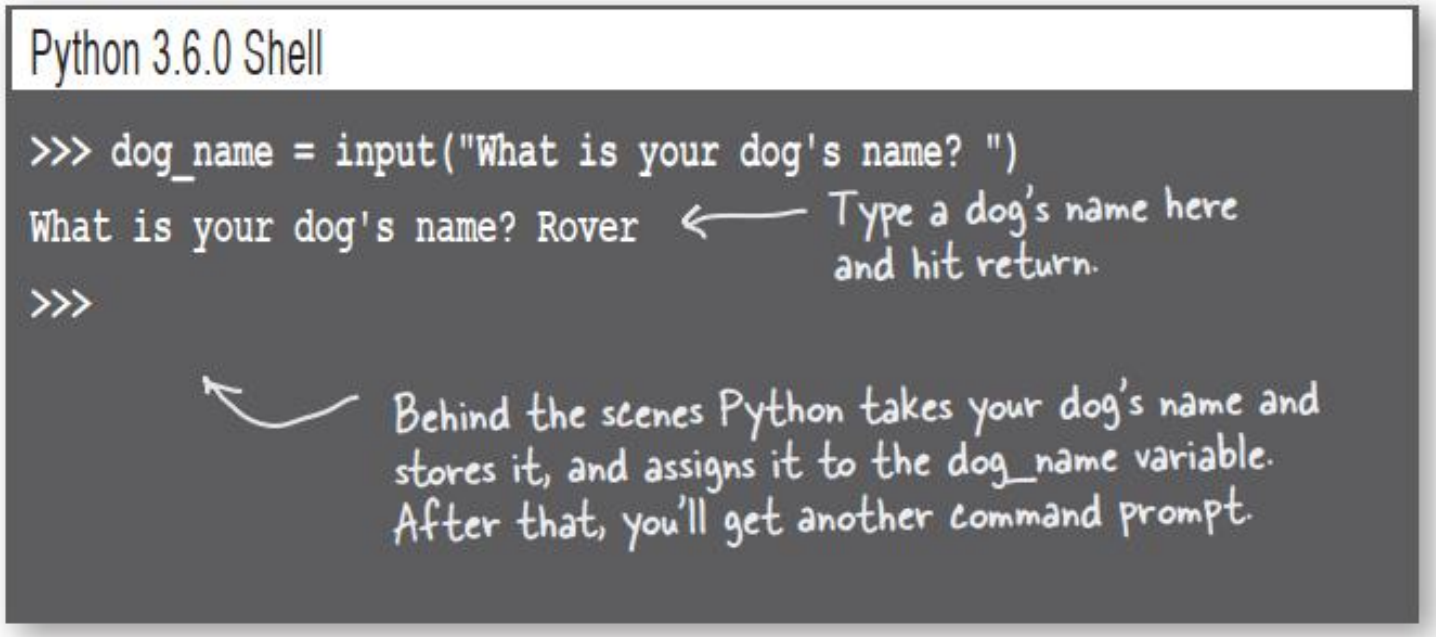
Dog Age Calculator Pseudocode

1. **Ask the user for the dog's name.**
2. **Ask the user for the dog's age.**
3. **Multiply the dog's age by the number 7 to get the dog's age in human years.**
4. **Output to the user:**

   "Your dog"
   the dog's name
   "is"
   the dog's age in human years
   "years old in human years"

It's your turn. Write the code to get the dog's age using the `input` function, just as we did with the dog's name. Prompt the user with "What is your dog's age?" and store the result in a variable called `dog_age`. Make some notes as well about what each piece of your code does. Check your answer in the back of the chapter before moving on.

It’s time to run some code



Notice that an assignment statement doesn't evaluate to a value like, say, 1 + 1 does. Rather, as you already know, the assignment statement takes the value on the righthand side and assigns it to the variable on the lefthand side.

The variable `dog_name` should now hold the value `'Rover'`, or whatever dog name you entered. How can you show it does?

I noticed you can't keep your single and double quotes straight—sometimes you use text surrounded by single quotes and sometimes by double quotes. What's the deal?

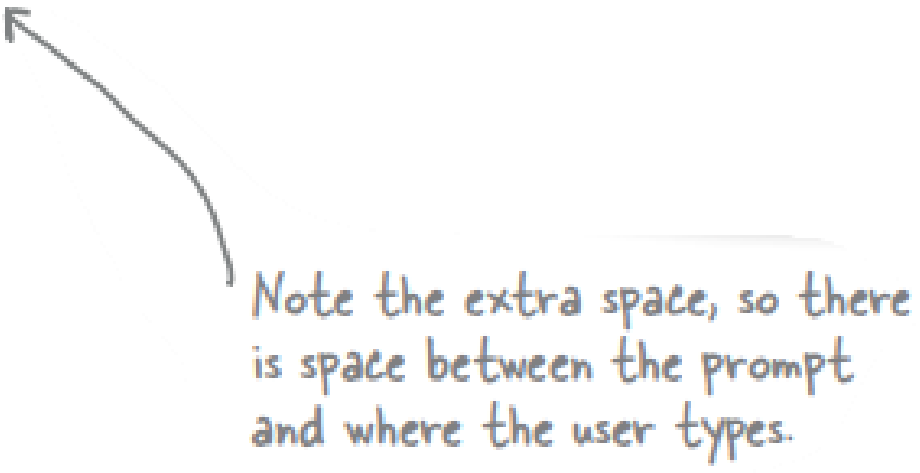
Good catch. First of all, remember we call text in quotes **strings**

```
dog_name = input("What is your dog's name? ")
```

To use a single quote as part of your string, just surround the text with double quotes.

Getting some code entered

```
dog_name = input("What is your dog's name? ")
dog_age = input("What is your dog's age? ")
```



A deep dive on variables

1- The first thing Python does is evaluate the righthand side of the assignment, which evaluates to the string `'Codie'`, and then it finds a free spot in your computer's memory where it stores the string.

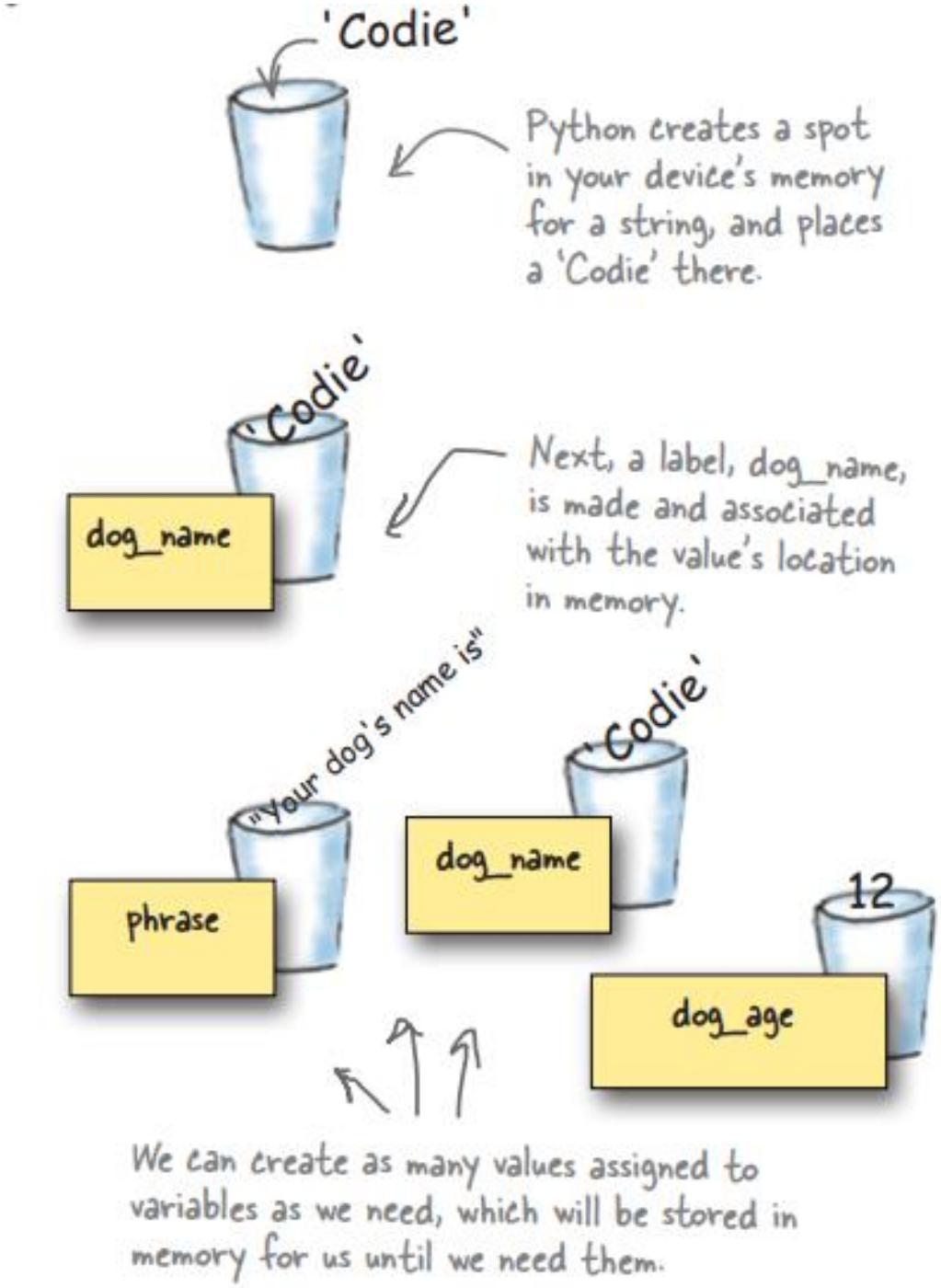
2- With the string `'Codie'` stored, Python then creates a label—think of it as a sticky note if you want—with the name **`dog_name`**, and puts it on the cup.

3- Of course we can create and store as many values as we need. How about two more:

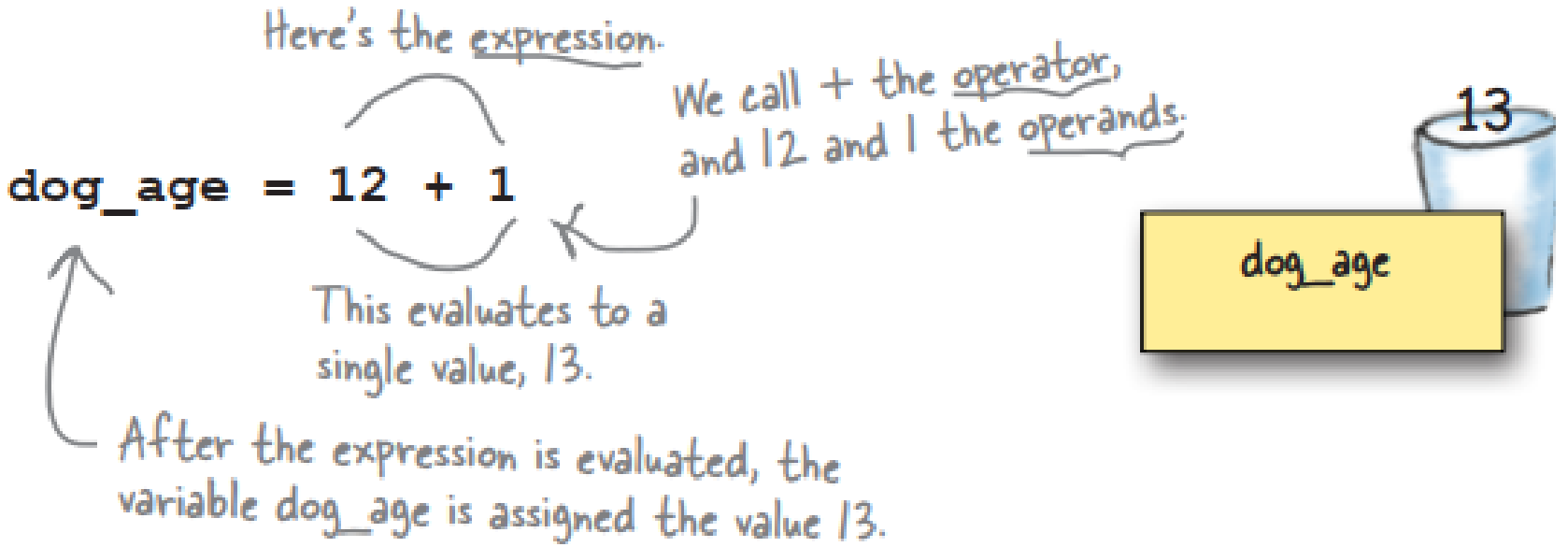
```
phrase = "Your dog's name is "
dog_age = 12
```

4- Anytime we need to retrieve the stored values, we can use the variables:

```
print(phrase)
print(dog_name)
```



Adding some expression

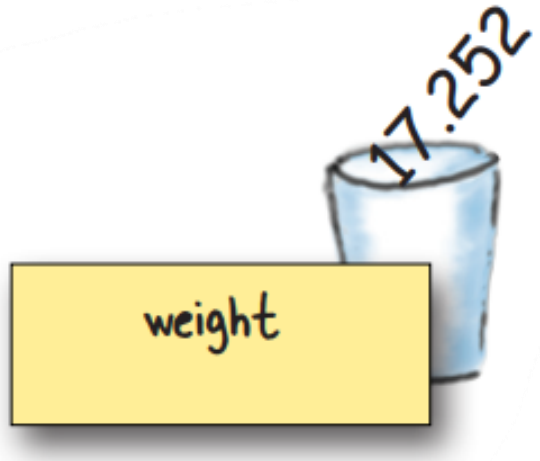


Adding some expression

Here we're using the multiplication operator.

```
weight = 38 * 0.454
```

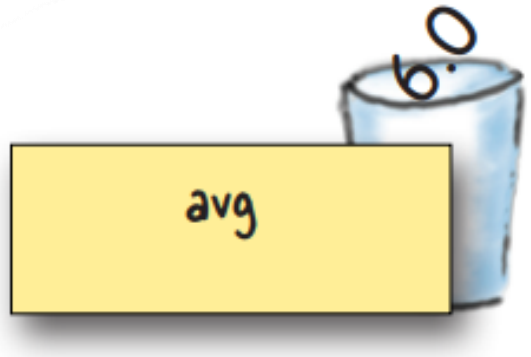
After this statement completes, weight has been assigned the value 17.252.



Adding some expression

```
avg = (12 + 5 + 1) / 3
```

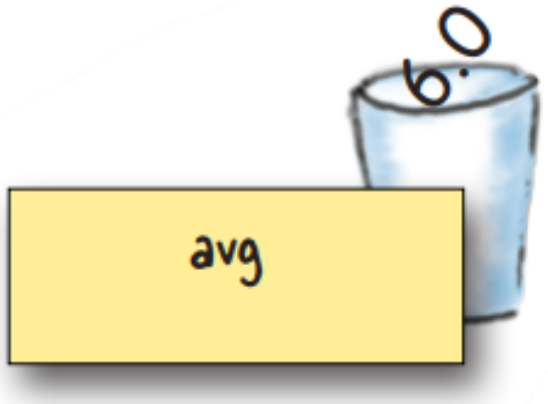
We can group operations together using parentheses.



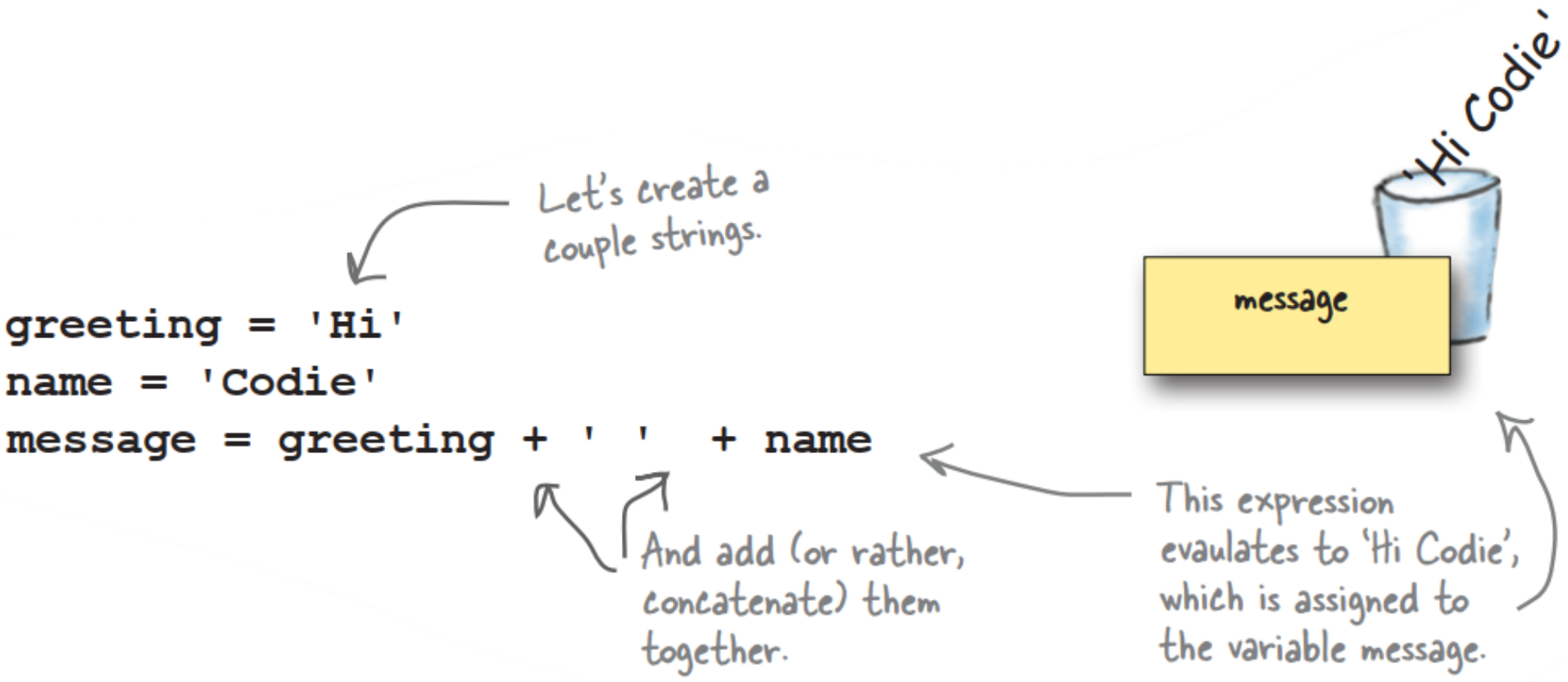
Adding some expression

```
codie = 12
fido = 5
sparky = 1
avg = (codie + fido + sparky) / 3
```

Anywhere we put a variable, it is replaced with its value to compute the expression.



Adding some expression

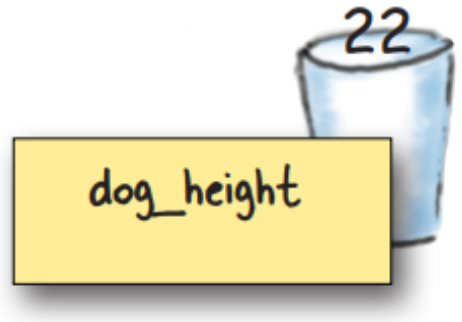


Variables are called VARY-ables for a reason

because their values usually *vary* over time

```
dog_height = 22
```

Here we're creating a new variable and assigning it the value 22.




```
dog_height = 22 + 1
```

As usual we evaluate the righthand side, which evaluates to 23, and then assign that value to dog_height. So, dog_height changes from 22 to 23.


```
dog_height = dog_height + 2
```

❶ Remember we can use a variable anywhere we use a value, so on the righthand side we add 2 to the current value of dog_height, or, 23 + 2 = 25.

❷ We then take 25 and make it the new value of dog_height.

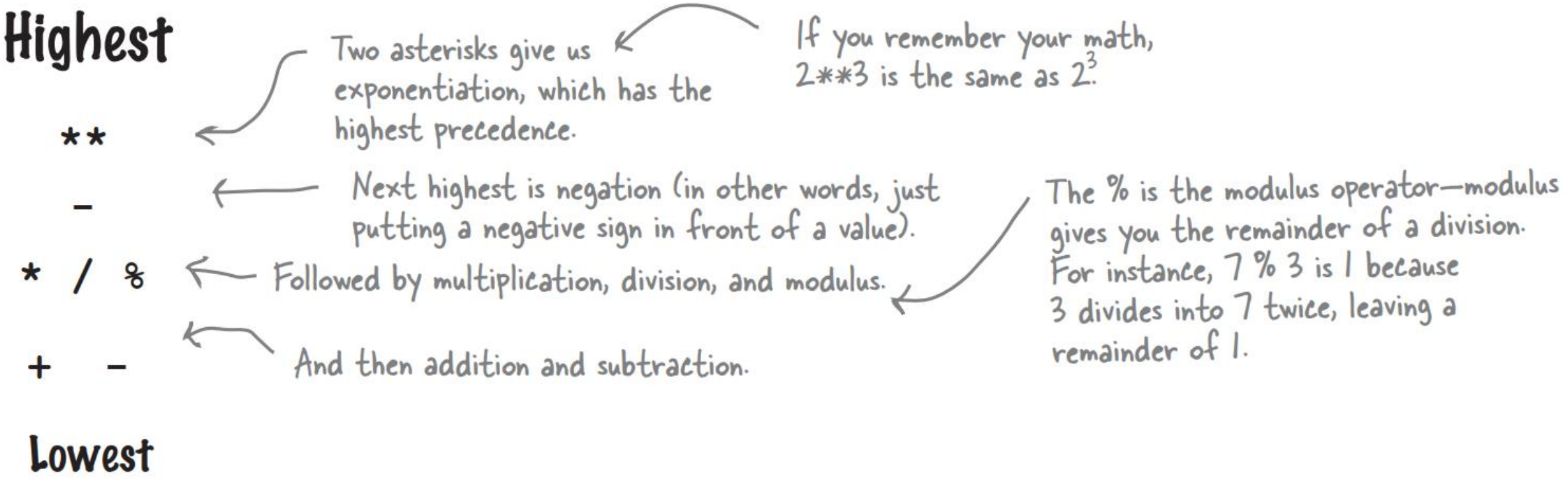
Better living through operator precedence

Evalute this expression:

```
mystery_number = 3 + 4 * 5
```

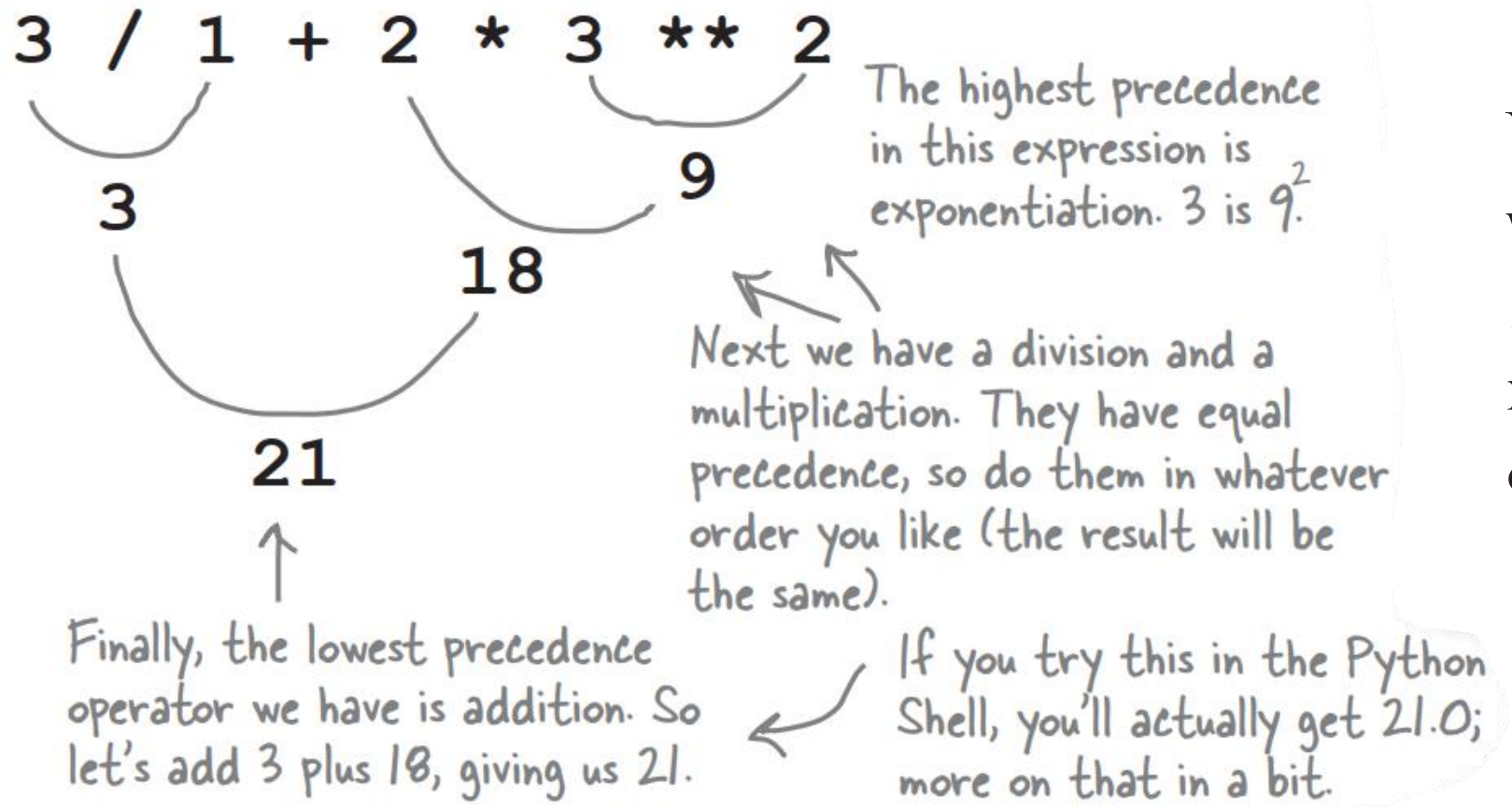
Is `mystery_number` 35? Or is it 23?

How did we know the right order of evaluation? *Operator precedence*. Operator precedence tells you the order in which operations should be applied.

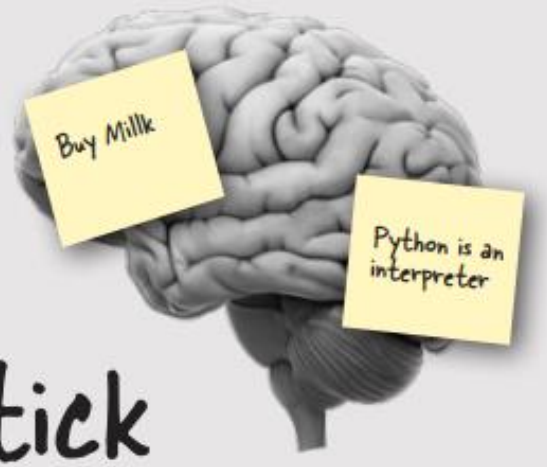


Computing with operator precedence

Let's evaluate an expression to understand how operator precedence is applied. Here's an expression to evalute:

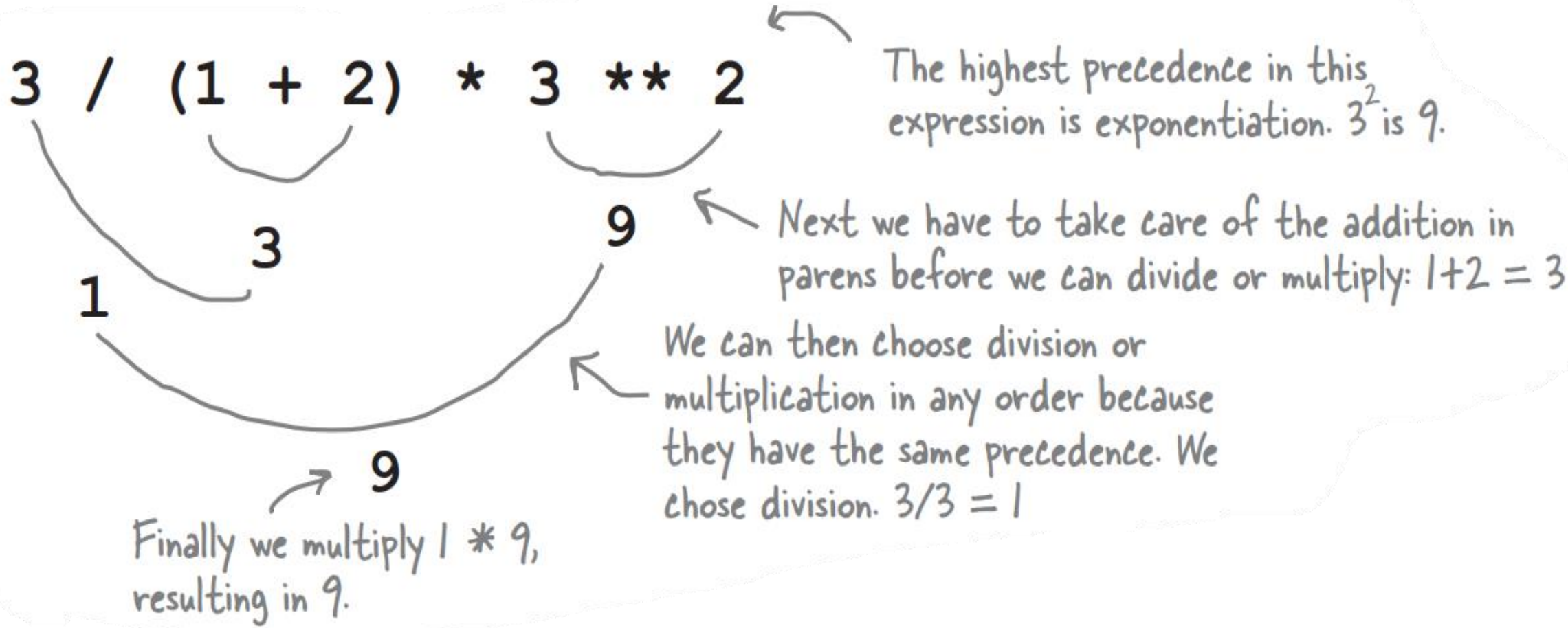


What if you really wanted to add 1 to 2 *before* the division and multiplication occurred?



Make it stick

Remember the mnemonic **PEMDAS** for **P**arentheses, **E**xponents, **M**ultiplication, **D**ivision, **A**ddition, and **S**ubtraction. If you follow that order of evaluation, left to right, you'll always evaluate expressions correctly. If you have trouble remembering PEMDAS, you could always try "Please Excuse My Dear Aunt Sally."

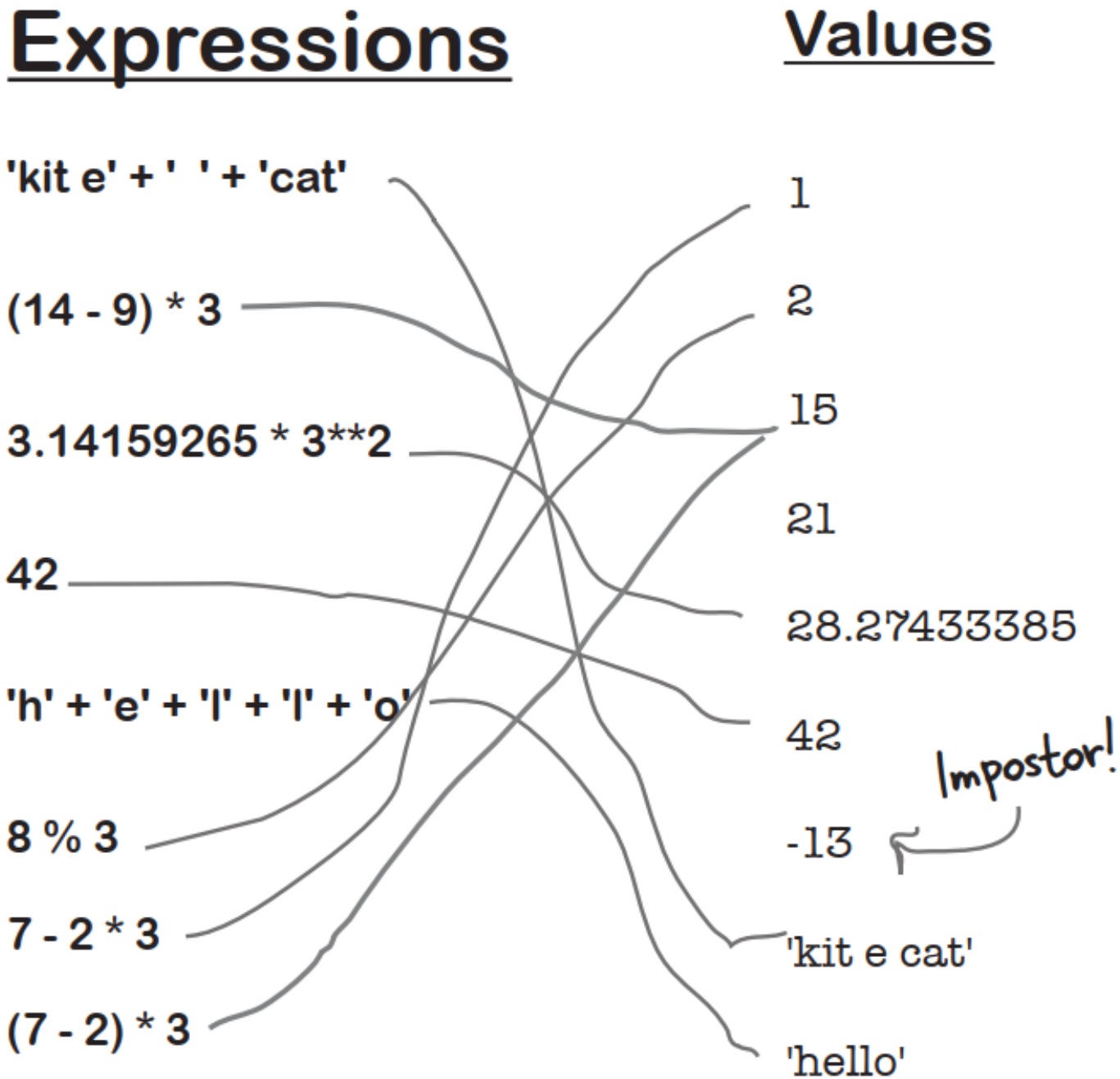

3 / (1 + 2) * 3 ** 2
The highest precedence in this expression is exponentiation. 3^2 is 9.
9
Next we have to take care of the addition in parens before we can divide or multiply: 1+2 = 3
3
1
We can then choose division or multiplication in any order because they have the same precedence. We chose division. 3/3 = 1
9
Finally we multiply 1 * 9, resulting in 9.

```
(((3 / 1) + 2) * 3) ** 2
```

You can also add parens that don't necessarily change evaluation order, but improve your code's readability.

This expression first does division, then addition, then multiplication, then exponentiation, and evaluates to 225.

Expressions	Values
'kit e' + ' ' + 'cat'	1
(14 - 9) * 3	2
3.14159265 * 3**2	15
42	21
'h' + 'e' + 'l' + 'l' + 'o'	28.27433385
8 % 3	42
7 - 2 * 3	-13
(7 - 2) * 3	'kit e cat'
	'hello'


Expressions
Values
'kit e' + ' ' + 'cat'
(14 - 9) * 3
3.14159265 * 3**2
42
'h' + 'e' + 'l' + 'l' + 'o'
8 % 3
7 - 2 * 3
(7 - 2) * 3
1
2
15
21
28.27433385
42
-13
'kit e cat'
'hello'
Impostor!

Get ready for the classic shell game, usually played with cups and balls; in this game we're going to use variables and values. Using what you know about variables, values, and assignment, see if you can beat the cup game. Work through the code and see which cup has the number 1 in it at the end of the game. Will it be in cup 1, cup 2, or cup 3? Place your bet now!

```
cup1 = 0
cup2 = 1
cup3 = 0
cup1 = cup1 + 1
cup2 = cup1 - 1
cup3 = cup1
cup1 = cup1 * 0
cup2 = cup3
cup3 = cup1
cup1 = cup2 % 1
cup3 = cup2
cup2 = cup3 - cup3
```

Use your brain to evaluate this code and see, when it completes, which cup has the 1 in it. When you're done, check your answer at the end of the chapter (you can type the code in as well, but only to check yourself).

Exercise Solution

Get ready for the classic shell game, usually played with cups and balls; in this game we're going to use variables and values. Using what you know about variables, values, and assignment, see if you can beat the cup game. Work through the code and see which cup has the number 1 in it at the end of the game. Will it be in 1, 2, or 3? Place your bet now!

```
cup1 = 0                cup1 is 0
cup2 = 1                cup1 is 0, cup2 is 1
cup3 = 0                cup1 is 0, cup2 is 1, cup3 is 0
cup1 = cup1 + 1         cup1 is 1, cup2 is 1, cup3 is 0
cup2 = cup1 - 1         cup1 is 1, cup2 is 0, cup3 is 0
cup3 = cup1             cup1 is 1, cup2 is 0, cup3 is 1
cup1 = cup1 * 0         cup1 is 0, cup2 is 0, cup3 is 1
cup2 = cup3             cup1 is 0, cup2 is 1, cup3 is 1
cup3 = cup1             cup1 is 0, cup2 is 1, cup3 is 0
cup1 = cup2 % 1         cup1 is 0, cup2 is 1, cup3 is 0
cup3 = cup2             cup1 is 0, cup2 is 1, cup3 is 1
cup2 = cup3 - cup3      cup1 is 0, cup2 is 0, cup3 is 1
```

Winner!

What does this expression evaluate to? Or do you think this is an error in Python because we're multiplying a number times a string?

```
3 * 'ice cream'
```

You might want to try typing it into the Python Shell.

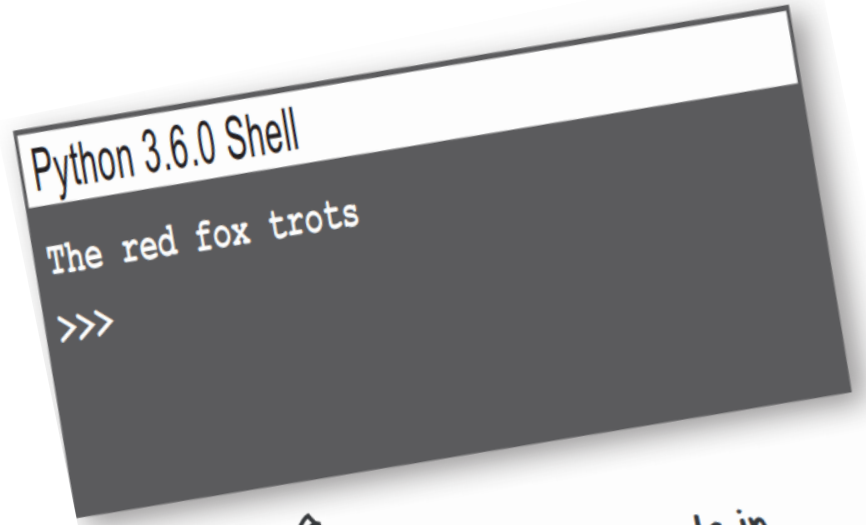
```
word1 = 'ox'
word2 = 'owl'
word3 = 'cow'
word4 = 'sheep'
word5 = 'flies'
word6 = 'trots'
word7 = 'runs'
word8 = 'blue'
word9 = 'red'
word10 = 'yellow'
word9 = 'The ' + word9
passcode = word8
passcode = word9
passcode = passcode + ' f'
passcode = passcode + word1
passcode = passcode + ' '
passcode = passcode + word6
print(passcode)
```

This prints your passcode.

Here's your passcode; all you have to do is work through the code to get it.

```
word1 = 'ox'
word2 = 'owl'
word3 = 'cow'
word4 = 'sheep'
word5 = 'flies'
word6 = 'trots'
word7 = 'runs'
word8 = 'blue'
word9 = 'red'
word10 = 'yellow'
word9 = 'The ' + word9
passcode = word8
passcode = word9
passcode = passcode + ' f'
passcode = passcode + word1
passcode = passcode + ' '
passcode = passcode + word6
print(passcode)
```

```
word9 is 'The red'
passcode is 'blue'
passcode is 'The red'
passcode is 'The red f'
passcode is 'The red fox'
passcode is 'The red fox '
password is 'The red fox trots'
```

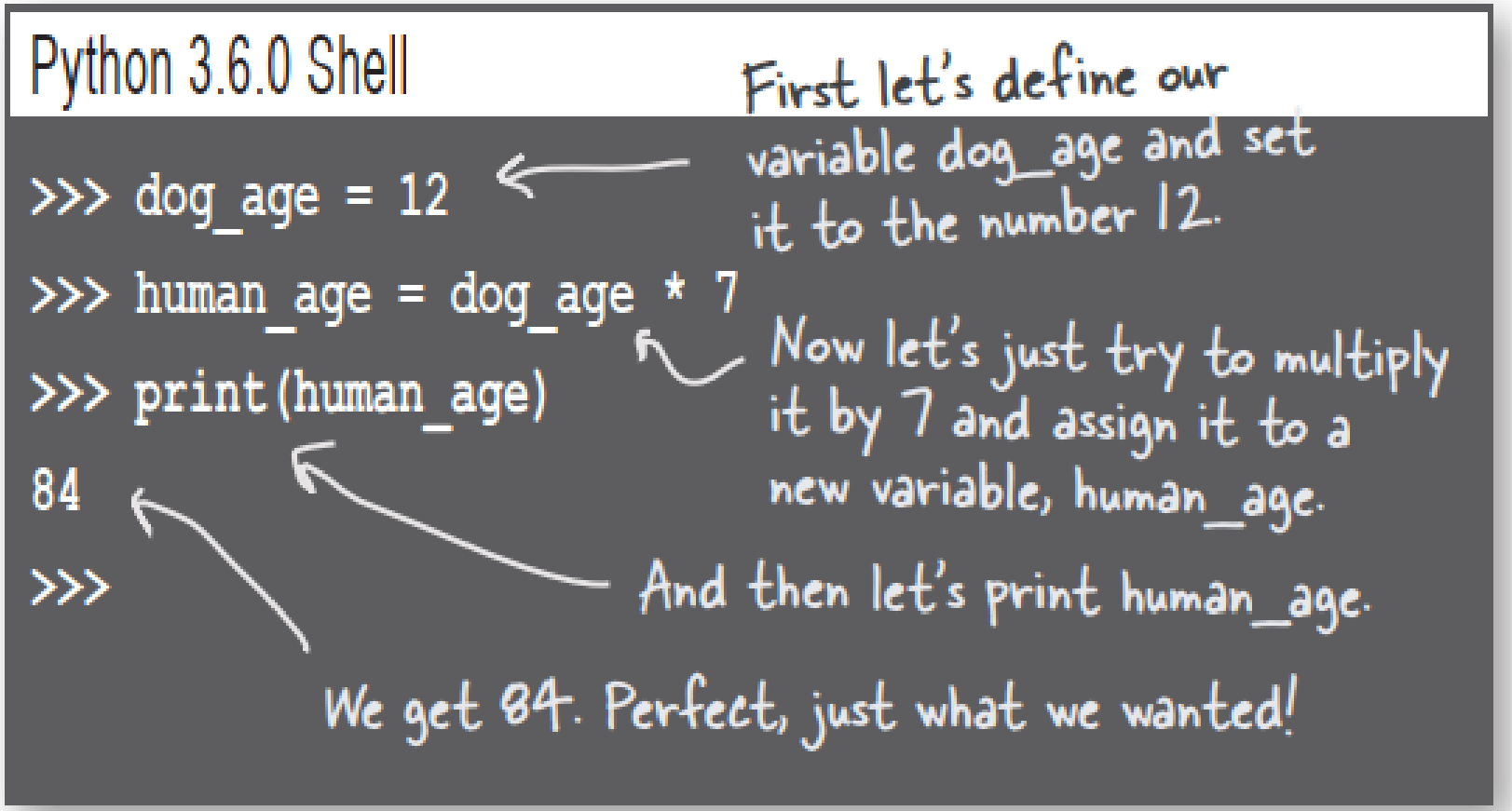


If you type the code in and run it, you'll get this.

The red fox trots

The passcode!

Step 3: Computing the dog's age



Dog Age Calculator Pseudocode

1. **Ask the user for the dog's name.**
2. **Ask the user for the dog's age.**
3. **Multiply the dog's age by the number 7 to get the dog's age in human years.**
4. **Output to the user:**

   "Your dog"
   the dog's name
   "is"
   the dog's age in human years
   "years old in human years"

Sharpen your pencil

Below you'll find the Dog Age Calculator code so far. Using the experiment above as a guide, add the code to compute the dog age in human years.

```
dog_name = input("What is your dog's name? ")
dog_age = input("What is your dog's age? ")
```

Add your new code here, then check your answer before moving on. →

A Test Drive

Let's test drive our code now that we've got step 3 coded. Go ahead and get the new code into your *dogcalc.py* file, save your code, and choose the **Run > Run Module** menu item. After that, head to the shell and enter your dog's name and age, and then check the output to see that we're calculating the human age correctly.

Here's the code again:

```
dog_name = input("What is your dog's name? ")
dog_age = input("What is your dog's age? ")
human_age = dog_age * 7
print(human_age)
```

Add this new code to your file.

Throughout the book you'll find the gray background signifies new code additions.

Could you have printed dog_age * 7 directly? Like print(dog_age * 7)? Give it a try.

```
Python 3.6.0 Shell
What is your dog's name? Codie
What is your dog's age? 12
12121212121212
>>>
```

Okay, that's not right!

BRAIN POWER

Think about the number `12121212121212`. Is there anything you can think of that explains how Python got this number?

Hint: how many 12s are in 12121212121212?

syntax errors

You'll know quickly if you have a syntax error in Python when you get the dreaded `'Syntax Error'` message from the interpreter. **Syntax errors are the equivalent of making grammatical errors**—in other words, you've typed something that violates the conventions for writing correct Python. The good news is syntax errors are usually easy to fix—just find the offending line of code and double-check your syntax.

syntax errors

You'll know quickly if you have a syntax error in Python when you get the dreaded `'Syntax Error'` message from the interpreter. **Syntax errors are the equivalent of making grammatical errors**—in other words, you've typed something that violates the conventions for writing correct Python. The good news is syntax errors are usually easy to fix—just find the offending line of code and double-check your syntax.

runtime errors

Runtime errors occur when you've written a syntactically correct program, but Python encounters a problem running your program. An example of a runtime error would be if at some point in your code you accidentally divided a number by zero (an invalid mathematical operation in any language). To fix runtime errors, look at the specific error you received and then track down where in your code you're causing the runtime condition to occur.

syntax errors

You'll know quickly if you have a syntax error in Python when you get the dreaded `'Syntax Error'` message from the interpreter. **Syntax errors are the equivalent of making grammatical errors**—in other words, you've typed something that violates the conventions for writing correct Python. The good news is syntax errors are usually easy to fix—just find the offending line of code and double-check your syntax.

runtime errors

Runtime errors occur when you've written a syntactically correct program, but Python encounters a problem running your program. An example of a runtime error would be if at some point in your code you accidentally divided a number by zero (an invalid mathematical operation in any language). To fix runtime errors, look at the specific error you received and then track down where in your code you're causing the runtime condition to occur.

Semantic errors

Semantic errors are also known as logic errors. With a semantic error your program will appear to operate normally—the interpreter won't complain that you've made a syntax error and at runtime you won't encounter any issues, but your program won't give you the results you expected. **These always occur because what you think you've told your program to do isn't actually what you're telling it to do.** Semantic errors can be some of the toughest errors to debug.

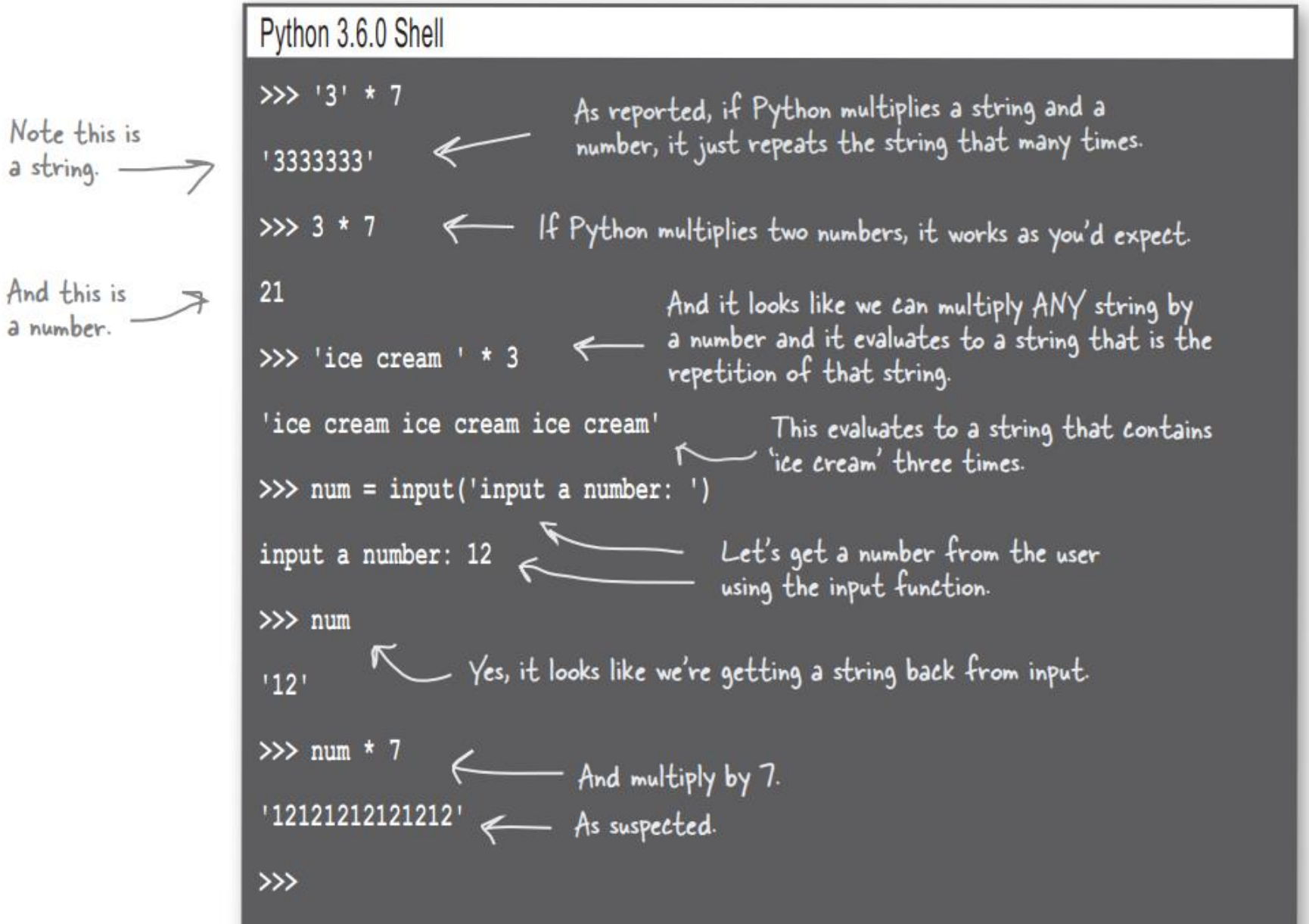
We currently have a bug in our Dog Age Calculator because we're getting `121212121212` instead of `84`. What kind of error is this?

- ☐ A. Runtime error
- ☐ B. Syntax error
- ☐ C. Semantic error
- ☐ D. Calculation error
- ☐ E. None of the above
- ☐ F. All of the above

Answer C because there are no syntax or runtime errors.

A little more debugging...

Let's do a little investigative debugging and see exactly what is happening. Using our handy Python Shell let's try a few things:

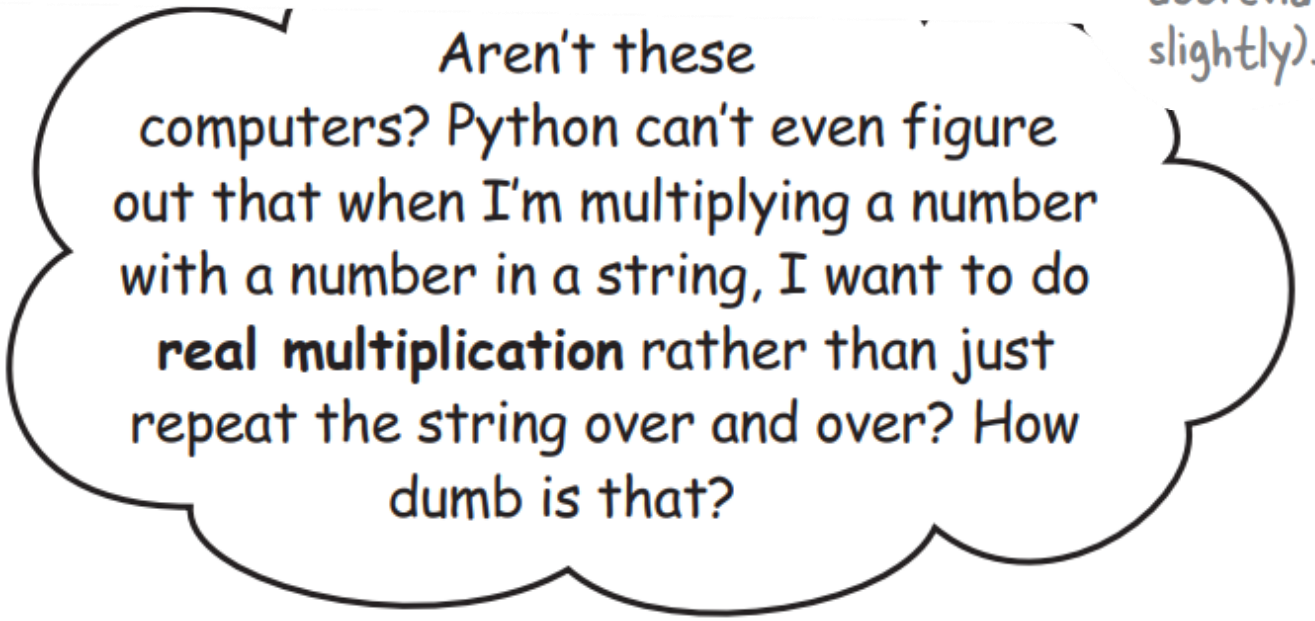


So why is Python treating the number we entered using the `input` function as a string?

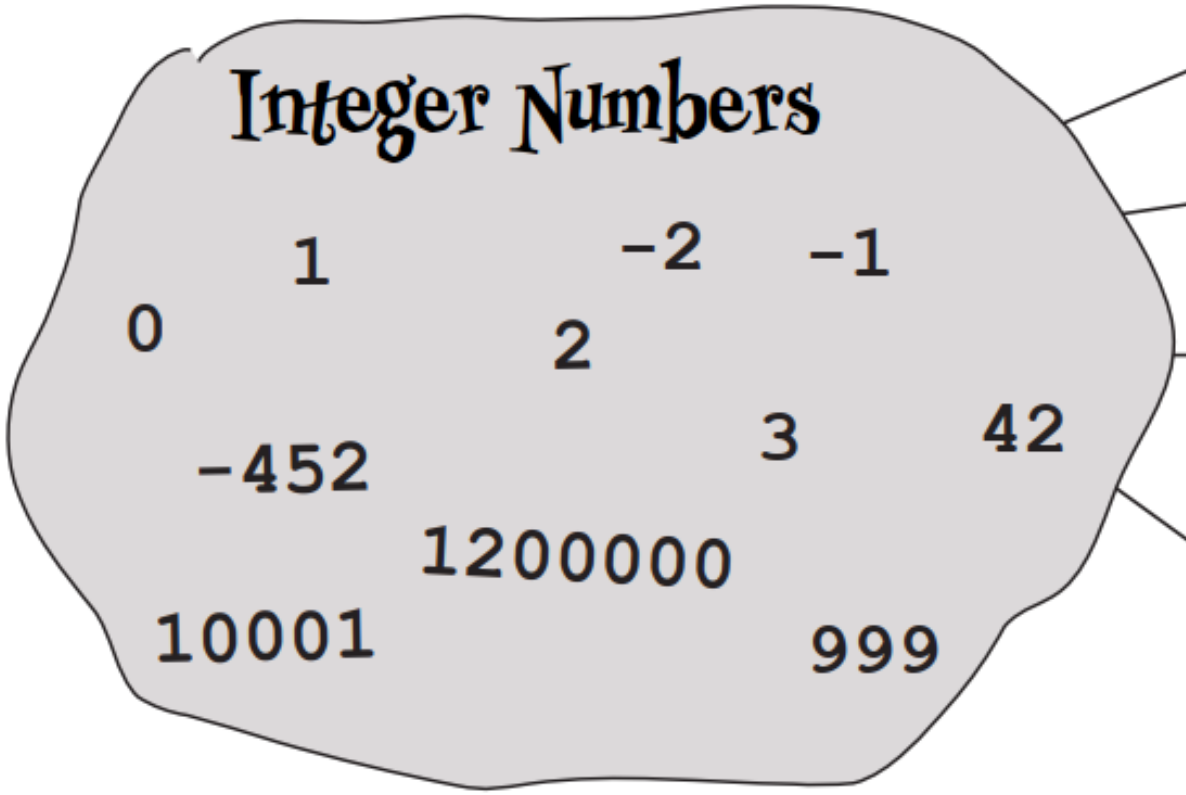
because the `input` function always returns a string.

`input(prompt)` **After the prompt argument is written to output, the function then reads a line from input, converts it to a string, and returns that.**

This definition is from the Python specification (we abbreviated it slightly).



What are Python types, anyway?



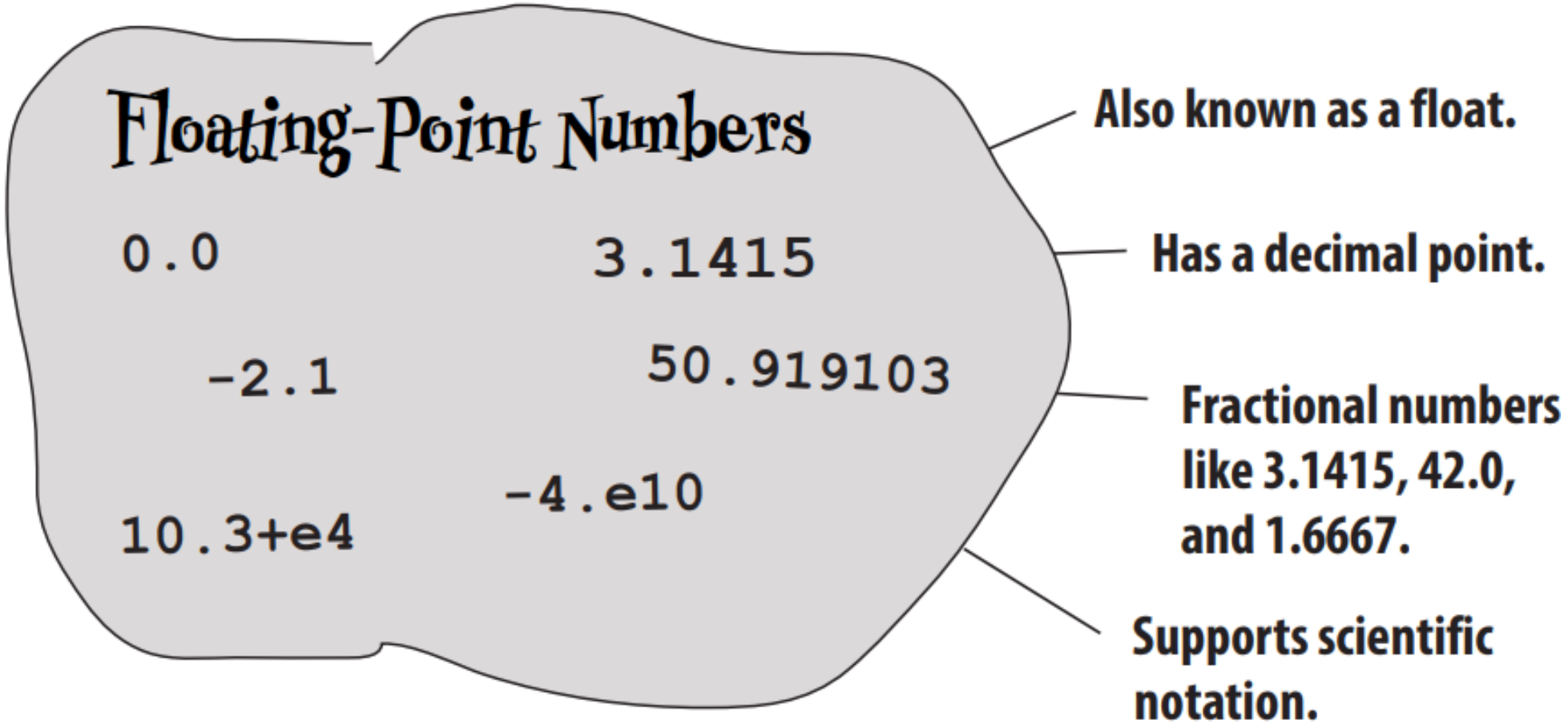
Also known as an int.

Has no decimal point.

Whole numbers like
–3, –2, –1, 0, 1, 2, 3, ...

Notice we don't use commas with Python numbers greater than 999.

What are Python types, anyway?



Python has other types too, like **booleans**, **lists**, **dictionaries**, and **objects**, that we'll be getting to later.

Fixing our code

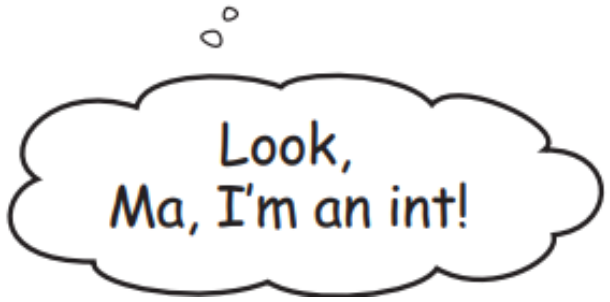
1. First, let's create a string representation of a number.

```
answer = '42'
```

Here the string '42' is assigned to the variable answer.

2. Next, call the `int` function and pass it the string.

```
answer = int('42')
```



Here the string '42' is converted to an integer and assigned to the variable answer.

Sharpen your pencil

So we know that the bug is caused by our assumption that `dog_age` is a number, when in fact Python is treating it as a string. Take the code below and add an `int` function to fix this problem.

There's a few ways to approach this, so be sure and check your work with the answer at the end of the chapter.

```
dog_name = input("What is your dog's name? ")
dog_age = input("What is your dog's age? ")
human_age = dog_age * 7
print(human_age)
```

Sharpen your pencil Solution

So we know that our bug is caused by our assumption that `dog_age` is a number, when in fact Python is treating it as a string. Take the code below and add an `int` function to fix this problem.

There are quite a few ways to add the int function. Here's a few:

```
dog_name = input("What is your dog's name? ")
dog_age = input("What is your dog's age? ")
dog_age = int(dog_age)
human_age = dog_age * 7
print(human_age)
```

First get dog_age as a string and then call the int function on it, and then reassign the result to the dog_age variable.

A Test Drive

So now that we've figured out the code to convert the string to an int (in the last exercise), let's get it added. First make the changes below to update your code to handle the conversion of the dog age to an integer, then save your code and choose the **Run > Run Module** menu item. After that, head to the console and enter your dog's name and age, and check the output to see that we're *finally* calculating the human age correctly.

Here's the code (changes are highlighted):

We added the int function here so that the dog_age value is converted to an integer just before it is multiplied by 7.

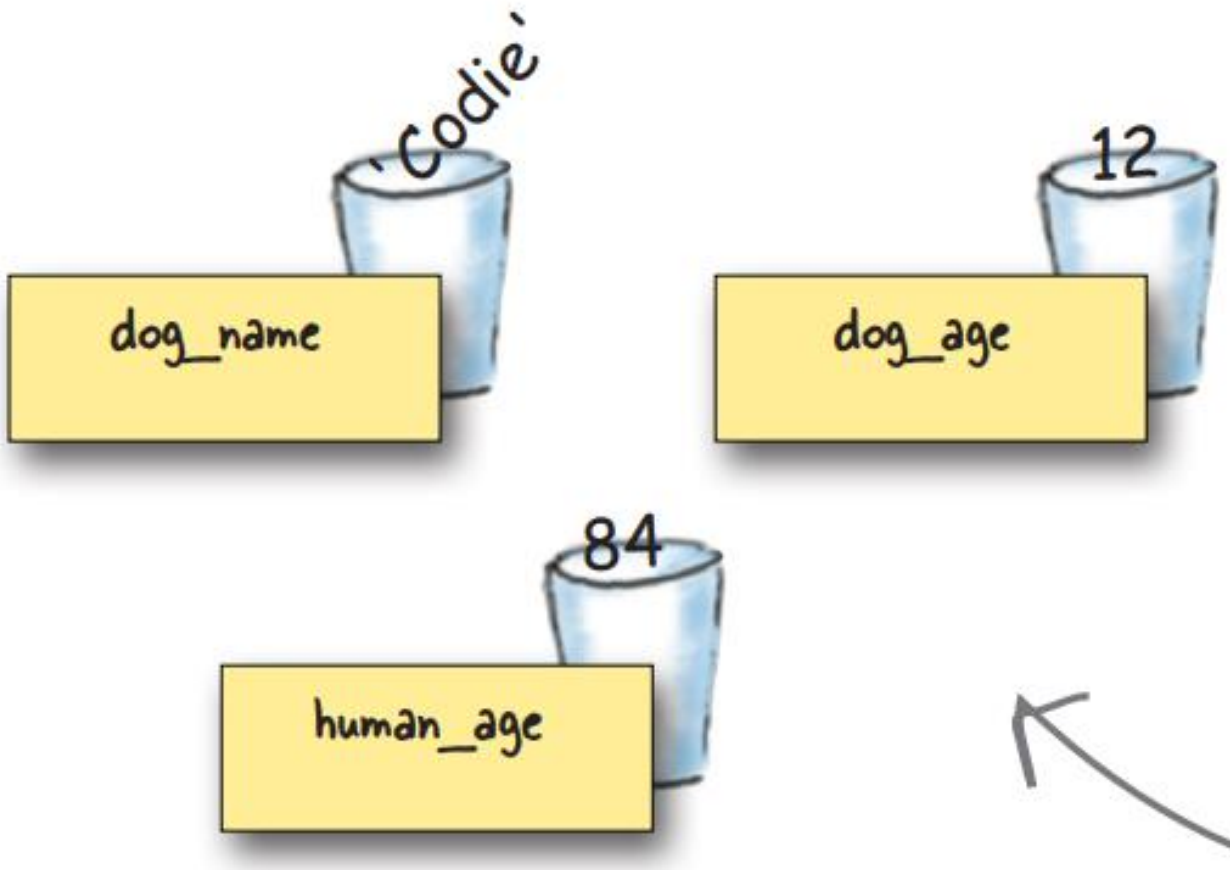
```
dog_name = input("What is your dog's name? ")
dog_age = input("What is your dog's age? ")
human_age = int(dog_age) * 7
print(human_age)
```

```
dog_name = input("What is your dog's name? ")
dog_age = int(input("What is your dog's age? "))
human_age = dog_age * 7
print(human_age)
```

Or finally, use int around the call to input. This way, dog_age is always assigned to an integer.

If you find these confusing, we'll be getting into how to use and call functions later in the book. For now, just know you can pass a string to the int function and have it convert the string to an integer.

Step 4: User-friendly output



Dog Age Calculator Pseudocode

1. **Ask the user for the dog's name.**
2. **Ask the user for the dog's age.**
3. **Multiply the dog's age by the number 7 to get the dog's age in human years.**
4. **Output to the user:**

   "Your dog"
   the dog's name
   "is"
   the dog's age in human years
   "years old in human years"

We've got all the values we need to finish the last step of our pseudocode.

we've just been providing `print` with one *argument*, like:

We refer to the values we provide to a function as its arguments. We also typically say that we pass those arguments to a function.

```
print('Hi there')
```

Until now we've only been passing single values to print, either strings or numbers.

or:

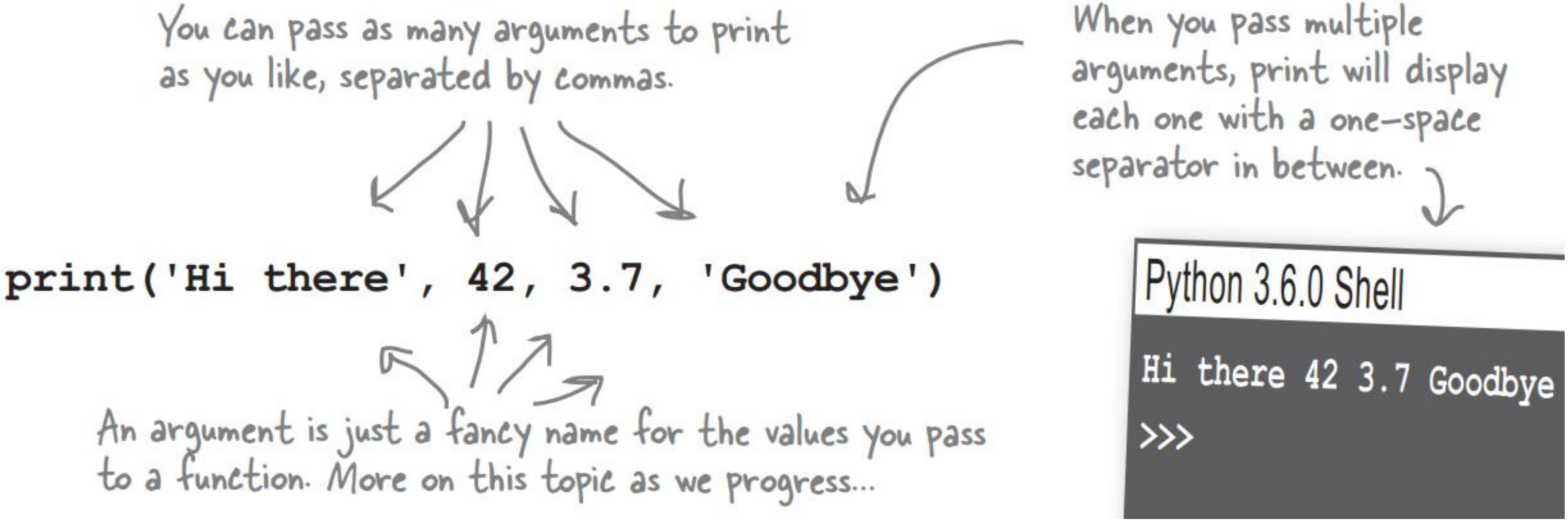
```
print(42)
```

or:

```
print('Good' + 'bye')
```

We've also used string concatenation to build up strings in the print function, but that still, ultimately, evaluates to just one argument.

But you can actually pass multiple arguments to `print`, like:



Sharpen your pencil

With your newfound print functionality, write the code to call to the `print` function that will produce the user-friendly output:

Output to the user:

"Your dog"
the dog's name
"is"
the dog's age in human years
"years old in human years"

Here's what you need to output.

```
dog_name = input("What is your dog's name? ")
dog_age = input("What is your dog's age? ")
human_age = int(dog_age) * 7
print(human_age)
print('Your dog',
      dog_name,
      'is',
      human_age,
      'years old in human years')
```

Remove this code.

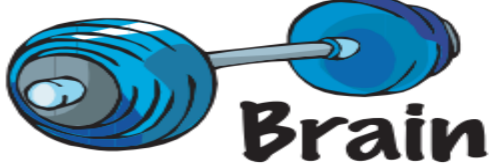
Add this code to your dogcalc.py file.

Our final output

Python 3.6.0 Shell

```
What is your dog's name? Codie
What is your dog's age? 12
Your dog Codie is 84 years old in human years
>>>
```

Just what we'd planned for!



Brain Building

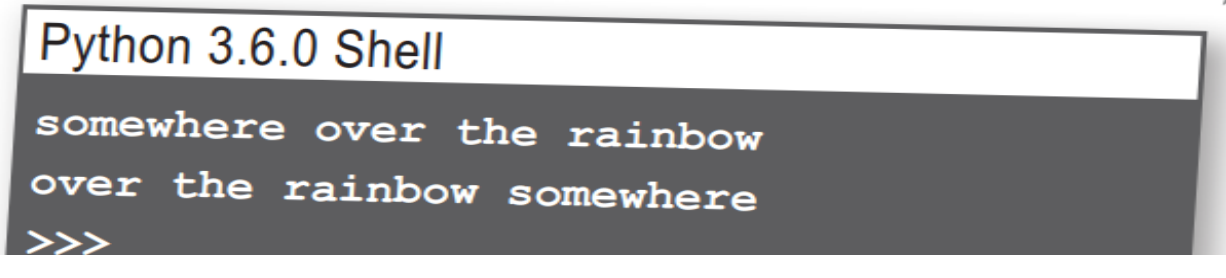
say you have two variables, `first` and `last`, like below. Can you swap their values? See if you can write the code.

```
first = 'somewhere'
last = 'over the rainbow'
print(first, last)
```

Write your code here.

```
print(first, last)
```

Here's the output you should get when you tune this code.



Brain Building Solution

Well, the chapter is almost over; you've just got the bullet points and a crossword to go. Why not do a few extra reps for that brain before you go?

Here's one for you: say you have two variables, `first` and `last`, like below. Can you swap their values? See if you can write the code:

```
first = 'somewhere'
last = 'over the rainbow'
print(first, last)
temp = first
first = last
last = temp
print(first, last)
```

A common technique is to use a temporary variable to store the first item while you set it to the second item. Then you set the second item to the temporary variable.

Trace through this until you understand how it works.

What if you left the quotes off of Codie's name? What would this evaluate to?

```
dog_name = Codie
```