INTRODUCTION TO PROGRAMMING IN PYTHON

Philo van Kemenade
General Assembly
London
23 February 2013

HELLO

Philo van Kemenade

- BSc Artificial Intelligence, University of Amsterdam
- MSc Cognitive Computing, Goldsmiths College

I use Python for:

- tracking video propagation in over 100gb of tweets
- building statistical models for natural language processing
- analysing experimental data from a database
- talking to robots :)

THIS SESSION

Programming

Python

Syntax

- Words
- Sentences
- Stories

INTRO

Programming

ABOUT PROGRAMMING...

Common Misconceptions:

- only ‘coders’ can program
- something magical is happening
- programming is hard

Take home

- programming language ≈ **natural language**
- **no secret magic**
- learning to program needs **active exercise**
- you need to **start somewhere**

ABOUT PROGRAMMING...

Why programming?

ABOUT PROGRAMMING...

Telling your computer what to do

Statements are elementary instructions that make up a program

ABOUT PROGRAMMING...

```
class TicTacToe(object):
    """A game of Tic Tac Toe"""
    def __init__(self):
        self.playing = True
        self.board = Board()
        self.turn = self.whoStarts()
        print "Welcome to Tic Tac Toe"

    def whoStarts(self):
        if random.randint(0,1) == 0:
            return "X"
        else:
            return "O"


  ...
```

words

sentences

stories

ABOUT PROGRAMMING...

```
class TicTacToe(object):
    """A game of Tic Tac Toe"""
    def __init__(self):
        self.playing = True
        self.board = Board()
        self.turn = self.whoStarts()
        print "Welcome to Tic Tac Toe"

    def whoStarts(self):
        if random.randint(0,1) == 0:
            return "X"
        else:
            return "O"


    ...
```

words

sentences

stories

ABOUT PROGRAMMING...

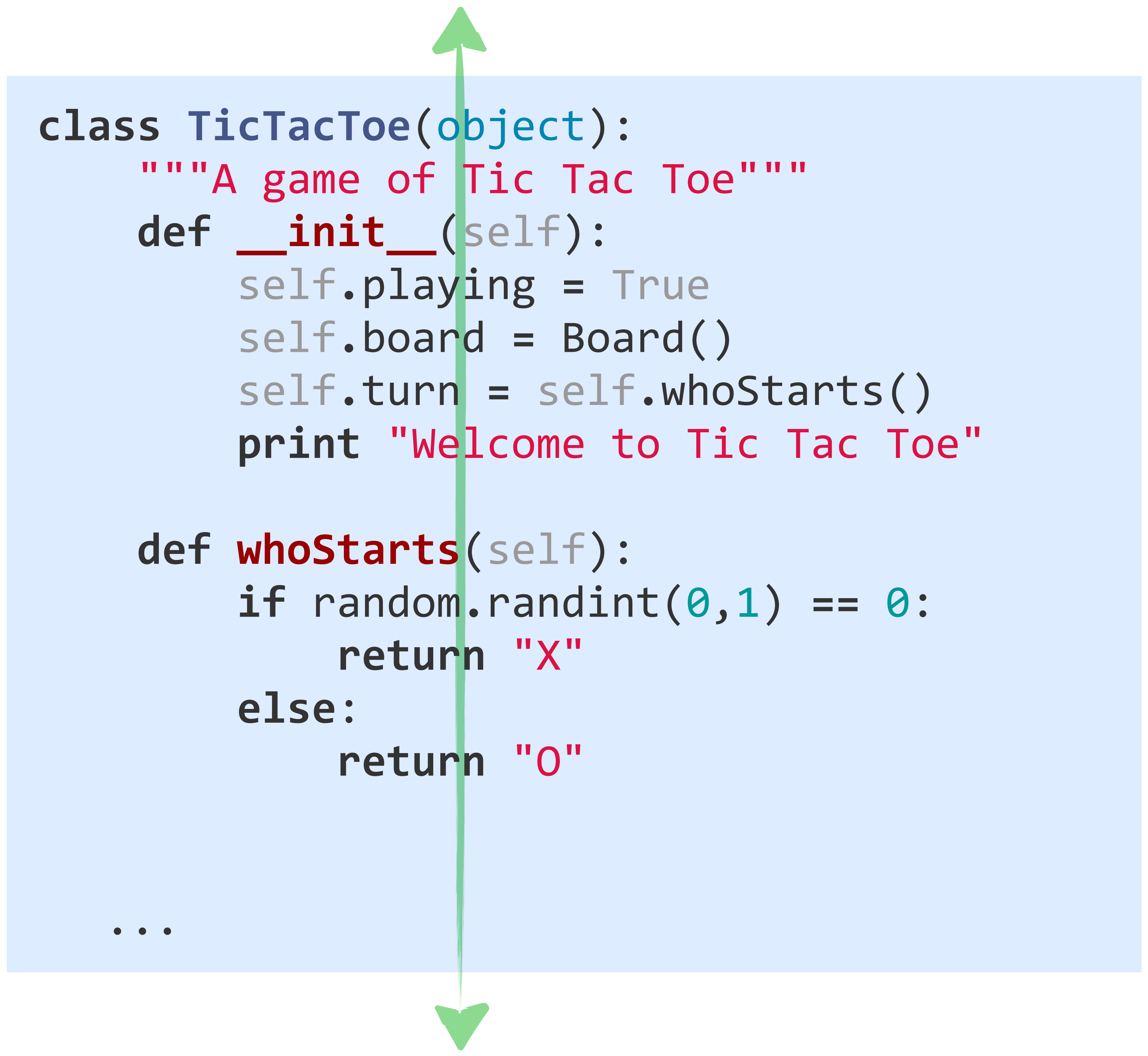
```
class TicTacToe(object):
    """A game of Tic Tac Toe"""
    def __init__(self):
        self.playing = True
        self.board = Board()
        self.turn = self.whoStarts()
        print "Welcome to Tic Tac Toe"

    def whoStarts(self):
        if random.randint(0,1) == 0:
            return "X"
        else:
            return "O"


  ...
```

words

sentences

stories

ABOUT PROGRAMMING...



```
class TicTacToe(object):
    """A game of Tic Tac Toe"""
    def __init__(self):
        self.playing = True
        self.board = Board()
        self.turn = self.whoStarts()
        print "Welcome to Tic Tac Toe"

    def whoStarts(self):
        if random.randint(0,1) == 0:
            return "X"
        else:
            return "O"


  ...
```

words

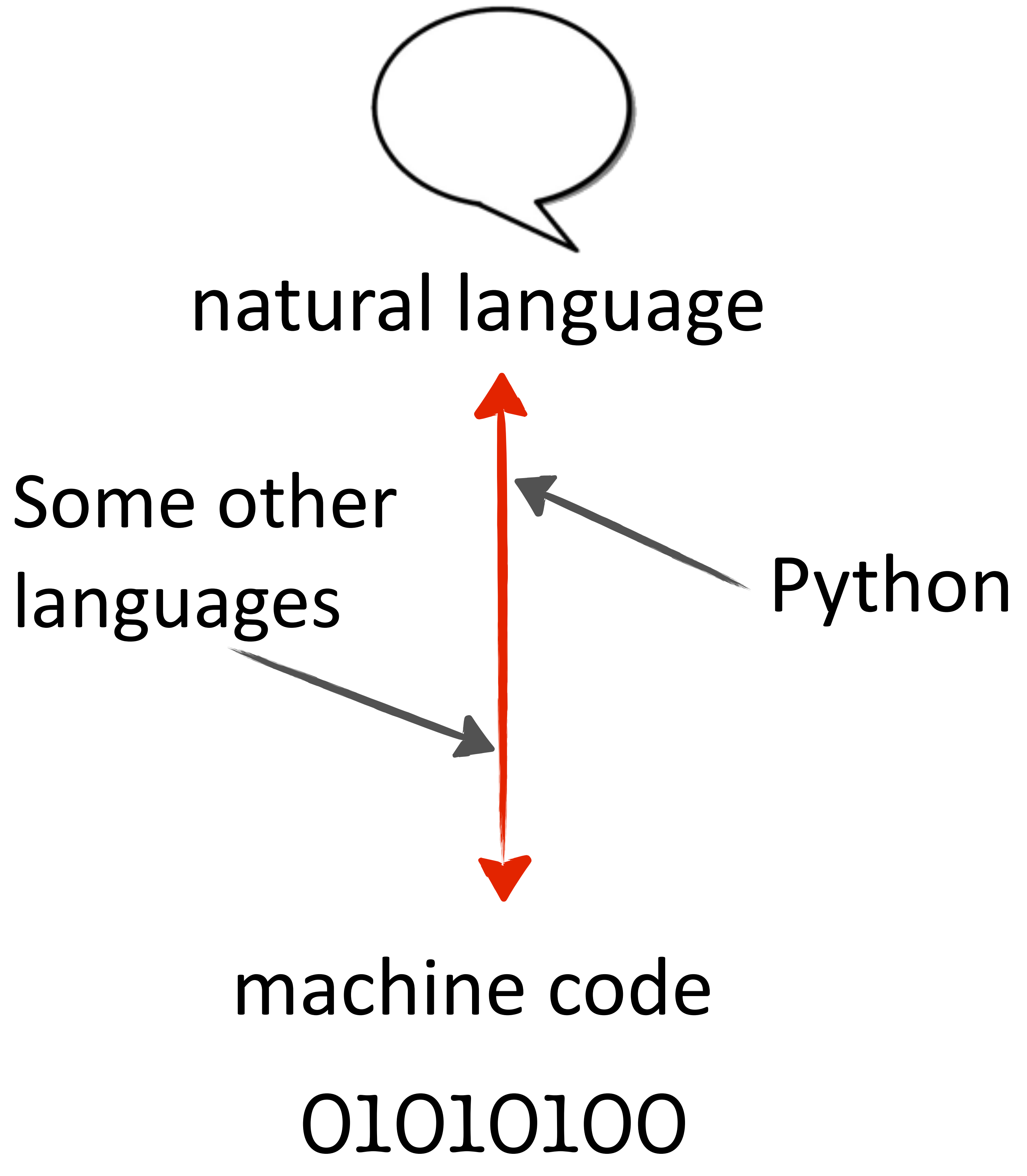
sentences

stories

INTRO

Python

PROGRAMMING IN PYTHON



one of Python's most attractive features

PROGRAMMING IN PYTHON

Features

- Easy to use
- Powerful & fast
- Connects to other languages and protocols
- Platform independent
- Strong community
- Free & open source

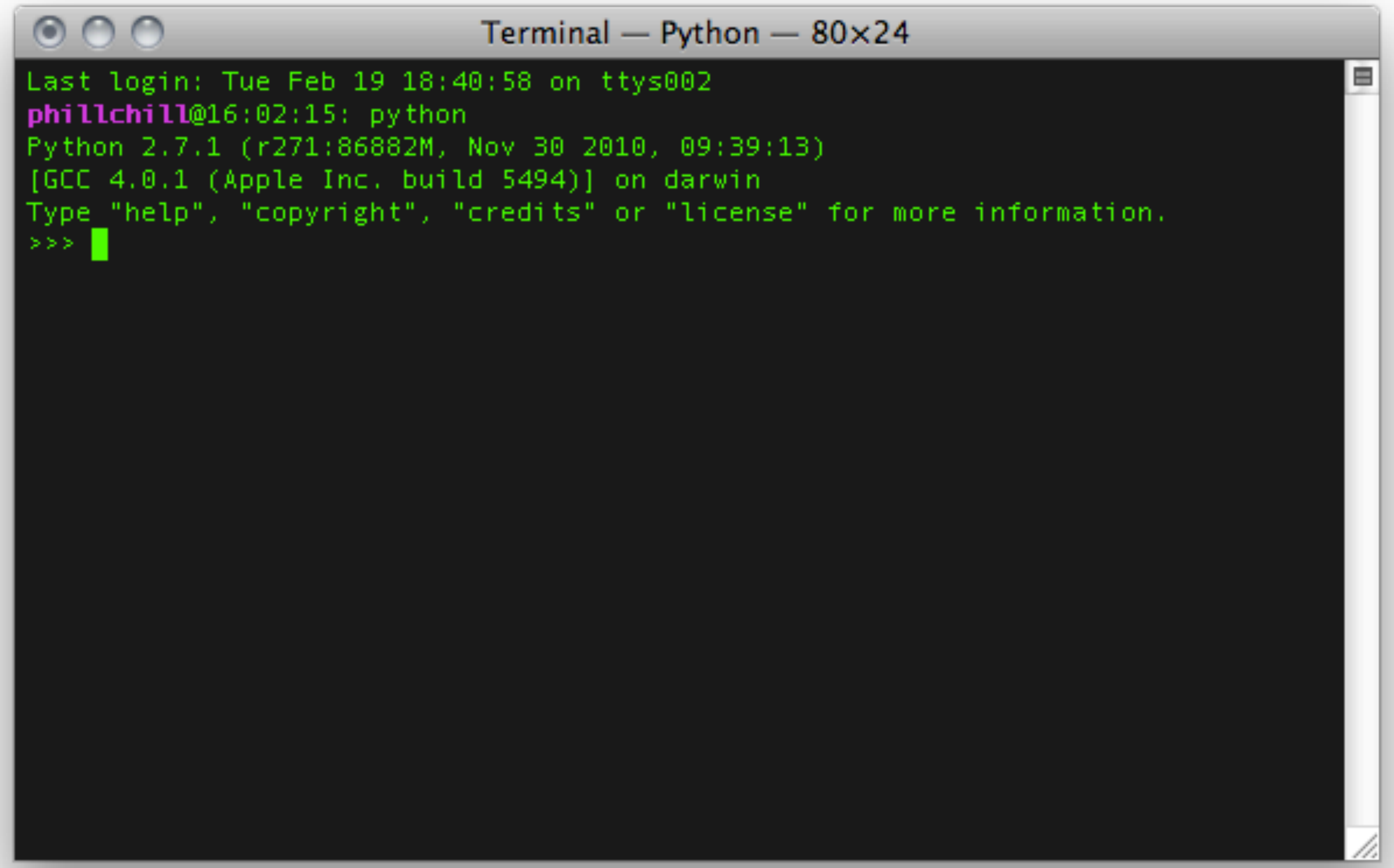
Documentation at www.python.org/doc/

PROGRAMMING IN PYTHON

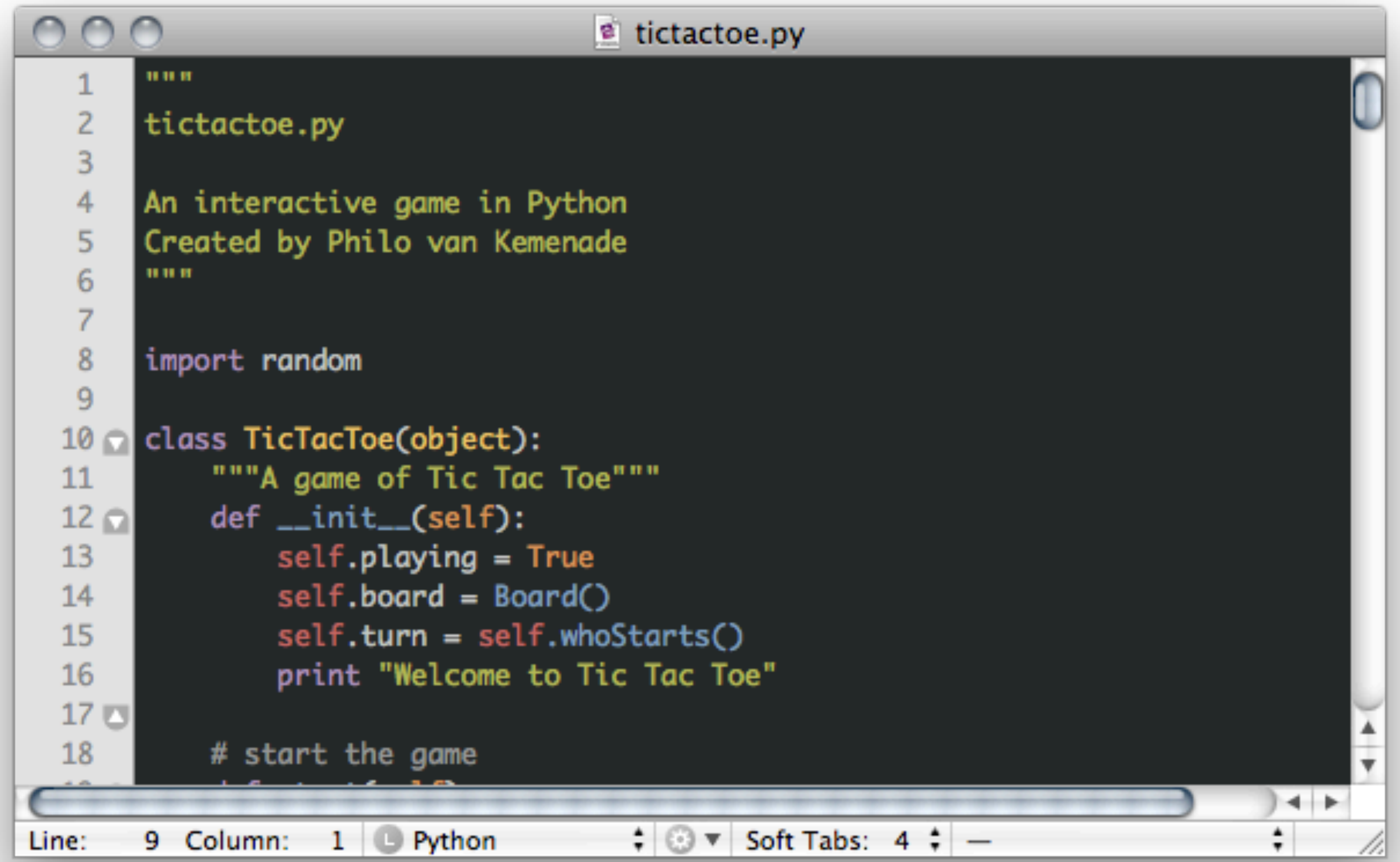
Two modes of operation

Interactive Mode
- Terminal-based
- Direct feedback
- Try out code

Scripting Mode
- Code in file
- Save & load programs
- More control



```
Last login: Tue Feb 19 18:40:58 on ttys002
phillchill@16:02:15: python
Python 2.7.1 (r271:86882M, Nov 30 2010, 09:39:13)
[GCC 4.0.1 (Apple Inc. build 5494)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```



```python
"""
tictactoe.py

An interactive game in Python
Created by Philo van Kemenade
"""

import random

class TicTacToe(object):
    """A game of Tic Tac Toe"""
    def __init__(self):
        self.playing = True
        self.board = Board()
        self.turn = self.whoStarts()
        print "Welcome to Tic Tac Toe"

    # start the game
```

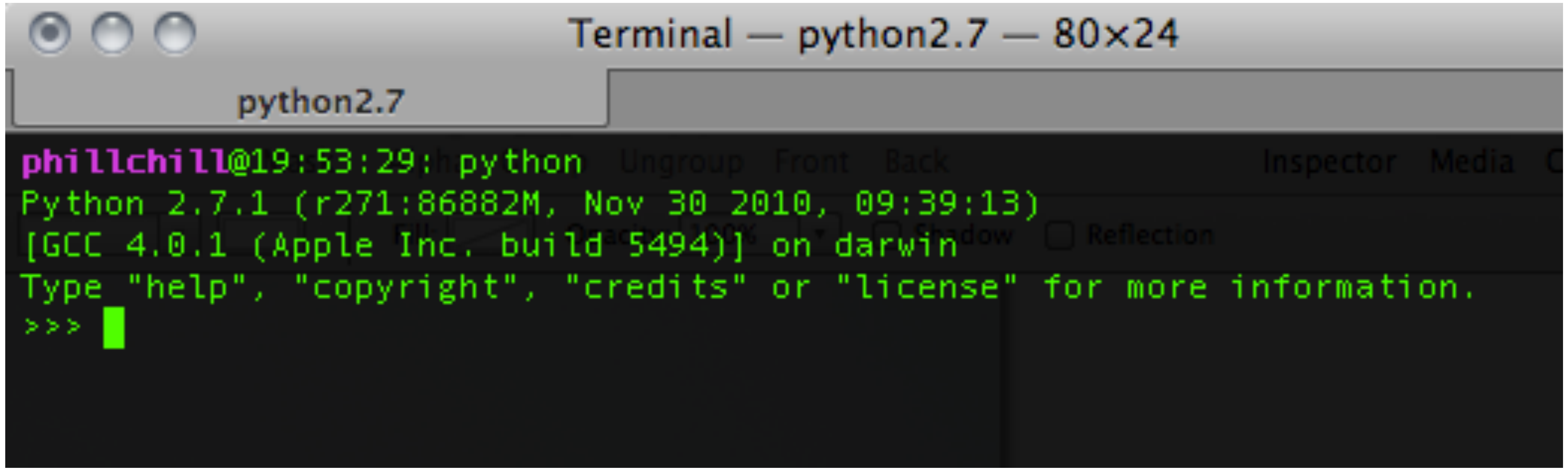
SETTING UP

Download python 2.7 from http://python.org/download/

Install Python

Open terminal / cmd window

Type “python” to start Python in interactive mode



```
phillchill@19:53:29: python
Python 2.7.1 (r271:86882M, Nov 30 2010, 09:39:13)
[GCC 4.0.1 (Apple Inc. build 5494)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

PROGRAMMING IN PYTHON

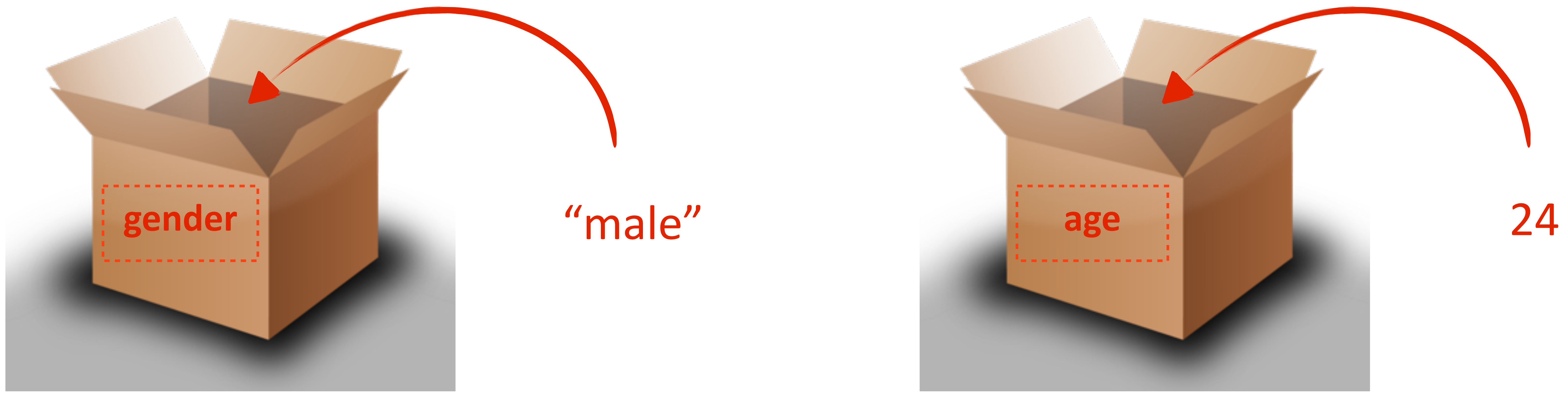
Words

FIRST WORDS: VARIABLES

Variables store *values* under a specified name

FIRST WORDS: VARIABLES

Variables store *values* under a specified name



FIRST WORDS: VARIABLES

Variables store values of different ***types***:

int - an integer, or whole number [1,5,9999, ...]

float - a floating point number (using a decimal point) [3.14, 1.68, 1.0, ...]

bool - boolean; binary true or false values [True, False]

string - a sequence of characters, comprising text [”a”, “London”, “X”]

USING WORDS: OPERATORS

You can process the values in your variables by ***operators***:

For example:

=	assignment: assign a value to a variable
==	comparison: are two variables equal?
!=	comparison: are two variables unequal?
<, >, <=, >=	less-than, greater-than, less or equal, greater or equal
+, -, *, /	mathematical operators
and, or	logical operators

EXAMPLE

Integers

```
>>> 1 + 1
2
>>> cats = 2
>>> cats
2
>>> dogs = 3
>>> cats == dogs
False
>>> cats < dogs
True
>>> dogs + 1
4
>>> dogs
3
>>> dogs = dogs + 1
>>> dogs
4
>>> pets = cats + dogs
>>> pets
6
```

Strings

```
>>> start = "Lon"
>>> start
'Lon'
>>> end = "don"
>>> start + end
'London'
>>> start + start + end
'LonLondon'
>>> town = 3 * start + end
>>> town
'LonLonLondon'
```

EXERCISE

Create two *string variables*:

`first` for your first name and `last` for your last name.

Can you make your full name by combining `first` and `last`?

What happens if we compare `first` and `last` with the ‘<‘ and ‘>’ *operators*?

Why?

(Cheating is encouraged)

RICHER VOCABULARY: COLLECTIONS OF DATA

Data can also be stored in a collection:

- List
- Dictionary
- Tuple
- Set

COLLECTIONS OF DATA: LISTS

We can store multiple values in a ***list:***

```
>>> l = [1,3,9,4,884328881]
>>> n = ['sex', 'drugs', 'rock', 'roll']
>>> m = l + n
>>> m
[1, 3, 9, 4, 884328881, 'sex', 'drugs', 'rock', 'roll']
```

A list is a ordered sequence of items (between [...]) that all have their own index:

```
>>> m[0]
1
>>> m[7]
'rock'
```

COLLECTIONS OF DATA: DICTIONARIES

Data can be stored associatively in a ***dictionary***:

```
>>> personX = {'first':'Philo', 'last':'van Kemenade', 'twitter':'@phivk'}
>>> personX['first']
'Philo'
>>> personX['age'] = 24
>>> personX
{'twitter': '@phivk', 'last': 'van Kemenade', 'age': 24, 'first': 'Philo'}
```

Dictionaries are collections of (*key:value*) pairs (between {...}).

Values are indexed by a unique key (e.g. string or int)

Dictionary items are unordered

EXERCISE

Create a *dictionary* to represent your neighbour in some attributes (for example hometown, hair colour, favorite movie).

Add a <u>list</u> of interests to the <u>dictionary</u>. What do you use as *key*, what do you use as *value*?

Bonus: For our tictactoe game, we'd like to represent an empty 3x3 board by 3 rows of 3 cells each. How could we represent this using lists?

(Cheating is encouraged)

PROGRAMMING IN PYTHON

Sentences

FUNCTIONS

Functions perform multiple tasks, collected under a specified name

Take input *argument(s)*, execute statement(s), (return output)

Input and output can be of all different types

Recognizable by pair of parentheses

```
>>> name = "Philo van Kemenade"
>>> length = len(name)
>>> print(length)
18
>>> type(length)
<type 'int'>
```

FUNCTIONS

You can *define* your own functions like this

```
>>> def multiply(a, b):
...     return a * b
...
>>> multiply(3,4)
12
```

FUNCTIONS

You can *define* your own functions like this

```
>>> def multiply(a, b):
...     return a * b
...
>>> multiply(3,4)
12
```

mind the *indentation*

CONDITIONAL STATEMENTS

Conditional statements enable you to deal with multiple options.

A part of your code is executed based on the truth value of a condtion.

You perform conditional checks with: `if`, `(elif)`, `else`

```
>>> boringlist = [1,2,3,4]
>>> for number in boringlist:
...     if number > 2:
...             print(number)
...     elif number < 2:
...             print("you're too small")
...     else:
...             print("2 is a nice number")
...
you're too small
2 is a nice number
3
4
```

FLOW OF CONTROL: LOOPS

You can use ***loops*** to repeat a statement.

A ***for-loop*** is useful when you know how many times you want to repeat an action (e.g. for every item in a list)

```
>>> boringlist = [0,1,2,3]
>>> for number in boringlist:
...     print(number)
...
0
1
2
3
```

Pro tip: use range([number]) to create a sequence from 0 until [number] to loop [number] times

FLOW OF CONTROL: LOOPS

A ***while-loop*** is useful when you don’t know when you want to stop looping yet.

A while-loop statement checks a *condition* and loops until the condition is no longer satisfied.

```
>>> ans = ""
>>> while ans != "because!":
...     ans = raw_input("why?\n")
...
why?
because you're not old enough
why?
because it's way past your bedtime
why?
because!
>>>
```

EXERCISE

Write a function that

- takes as input a tictactoe board (represented as ‘boardList’ of three ‘rowLists’, each containing three strings, each representing a cell. For example an empty board:

```
[[“_”, “_”, “_”],
 [“_”, “_”, “_”],
 [“_”, “_”, “_”]]
```

- prints the contents of the different rows in the shape of a 3x3 board

(Cheating is encouraged)

SYNTAX SO FAR

Questions?

PROGRAMMING IN PYTHON

Stories

WRITING SCRIPTS

You can also structure your code conveniently in a **file**

Such a file is a **program** or **script** and can look as simple as this:

```
print("Hello World")
```

Save your script as “[a_descriptive_name].py”

Navigate in terminal to the location of your file

Run “python [a_descriptive_name].py”

WRITING SCRIPTS

Or this:

```
# this is a comment

'''
Comments can also
span multiple lines
'''

# print() is a very useful function
# mind the quotes
print("Hello World")

# you can also use "print [what you want to print]" without parentheses
print "Hello Sun"
```

use comments

IMPORTING LIBRARIES

A ***library*** is a package of code that extends the native functionality of Python.
Use libraries to:

- plot graphs
- use the Twitter API
- read and write .csv files
- ...

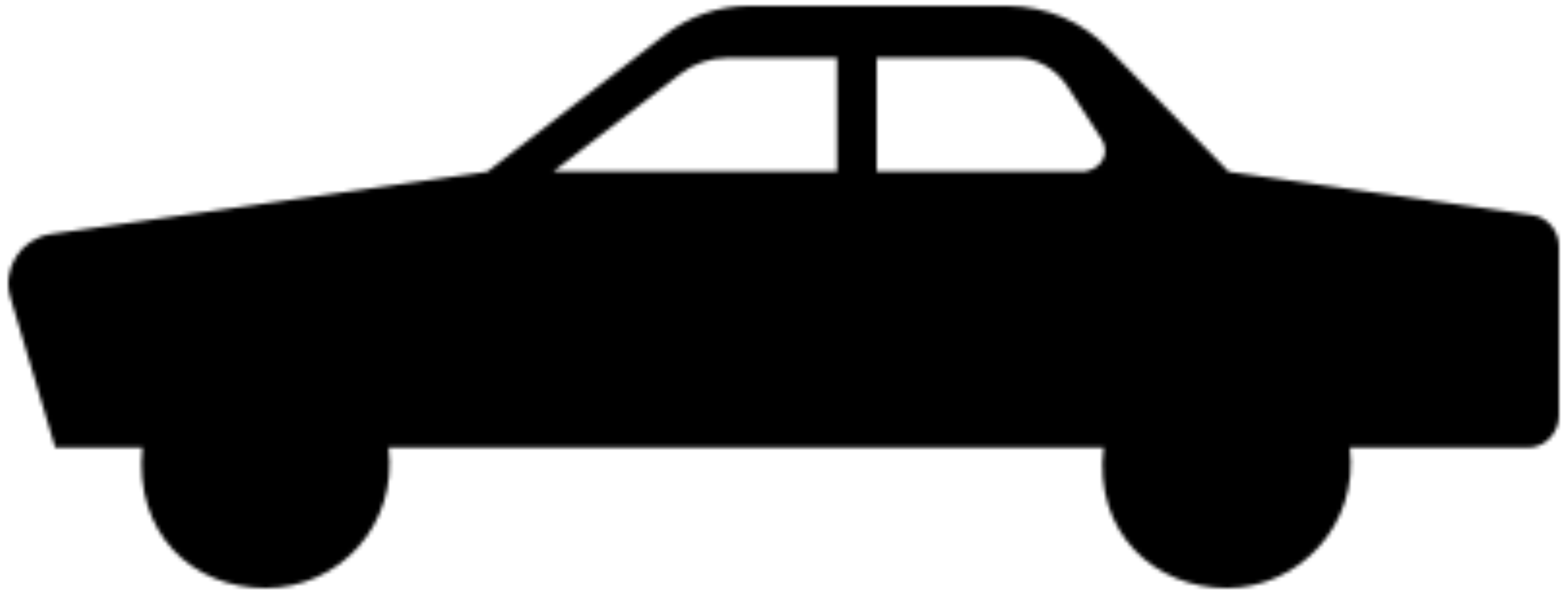
Importing a library is simple:

```
>>> import random
>>> random.randint(0,1)
1
>>> random.randint(0,1)
1
>>> random.randint(0,1)
0
```

STRUCTURED STORIES: CLASSES

Classes describe which common attributes (variables) and procedures (methods) something has

For example, a car...



http://commons.wikimedia.org/wiki/File:Car.svg

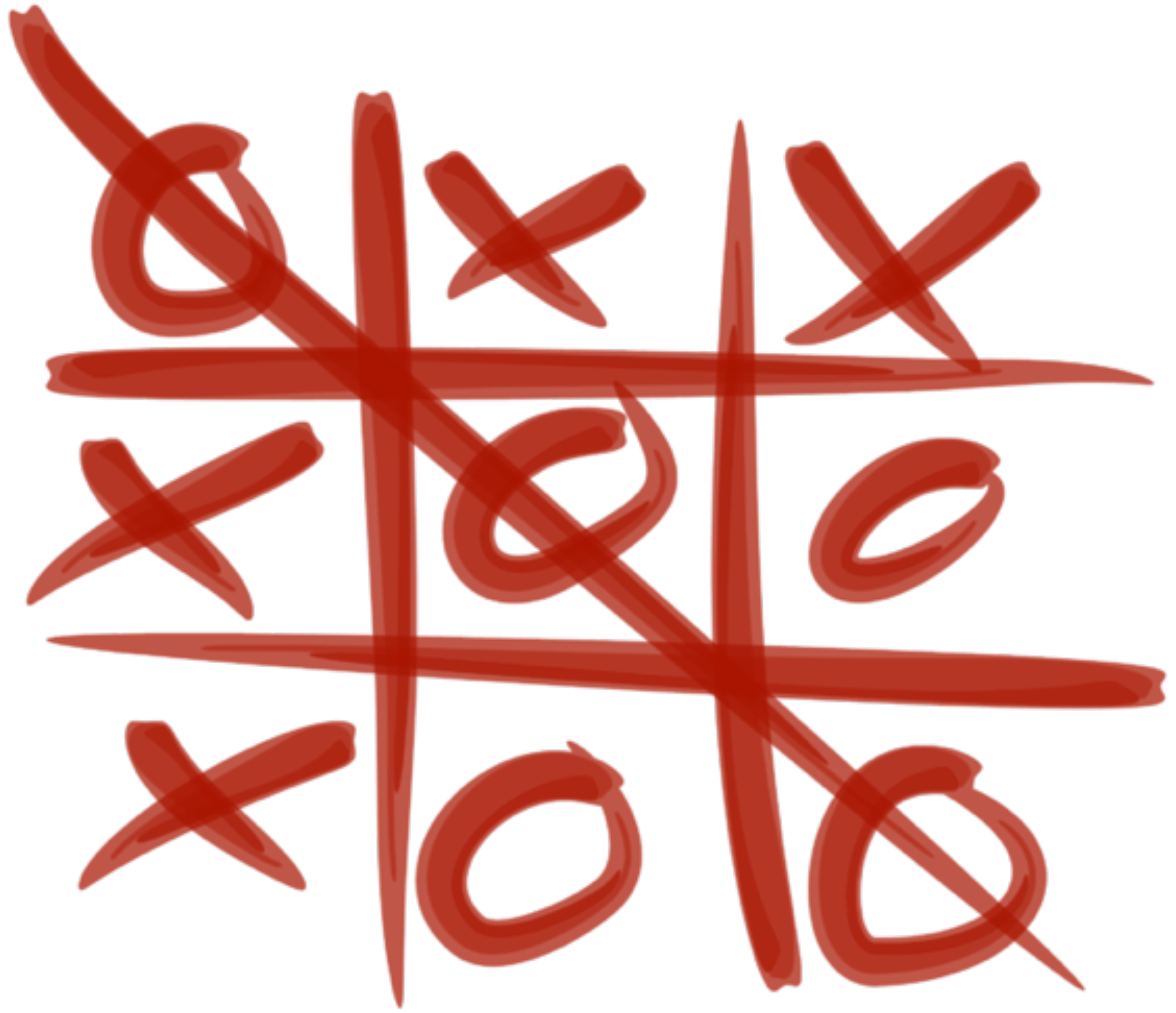
STRUCTURED STORIES: OBJECTS

Objects are specific instantiations of a *class.*

For example this dark green jaguar...

EXERCISE

Think of several classes that could be used to capture the dynamics of a tictactoe game.



Let’s have a look at tictactoe.py, which classes are used?

(Cheating is encouraged)

SUMMARY

USEFUL RESOURCES

www.google.com; “python” + your problem / question

www.python.org/doc/; official python documentation, useful to find which functions are available

http://docs.python.org/tutorial/; official tutorial if you want to explore more detail

www.stackoverflow.com; huge gamified forum with discussions on all sorts of programming questions, answers are ranked by community

http://www.codecademy.com/tracks/python; interactive exercises that teach you coding by doing, step by step