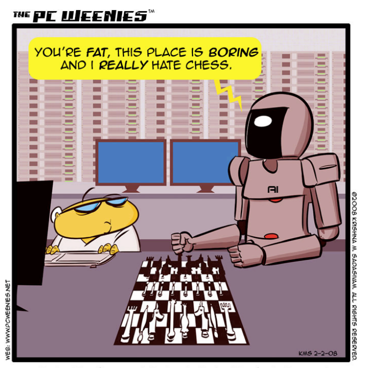


HOW YOU'LL KNOW WHEN YOU'VE TRULY SUCCEEDED IN THE FIELD OF A.I. RESEARCH.

http://www.bbspot.com/comics/PC-Weenies/2008/02/3248.php

CS311: Artificial Intelligence

# Intro to AI & Intro to Python

CS311
David Kauchak
Spring 2013

*Adapted from notes from*:
Sara Owsley Sood

## Who are you and why are you here?

Name/nickname

Major and year

What is AI? (or what do you think AI is? ☺)

Why are you taking this course?

What topics would you like to see covered?

## Course goals

Be able to answer the question "What is AI"?

Learn and apply basic AI techniques…

…to solve real-world (current) problems, and in the process…

…appreciate how HARD AI really is (and why)

## AI is a huge field

So, what is AI…

One definition:

"*Building programs that enable computers to do what humans can do.*"

For example:
read, walk around, drive, play games, solve problems, learn, have conversations…

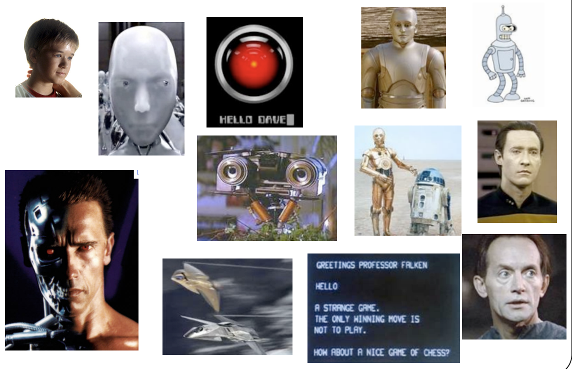
## How do we measure success?

"Building programs that enable computers to do what humans can do."

there are many interpretations of this goal…

| | human | rational |
|---|---|---|
| thinking | **Think like a human** Cognitive Modeling | **Think rationally** Logic-based Systems |
| vs. | | |
| acting | **Act like a human** Turing Test | **Act rationally** Rational Agents |

## How is AI viewed in popular media?



## What challenges are there?

AAAI
Association for the Advancement of Artificial Intelligence
MEDICAL CENTER
See the AI timeline and more at www.aaai.org/AILandscape
The AI Landscape

## What challenges are there?

Perception
- perceive the environment via sensors

Computer vision (perception via images/video)
- process visual information
- object identification, face recognition, motion tracking

Natural language processing and generation
- speech recognition, language understanding
- language translation, speech generation, summarization

## What challenges are there?

Knowledge representation
- encode known information
- water is wet, the sun is hot, Dave is a person, …

Learning
- learn from environment
- What type of feedback? (supervised vs. unsupervised vs. reinforcement vs …)

Reasoning/problem solving
- achieve goals, solve problems
- planning
- How do you make an omelet? I'm carrying an umbrella and it's raining… will I get wet?

Robotics
- How can computers interact with the physical world?

## What can we currently do?

Understand spoken language?
- speech recognition is really good, if:
  - restricted vocabulary
  - specific speaker with training
- Gotten quite good in the last few years and shows up in lots of places:
  - Mac has built-in dictation software
  - Siri is pretty good (though there's more than speech recognition going on there)
  - Google allows you to search via voice command
- What does the spoken language actually mean (language understanding)?
  - much harder problem!
  - many advances in NLP in small things, but still far away from a general solution

## What can we currently do?

Speak?
- Understandable, but you wouldn't confuse it for a person
- Can do accents, intonations, etc.
- Better with restricted vocabulary
- Loquendo
  - http://tts.loquendo.com/ttsdemo/default.asp
- Dealing with facial expression is challenging

Kismet (MIT)

## What can we currently do?

Drive a car?

- Freeway driving is relatively straightforward
- Off-road a bit harder
  - see DARPA grand challenges (2004, 2005)
- And urban driving is even trickier
  - See DARPA urban challenge (2007)
  - Google's autonomous vehicle

Hint: there's a connection here

## What can we currently do?

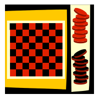
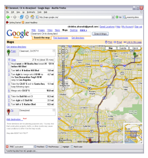
Identify emotion?

- This is hard!
- Some success in text
  - movie reviews
  - blogs
  - twitter
  - dealing with sarcasm is hard
- Some success with faces
  - strongly biased by training data
  - works best when exaggerated

## What can we currently do?

Reasoning?

- Success on small sub-problems

- General purpose reasoning is harder
  - Wolfram Alpha
  - OpenCyc

## What can we currently do?

Walk?

- Robots have had a variety of locomotion methods
- Walking with legs, is challenging
  - Differing terrains, stairs, running, ramps, etc.
  - Recently, a number of successes
    - Honda's Asimo
      - http://www.youtube.com/watch?v=W1czBcnX1Ww
    - Sony QRIO
      - http://www.youtube.com/watch?v=9vwZ5FQEUFg
    - Boston Dynamic's Big Dog
      - http://www.youtube.com/watch?v=W1czBcnX1Ww

## When will I have my robot helper?

## What can we currently do?

Vacuum? Clean the floor? Mow the lawn?

## What can we currently do?

Fold a pile of towels?

UC Berkeley towel folding robot:

http://www.youtube.com/watch?v=gy5g33S0Gzo

## Administrivia

go/cs311

- Office hours, schedule, assigned readings, problem sets
- Everything will be posted here

Read the "administrivia" page!

- ~4 programming assignments (in Python)
- ungraded written assignments
- quizzes
- 2 exams (dates are tentative)
- final project for the last month
  - teams of 2-3 people
  - research-like with write-up and presentation
- class participation
- readings

- Academic honesty and collaboration

## Python basics

```
>>> x = 5 #no variable types or declarations
>>> y = 10.0 #python has implicit typing
>>> type(y)  # gets the type of an expression
<type 'float'>
>>> type(x)
<type 'int'>
>>> x + y  #meaning that the type is implied by whatever
           #you do to the variable
15.0
>>> type(x + y)
<type 'float’>
>>> x = 'hi there'
>>> x + y
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int' objects
>>> x + str(y)
'hi there10'
```

## More python fun

```
>>> x[0] ##treat a string as a list
'h'
>>> x[0:4] ##substring of a string is a sublist - slices
'hi t’
>>> x[-1]
‘e’
>>> x[-3:]
‘ere’
>>> myL = [2]*5 ##can do powerful things
>>> myL   ##what do you think this will do?
[2, 2, 2, 2, 2]
>>> myL[2] = 0
>>> x = 1
>>> y = ‘hello’
>>> x, y = y, x
>>> x
‘hello’
>>> y
1
```

## Defining functions

look at *functions.py*

- **def** defines a new function
- **:** indicates the beginning of a new block of text
  - no curly braces (for better or worse)
  - a block is indicated by it’s indentation
  - DON’T MIX SPACES AND TABS
    - Whatever editor you use look up how to “Auto-expand” tabs into spaces!
- == for equality
- **True** and **False**

## External files

You can edit in your favorite text editor (for example Aquamacs or TextWrangler).

You could also use an IDE like IDLE or WingIDE if you’d like.

When you’re ready, you can import the commands in the file using **import**

```
>>> import functions  # don’t include the .py
>>> L = [1,2,3,4]
>>> functions.listSearch(L, 5)
False
>>> functions.listSearch(L, 1)
True
```

## External files

You can edit in your favorite text editor (for example Aquamacs or TextWrangler).

You could also use an IDE like IDLE or WingIDE if you'd like.

If you don't want to have to type <module>. before the commands:

```
>>> from functions import *  # don't include the .py
>>> L = [1,2,3,4]
>>> listSearch(L, 5)
False
>>> listSearch(L, 1)
True
```

## STDOUT and STDIN

look at *IO.py*

- **print** prints to the console (you can use it in functional form with () or without)
- **help(raw_input)**
  - takes an optional prompt which is printed to the user
  - returns a string
  - we typecast it to an int using **int()**
    - can get a typecast error if user doesn't enter an int
- notice that this is a script!
  - Python executes from the top of the file to the bottom
  - Functions must be declared before calling
  - The last line is the method call to the function

## Defining classes

look at **polygon** class in *classes.py*

- **class <classname>:** again, denotes a new block of code
- **self** should be the first argument to any class method
  - It allows you to use the '.' notation for calling methods
  - It gives you a handle to the data associated with the current object
  - When you call the method, you don't actually specify it
- **__init__** is similar to a constructor
  - **p = polygon(…)**
- **__str__** is similar to toString in Java and is called whenever an object is **print**ed
- we can define optional parameters using '=' in the parameter list
  - must be the last parameters in the list
  - if not specified, they get the default value
  - can specify using the name if desired

## Defining classes

look at **box** class in *classes.py*

- can define multiple classes in a file (and filename doesn't matter, remember these are just scripts. You could type these into the interpreter if you'd like)
- **isinstance** function is similar to **instanceof** in Java

## **`dir`** and **`help`**!

provide all of the methods and data members available to an object

```
help(listSearch)
dir("foo")
help("foo".split)
dir(str)
help(str.split)
dir(42)
dir([])
```

No memorizing! Just use dir & help…

(and online documentation ☺)

## Comments and docstrings

# denotes a comment (use them!)

""" denotes a multi-line string. When put at the top of a file or as the first line in a function definition these provide documentation via help (use them!)

**help**(classes)