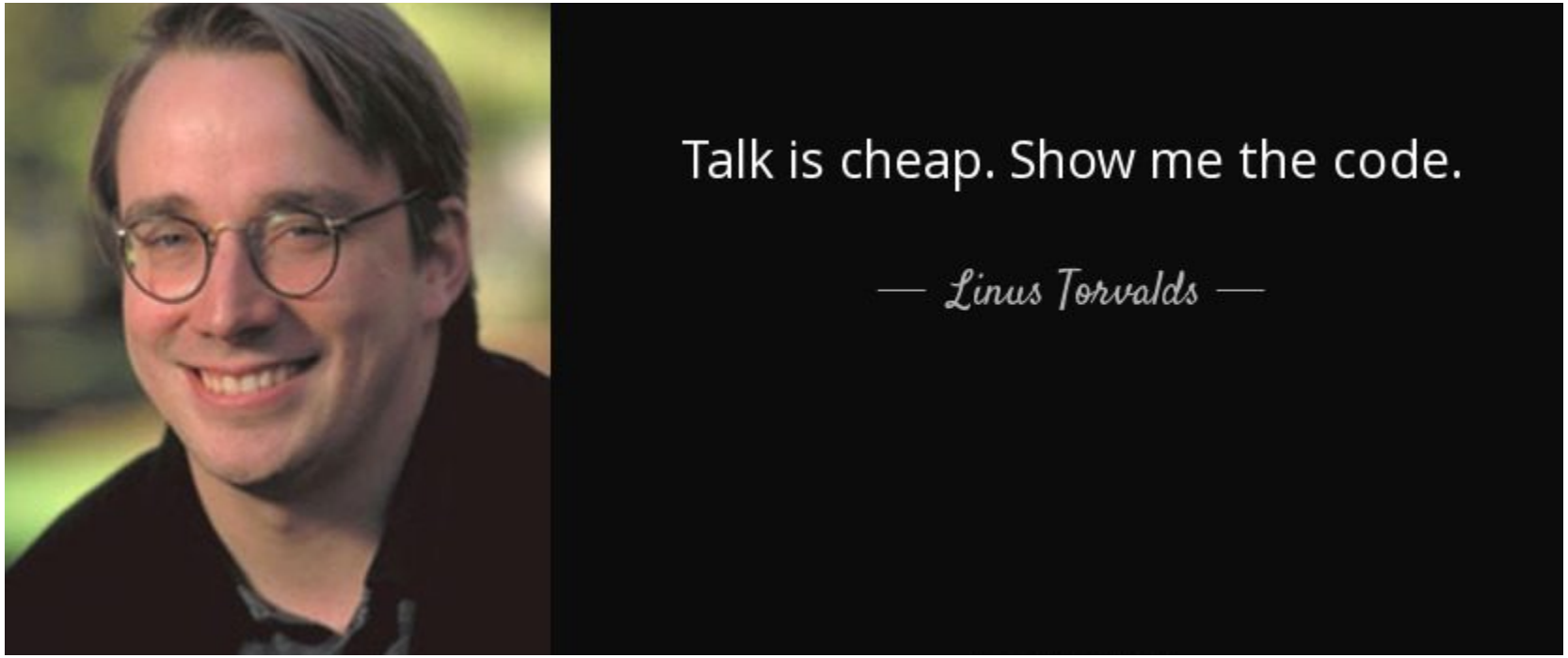
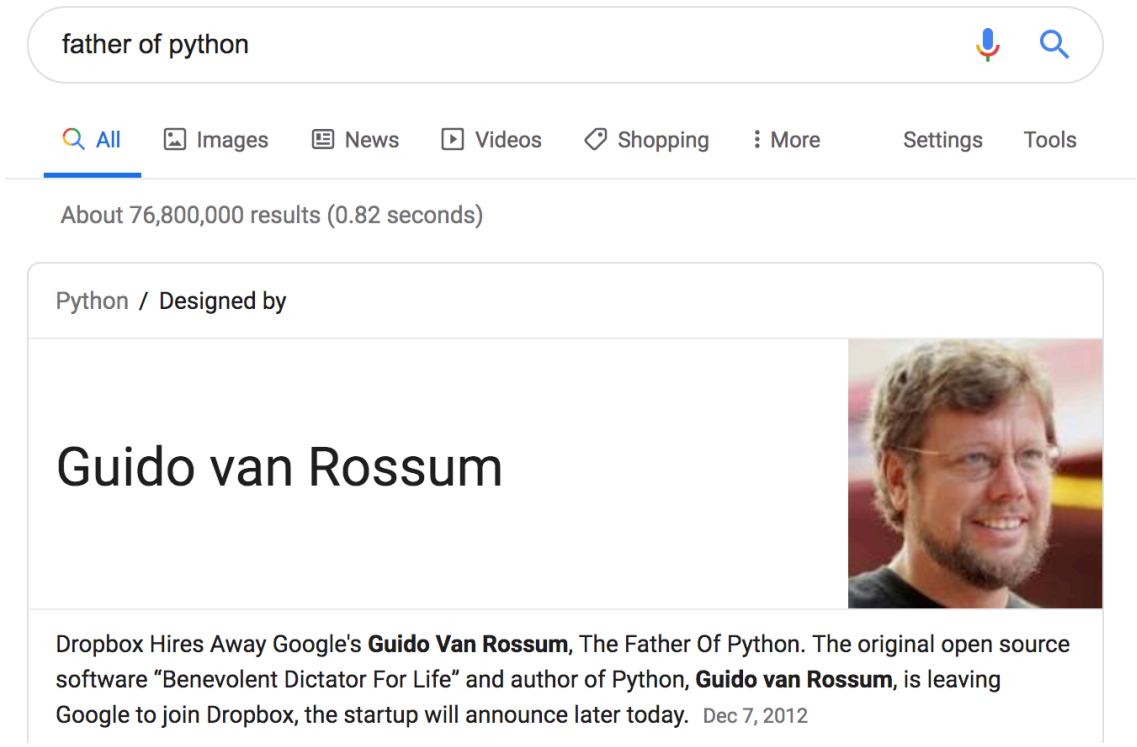
Python Programming



Python Programming



"Life is short (You need Python)" *-- Bruce Eckel*

Python Programming

```
>>> p = (4, 5)
>>> x, y = p
>>> x
4
>>> y
5
>>>

>>> data = [ 'ACME', 50, 91.1, (2012, 12, 21) ]
>>> name, shares, price, date = data
>>> name
```

Source: Beazley, David; Jones, Brian K. (2013). *Python Cookbook* (3rd ed.).

Python Programming

```
'ACME'
>>> date
(2012, 12, 21)

>>> name, shares, price, (year, mon, day) = data
>>> name
'ACME'
>>> year
2012
>>> mon
12
>>> day
21
>>>
```

Source: Beazley, David; Jones, Brian K. (2013). *Python Cookbook* (3rd ed.).

Python Programming

If there is a mismatch in the number of elements, you'll get an error. For example:

```
>>> p = (4, 5)
>>> x, y, z = p
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: need more than 2 values to unpack
>>>
```

Source: Beazley, David; Jones, Brian K. (2013). *Python Cookbook* (3rd ed.).

Python Programming

```
>>> s = 'Hello'
>>> a, b, c, d, e = s
>>> a
'H'
>>> b
'e'
>>> e
'o'
>>>
```

Source: Beazley, David; Jones, Brian K. (2013). *Python Cookbook* (3rd ed.).

Python Programming

```
def drop_first_last(grades):
    first, *middle, last = grades
    return avg(middle)
```

```
>>> record = ('Dave', 'dave@example.com', '773-555-1212', '847-555-1212')
>>> name, email, *phone_numbers = user_record
>>> name
'Dave'
>>> email
'dave@example.com'
>>> phone_numbers
['773-555-1212', '847-555-1212']
>>>
```

Source: Beazley, David; Jones, Brian K. (2013). *Python Cookbook* (3rd ed.).

Python Programming

```
records = [
      ('foo', 1, 2),
      ('bar', 'hello'),
      ('foo', 3, 4),
]
```

```
def do_foo(x, y):
    print('foo', x, y)

def do_bar(s):
    print('bar', s)

for tag, *args in records:
    if tag == 'foo':
        do_foo(*args)
    elif tag == 'bar':
        do_bar(*args)
```

Source: Beazley, David; Jones, Brian K. (2013). *Python Cookbook* (3rd ed.).

Python Programming

```
>>> line = 'nobody:*:-2:-2:Unprivileged User:/var/empty:/usr/bin/false'
>>> uname, *fields, homedir, sh = line.split(':')
>>> uname
'nobody'
>>> homedir
'/var/empty'
>>> sh
'/usr/bin/false'
>>>
```

Source: Beazley, David; Jones, Brian K. (2013). *Python Cookbook* (3rd ed.).

Python Programming

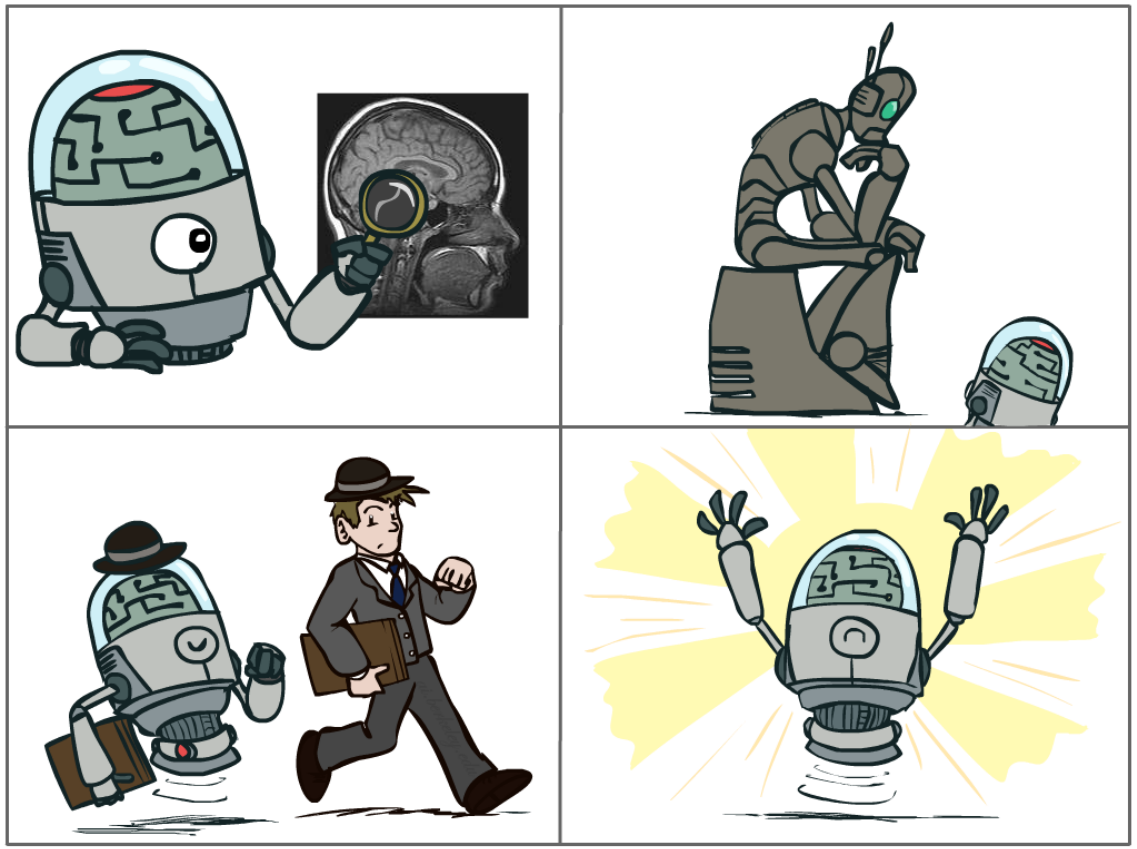
```
>>> items = [1, 10, 7, 4, 5, 9]
>>> head, *tail = items
>>> head
1
>>> tail
[10, 7, 4, 5, 9]
>>> def sum(items):
...     head, *tail = items
...     return head + sum(tail) if tail else head
...
>>> sum(items)
36
>>>
```

Source: Beazley, David; Jones, Brian K. (2013). *Python Cookbook* (3rd ed.).

What is AI?

The science of making machines that:

Think like people



Think rationally

Act like people

Act rationally

Fundamental question for this lecture
(and really this whole AI field!):

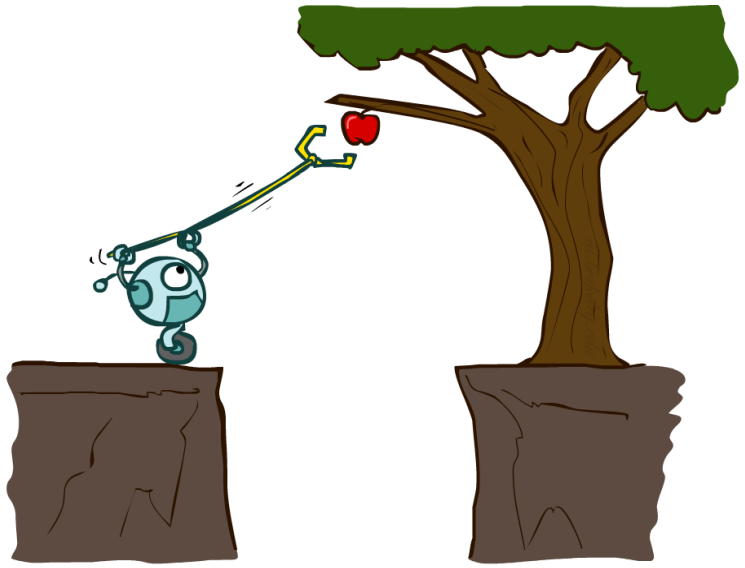
How do you turn a real-world problem into an AI solution?

AI – Agents and Environments

Much (though not all!) of AI is concerned with **agents** operating in **environments.**

Agent – an entity that *perceives* and *acts*

Environment – the problem setting



Fleshing it out

Performance – measuring desired outcomes

Environment – what populates the task's world?

Actuators – what can the agent act with?

Sensors – how can the agent perceive the world?

PEAS in a taxi

Automated taxi driver

Performance – Safe, fast, legal, comfortable trip, maximize profits

Environment – Roads, other traffic, pedestrians, customers

Actuators – Steering, accelerator, brake, signals, horn, display

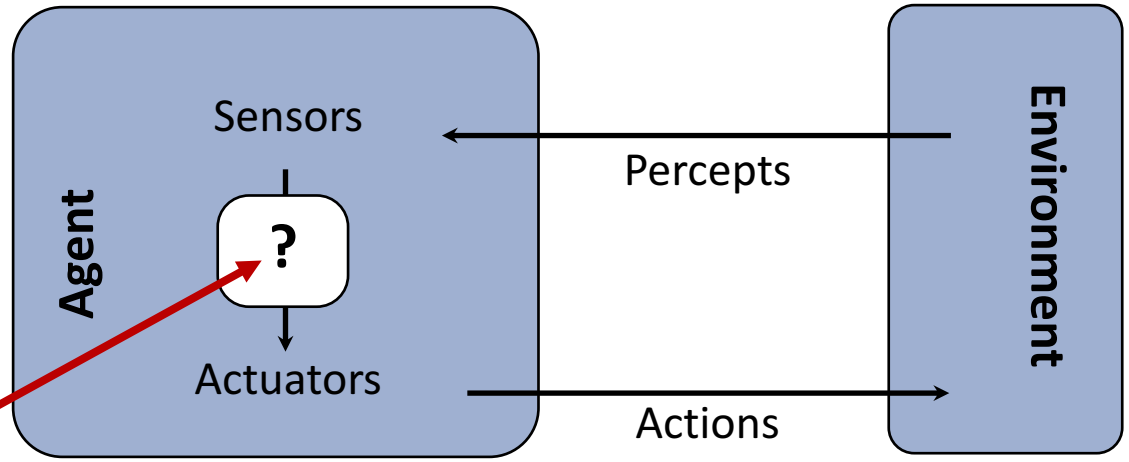
Sensors - Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, microphone/keyboard

What makes an Agent?

Agent – an entity that perceives its environment through sensors, and acts on it with actuators.

Percepts are constrained by
Sensors + Environment

Actions are constrained by
Actuators + Environment



Agent Function – how does it choose the action?

What makes one rational?

Actually pretty simple:

A **rational agent** always acts to **maximize its expected performance measure**, given current state/percept

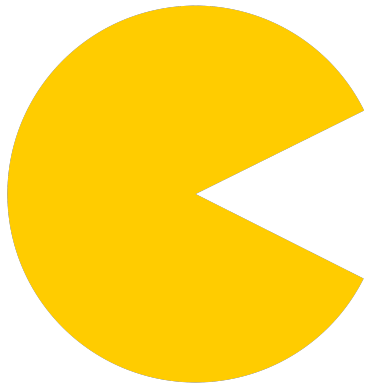
Our sample agents

Pacman

Percepts – squares around Pacman

Actions – move U/D/L/R

Environment – map with walls, dots, and ghosts



Spam detector

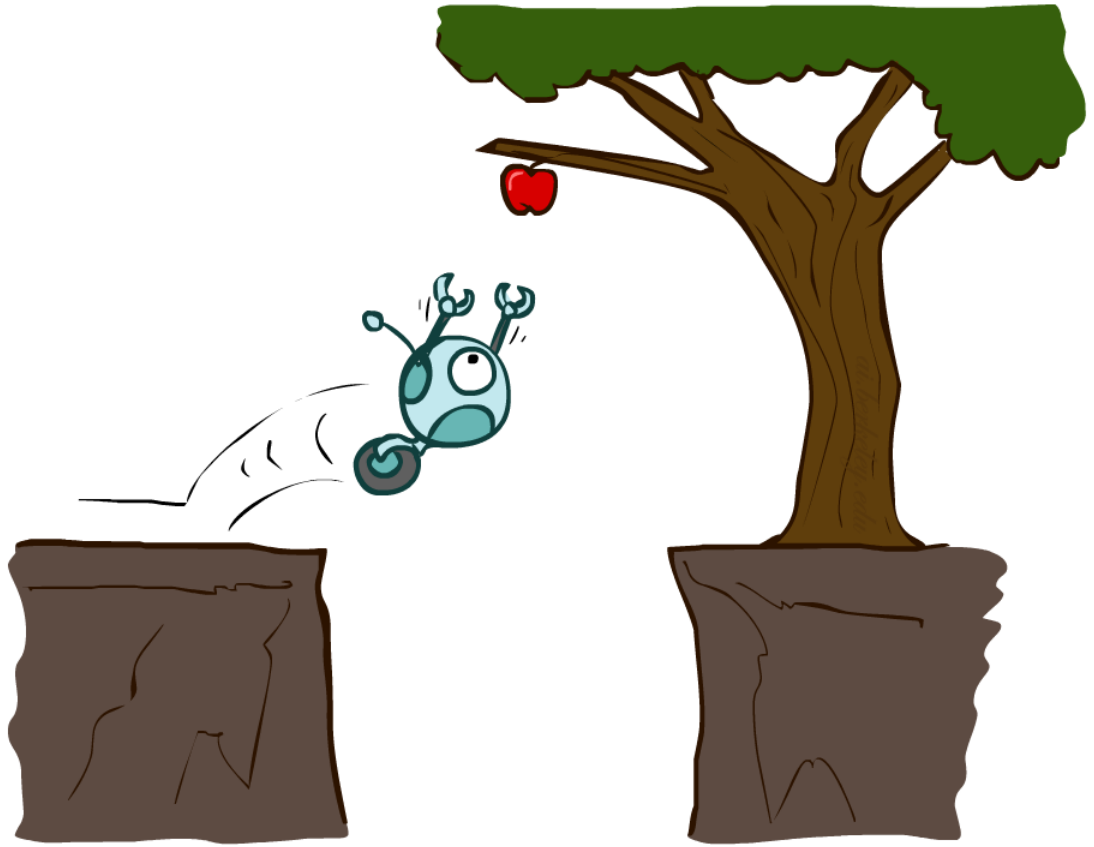
Percepts – sender, subject line, body of current email

Actions – mark Spam/Not Spam

Environment – your email inbox

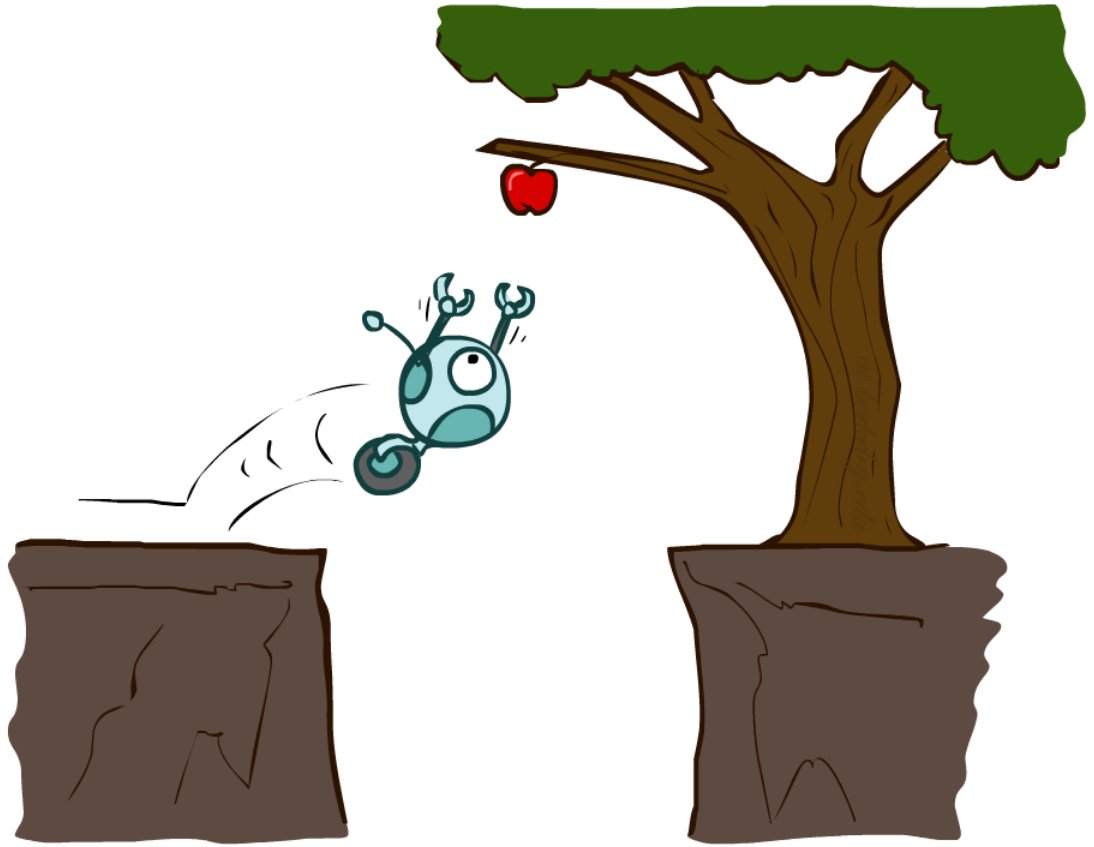
Reflex Agents

- Reflex agents:
  - Choose action based on current percept (and maybe memory)
  - May have memory or a model of the world's current state
  - Do not consider the future consequences of their actions
  - Consider how the world IS



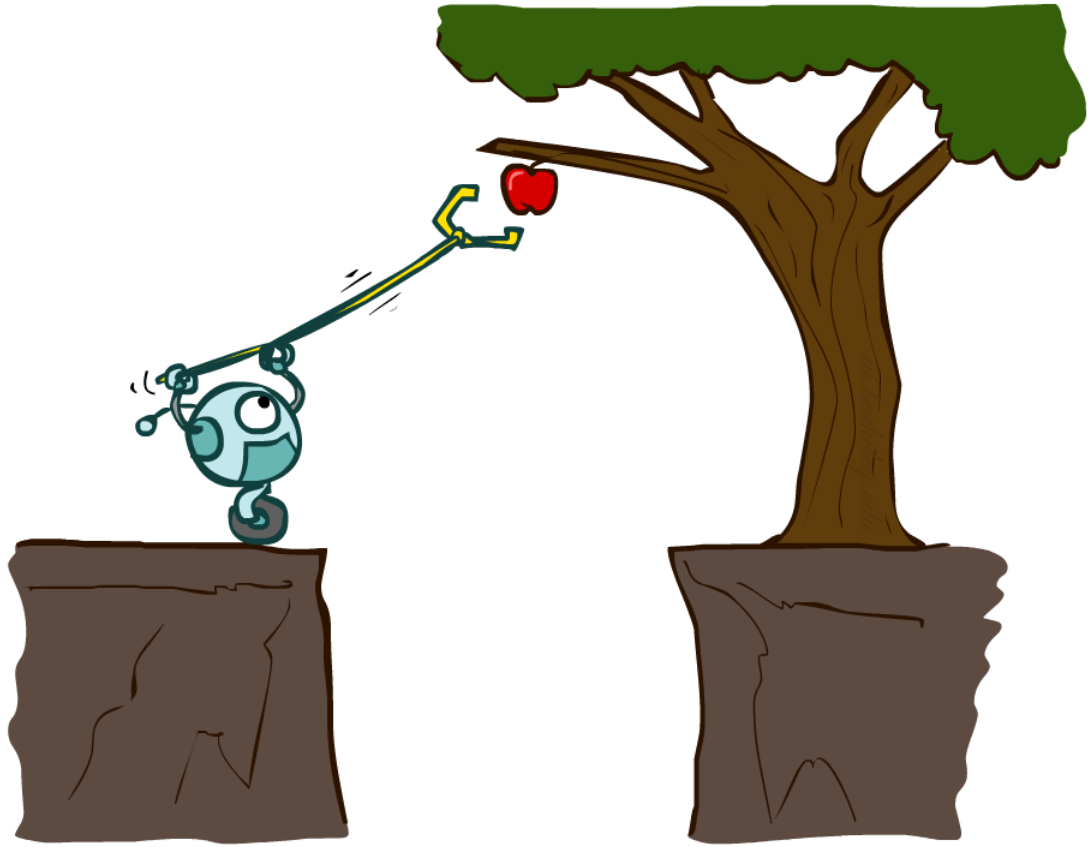
Reflex Agents

- Reflex agents:
  - Choose action based on current percept (and maybe memory)
  - May have memory or a model of the world's current state
  - Do not consider the future consequences of their actions
  - Consider how the world IS
- Can a reflex agent be rational?



Planning Agents

- Planning agents:
  - Ask “what if”
  - Decisions based on (hypothesized) consequences of actions
  - Must have a model of how the world evolves in response to actions
  - Must formulate a goal (test)
  - Consider how the world WOULD BE



Goal-based Agents

Chooses action (sequence) to get from current state to some goal

Pacman

Percepts – squares around Pacman

Actions – move U/D/L/R

Environment – map with walls, dots, and ghosts

Goal:

Spam detector

Percepts – sender, subject line, body of current email

Actions – mark Spam/Not Spam

Environment – your email inbox

Goal:

???

Utility-based Agents

Chooses action (sequence) to get from current state to some goal
with maximum utility along the way

Pacman

Percepts – squares around Pacman

Actions – move U/D/L/R

Environment – map with walls, dots, and ghosts

Goal:

…in as short a path as possible!

Spam detector

Percepts – sender, subject line, body of current email

Actions – mark Spam/Not Spam

Environment – your email inbox

Goal:

Summary

Reflex agents

Act on current state (and maybe past)

Simple – current p

Model – current p

of rest of

Goal-based agents

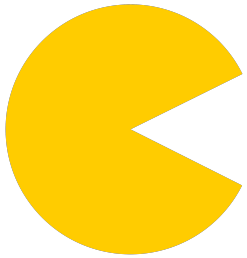
From current state to desired future

Goal only – find *any* action(s) to

ch the goal

st action(s) to

reach the goal



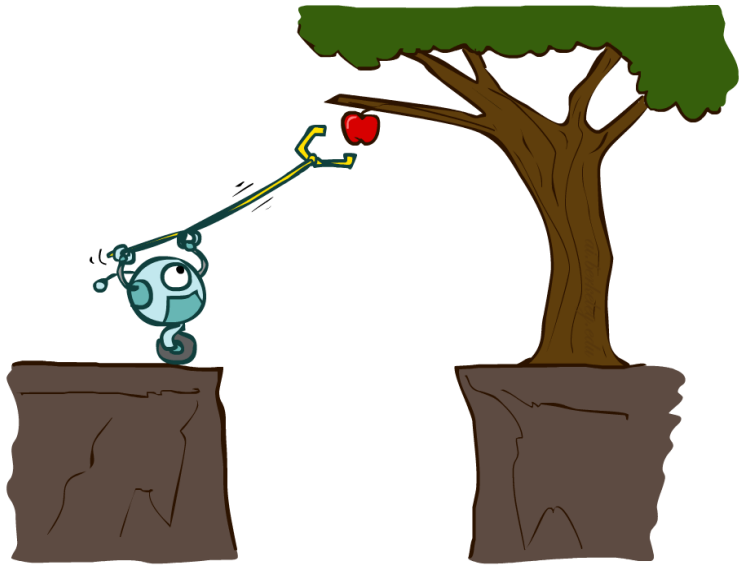
Can also have a **Learning Agent** – we'll talk about these later in the course!

AI – Agents and Environments

Much (though not all!) of AI is concerned with **agents** operating in **environments.**

Agent – an entity that *perceives* and *acts*

Environment – the problem setting



Kinds of task environments

6 common properties to distinguish tasks (not exhaustive)

- **Fully observable** vs **Partially observable**
- **Single agent** vs **Multiagent**
- **Deterministic** vs **Stochastic**
- **Episodic** vs **Sequential**
- **Static** vs **Dynamic**
- **Discrete** vs **Continuous**

Fully observable vs partially observable

Fully observable – agent is able to sense everything in the environment

ACROSS

1 See 24-Across

6 They radiate outward from an earthquake's epicenter

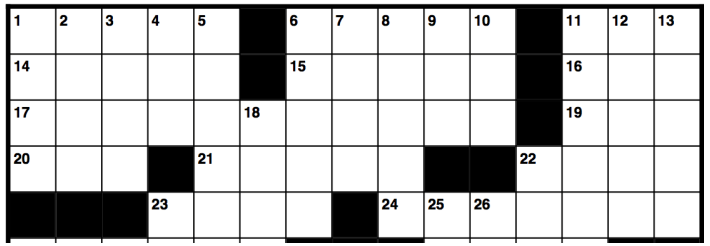
11 The "F" of "T.G.I.F.!": Abbr.

45 ___ fire under (urged to take action): 2 wds.

47 Daniel Defoe's "Robinson ___"

49 Vibrations caused by earthquakes

52 Low in fat

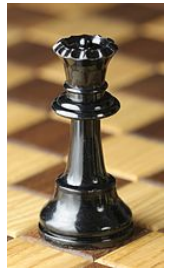


Partially observable – noisy, inaccurate, or incomplete sensors

Single agent vs Multiagent

Single agent – self-explanatory

Multiagent – task involves more than one agent, each with its own performance measure



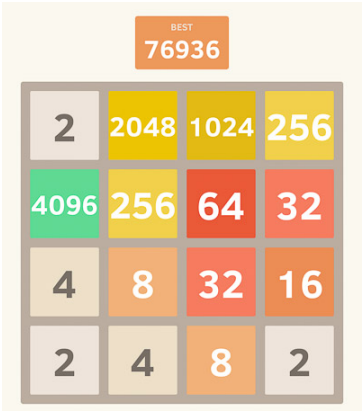
May be **competitive** (measures are opposed)

or **cooperative** (measures align)

Deterministic vs Stochastic

Deterministic – next state of the world is fully determined by current state + agent action

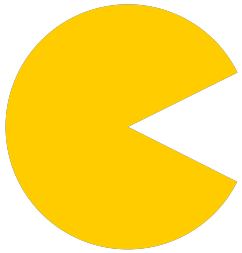
Stochastic – it's not deterministic



Episodic vs Sequential

Episodic – Each step/decision is independent of the previous ones

Sequential – Each step/decision affects later ones



Static vs Dynamic

Static – world doesn't change while agent is choosing an action

Dynamic – decision time matters!

Discrete vs Continuous

Discrete – possible states/actions are distinct; world changes discretely

Continuous – states/actions take on continuous values

These help determine how to approach problems

Static -> can focus on getting really high accuracy/utility

Dynamic -> trade some utility for higher efficiency (speed!)

Episodic -> reflex agent with a great model

Sequential -> need a goal-oriented agent

Stochastic -> need robustness to uncertainty/failure (robots!)

Deterministic -> can focus on efficiency and exactness (Internet crawler)

Next up

Defining search problems – how to choose the right action sequence?

Uninformed search approaches – simple reflex agents for searching