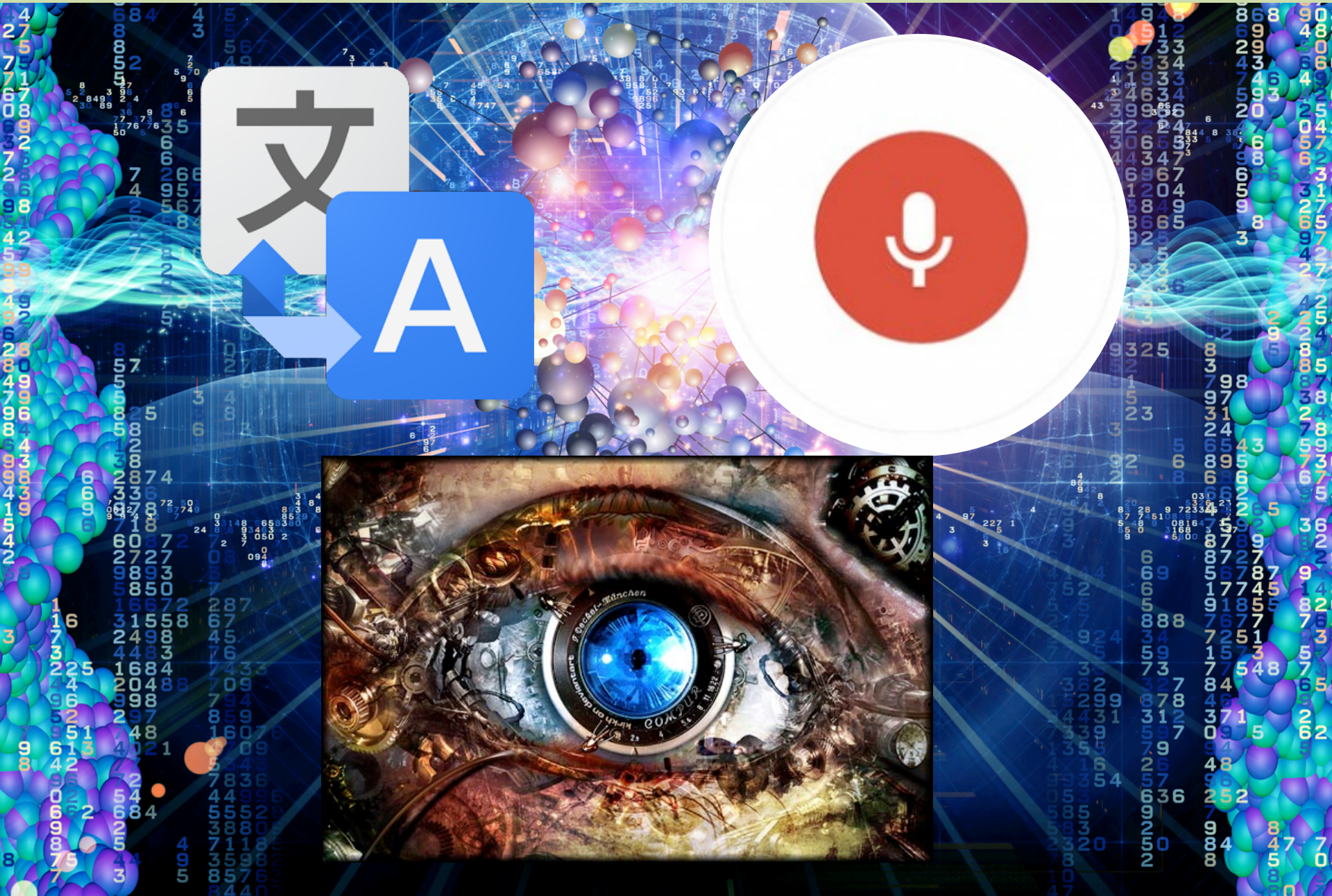
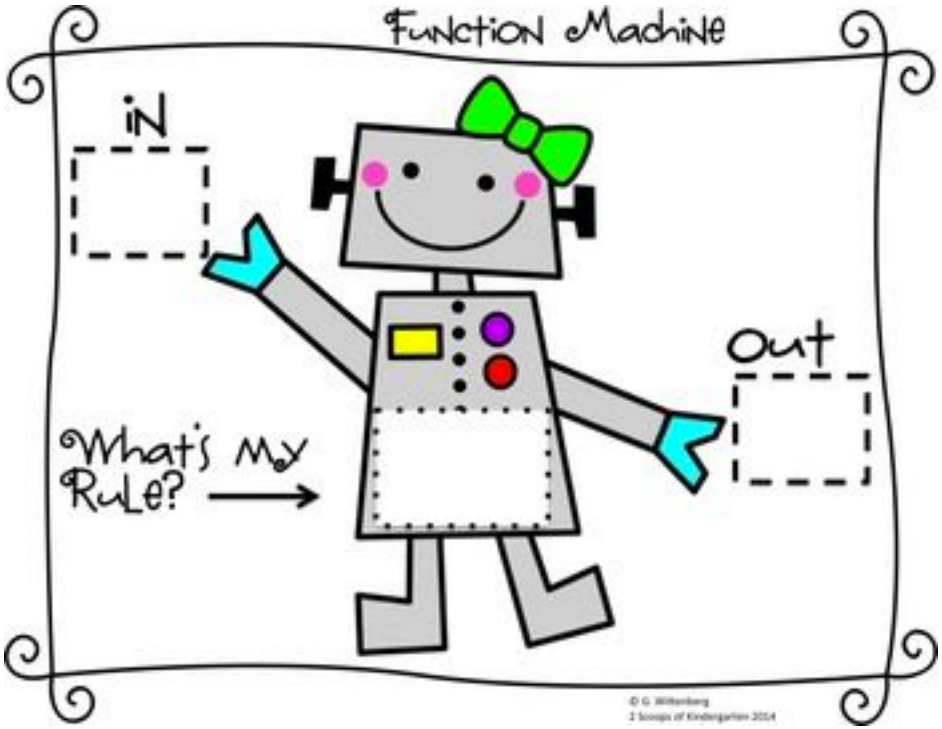
What's Going On With AI?



Functions

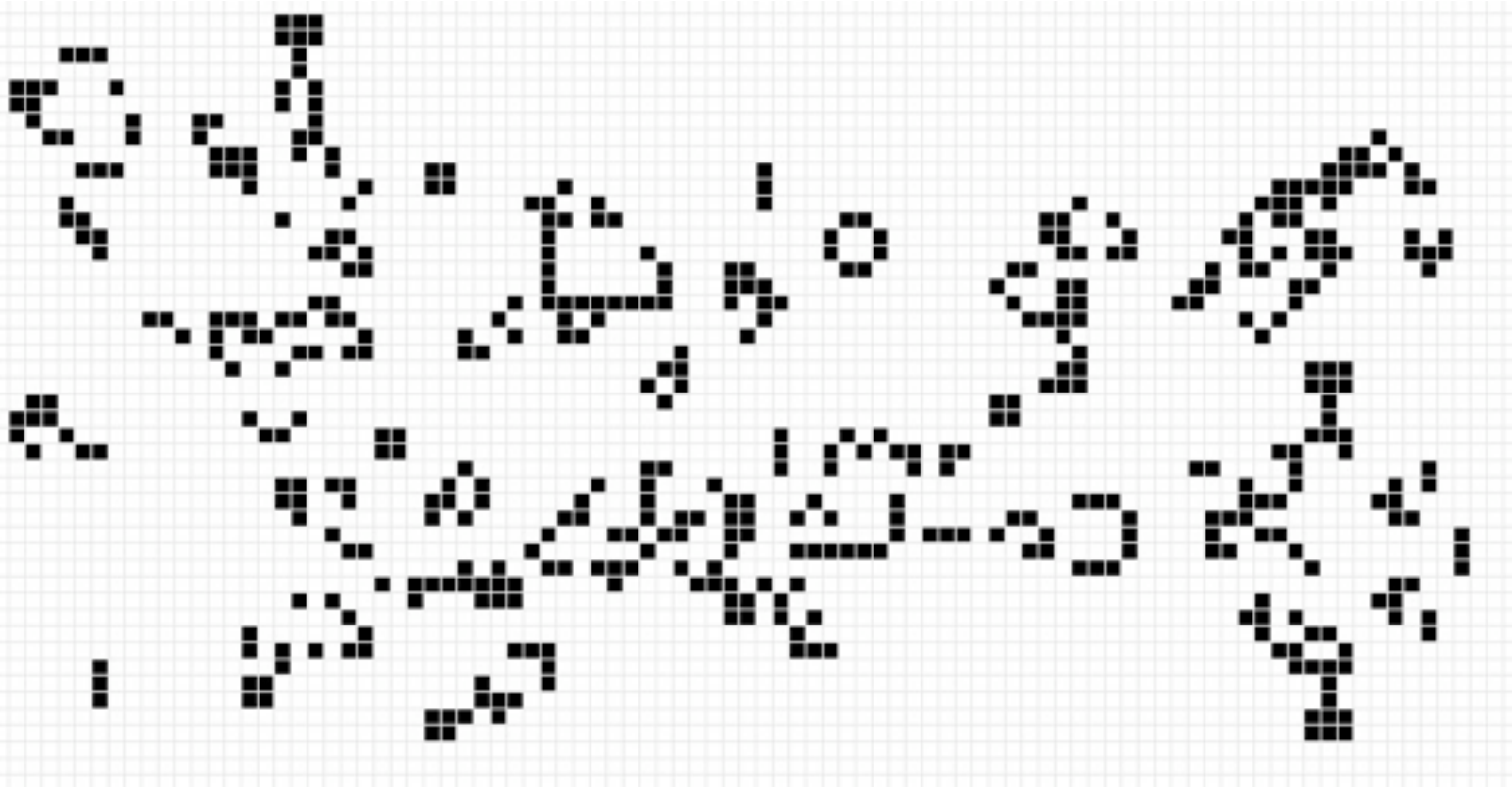


CS 106B
Lecture 1
Jan 4, 2015

Announcements

- You can start Life now. Due Jan 15th
- I would recommend getting QT creator installed on your computer.
- Monday extra office hours with Megan! Gates 104 from 10am to 11am.
- Chris office hours are moved to Friday 2:30pm to 3:30pm

Life



Life Simulated

Recap from Monday

Class Website

CS106B Lectures Handouts Assignments Overview

CS106B: Programming Abstractions

Winter 2016

Monday, Wednesday, Friday 11:30am to 12:30pm in Dinklespiel Auditorium

RESOURCES

- Course Overview
- Submitter
- Piazza
- QT Creator
- Stanford C++ Lib
- Past Quarters

ASSIGNMENTS

Assignment 1: Life

EXAMS

Midterm
Tuesday, Feb 9th

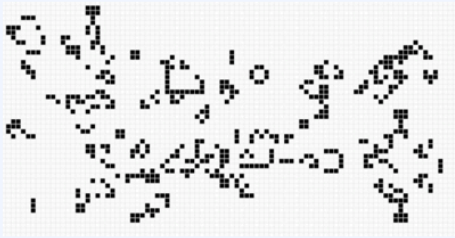
Final
Monday, March 14th

STAFF

NEWSFEED

Life Assignment Out
a minute ago

Assignment 1, Life goes out today in class. In this program you will implement Conways Game of Life, an elegant cartoon model of cell life that has inspired an entire field of thinking on complex structures that can arise from simple rules.



One time office hour change.
a day ago

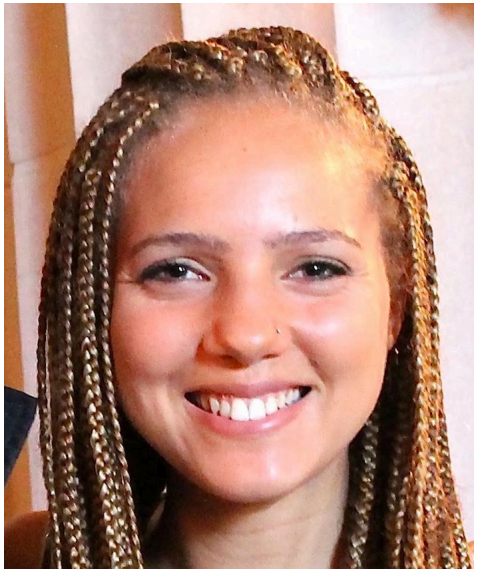
Chris' Wednesday office hours for this week have been changed from Wednesday (5th of Jan) to Friday (7th of Jan) from 2:30pm to 3:30pm. This is a one time change. Things go back to normal next week. Come with early question on Life :).

cs106b.stanford.edu

Course Staff

Instructor: Chris Piech

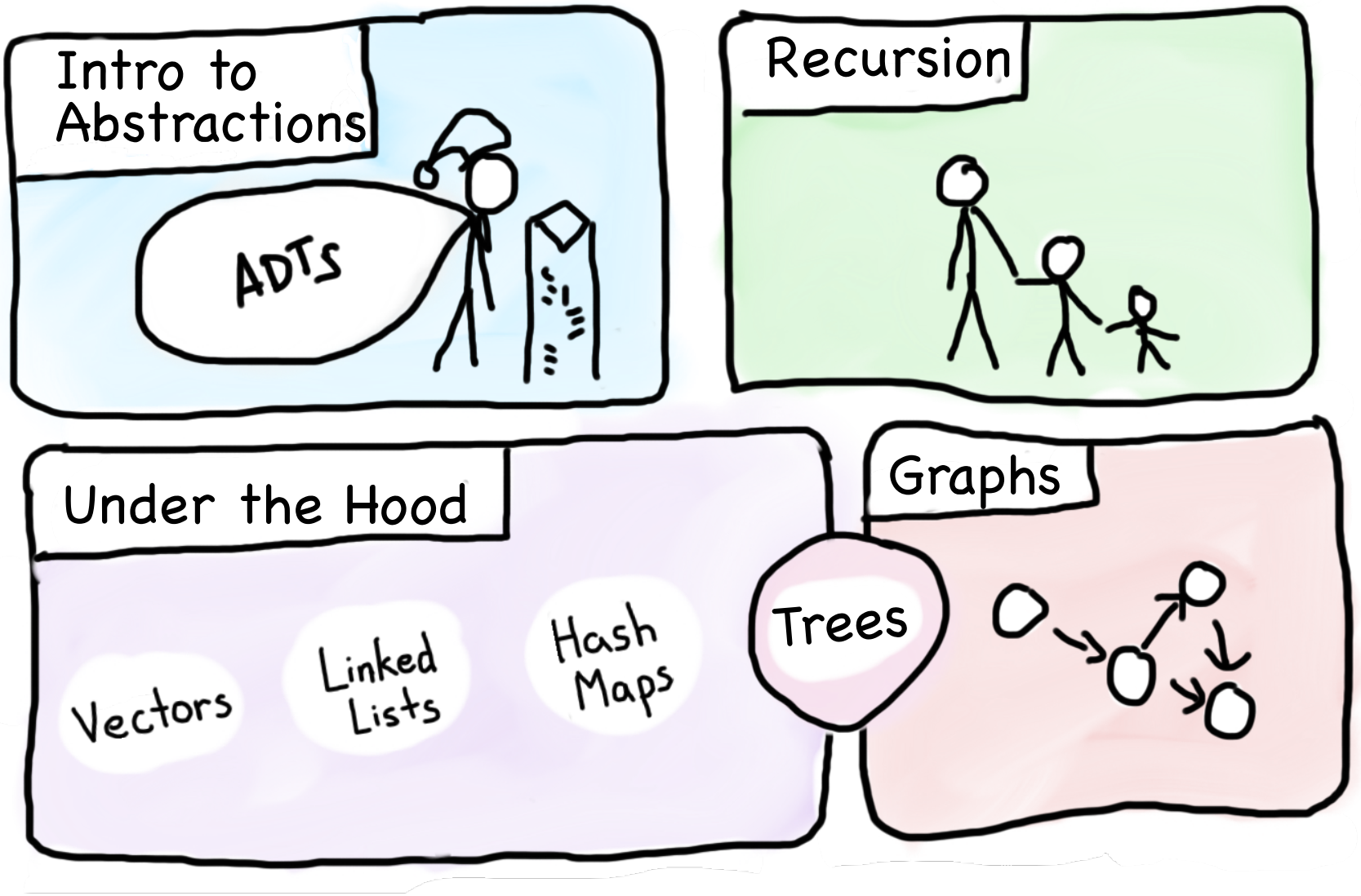
Office hours:
Mon 1:30 to 2:30pm
Wed 2:30 to 3:30pm



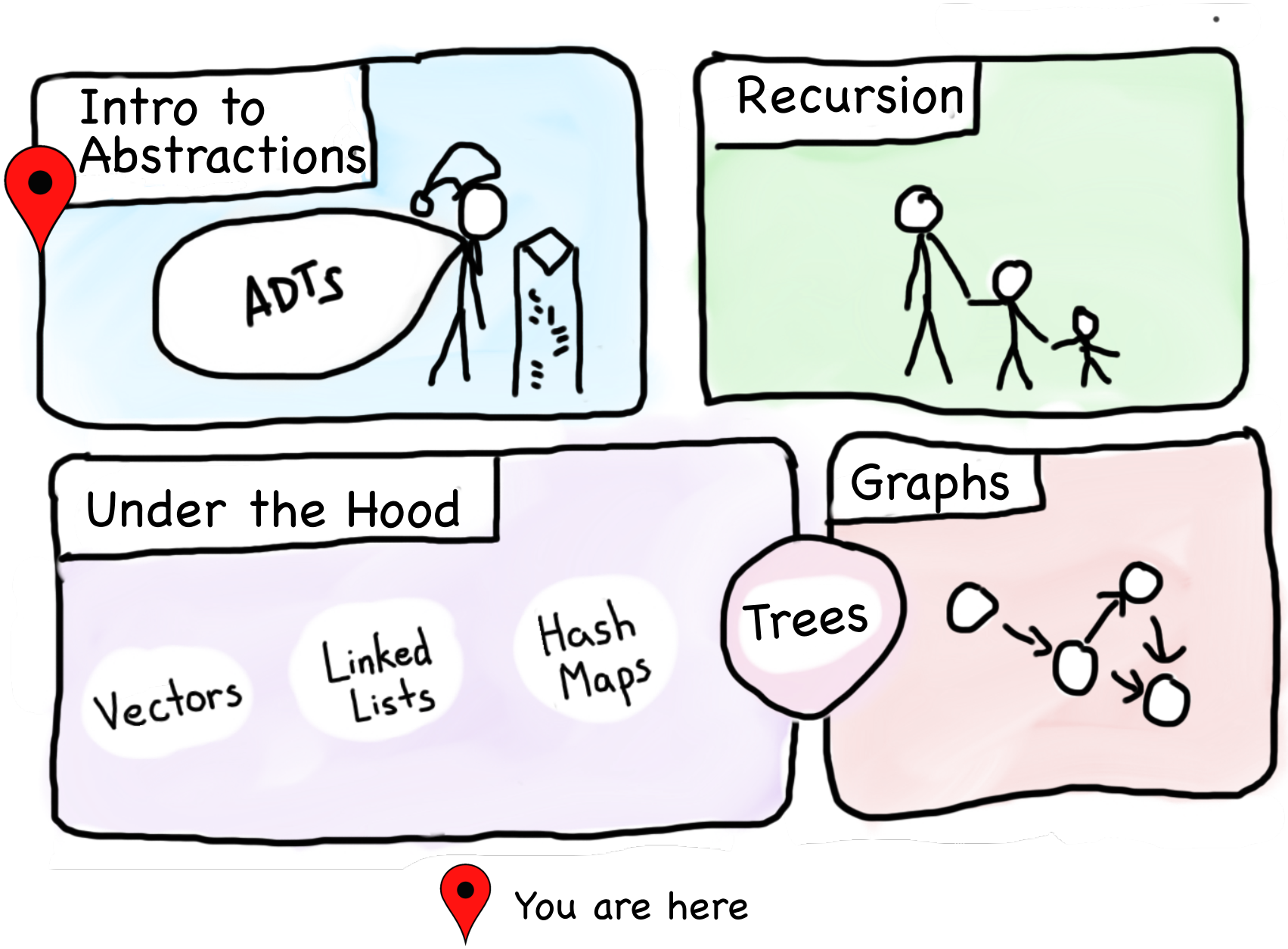
Head TA: Megan Faulk

Office hours:
Tue, Th 9:00 to 10:30am

Course Syllabus



Course Syllabus



Today's Goals

1. Intro to C++
2. Functions in C++

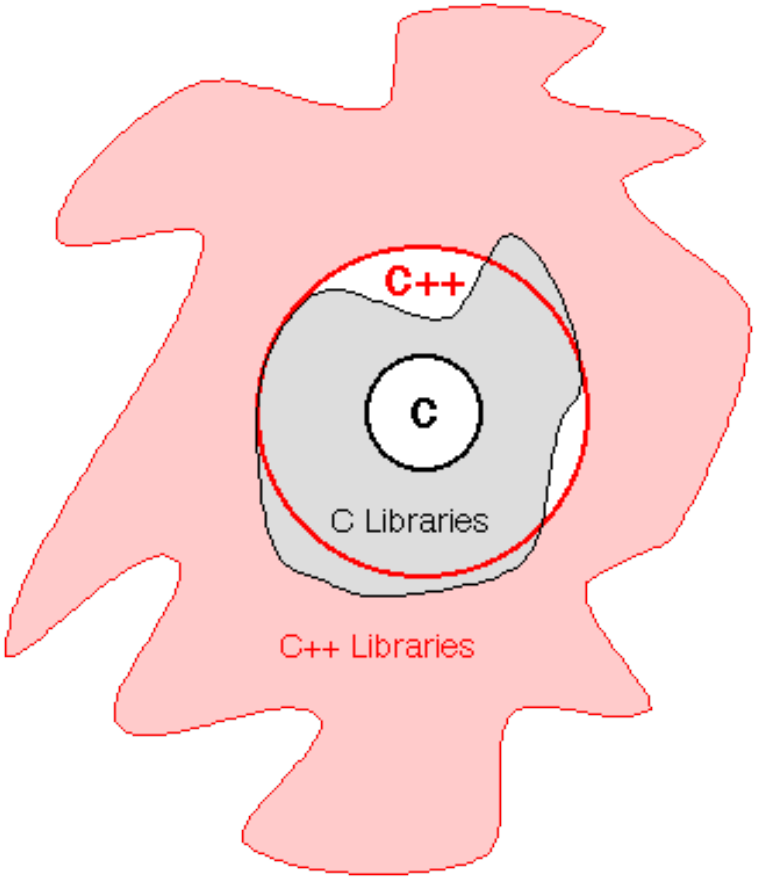
Part 1: C++

Similar to Java

C++ from the Java Programmer's Perspective

(But it's ok if you don't know java!)

C++



Hello World

```
/*
 * The canonical hello world program in C++
 */
```

```
#include <iostream>  // A true classic.
#include "console.h" // A Stanford Library!
using namespace std; // The best namespace.

int main() {
   cout << "hello, world!" << endl;
   return 0;
}
```

Hello World

```
/*
 * The canonical hello world program in C++
 */

#include <iostream>  // A true classic.
#include "console.h" // A Stanford Library!
using namespace std; // The best namespace.

int main() {
   cout << "hello, world!" << endl;
   return 0;
}
```

Hello World

```
/*
 * The canonical hello world program in C++
 */

#include <iostream>  // A true classic.
#include "console.h" // A Stanford Library!
using namespace std; // The best namespace.

int main() {
   cout << "hello, world!" << endl;
   return 0;
}
```

Hello World

```
/*
 * The canonical hello world program in C++
 */

#include <iostream>  // A true classic.
#include "console.h" // A Stanford Library!
using namespace std; // The best namespace.

int main() {
    cout << "hello, world!" << endl;
    return 0;
}
```

Most Primitives are the Same

Java	C++
`int`	`int`
`double`	`double`
`float`	`float`
`boolean`	`bool`
`char`	`char`

Most Operators are the Same

Operator	Meaning
==	Equals comparison
||	Logical or
<=	Less than comparison
=	assignment

Most Control flow is the Same

Control	Meaning
`while(cond) { body }`	Repeat *body* until *cond* is false.
`for(int i = 0; i < N; i++) { body }`	Repeat body *N* times.
`if(cond) { body A } else { body B }`	Either run *bodyA* or *bodyB* depending on cond.

Wahoo!

Is it all the same?

No.

Strings are a little different

```
#include <string>
...
string s = "hello";
```

A string is a (possibly empty) sequence of characters. Strings in C++ are conceptually similar to strings in Java.

Minor differences:

- Different names for similar methods.
- Different behavior for similar methods

Major differences

- Strings are mutable (can be changed) in C++
- There are two types of strings in C++

String Operators

Like in Java, you can concatenate strings using + or +=:

```
string s1 = "Chr";
s1 += "is";                 // "Chris”
```

Unlike in Java, you can compare strings using relational operators:

```
string s2 = "Megan";           // == !=  < <=  > >=
if (s2 > s1 && s2 != "Keith") {  // true
    ...
}
```

Unlike in Java, strings are mutable and can be changed:

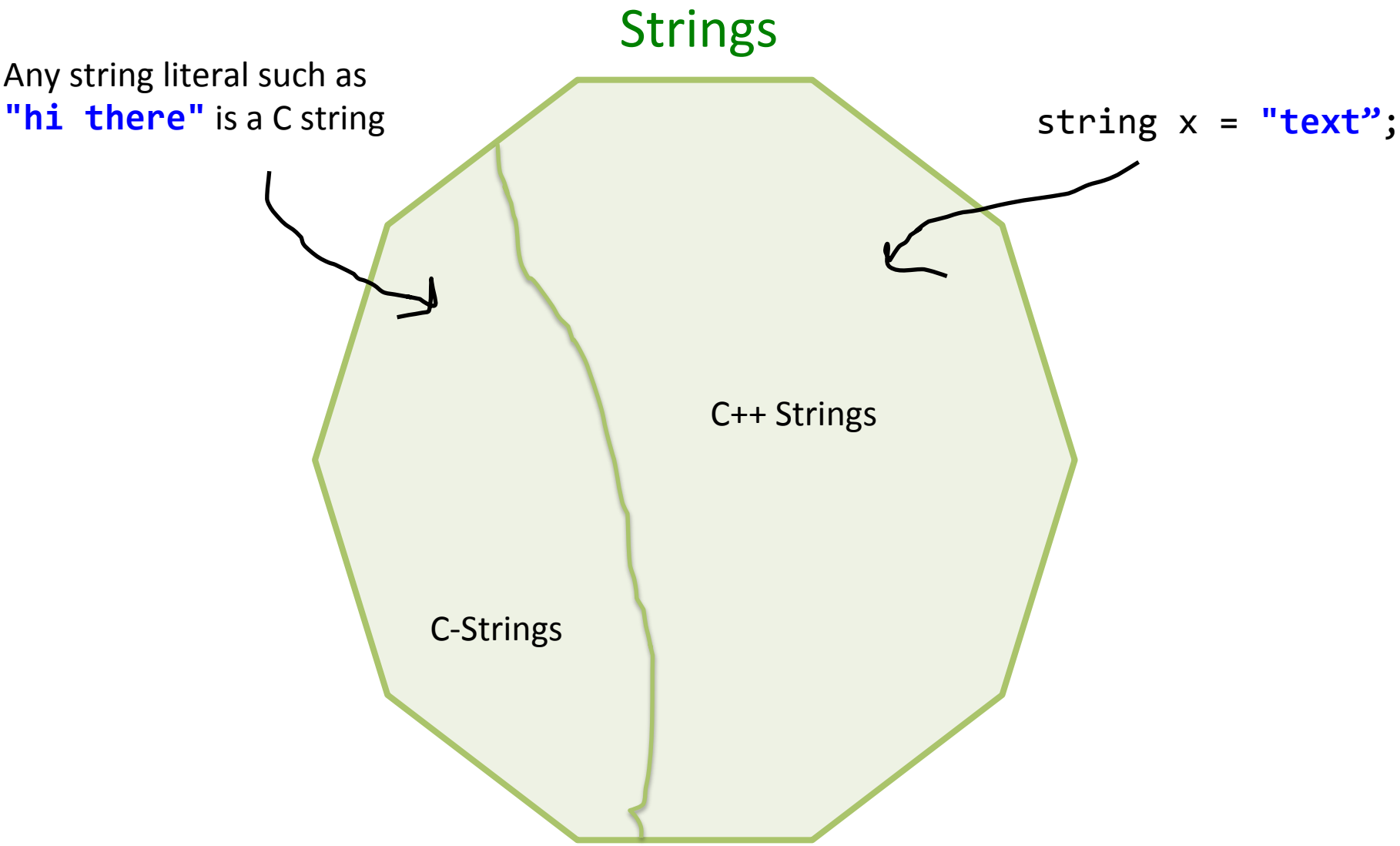
```
s1.append(" Piech")      // s1 is ”Chris Piech"
s1.erase(3, 2);          // s1 is ”Chr Piech"
s1[6] = 'o';             // s1 is “Chr Piech"
```

String Functions

Member function name	Description
`s.append(str)`	add text to the end of a string
`s.compare(str)`	return -1, 0, or 1 depending on relative ordering
`s.erase(index, length)`	delete text from a string starting at given index
`s.find(str)` `s.rfind(str)`	first or last index where the start of ***str*** appears in this string (returns `string::npos` if not found)
`s.insert(index, str)`	add text into a string at a given index
`s.length()` or `s.size()`	number of characters in this string
`s.replace(index, len, str)`	replaces *len* chars at given index with new text
`s.substr(start, length)` or `s.substr(start)`	the next *length* characters beginning at *start* (inclusive); if *length* omitted, grabs till end of string

```
string name = "Donald Knuth";
if (name.find("Knu") != 0) {
    name.erase(7, 5);              // "Donald"
}
```

Two Kinds of Strings



Aside: Common Bugs

C-String Bugs

```
string s = "hi" + "there”;
```

Does not compile; C strings can't be concatenated with +.

```
string s = string("hi") + "there";
string s = "hi";      // "hi" is auto-converted to string
s += "there”;
```

These both compile and work properly.

```
int n = (int) "42”;
```

Bug; sets n to the memory address of the C string "42" (ack!).

```
int n = stringToInteger("42");
```

Works, because of explicit conversion.

Concatenation Bugs

```
string s = "hi" + '?';  // C-string + char
string s = "hi" + 41;   // C-string + int
```

Bug; produces garbage, not "hi?" or "hi42". *(memory address stuff)*

```
string s = "hi";    // char '?' is concatenated to string
s += '?';           // "hi?”
```

Works, because of auto-conversion.

```
s += 41;            // "hi?)”
```

Adds character with ASCII value 41, ') ', doesn't produce "hi?41".

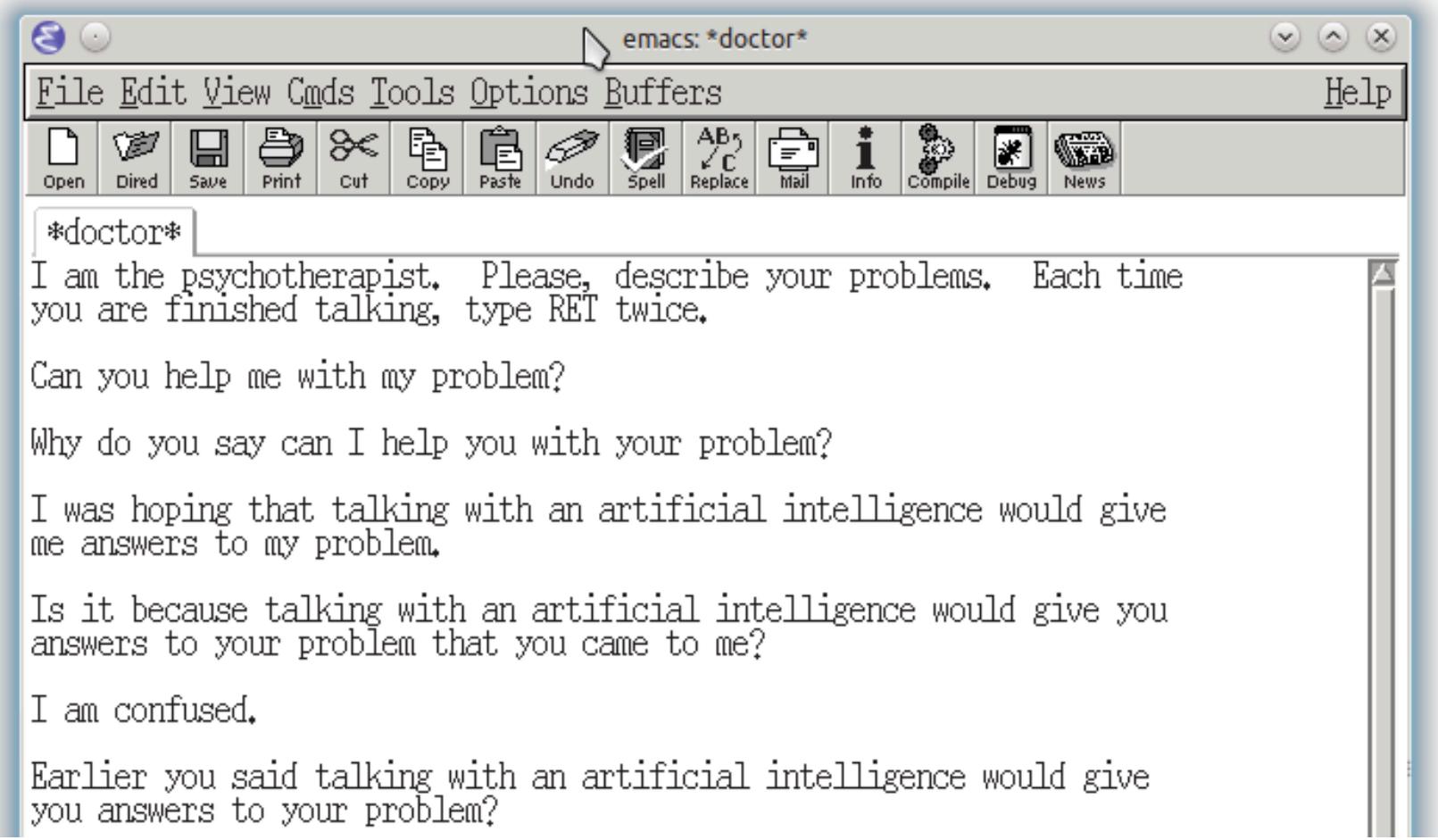
```
s += integerToString(41);  // "hi?41”
```

Works, because of conversion from `int` to `string`.

At this point seen some basics…

Ready to Dive In?

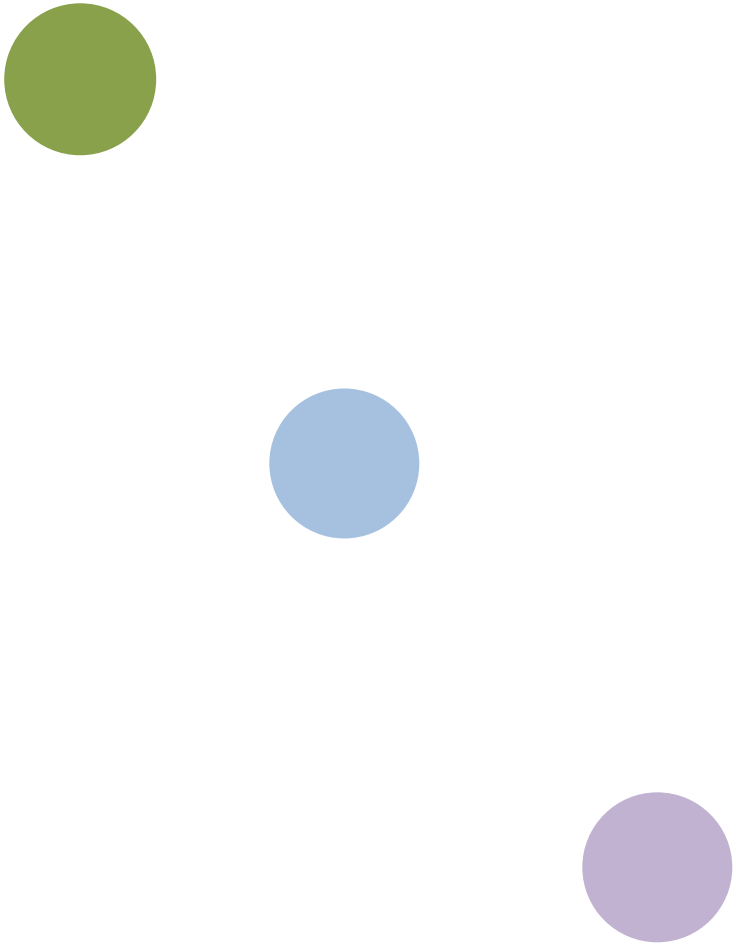
Doctor Eliza



Eliza *Beta*

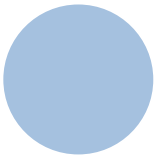
1. Until the end of time:
    1. Get a prompt from the user
    2. If prompt is hello, respond
    3. If prompt is a question ask why they asked
    4. If prompt is a statement challenge its truth

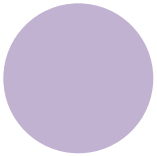
Points



Points

Basics of C++ and Java are very similar. Slight difference in Strings.





Great start to C++

Part 2: Functions

What is a Function?

A First C++ Function

```
#include <iostream>
#include "console.h"
using namespace std;

int main() {
    cout << "|-5| = " << absoluteValue(-5) << endl;
    return 0;
}

int absoluteValue(int n) {
    if(n < 0) {
        return -n;
    }
    return n;
}
```

A First C++ Function

```
#include <iostream>
#include "console.h"
using namespace std;

int main() {
    cout << "|-5| = " << absoluteValue(-5) << endl;
    return 0;
}

int absoluteValue(int n) {
    if(n < 0) {
        return -n;
    }
    return n;
}
```

Return

Name

Parameters

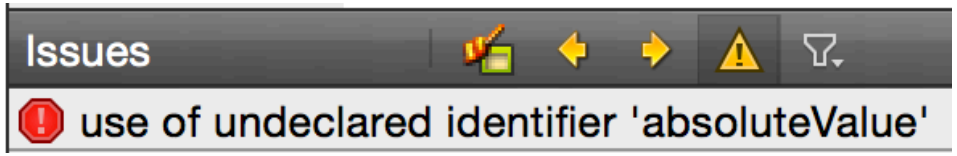
Body

A First C++ Function

```
#include <iostream>
#include "console.h"
using namespace std;

int main() {
    cout << "|-5| = " << absoluteValue(-5) << endl;
    return 0;
}

int absoluteValue(int n) {
    if(n < 0) {
        return -n;
    }
    return n;
}
```

A First C++ Function (Error)

```
#include <iostream>
#include "console.h"
using namespace std;

int main() {
    cout << "|-5| = " << absoluteValue(-5) << endl;
    return 0;
}

int absoluteValue(int n) {
    if(n < 0) {
        return -n;
    }
    return n;
}
```



A First C++ Function (Fix 1)

```
#include <iostream>
#include "console.h"
using namespace std;

int absoluteValue(int n) {
    if(n < 0) {
        return -n;
    }
    return n;
}

int main() {
    cout << "|-5| = " << absoluteValue(-5) << endl;
    return 0;
}
```

Fix: Change order of functions

A First C++ Function (Fix 2)

```
#include <iostream>
#include "console.h"
using namespace std;

int absoluteValue(int n);

int main() {
    cout << "|-5| = " << absoluteValue(-5) << endl;
    return 0;
}

int absoluteValue(int n) {
    if(n < 0) {
        return -n;
    }
    return n;
}
```

Fix: Add a function "prototype"

C++ Design?

Question: Why does C++ have the function prototype syntax?

Why not just have a rule that you must set up the ordering so you define your functions before using them, as in the "FIX 1" example?

A. **C++ could have done that**, but such a rule would be *too cumbersome* for programmers to follow.

B. **C++ could have done that**, but good programming *style* dictates "top-down" approach that logically puts main() first and helper functions it calls to follow.

C. **C++ could *not* have done that**, because sometimes there is *no way* to order the functions so that all functions are defined before being used.

D. **None of the above**

C++ Design?

Question: Why does C++ have the function prototype syntax?

Why not just have a rule that you must set up the ordering so you define your functions before using them, as in the "FIX 1" example?

A. **C++ could have done that**, but such a rule would be *too cumbersome* for programmers to follow.

Good logic

B. **C++ could have done that**, but good programming *style* dictates "top-down" approach that logically puts main() first and helper functions it calls to follow.

Good logic

C. **C++ could *not* have done that**, because sometimes there is *no way* to order the functions so that all functions are defined before being used.

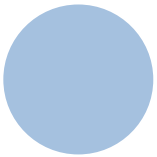
Correct

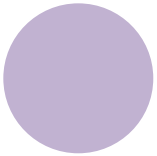
D. **None of the above**

No

Points

Basics of C++ and Java are very similar. Slight difference in Strings.

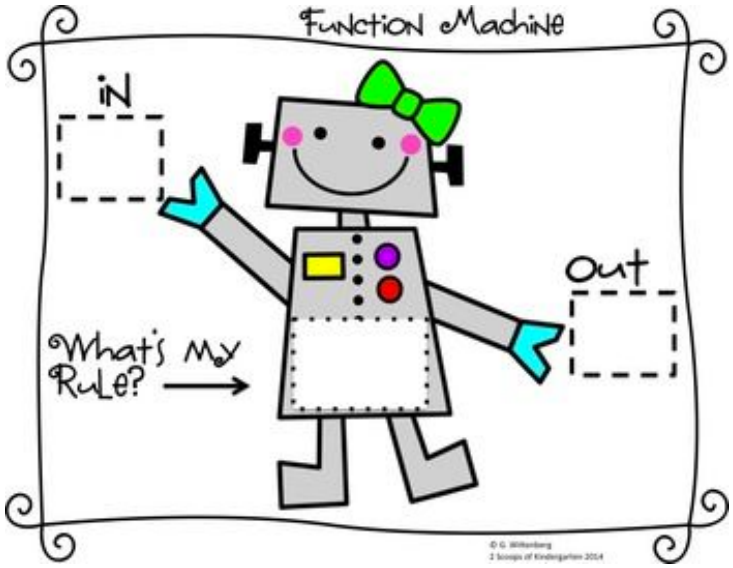


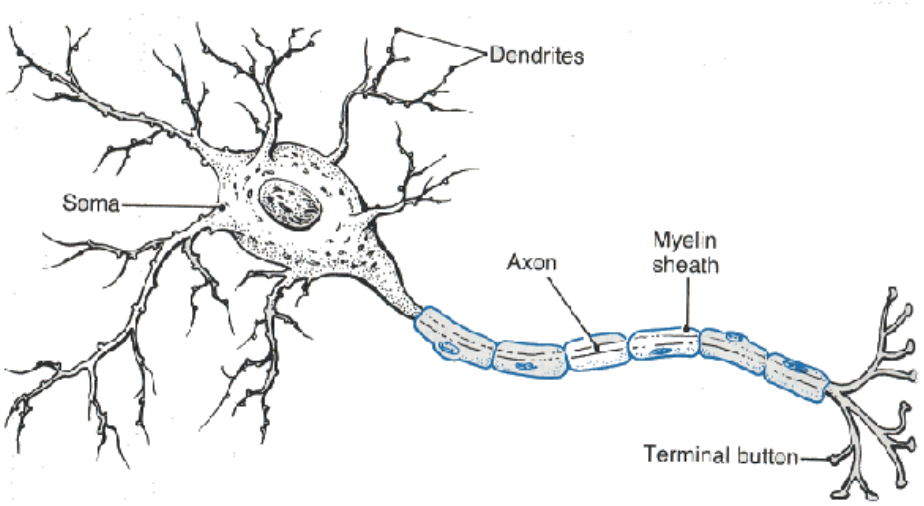


Points

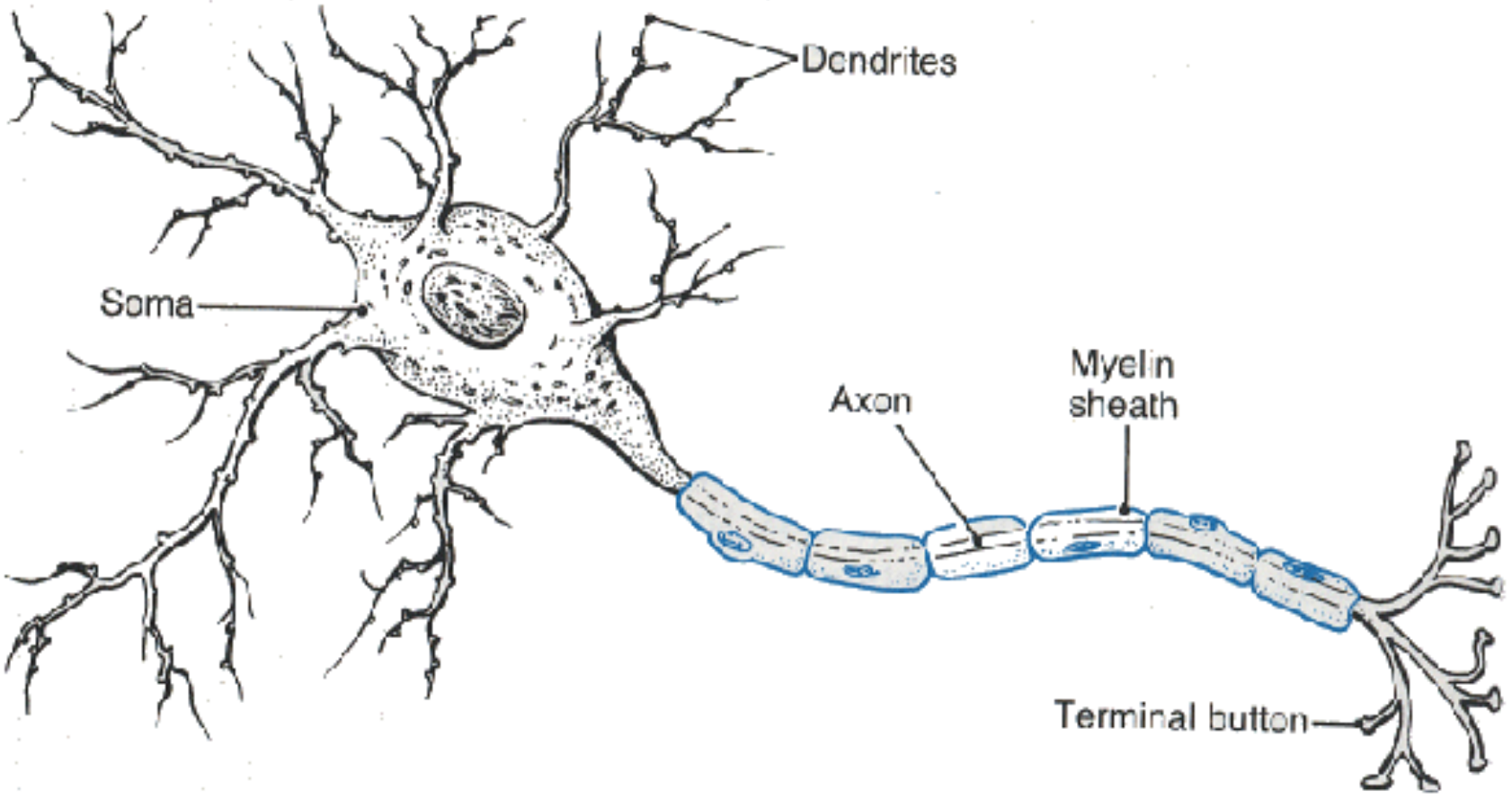
- Basics of C++ and Java are very similar. Slight difference in Strings.
- Prototype your functions

Lets Write a Function

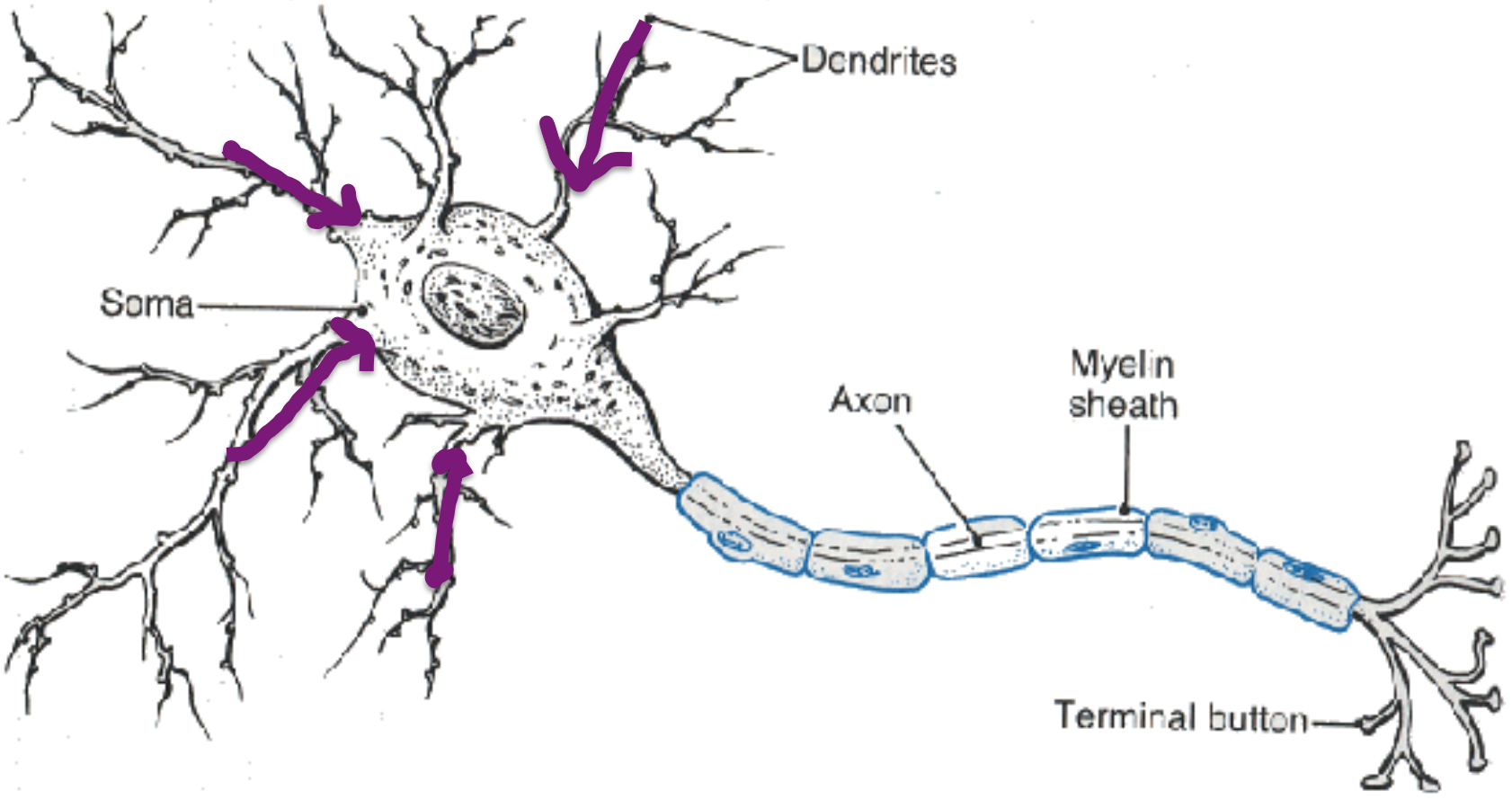




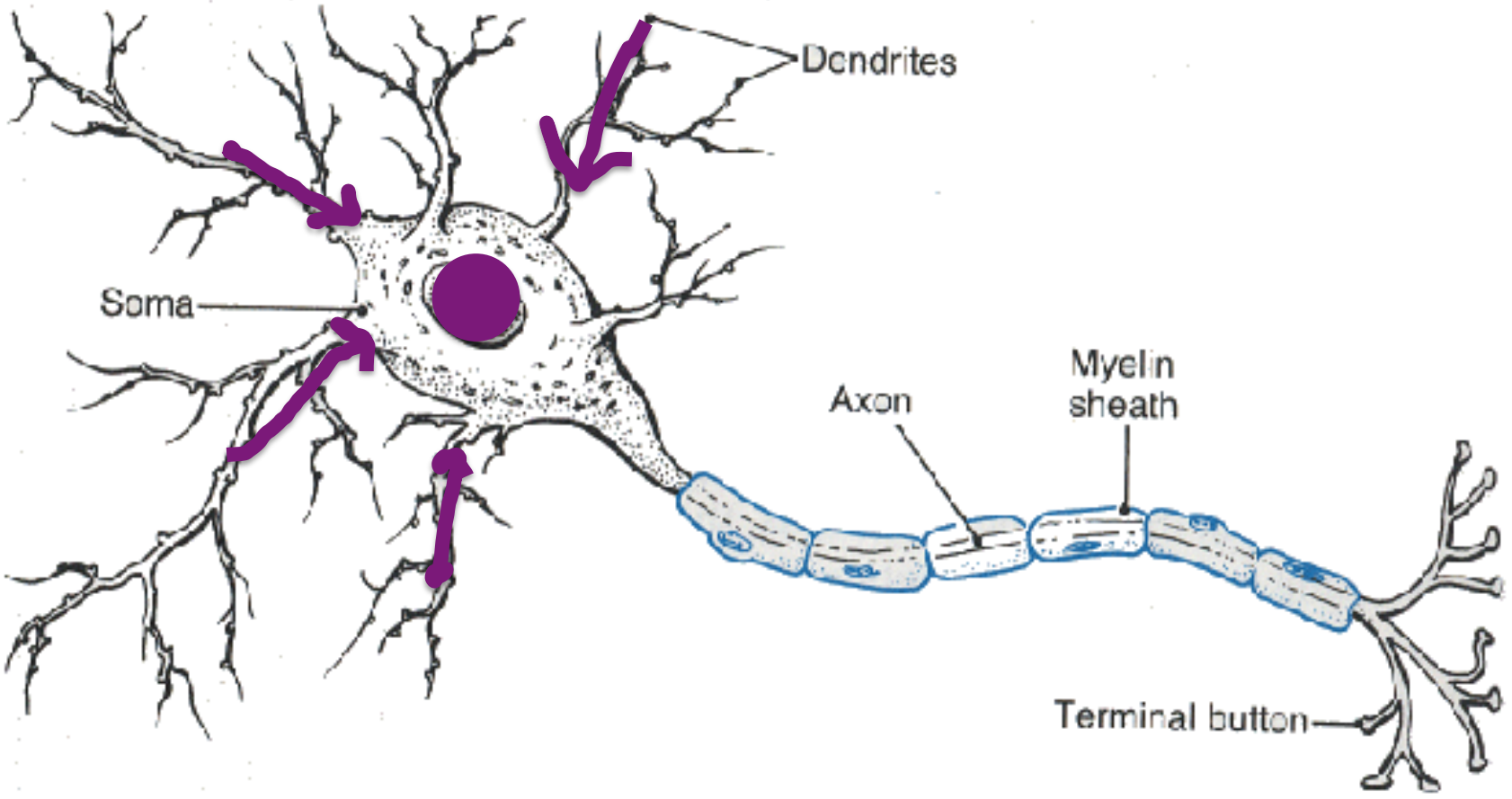
Neuron



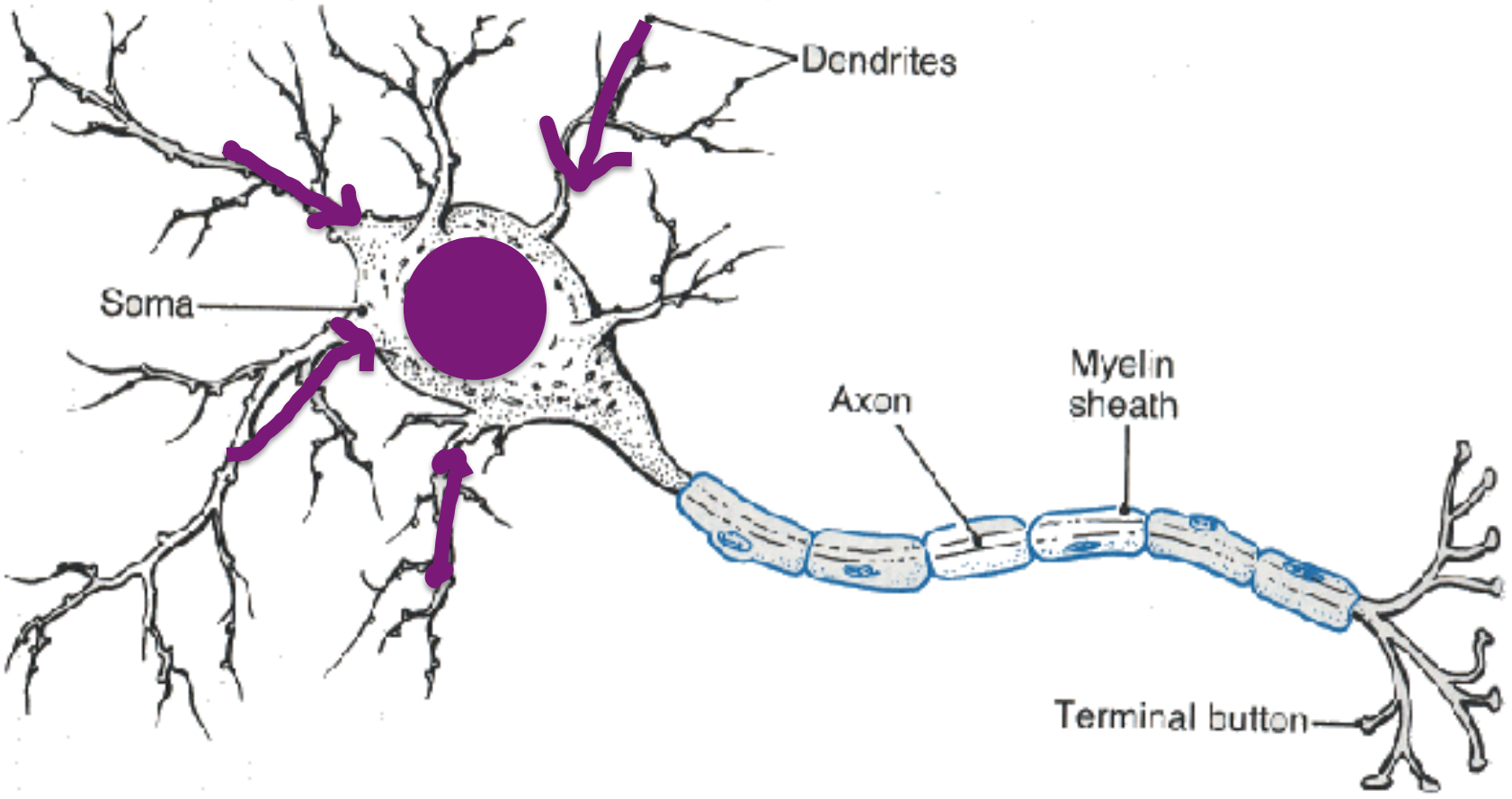
Neuron



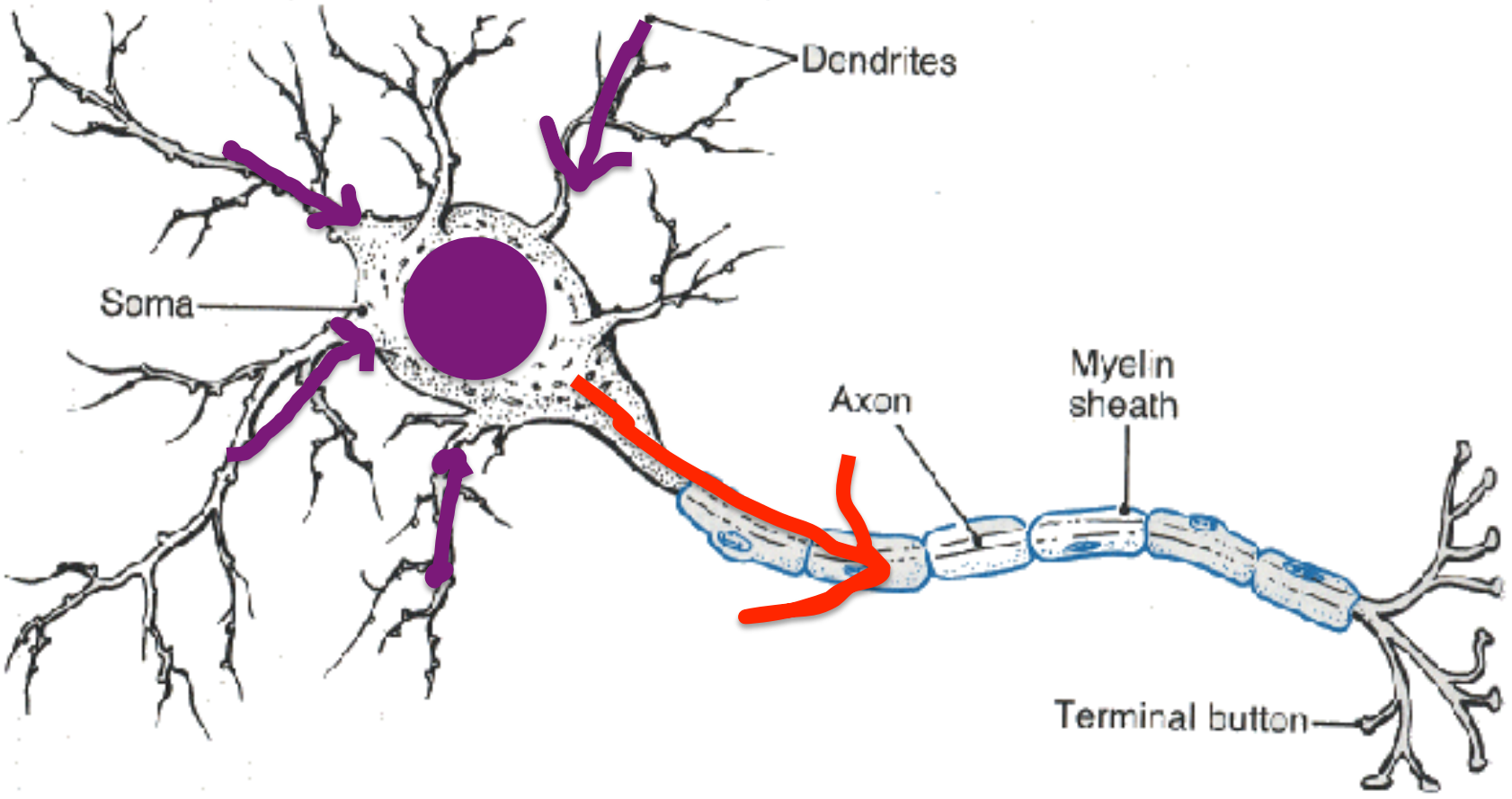
Neuron



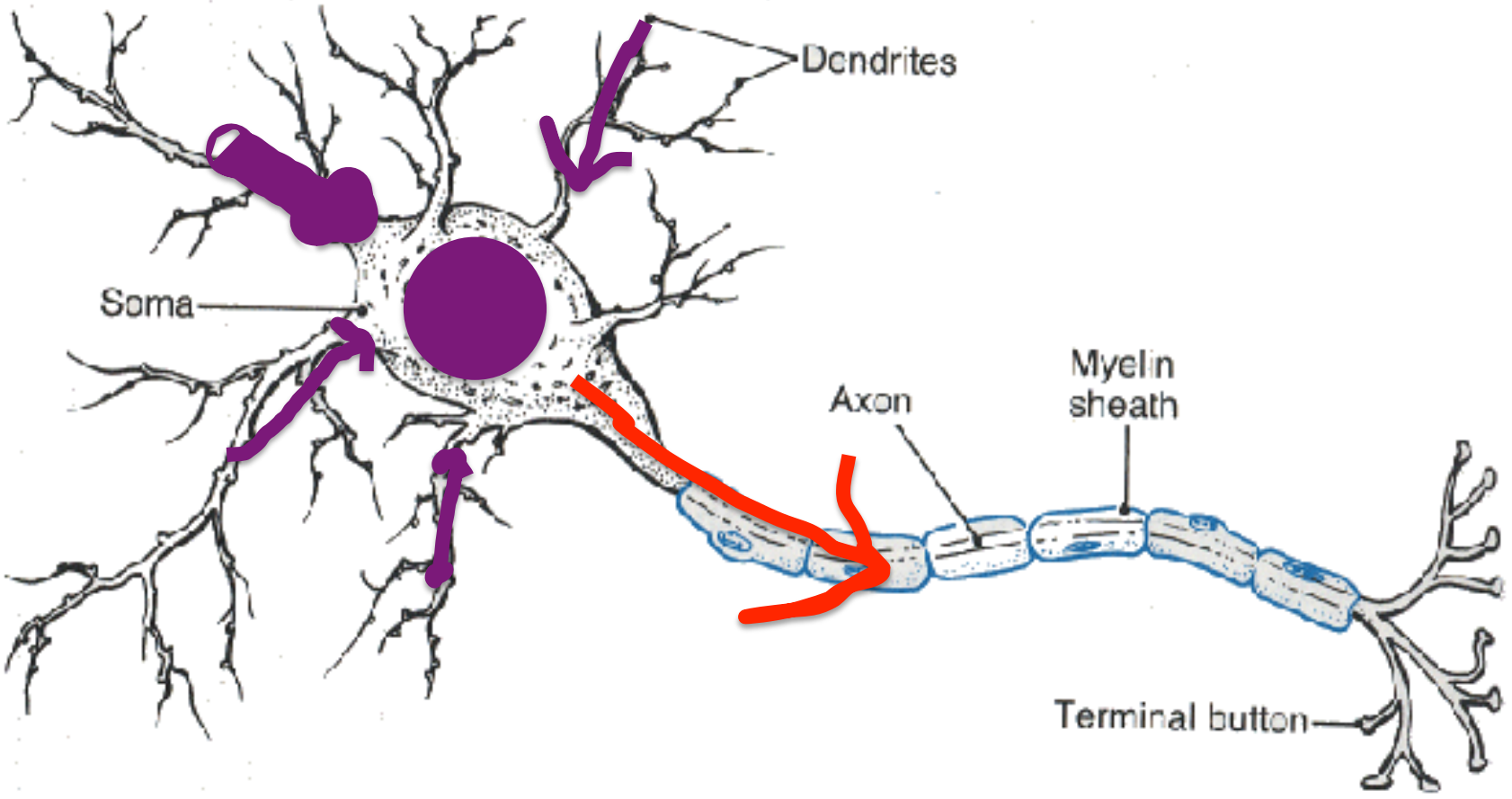
Neuron



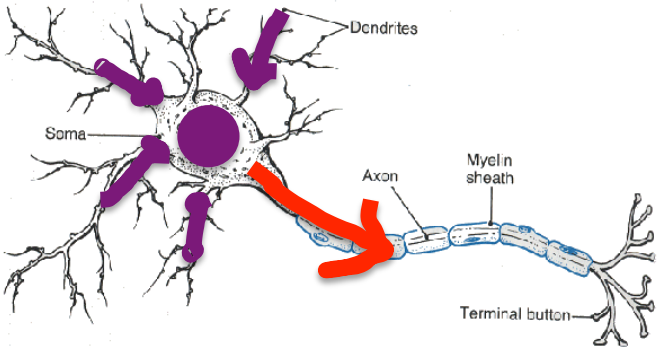
Neuron

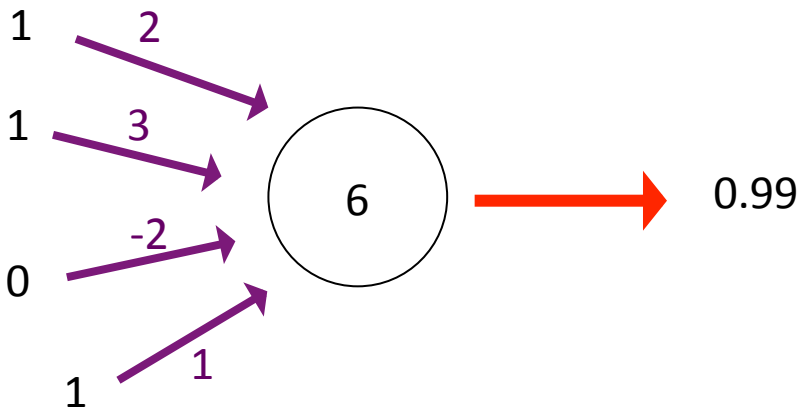


Some Inputs are More Important

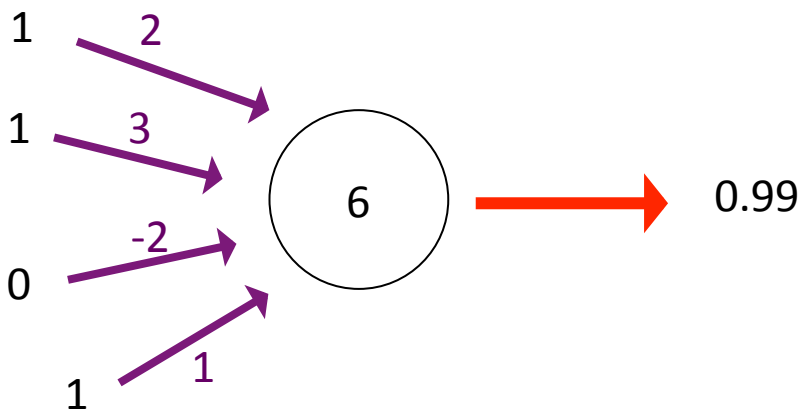


Artificial Neuron



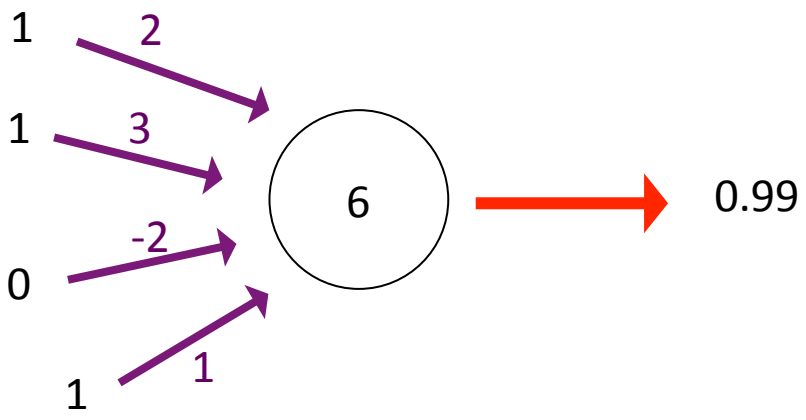


Artificial Neuron



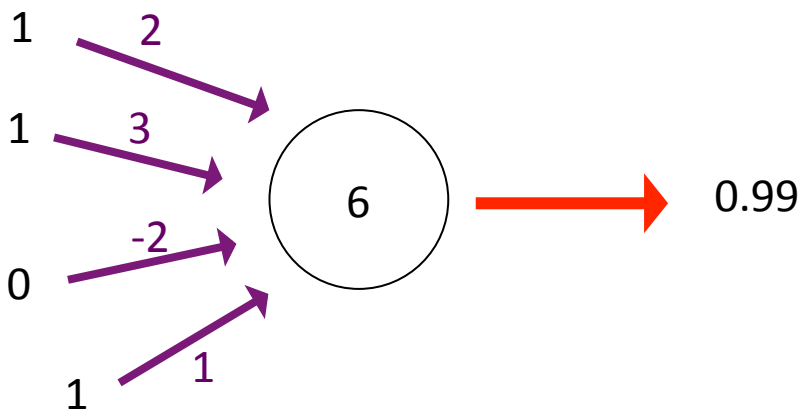

```
InputSum  = input1 * weight1 +
            input2 * weight2 +
            input3 * weight3 +
            input4 * weight4
```

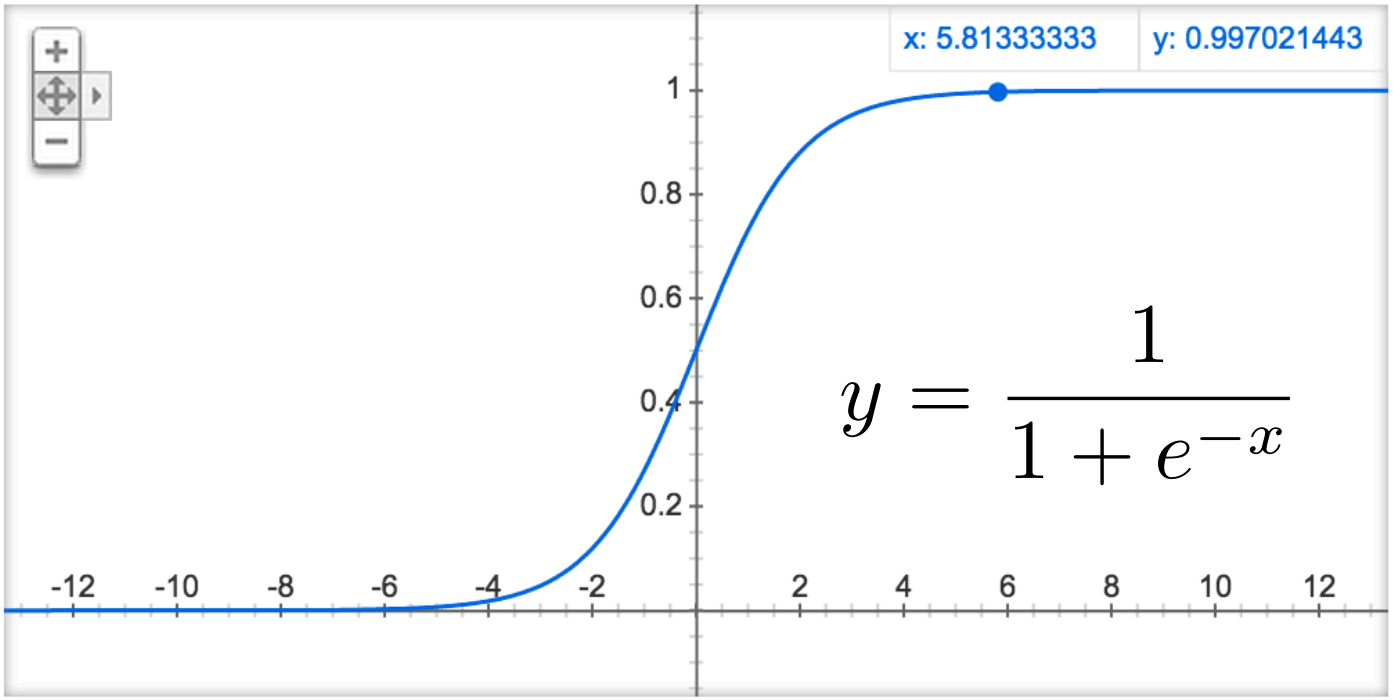
Artificial Neuron



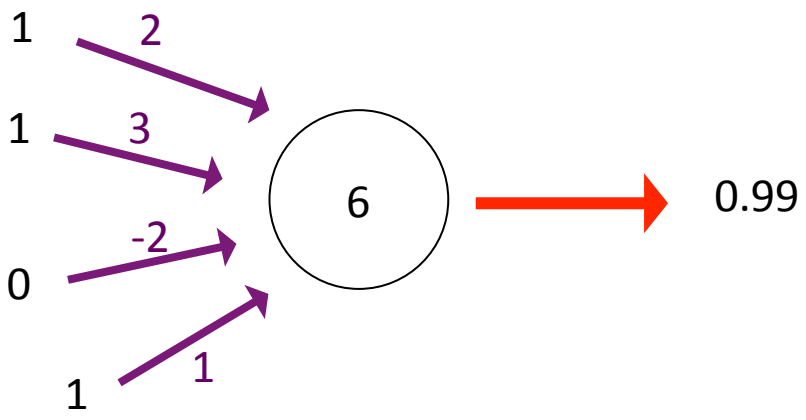

```
InputSum  = 1 * 2 +
            1 * 3 +
            0 * -2 +
            1 * 1
```

Sigmoid Function



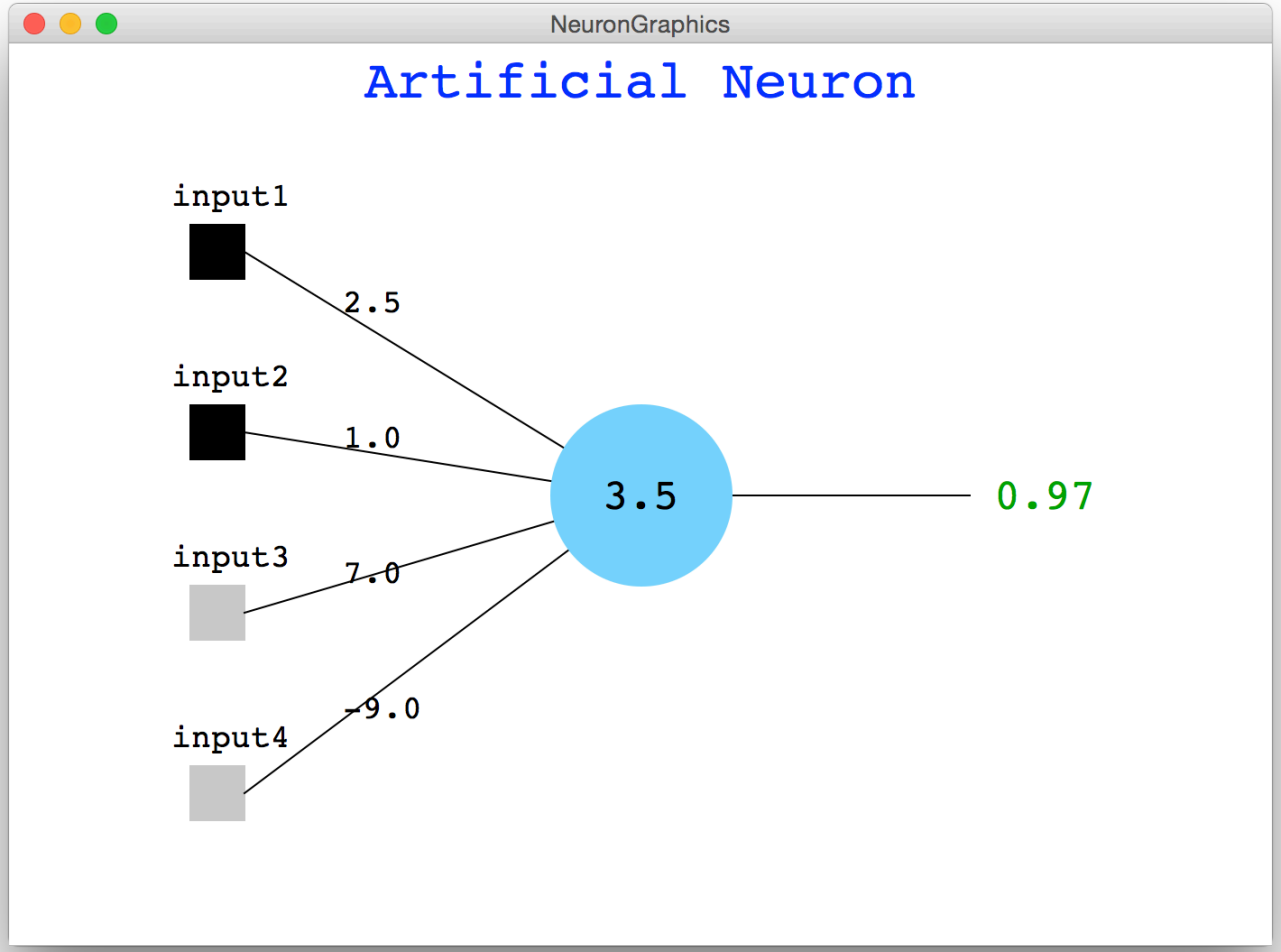


Artificial Neuron

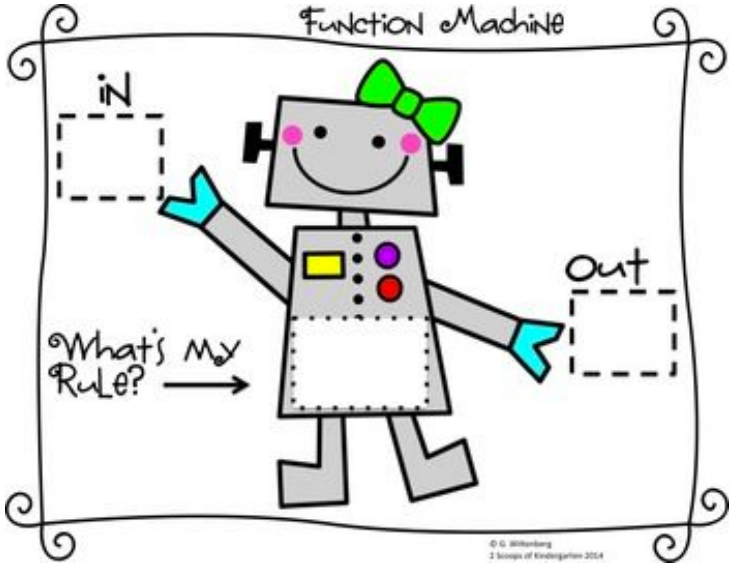


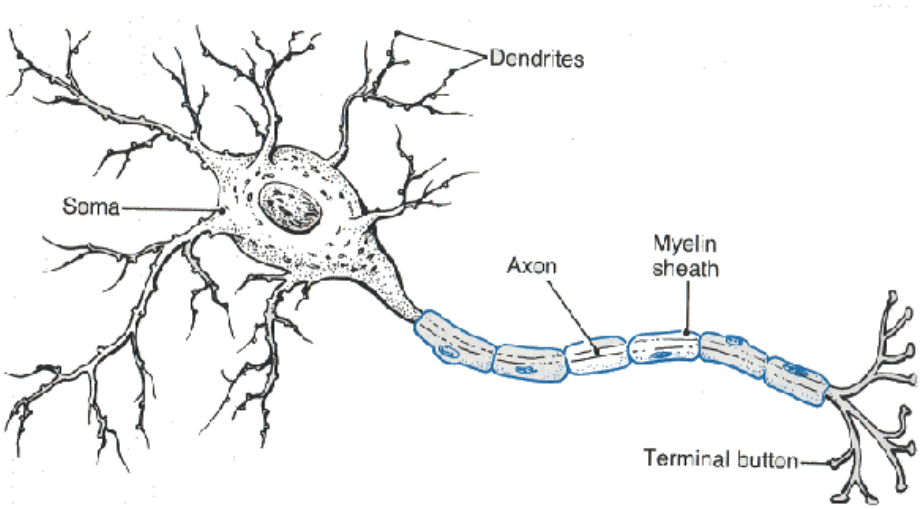

```
InputSum  = input1 * weight1 +
            input2 * weight2 +
            input3 * weight3 +
            input4 * weight4;
Output    = sigmoid(InputSum);
```

Artificial Neuron Demo

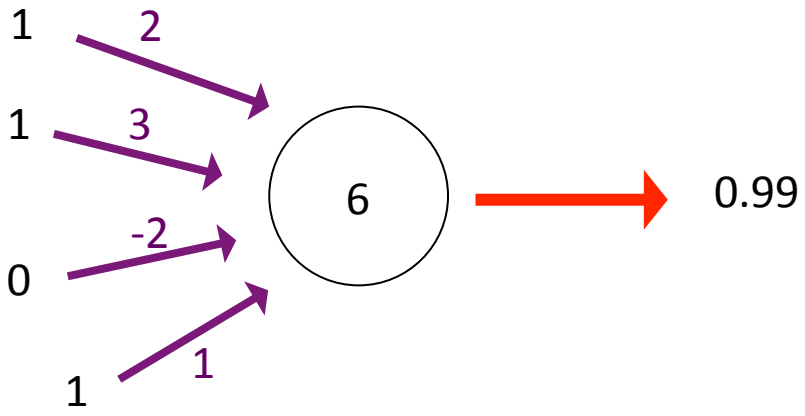


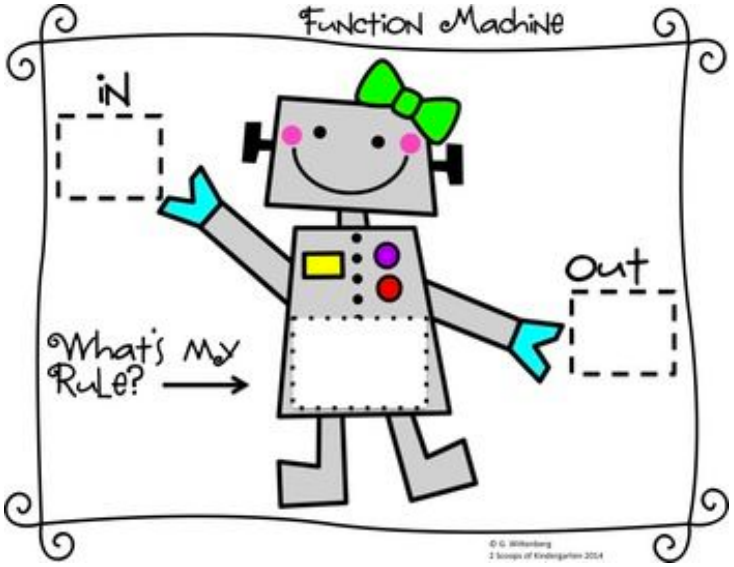
Let's Write a Function





Lets Write a Function






```
InputSum  = input1 * weight1 +
            input2 * weight2 +
            input3 * weight3 +
            input4 * weight4;
Output    = sigmoid(InputSum);
```

```
double neuron(double x1, double x2);
```

Pass by Value

Value semantics: In Java and C++, when variables (`int`, `double`) are passed as parameters, their values are copied.

Modifying the parameter will not affect the variable passed in.

```
void grow(int age) {
    age = age + 1;
    cout << "grow age is " << age << endl;
}

int main() {
    int age = 20;
    cout << "main age is " << age << endl;
    grow(age);
    cout << "main age is " << age << endl;
    return 0;
}
```

Output:

```
main age is 20
grow age is 21
main age is 20
```

Pass by Reference (2.5)

Reference semantics: In C++, if you declare a parameter with an & after its type, instead of passing a copy of its value, it will link the caller and callee functions to the same variable in memory.

Modifying the parameter *will* affect the variable passed in.

```
void grow(int& age) {
    age = age + 1;
    cout << "grow age is " << age << endl;
}

int main() {
    int age = 20;
    cout << "main age is " << age << endl;
    grow(age);
    cout << "main age is " << age << endl;
    return 0;
}
```

Output:

```
main age is 20
grow age is 21
main age is 21
```

Pass by Reference Example

With reference parameters, you can write a swap function.

```
/*
 * Places a's value into b and vice versa.
 */
void swap(int& a, int& b) {
    int temp = a;
    a = b;
    b = temp;
}
```

A Mystery

- What is the output of this code?

```
void mystery(int& b, int c, int& a) {
    b--;
    a++;                                    // A. 5 3 13
    c += a;                                 // B. 5 2 8
}                                           // C. 4 3 7
                                            // D. 4 2 9
int main() {                                // E. other
    int a = 5;
    int b = 2;
    int c = 8;
    mystery(a, b, c);
    cout << a << " " << b << " " << c << endl;
    return 0;
}
```

A Mystery

- What is the output of this code?

```
void mystery(int& b, int c, int& a) {
    b--;
    a++;
    c += a;
}

int main() {
    int a = 5;
    int b = 2;
    int c = 8;
    mystery(a, b, c);
    cout << a << " " << b << " " << c << endl;
    return 0;
}
```

A Mystery

- What is the output of this code?

```
void mystery(int& b, int c, int& a) {
    b--;
    a++;
    c += a;
}

int main() {
    int a = 5;
    int b = 2;
    int c = 8;
    mystery(a, b, c);
    cout << a << " " << b << " " << c << endl;
    return 0;
}
```

A Mystery

- What is the output of this code?

b = 5 c = 2 a = 8

```
void mystery(int& b, int c, int& a) {
    b--;
    a++;
    c += a;
}

int main() {
    int a = 5;
    int b = 2;
    int c = 8;
    mystery(a, b, c);
    cout << a << " " << b << " " << c << endl;
    return 0;
}
```

A Mystery

- What is the output of this code?

b = 5 c = 2 a = 8

```
void mystery(int& b, int c, int& a) {
    b--;
    a++;
    c += a;
}

int main() {
    int a = 5;
    int b = 2;
    int c = 8;
    mystery(a, b, c);
    cout << a << " " << b << " " << c << endl;
    return 0;
}
```

Now b = 4

A Mystery

- What is the output of this code?

b = 5 c = 2 a = 8

```
void mystery(int& b, int c, int& a) {
    b--;   Now b = 4
    a++;   Now a = 9
    c += a;
}

int main() {
    int a = 5;
    int b = 2;
    int c = 8;
    mystery(a, b, c);
    cout << a << " " << b << " " << c << endl;
    return 0;
}
```

A Mystery

- What is the output of this code?

b = 5 c = 2 a = 8

```
void mystery(int& b, int c, int& a) {
    b--;   Now b = 4
    a++;   Now a = 9
    c += a; Now c = 11
}

int main() {
    int a = 5;
    int b = 2;
    int c = 8;
    mystery(a, b, c);
    cout << a << " " << b << " " << c << endl;
    return 0;
}
```

Output Parameters

What is the minimum and maximum non-creepy age to date?

http://xkcd.com/314/

Output Parameters

```
void datingRange(int age, int& min, int& max);

int main() {
    int young;
    int old;
    datingRange(48, young, old);
    cout << "A 48-year-old could date someone from "
         << young
         << " to "
         << old
         << " years old."
         << endl;
    return 0;
}

void datingRange(int age, int& min, int& max) {
    min = age / 2 + 7;
    max = (age - 7) * 2;
}


// A 48-year-old could date someone from
// 31 to 82 years old.
```

Pass by Reference Pros / Cons

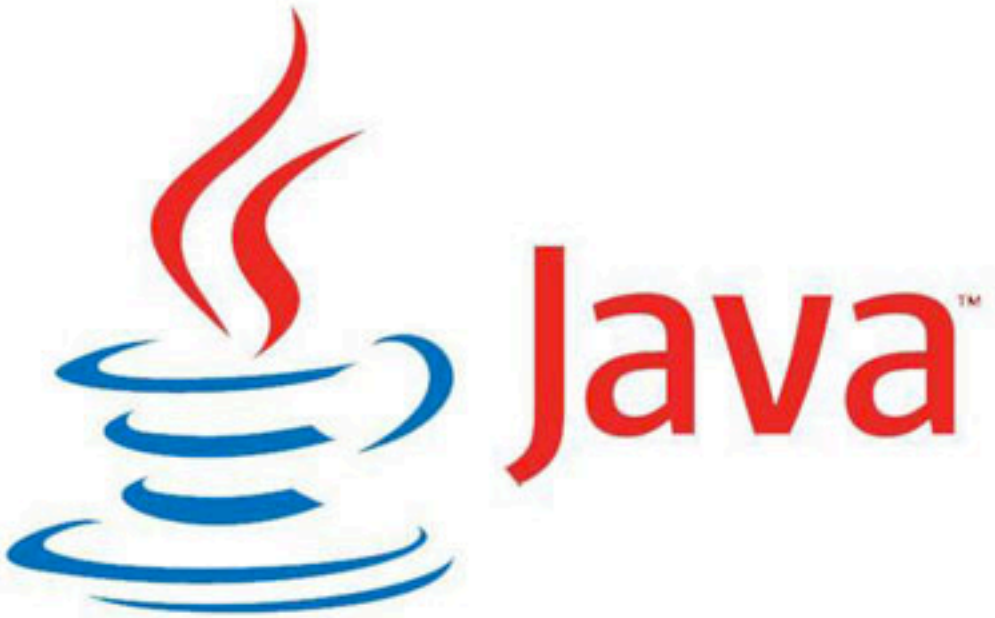
Benefits of reference parameters:

- a useful way to be able to 'return' more than one value
- often used with objects, to avoid making bulky copies when passing

Downsides of reference parameters:

- hard to tell from call whether it is ref; can't tell if it will be changed
  - `foo(a, b, c);  // will foo change a, b, or c?  :-/`
- (very) slightly slower than value parameters
- can't pass a literal value to a ref parameter
  - `grow(39);  // error`

Insight into Java



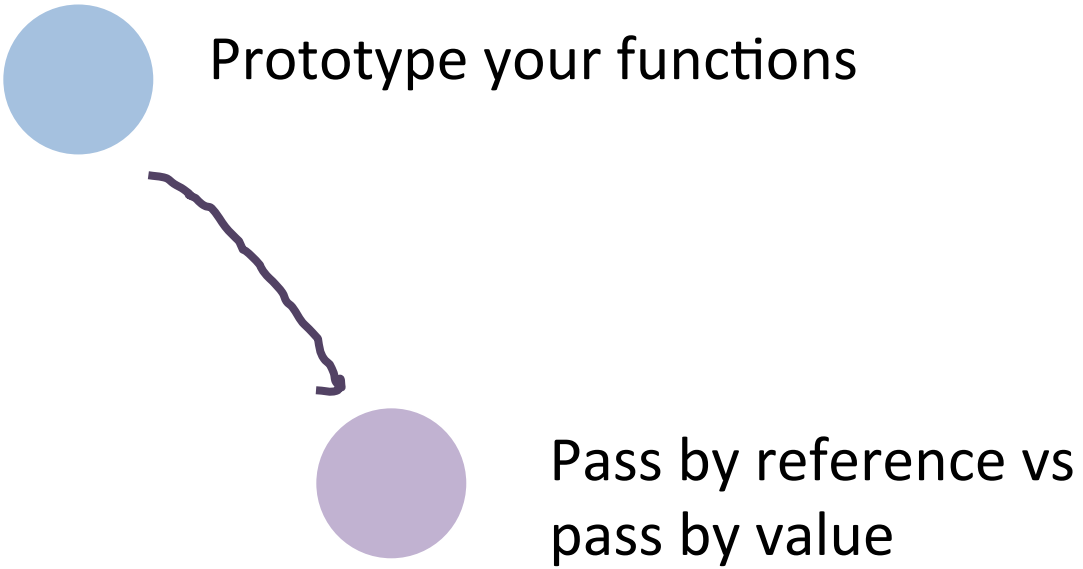
In Java, primitives are passed by value, and objects are passed by reference

Points

- Basics of C++ and Java are very similar. Slight difference in Strings.
- Prototype your functions

Points

- Basics of C++ and Java are very similar. Slight difference in Strings.



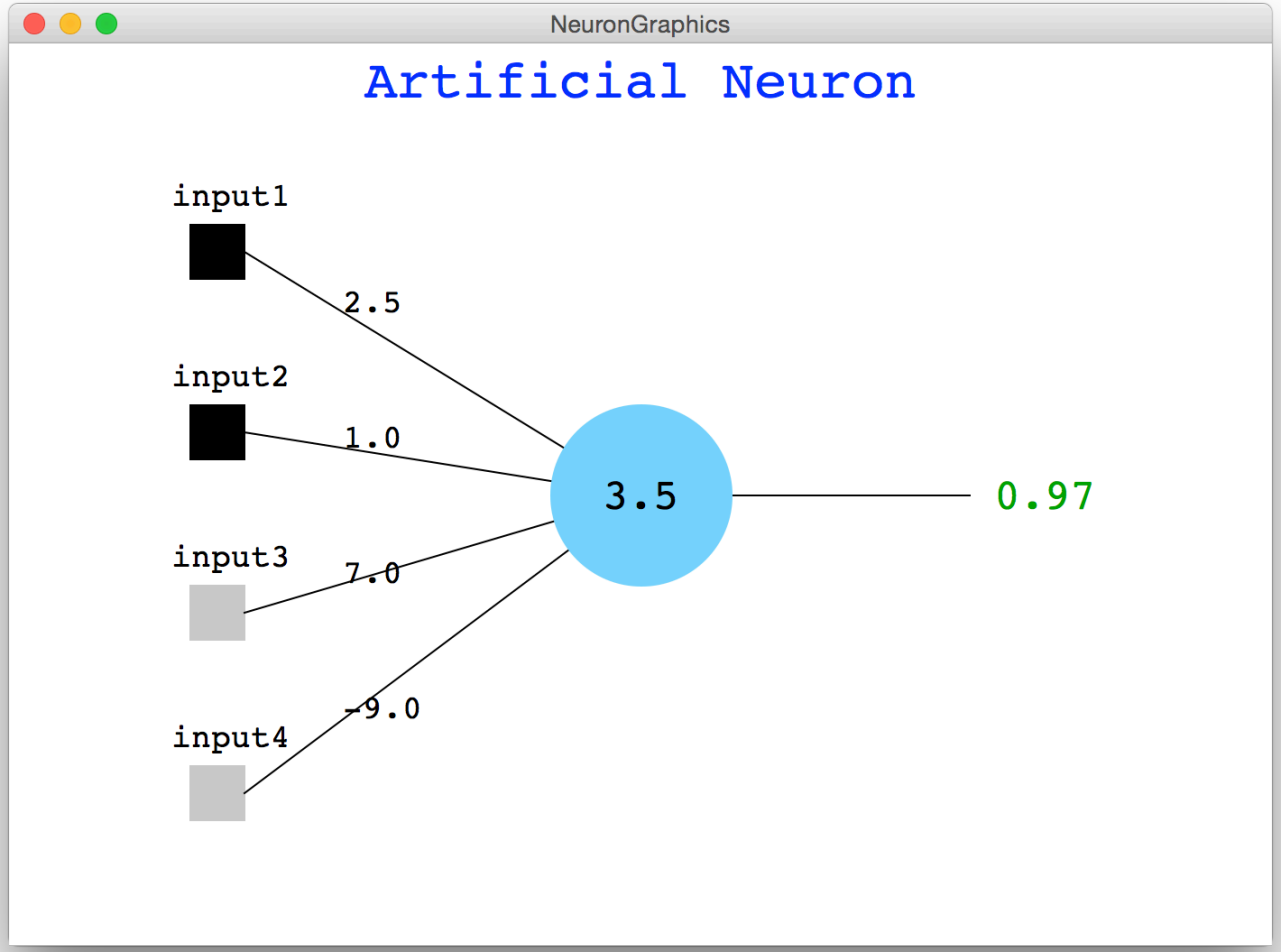
Today's Goals

1. Intro to C++
2. Functions in C++

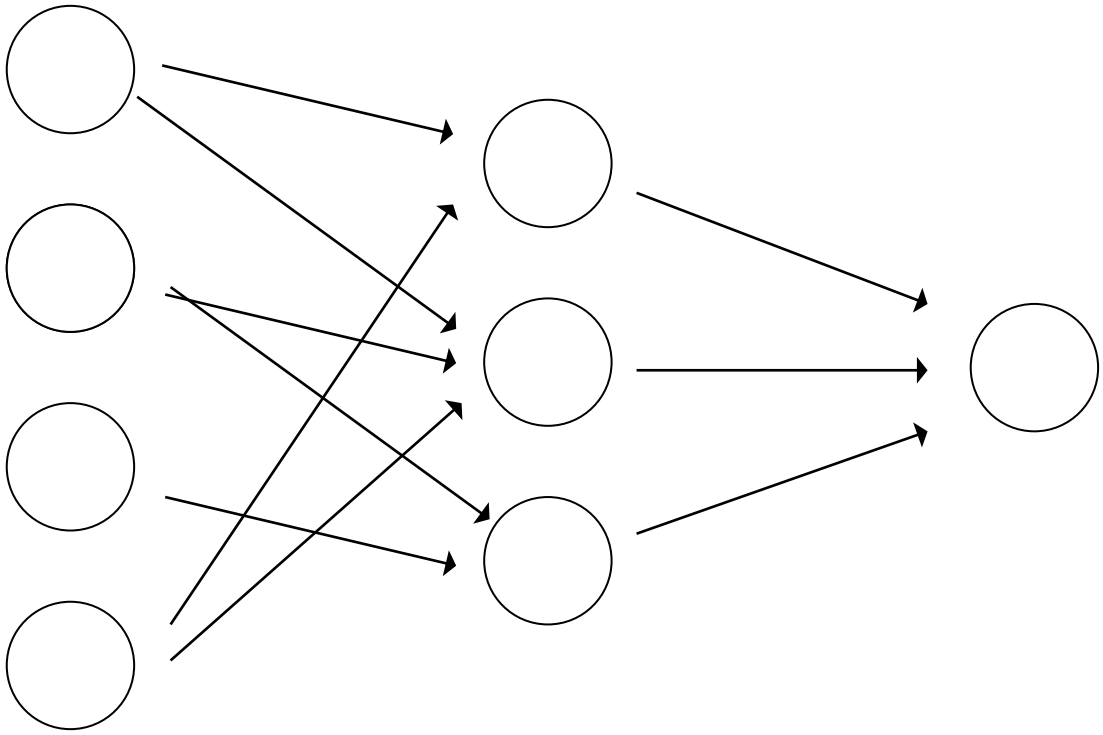
Story of Modern AI

Whats up with AI?

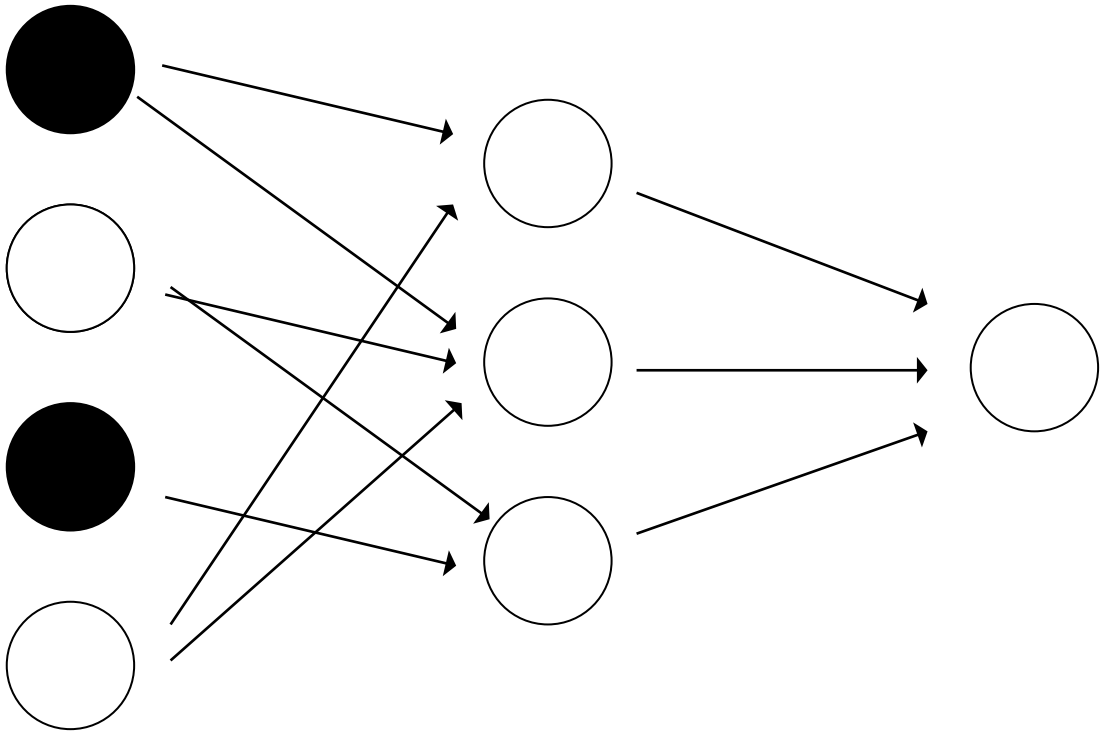
Remember Me?


NeuronGraphics
Artificial Neuron
input1
input2
input3
input4
2.5
1.0
7.0
-9.0
3.5
0.97

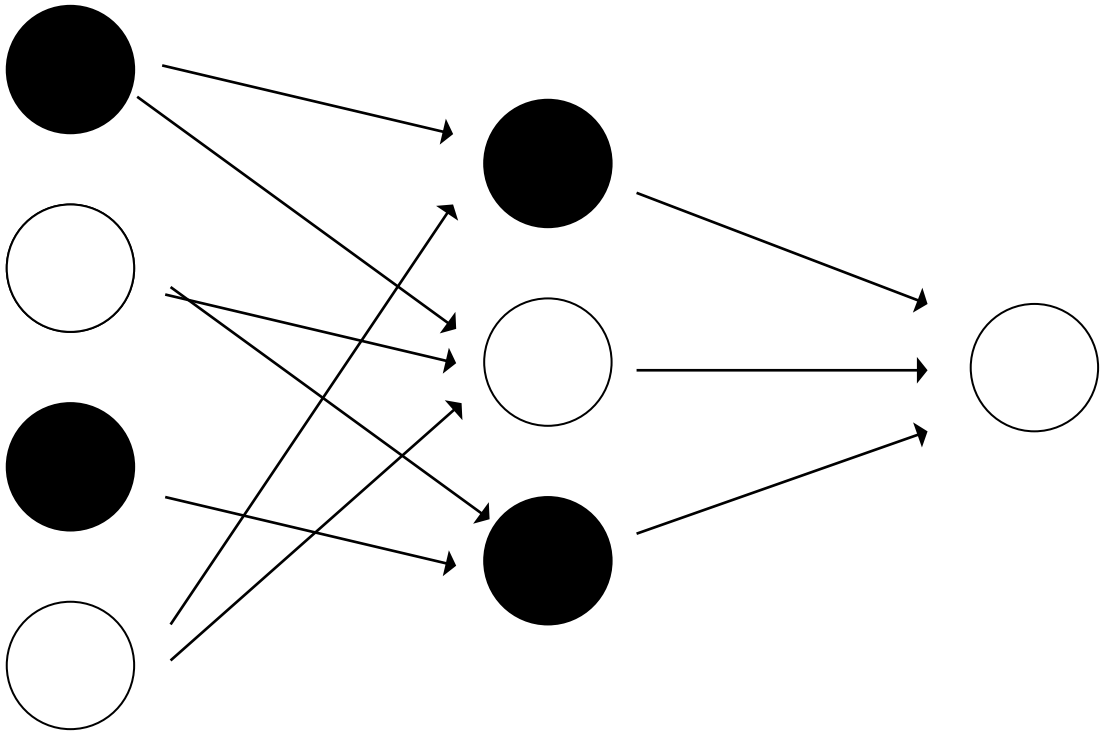
Put Many Together



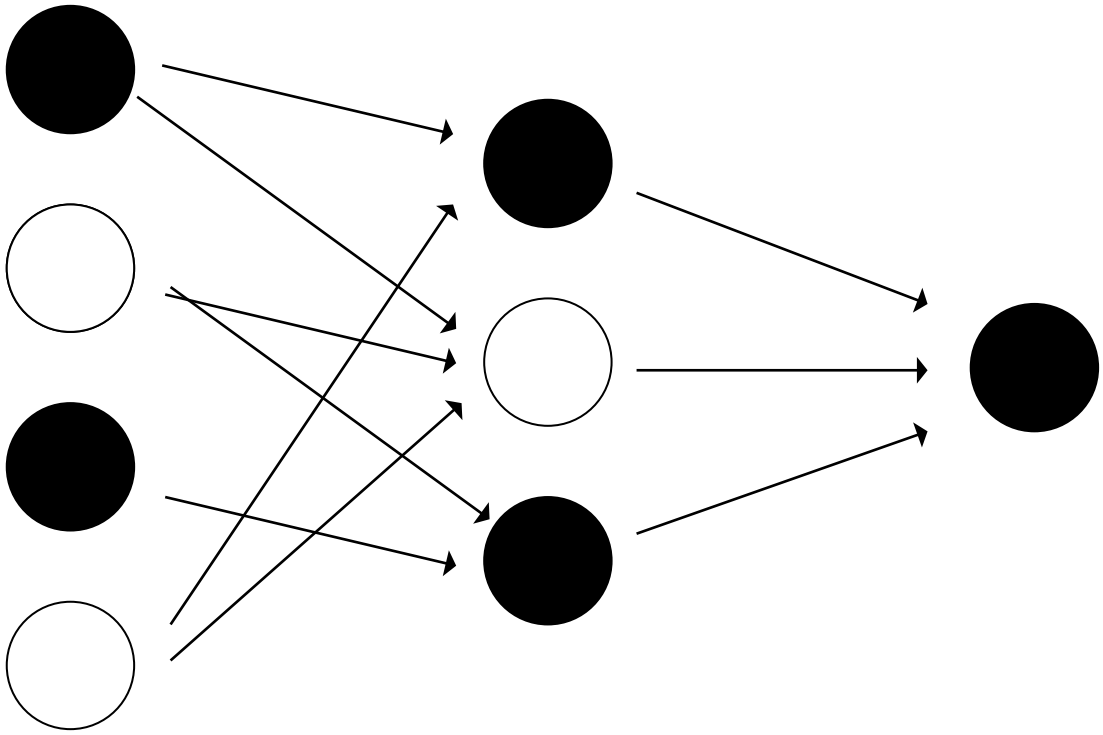
Put Many Together



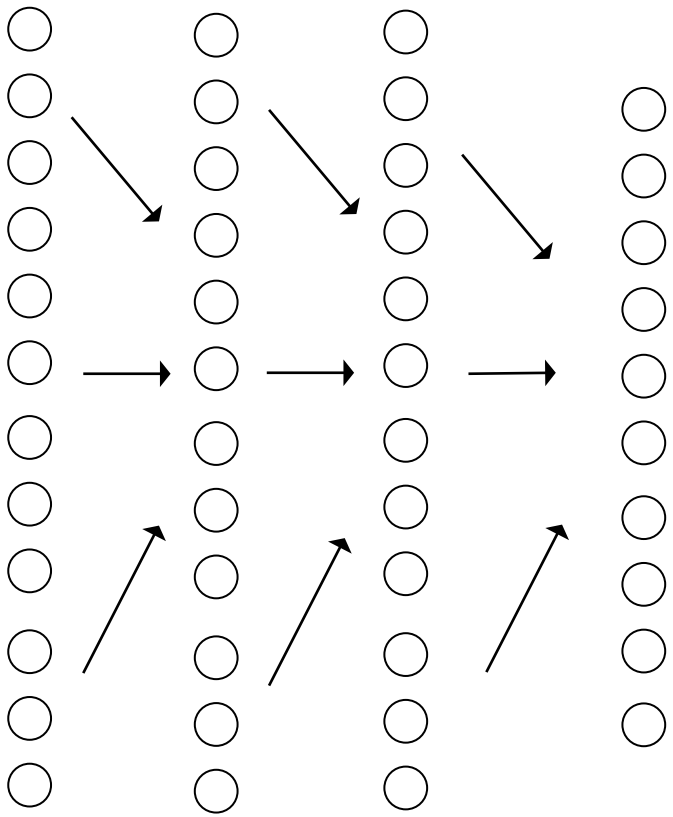
Put Many Together



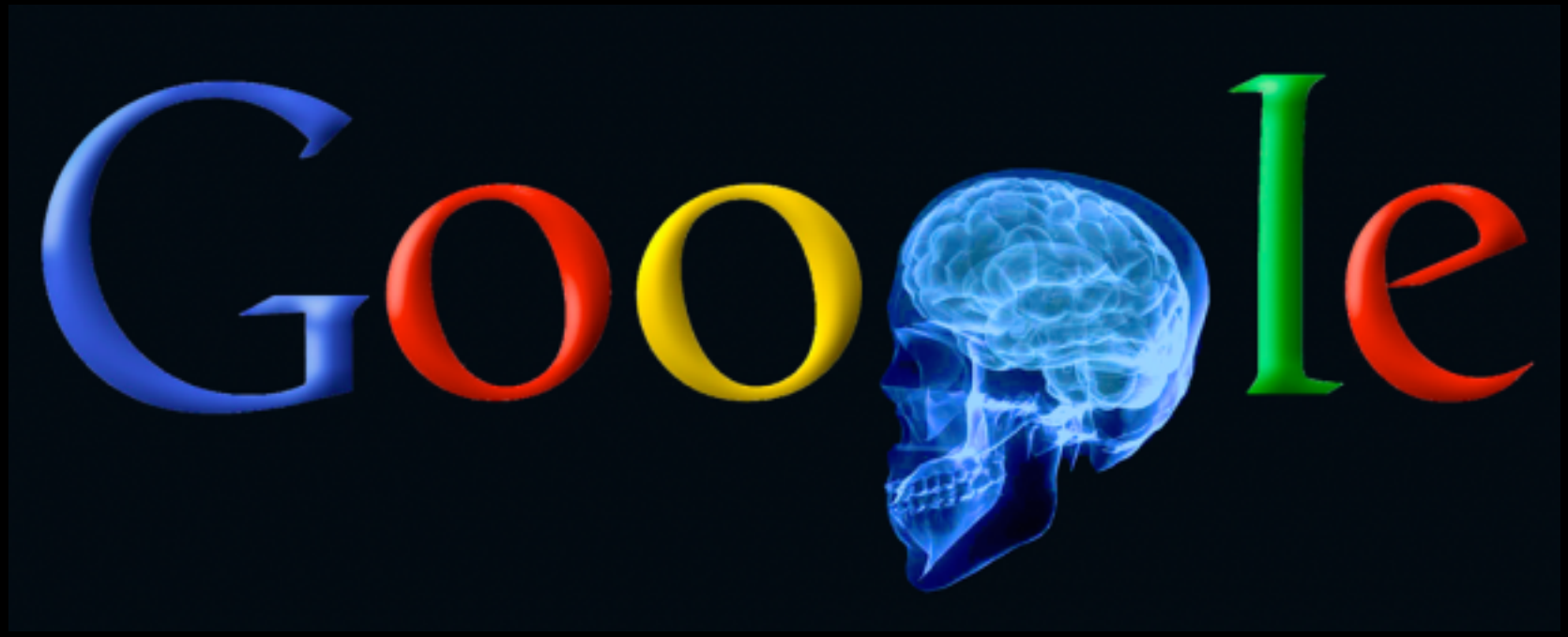
Put Many Together



Put Many Together

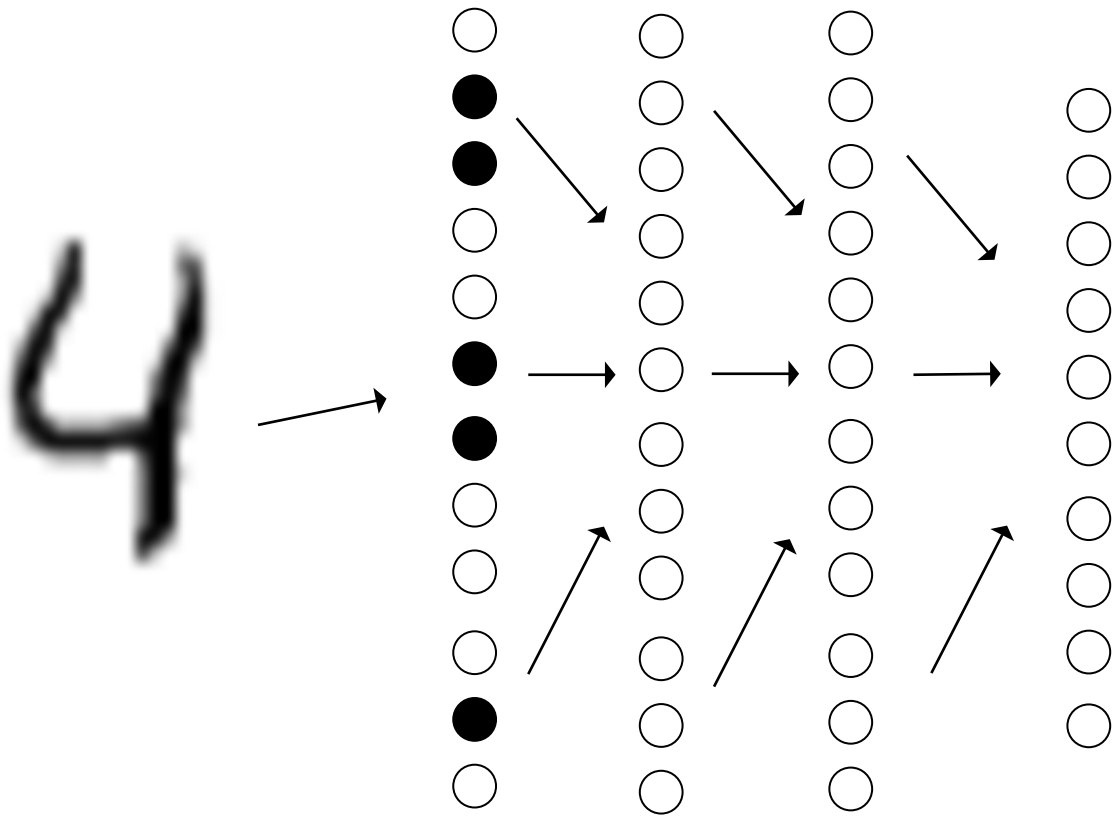


Google Brain

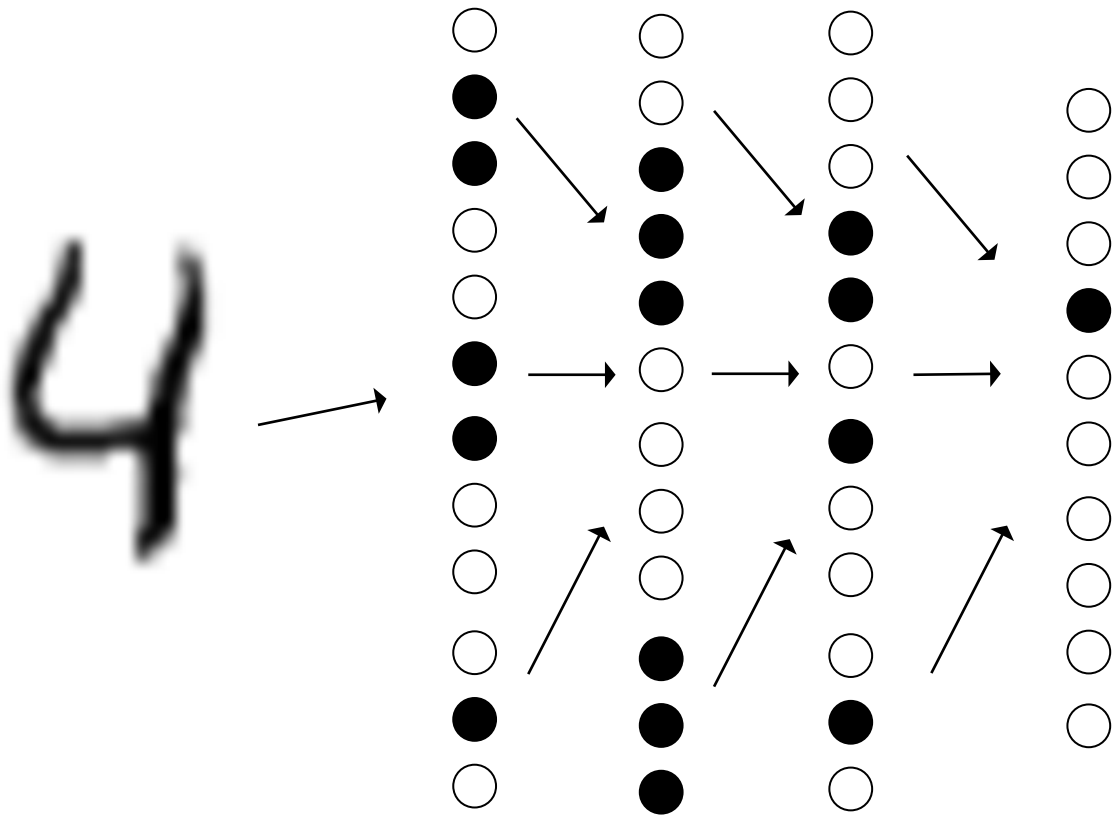


1 Trillion Artificial Neurons

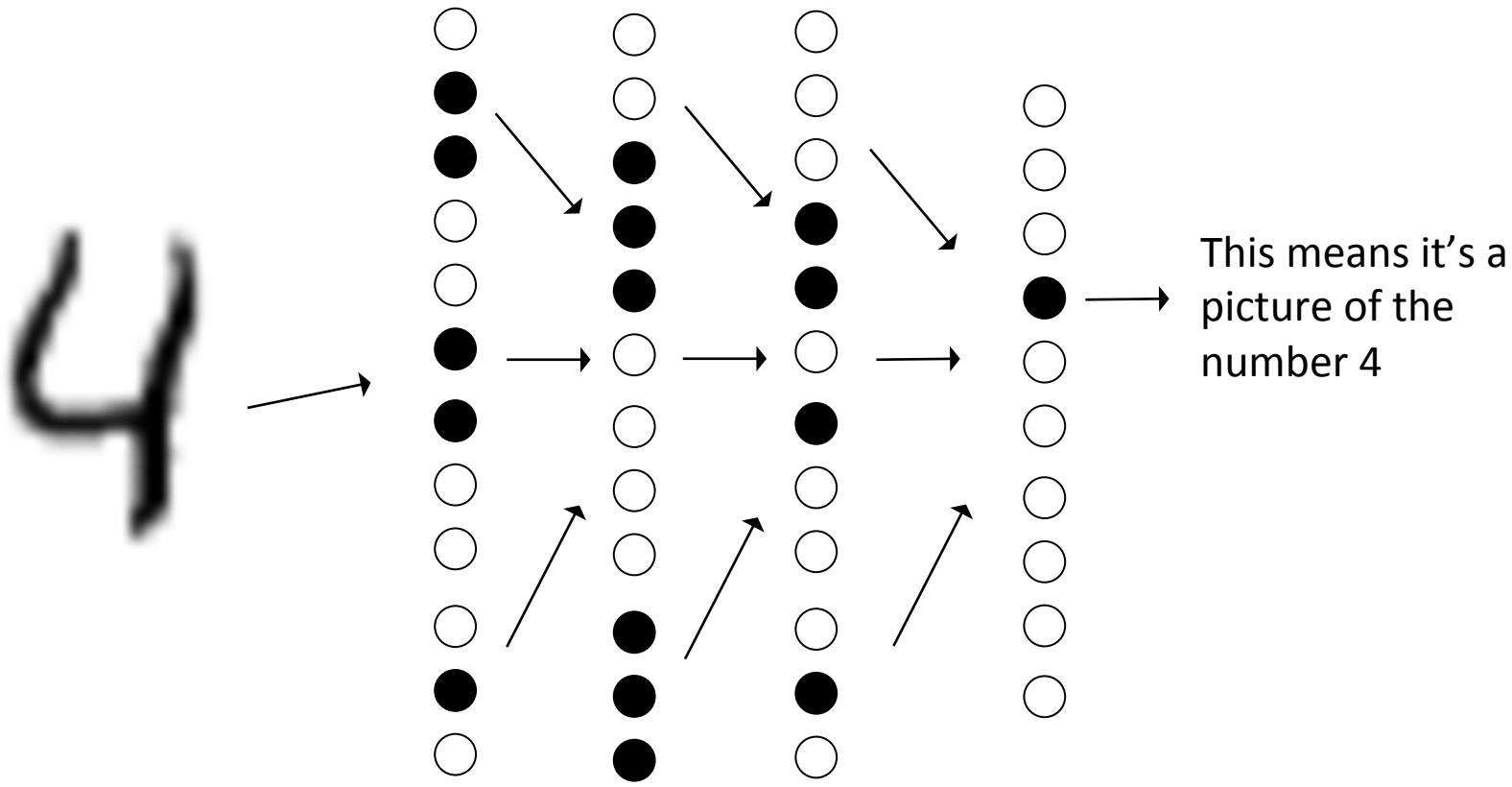
Make It Useful



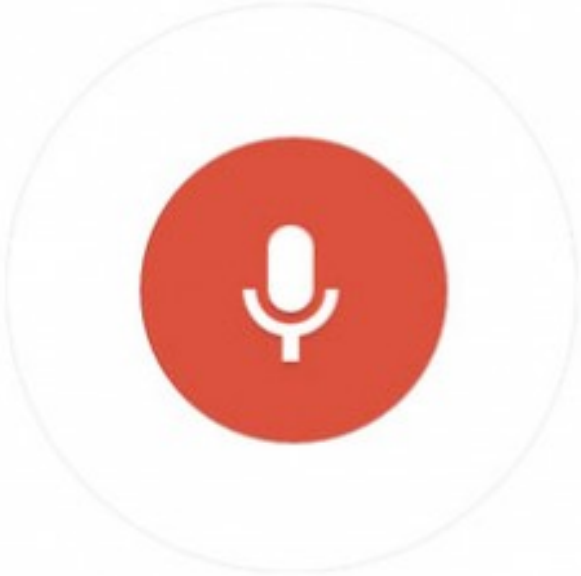
Make It Useful



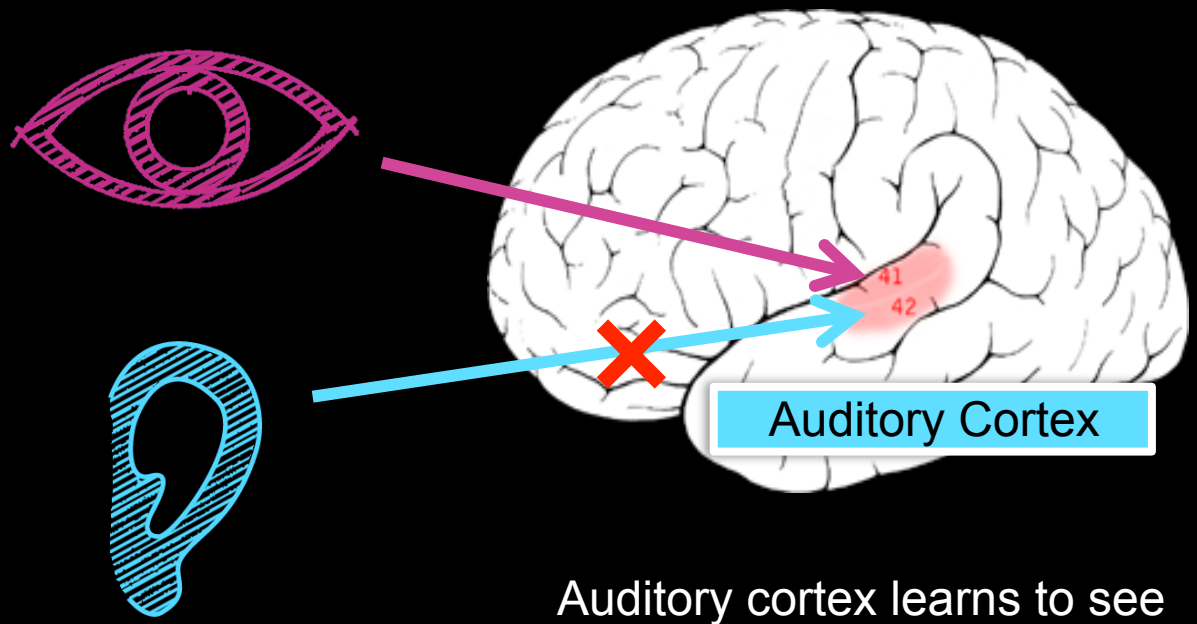
Make It Useful



Big Milestones



One Algorithm Hypothesis

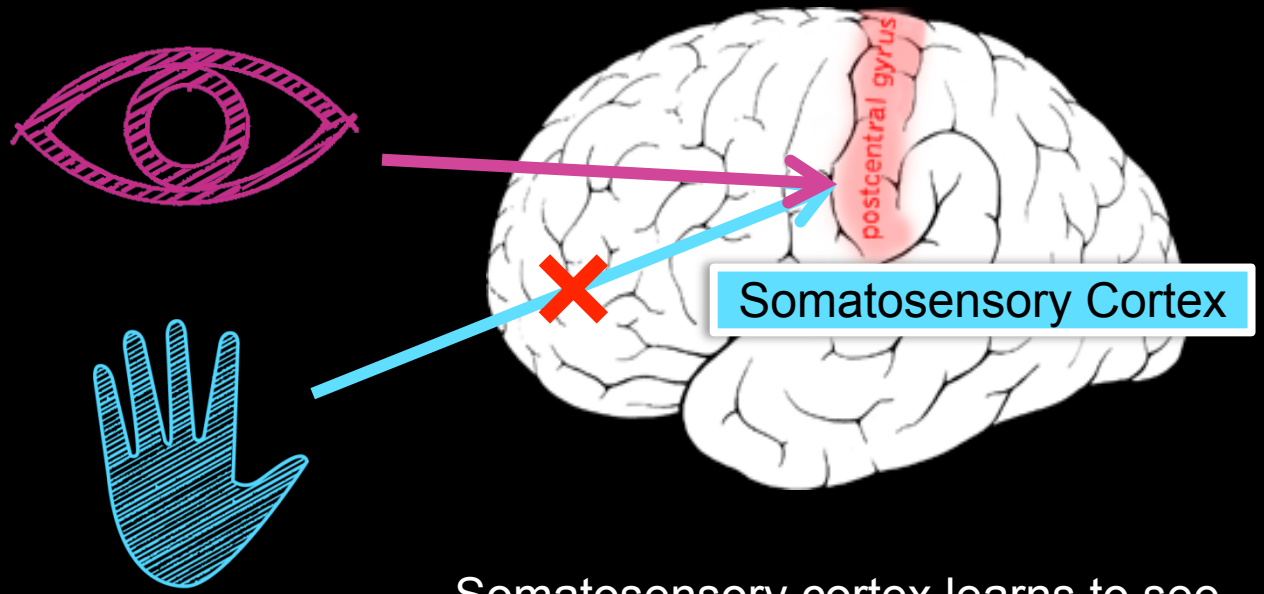


Auditory cortex learns to see

[Roe et al., 1992]

[Andrew Ng]

One Algorithm Hypothesis

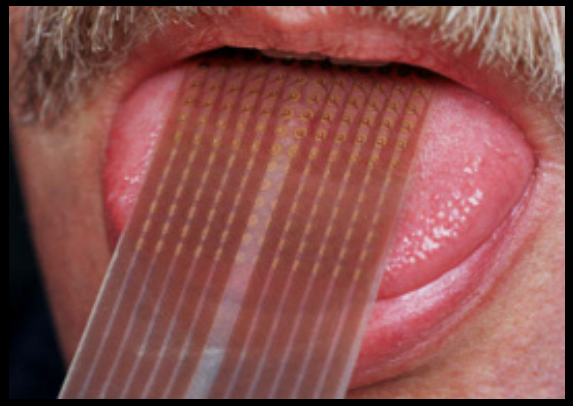


Somatosensory cortex learns to see

[Metin & Frost, 1989]

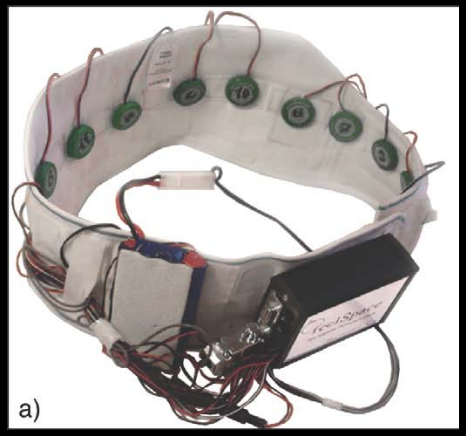
[Andrew Ng]

Sensor Representations

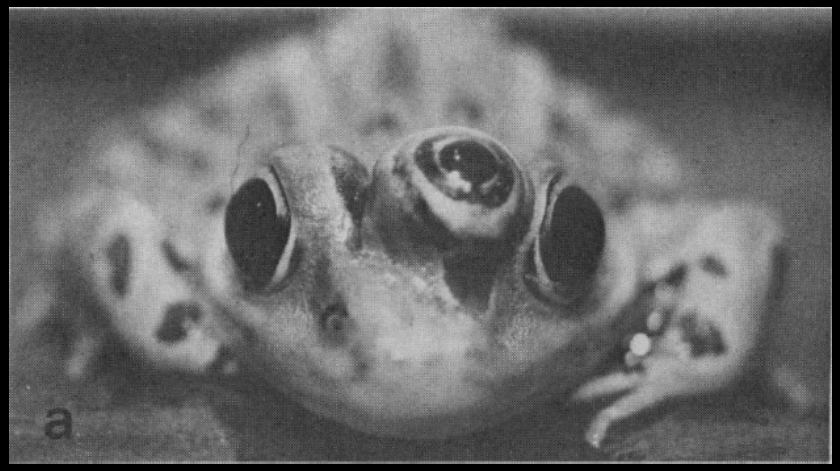


Seeing with your tongue

Human echolocation (sonar)



Haptic belt: Direction sense



Implanting a 3rd eye

[BrainPort; Welsh & Blasch, 1997; Nagel et al., 2005; Constantine-Paton & Law, 2009]

Know It So You Can Beat It

Little math

The End.


The End?