15-112
Fundamentals of Programming

Lecture 1:
Introduction + Basic Building Blocks of Programming

David Kosbie
koz@andrew.cmu.edu

Anil Ada
aada@cs.cmu.edu

January 12, 2016

Why are you here?

To learn programming.

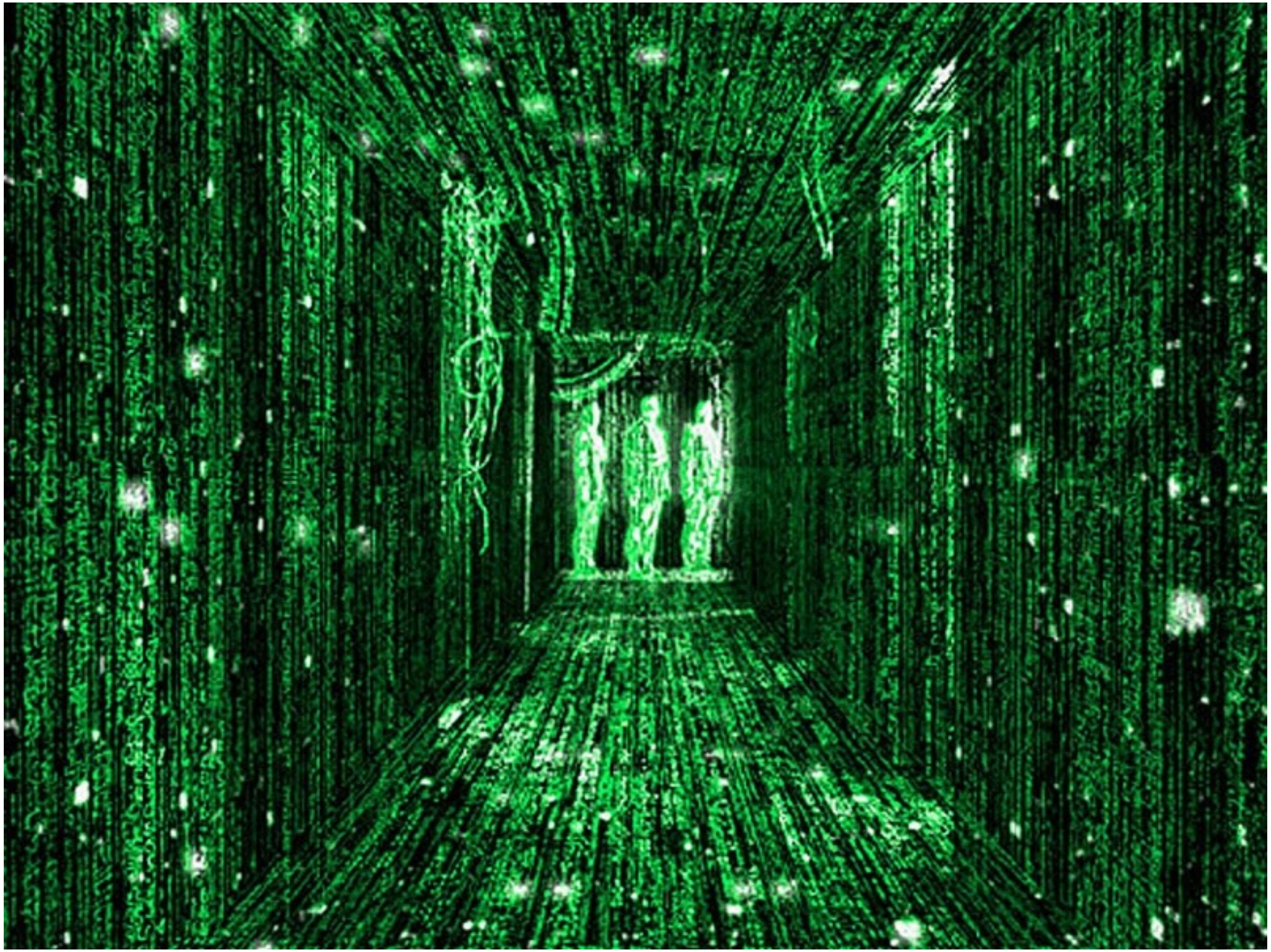
AND to be the next Mark Zuckerberg.

Reality

We actually live in a basement

in our parents' house!

Actually, programming is cool.



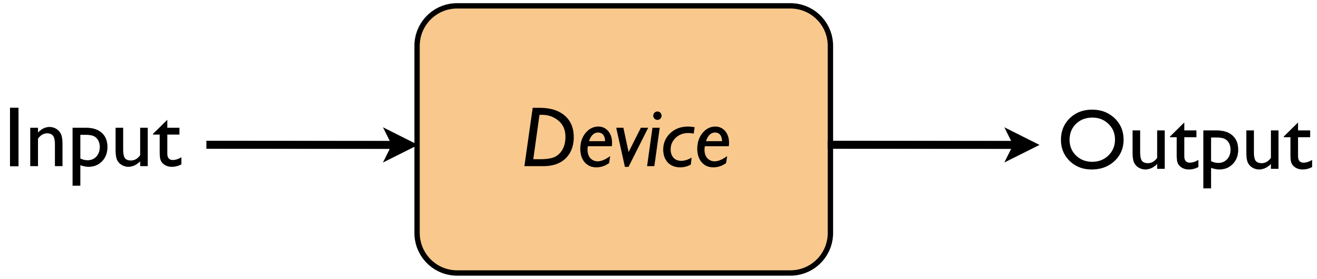
What is programming (coding) ?

What is *computer* programming ?

What is a computer?

Any device that manipulates/processes data (information)

Usually



We call this process computation.

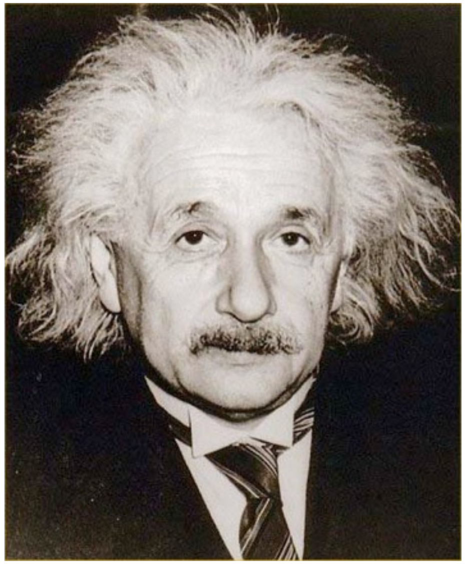
Calculation: processing/manipulating numbers.
(computation restricted to numbers)

Examples

Push
Push

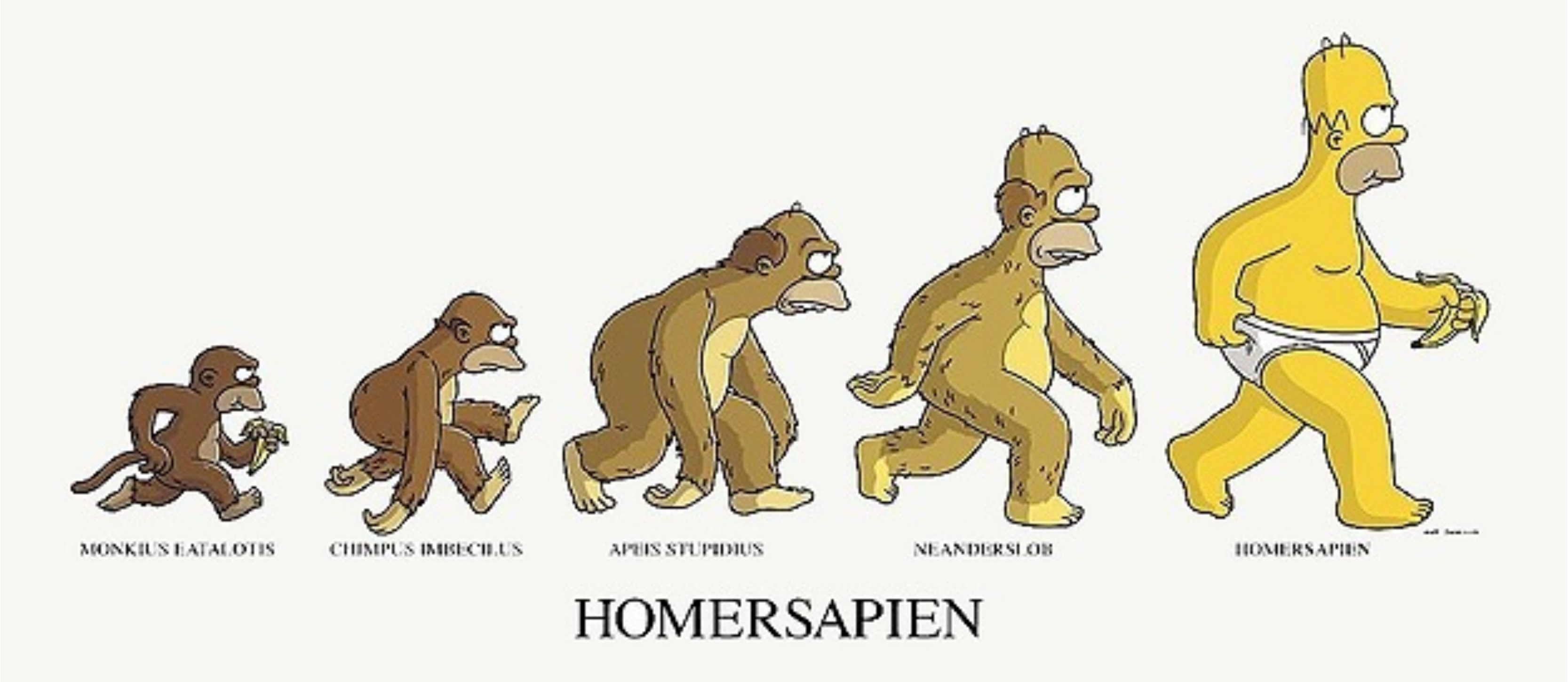
CASHO
TWO WAY POWER
ELECTRONIC CALCULATOR
CS-270
12345678.
8-DIGIT
MC
MR
M−
M+
OFF
ON
CE
C
%
√
×
÷
−
+
=
+/−

disk II
disk II



Examples: Nature (?)

Evolution



The computational lens

Computational physics

Computational biology

Computational chemistry

Computational neuroscience

Computational finance

…

A more refined definition of “computer”

- Restricted to electronic devices

A more refined definition of "computer"

- Restricted to electronic devices

- "Universal"
  programmable to do any task.

A more refined definition of "computer"

- Restricted to electronic devices

- "Universal"
  programmable to do any task.

An electronic device that can be programmed to carry out a set of basic instructions in order to acquire data, process data and produce output.

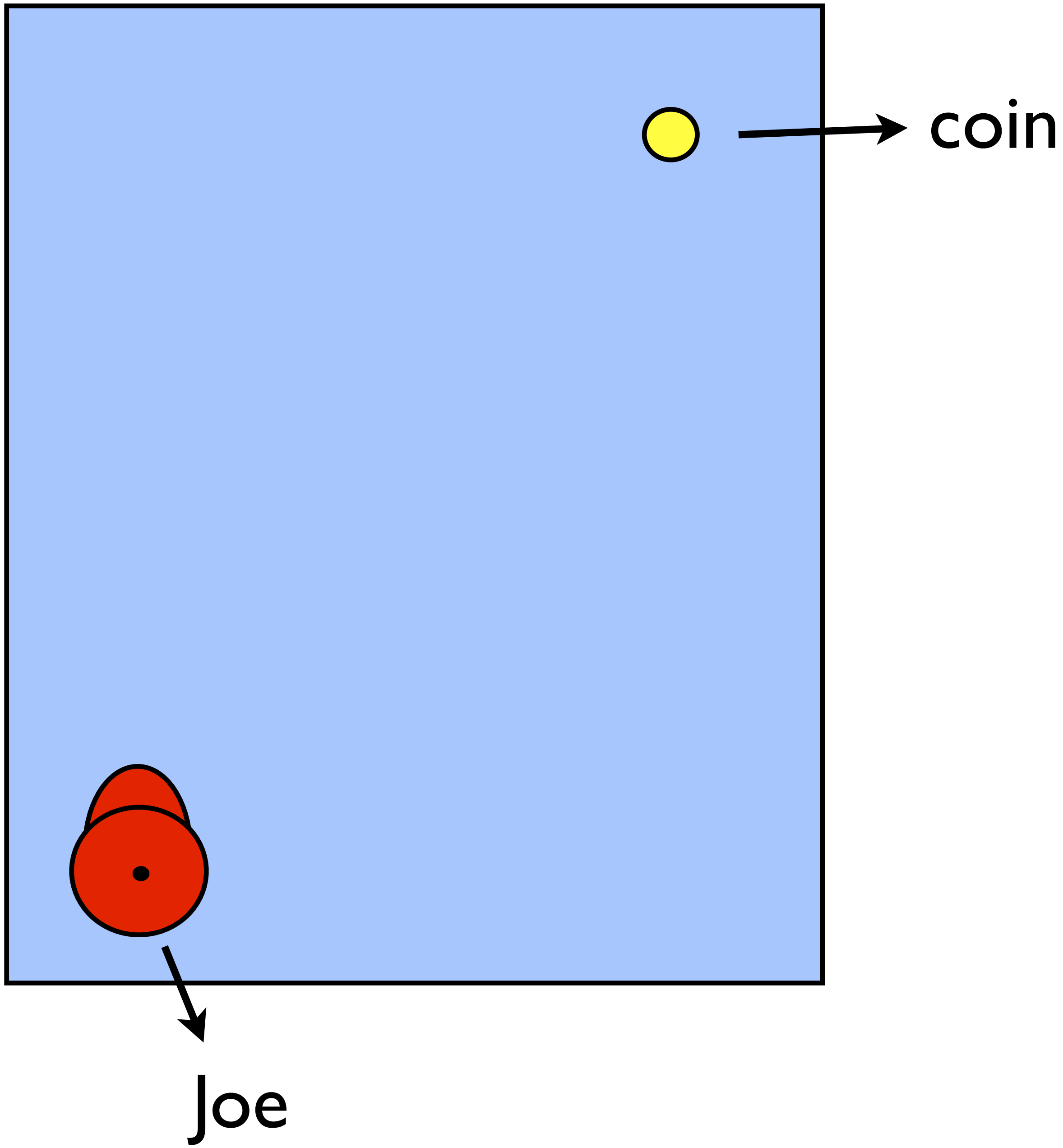
What is a computer program ?

A set of instructions that tells the computer how to manipulate data (information).

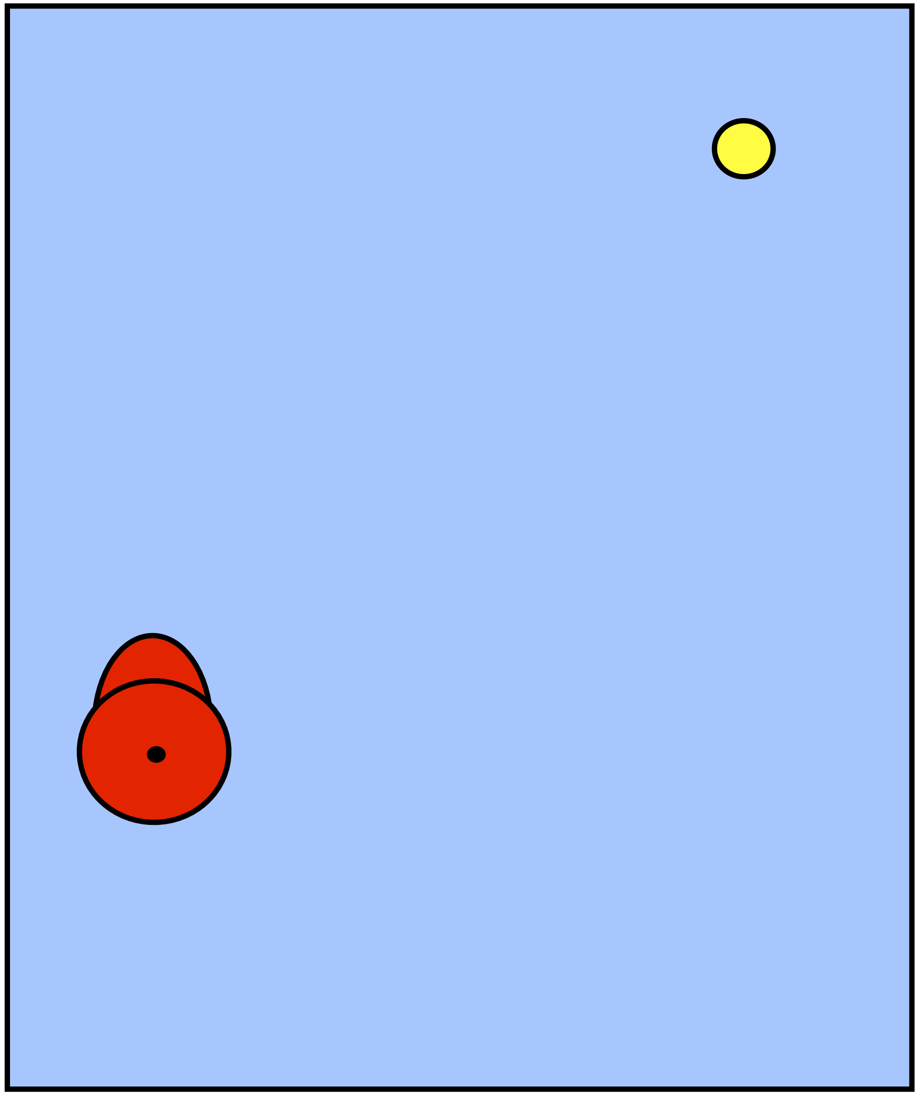
Who is a computer programmer ?

The person who writes the set of instructions.

Example of program: Joe the robot

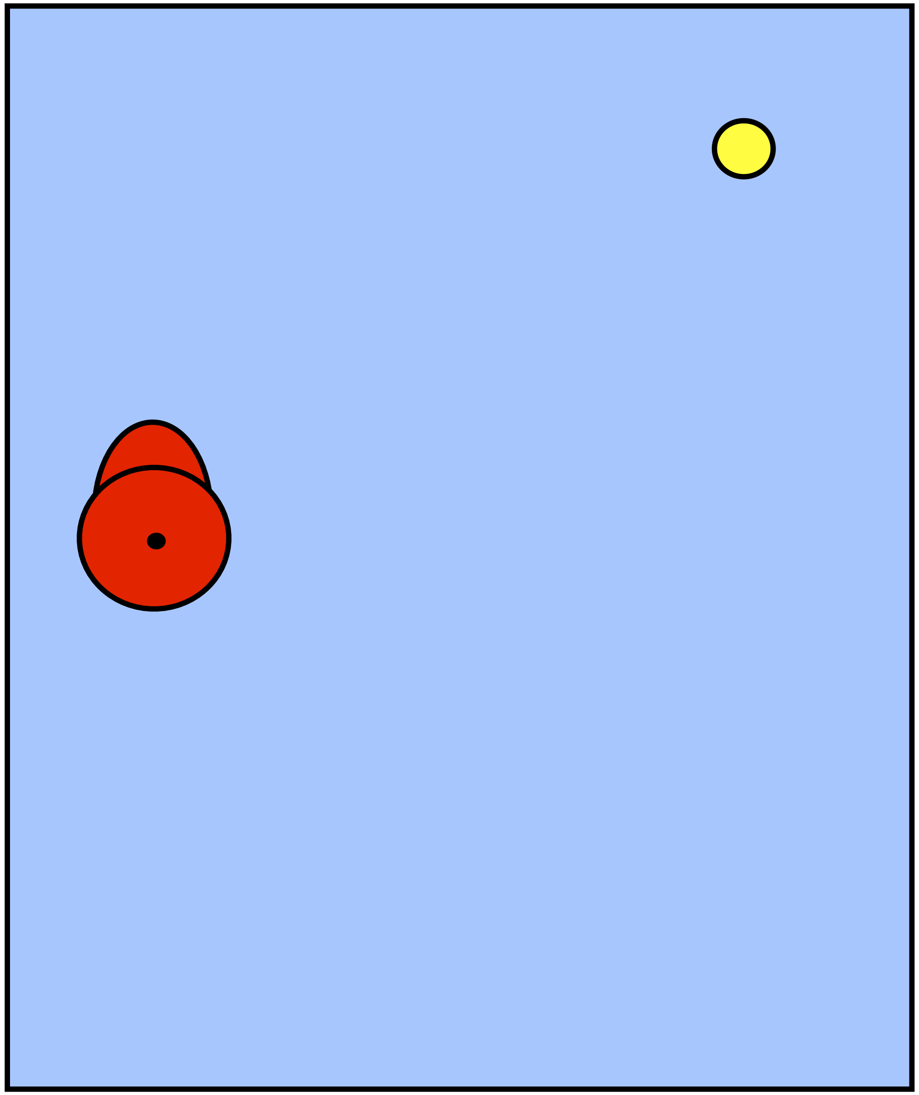


Example of program: Joe the robot



Move 1 step forward

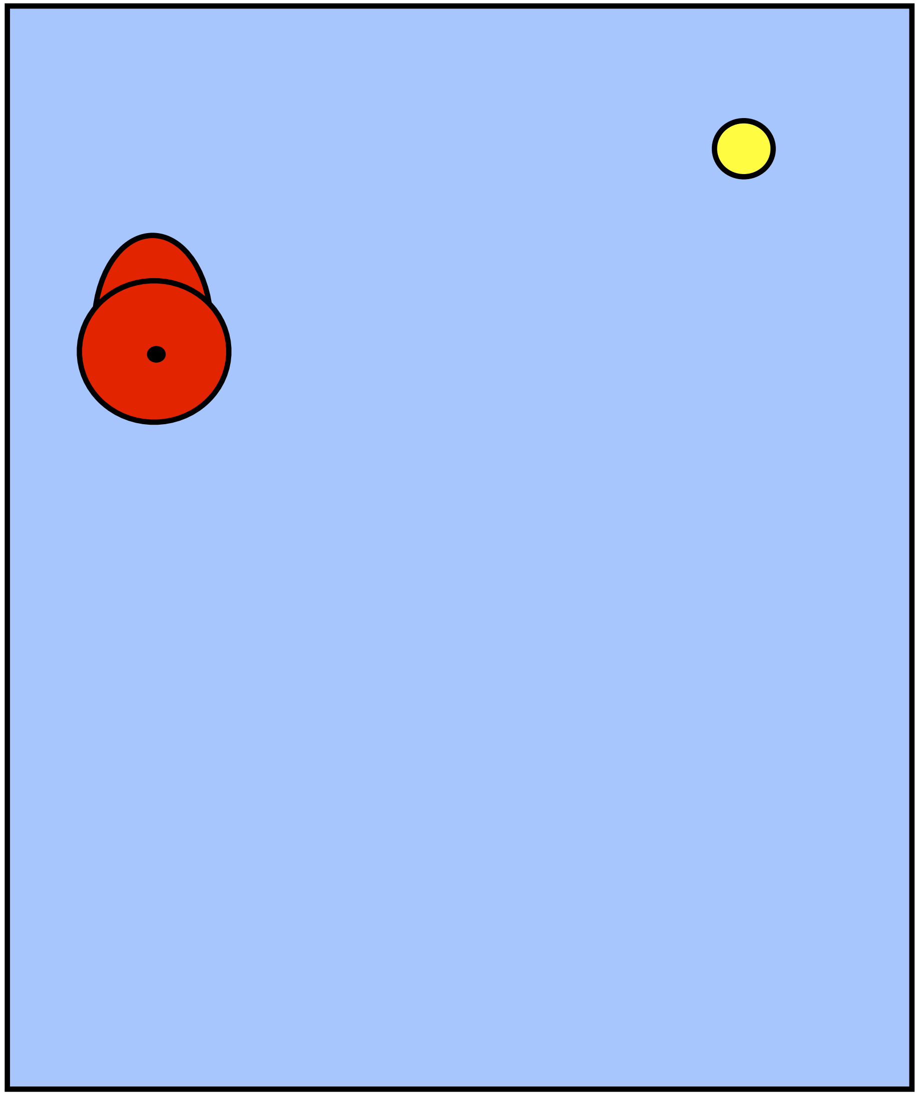
Example of program: Joe the robot



Move 1 step forward

Move 1 step forward

Example of program: Joe the robot

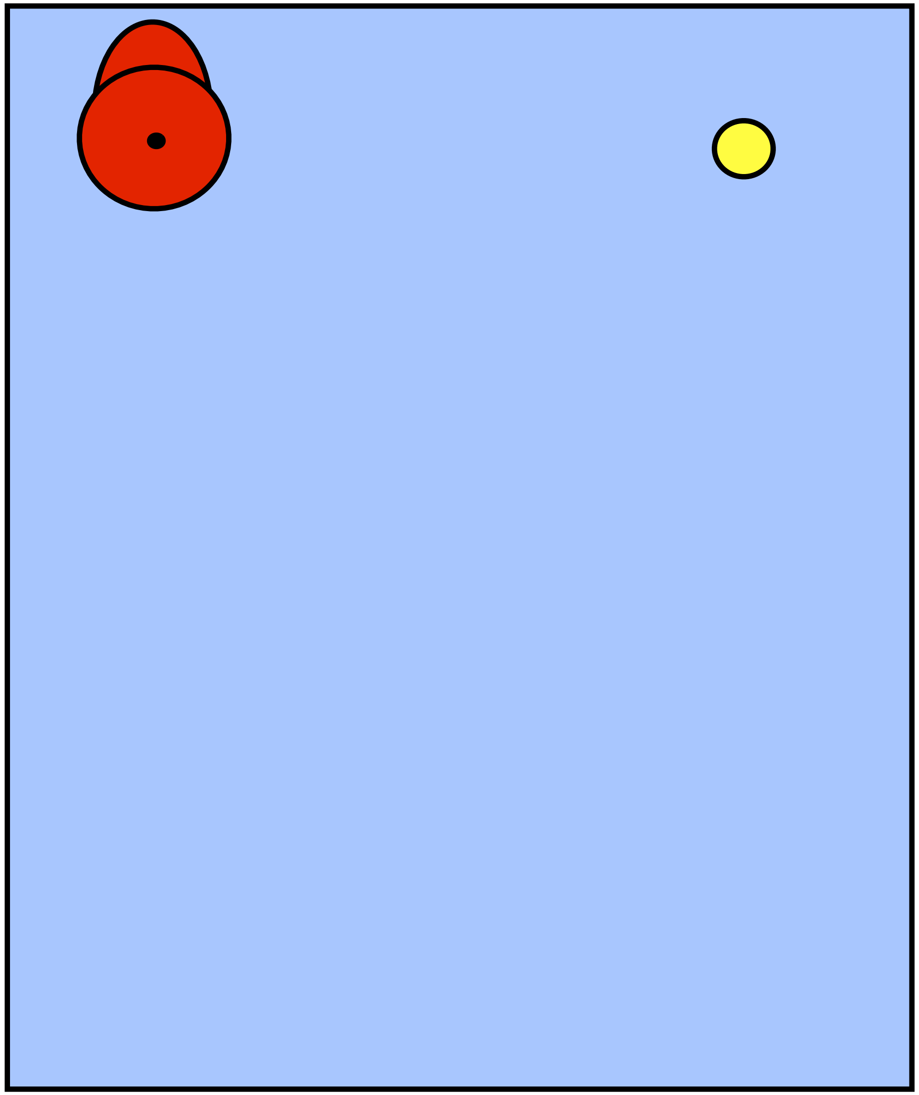


Move 1 step forward

Move 1 step forward

Move 1 step forward

Example of program: Joe the robot



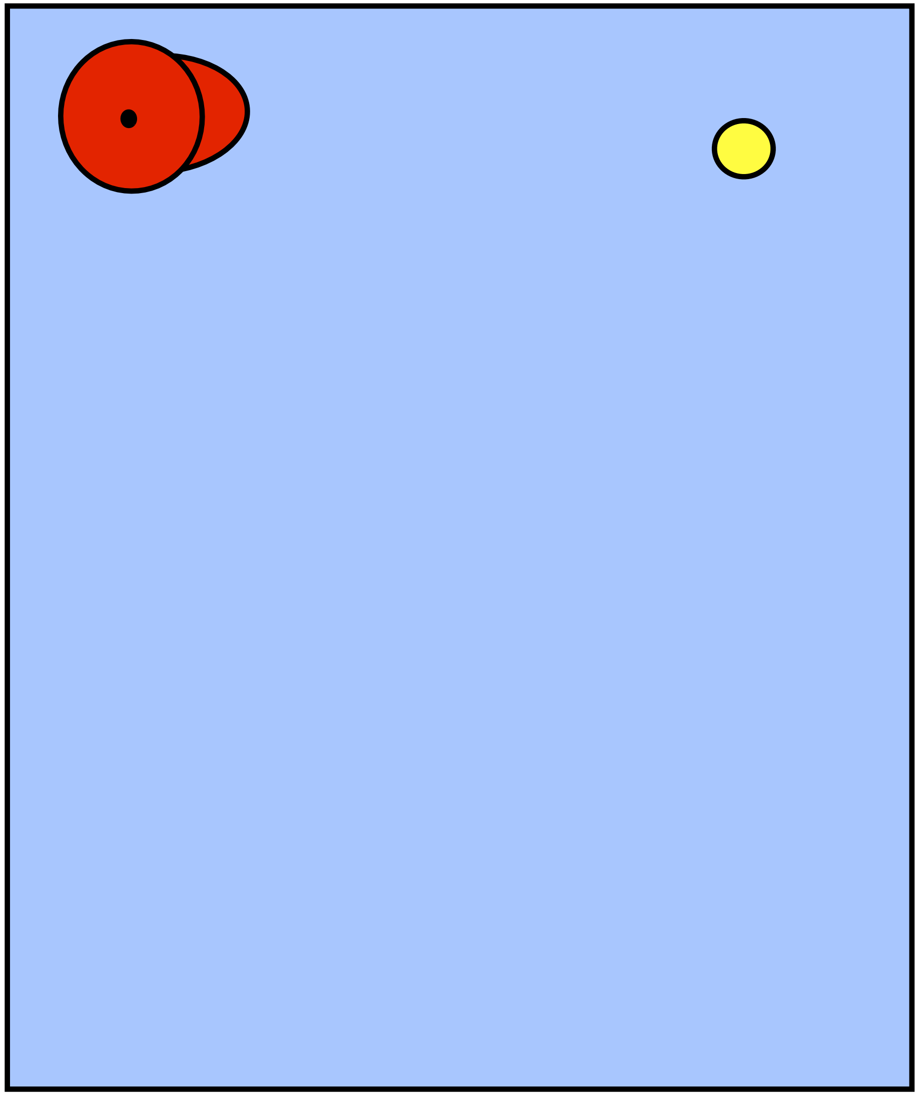
Move 1 step forward

Move 1 step forward

Move 1 step forward

Move 1 step forward

Example of program: Joe the robot



Move 1 step forward

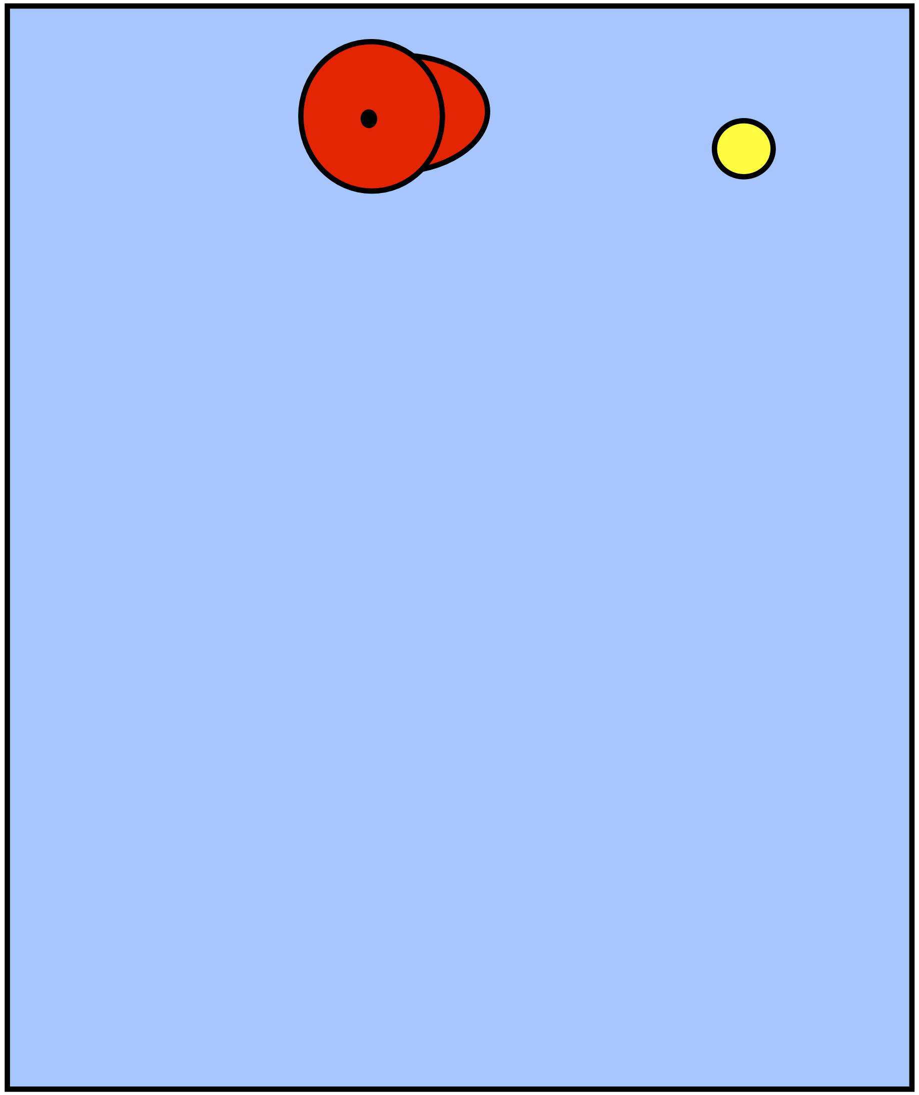
Move 1 step forward

Move 1 step forward

Move 1 step forward

Turn right

Example of program: Joe the robot



Move 1 step forward

Move 1 step forward

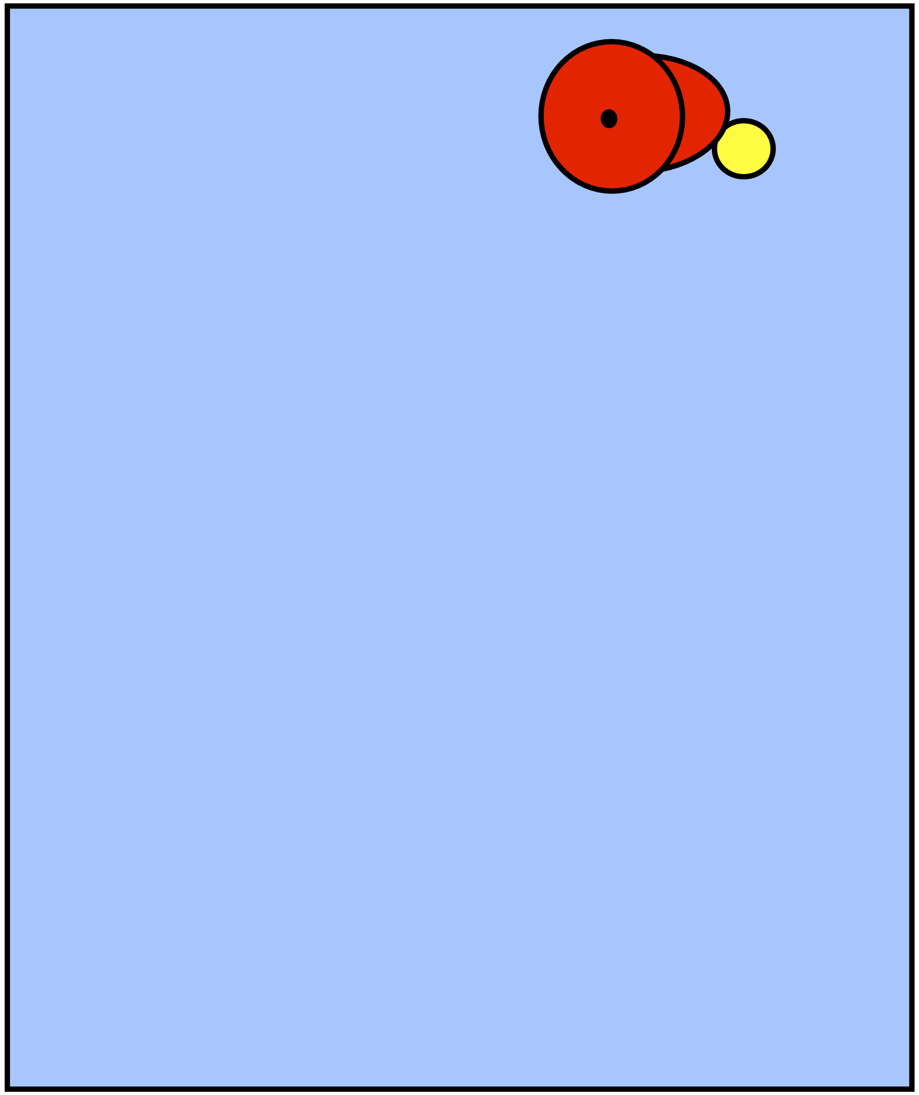
Move 1 step forward

Move 1 step forward

Turn right

Move 1 step forward

Example of program: Joe the robot



Move 1 step forward

Move 1 step forward

Move 1 step forward

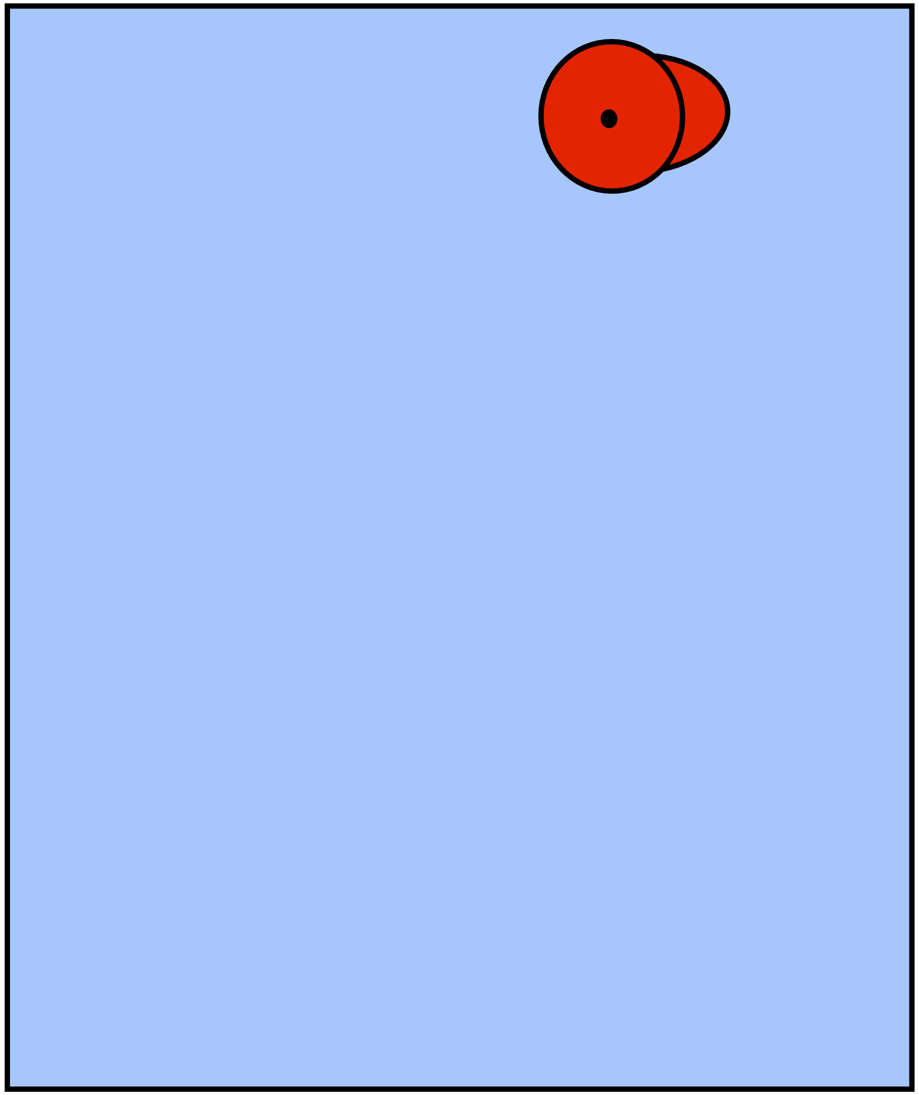
Move 1 step forward

Turn right

Move 1 step forward

Move 1 step forward

Example of program: Joe the robot



Move 1 step forward

Move 1 step forward

Move 1 step forward

Move 1 step forward

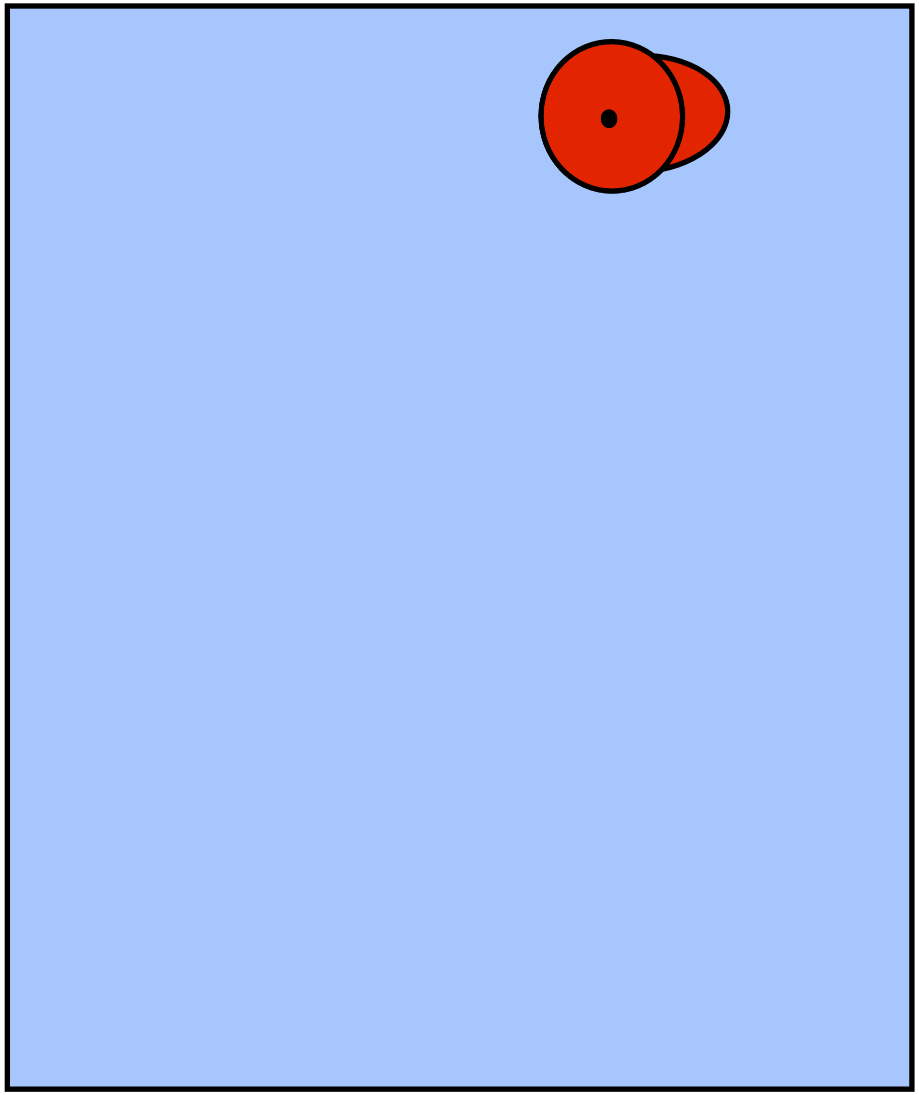
Turn right

Move 1 step forward

Move 1 step forward

Pick up coin

Example of program: Joe the robot



```
Repeat 4 times:
    Move 1 step forward
Turn right
Repeat 2 times:
    Move 1 step forward
Pick up coin
```

Another example: cooking

More appropriate to call this an algorithm.

Melt butter with olive oil.

Add garlic.

Cook until lightly browned.

Stir in green beans.

Season with salt and pepper.

Cook until beans are tender.

Sprinkle with parmesan cheese.

In this course:

This course is about learning to write programs for these:

You will be their master.

Wait a minute!
Are you telling me Angry Birds is just a set of instructions?

Examples of Programs

Operating Systems

Windows
MacOS
Unix

Applications

Internet Explorer
iTunes
Warcraft

Web Sites

Facebook
Twitter
Wikipedia

There are thousands (sometimes millions) of lines of code (instructions) that tell the computer **exactly** what to do and when to do it.

What you will learn in this course:

We will lay the foundations of programming.

1. How to think like a computer scientist.

2. Principals of good programming.

3. Programming language: Python

What you will learn in this course:

1. How to think like a computer scientist.

   Solving problems

   Finding an efficient (preferably most efficient) solution.

What you will learn in this course:

We will lay the foundations of programming.

1. How to think like a computer scientist.

2. Principals of good programming.

3. Programming language: Python

What you will learn in this course:

2. Principals of good programming.

Does your program work correctly?

Is it efficient?

These are not the only important things:

Is your program (code) easy to read? easy to understand?

Can it be reused easily? extended easily?

Is it easy to fix errors (bugs)?

Are there redundancies in the code?

What you will learn in this course:

We will lay the foundations of programming.

1. How to think like a computer scientist.

2. Principals of good programming.

3. Programming language: Python

What you will learn in this course:

3. Programming language: Python

There are many human languages.
Can give instructions in English or Spanish or French, etc.

Similarly, there are many programming languages.

There are a lot of similarities, but also important differences.

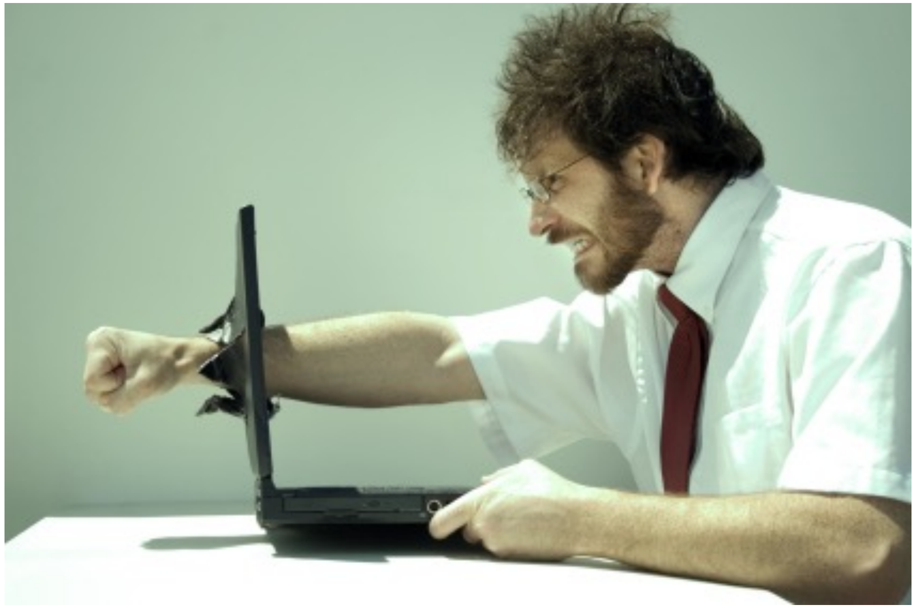
Programming is fun!

Sky is the limit.

It is a creative process.

When your program does what it is supposed to do:

When it doesn't:



The destination

Term Projects

Keys to success in this course

How do you learn programming? **By doing.**

Understand the method: learning by immersion.

Understand the challenge. Embrace the challenge.

Time management!

Help us help you.

Ask questions in class, in office hours, on Piazza.

You will learn the most from your CAs. Use them.

Keys to success in this course

Most importantly:

Have fun!

A typical week

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		🟦				

Lecture:

Cover main content for the week

A typical week

Sun	Mon	Tue	Wed	Thu	Fri	Sat
			🟦			

Recitation:

Go over practice problems.

A typical week

Sun	Mon	Tue	Wed	Thu	Fri	Sat
				🟦		

Lecture:

Go over more problems/examples.

A typical week

Sun	Mon	Tue	Wed	Thu	Fri	Sat
					🟦	

“Recitation”:

Quiz: a subset of the problems seen on Wed

+

questions related to previous week’s material.

Start working on the homework.

A typical week

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						🟦

Work on the homework.

A typical week

Sun	Mon	Tue	Wed	Thu	Fri	Sat
🟦						

Homework due at 10pm.

This week

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		🟦				

Lecture:

We don’t have time to cover everything in lecture.

Go through the notes yourself.

Course Webpage

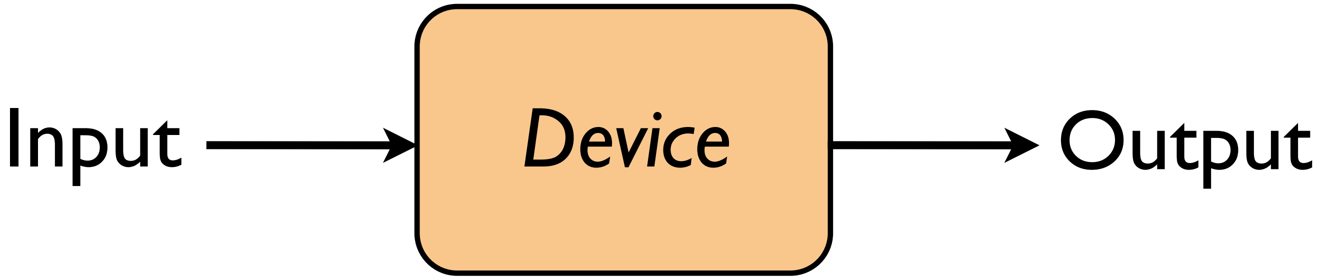
www.cs.cmu.edu/~112

What we know so far:

What is a computer?

A programmable device that manipulates data/information

Usually



What is a computer program ?

A set of instructions that tells the computer how to manipulate data/information.

This Lecture (and Next (and Next...))

How do these instructions look like?
(What kind of instructions are allowed?)

How can I use these instructions to write programs?
(How do I approach programming, where do I start?)

Basic Building Blocks

Statements
Tells the computer to do something.

Data Types
Data is divided into different types.

Variables
Allows you to store data and access stored data.

Operators
Allows you to manipulate data.

Conditional Statements
Executes statements if a condition is satisfied.

Functions
Mini self-contained programs.

Basic Building Blocks

Statements

Tells the computer to do something.

BASICS OF HANDLING DATA

Data Types

Data is divided into different types.

Variables

Allows you to store data and access stored data.

Operators

Allows you to manipulate data.

Conditional Statements

Executes statements if a condition is satisfied.

Functions

Mini self-contained programs.

Basics of handling data

Printing values on the screen

```
print(“Hello World”)
```

```
Hello World
```

```
print(911)
```

```
911
```

```
print(1, 2, 3)
```

```
1 2 3
```

```
print(3.14, “is not an integer”)
```

```
3.14 is not an integer.
```

Basics of handling data

Data/value types

```
print(“Hello World”)
```

→ string

```
Hello World
```

```
print(911)
```

→ integer

```
911
```

```
print(1, 2, 3)
```

```
1 2 3
```

```
print(3.14, “is not an integer”)
```

→ float

```
3.14 is not an integer.
```

Basics of handling data

Variables and Assignment Statements

variable-name = value

```
x = 5
y = “Hello World”
print(x)
print(y)
```

```
x <-- 5
y <-- “Hello World”
```

```
x = 5
y = x
x = 0
print(y)
```

In an **assignment statement:**

1. Evaluate RHS.
2. Assign the value to the variable.

Basics of handling data

Operators

x = 3 + 5	x stores 8
print(“Hello” + “ World”)	Hello World
print(1.5 + 1.5)	3.0
x = 2 * x + 1	x stores 17
x = “Hi!” * 2	x stores “Hi!Hi!”

Expression: - a valid combination of data and operators

- evaluates to a value

Expressions are evaluated first.

Basics of handling data

Data type conversion

```
x = 251 + “ is a prime number.”

print(x)
```

Error: cannot add an integer and a string

```
x = str(251) + “ is a prime number.”

print(x)                                   251 is a prime number
```

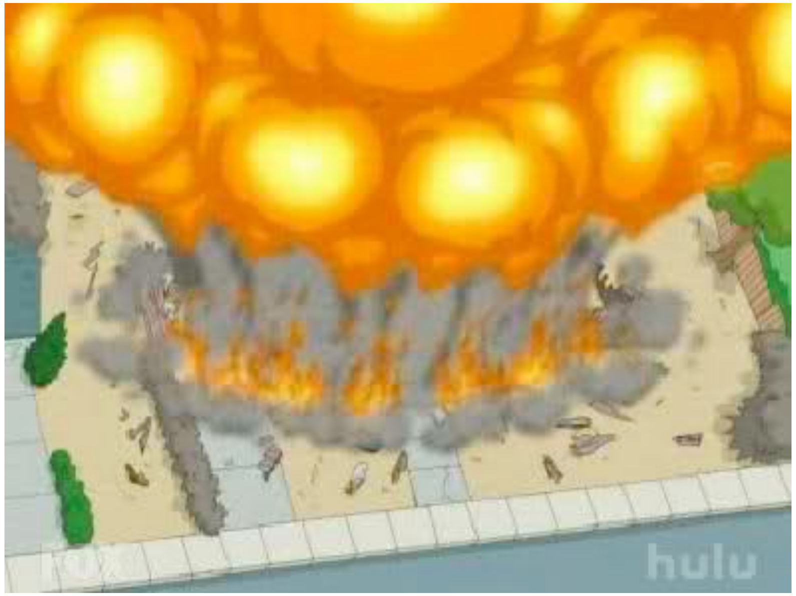
```
print(“123” + “456”)                             123456

print(int(“123”) + int(“456”))                   579
```

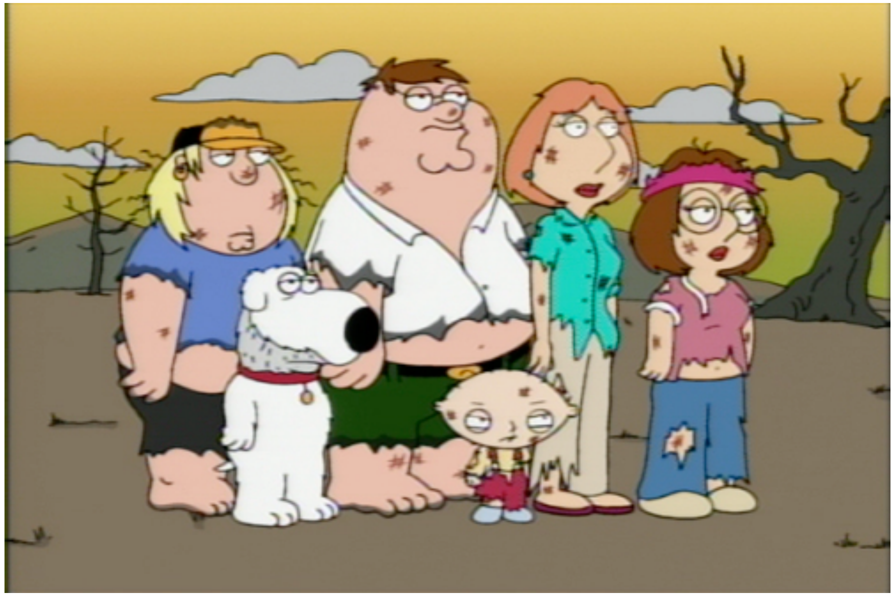
A useful Python program

Help Peter!

His cookbook is in Fahrenheit, but his stove is in Celsius.

Last time:





Fahrenheit to Celsius Converter

The formula: $C = (F - 32) * (5 / 9)$

```
x = input(“Enter degrees in Fahrenheit: ”)   here x will be a string
x = float(x)
y = (x - 32) * (5/9)
print(y)
```

Careful: Easy to make errors!

Try to modify the examples:

- Misspell some of the words.
- Write in upper case.
- Put two statements on one line.
- Divide one statement over two lines.
- ...

Try to run and see what kind of errors you get.

Types of Programming Errors (Bugs)

3 types

Compile-time errors:

The compiler finds problems with syntax or other basic issues. e.g. typed “Print” rather than “print”

Run-time errors:

A problem occurs during program execution, and causes the program to terminate abnormally (*crash*).

e.g. division by 0.

Logical errors:

The program runs, but produces incorrect results.

e.g. maybe in your program you used a wrong formula:

```
celsius = (5 / 9) * fahrenheit - 32
```

A more sophisticated example

```
print(“1. Convert Celsius to Fahrenheit”)
print(“2. Convert Fahrenheit to Celsius”)

option = int(input(“Enter 1 or 2: ”))
degrees = float(input(“Enter degrees: ”))

if(option == 1):
    print(degrees * (9/5) + 32)
else:
    print((degrees - 32) * (5/9))
```

A more sophisticated example

```
print(“1. Convert Celsius to Fahrenheit”)
print(“2. Convert Fahrenheit to Celsius”)

option = int(input(“Enter 1 or 2: ”))
degrees = float(input(“Enter degrees: ”))

if(option == 1):
    print(degrees * (9/5) + 32)
else:
    print((degrees - 32) * (5/9))
```

Evaluates to **True** or **False**

So **True** and **False** are values.

What is their type?

Boolean

Data Types

Python name	Description	Values
NoneType	absence of value	None
bool (boolean)	Boolean values	True, False
int (integer)	integer values	-2^{63} to $2^{63}-1$
long	large integer values	all integers
float	fractional values	e.g. 3.14
complex	complex values	e.g. 1+5j
str (string)	text	e.g. "Hello World!"

•••

Basic Building Blocks

Statements
Tells the computer to do something.

Data Types
Data is divided into different types.

Variables
Allows you to store data and access stored data.

Operators
Allows you to manipulate data.

BASICS OF HANDLING DATA

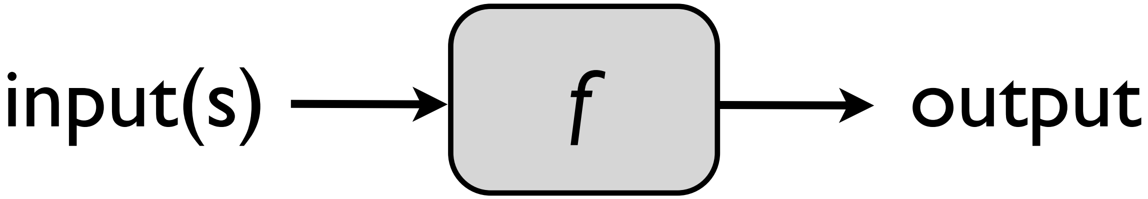
Conditional Statements
Executes statements if a condition is satisfied.

Functions
Mini self-contained programs.

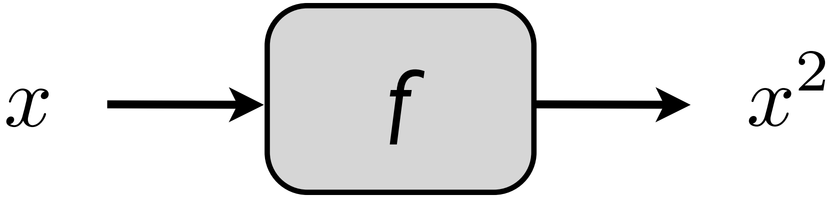
Functions

Functions in math

A function in math:



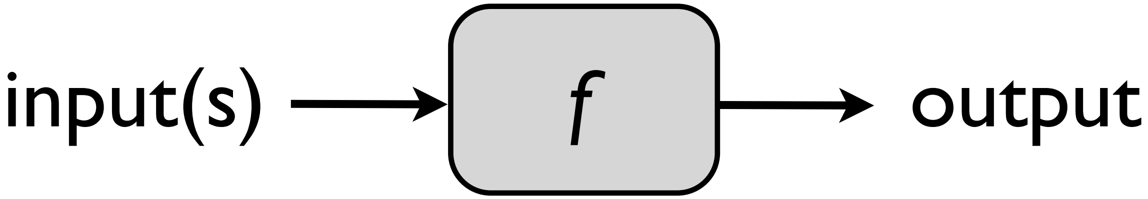
$$f(x) = x^2$$



$f(2) + f(5)$ evaluates to 29

Functions in math

A function in math:

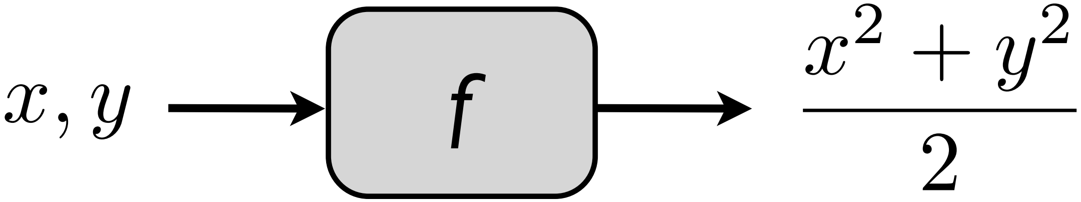


input(s)

f

output

$$f(x, y) = \frac{x^2 + y^2}{2}$$



x, y

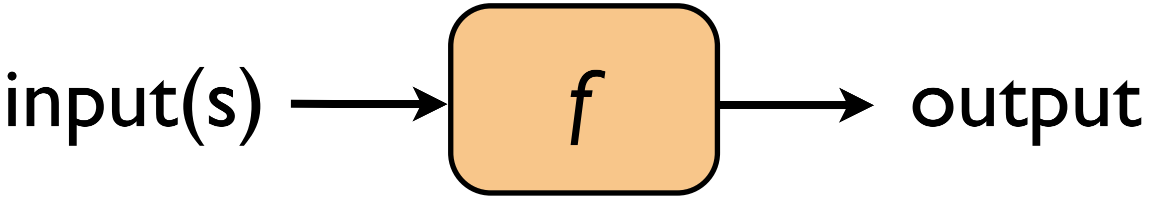
f

$$\frac{x^2 + y^2}{2}$$

$f(2, 4) + 5$ evaluates to 15

Functions in Python

A function in Python:



But now, inputs and output can be any type of data.

Functions in Python

```
def square(x):
    y = x*x
    return y
```

function definition

```
print(square(5))
```

Functions in Python

```
def square(x):
    y = x*x
    return y

print(square(5))
```

function body (must be indented)

Functions in Python

```
def square(x):  parameter
    y = x*x
    return y

print(square(5))
```

Functions in Python

```
def square(x):
    y = x*x
    return y

print(square(5))
```

function call

Functions in Python

```
def square(x):
    y = x*x
    return y

print(square(5))
```

argument

Functions in Python

```
def square(x):
    y = x*x
    return y

print(square(5))
```

```
def square(x):
    return x*x

print(square(5))
```

```
def square(x):
    return x**2

print(square(5))
```

```
def f(x, y):
    return (square(x) + square(y))/2

print(f(2, 3))
```

Functions in Python

```
def greetUser(name):
    print(“Hello”, name)
```

```
greetUser(“David”)
```

```
Hello David
```

Does this function return anything?

It actually returns None.

```
print(greetUser(“David”))
```

```
Hello David
None
```

Functions in Python

```
def greetEveryone():
    print(“Hello everyone!”)
```

```
greetEveryone()                          Hello everyone!

greetEveryone(“David”)                   ERROR
```

```
def isPositive(x):
    return (x > 0)
```

```
print(isPositive(-1))                    False
```

Functions in Python

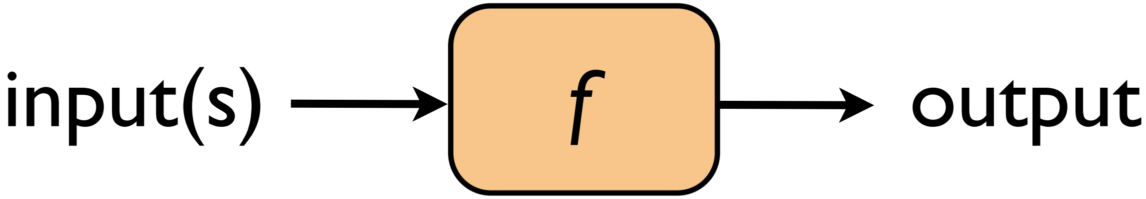
```
def isPositive(x):
    print(“Hello.”)
    return (x > 0)
    print(“Bye.”)

print(isPositive(-1))
```

```
Hello.
False
```

```
def celsiusToFahrenheit(degrees):
    return (degrees - 32) * (5 / 9)

def fahrenheitToCelsius(degrees):
    return degrees * (9 / 5) + 32
```

Functions in Python

A function in Python:



input(s)

f

output

In fact,

Python program = a function + other “helper” functions

```
def celsiusToFahrenheit(degrees):
    return (degrees - 32) * (5 / 9)

def fahrenheitToCelsius(degrees):
    return degrees * (9 / 5) + 32
```

Never define variables outside of a function!!!

Functions in Python

Example problem:

Write a function that takes 2 integers as input and returns the maximum of the ones digit of the numbers.

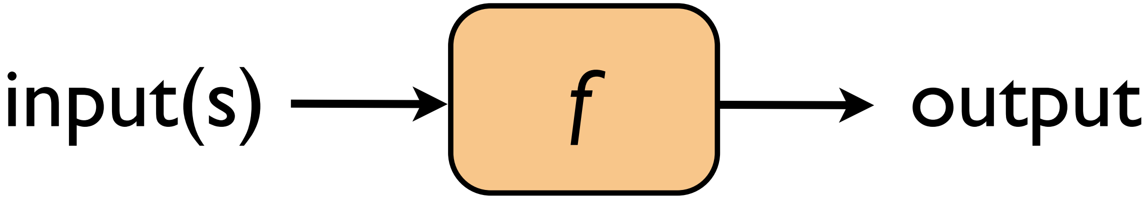
```
def max(x, y):
    if(x >= y):
        return x
    else:
        return y

def onesDigit(x):
    return x%10

def largerOnesDigit(x, y):
    return max(onesDigit(x), onesDigit(y))
```

helper functions

the remainder when x is divided by 10

Functions in Python

A function in Python:



input(s)

f

output

In fact,

Python program = a function + other “helper” functions

We don’t want to worry about how the input is acquired or how the output will be used.

We only use print( ) for debugging purposes.
We almost never use input( ).

Built-in functions

```
print(abs(-5))
print(max(2, 3))
print(min(2, 3))
print(pow(2, 3))            raise to the given power (pow(x,y) == x**y)
print(round(2.354, 1))     round with the given number of digits


print(type(5))
print(type(“hello”))
print(type(True))


print(int(2.8))


import math
print(math.factorial(10))
print(math.pi)
```

Python fun fact

```
x = 0.1 + 0.1 + 0.1        x actually stores 0.30000000000000004
y = 0.3
print(x == y)              prints False
```

```
def almostEqual(x, y):
   epsilon = 10**(-10)
   return (abs(x - y) < epsilon)
```