SAMS
Programming - Section C

Lecture 1:
Introduction + Basic Building Blocks of Programming

Anil Ada
aada@cs.cmu.edu

July 3, 2017

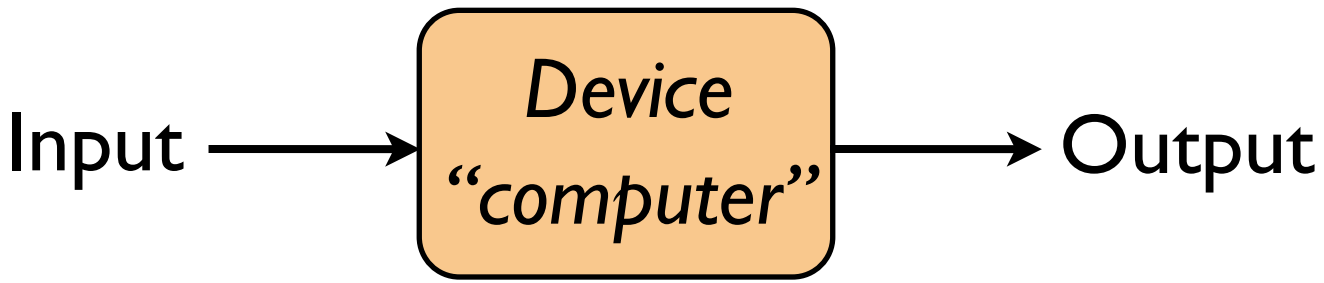
What is programming (coding) ?

What is *computer* programming ?

What is a computer ?

What is a computer?

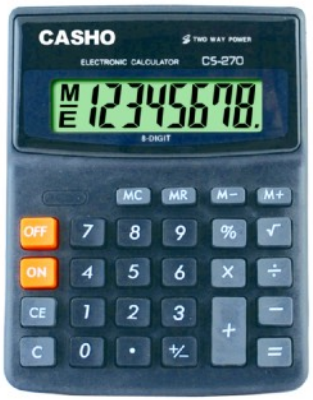
Any device that manipulates/processes data (information)



We call this process **computation**.

calculation: manipulation of numbers.
(i.e., computation restricted to numbers)

Examples

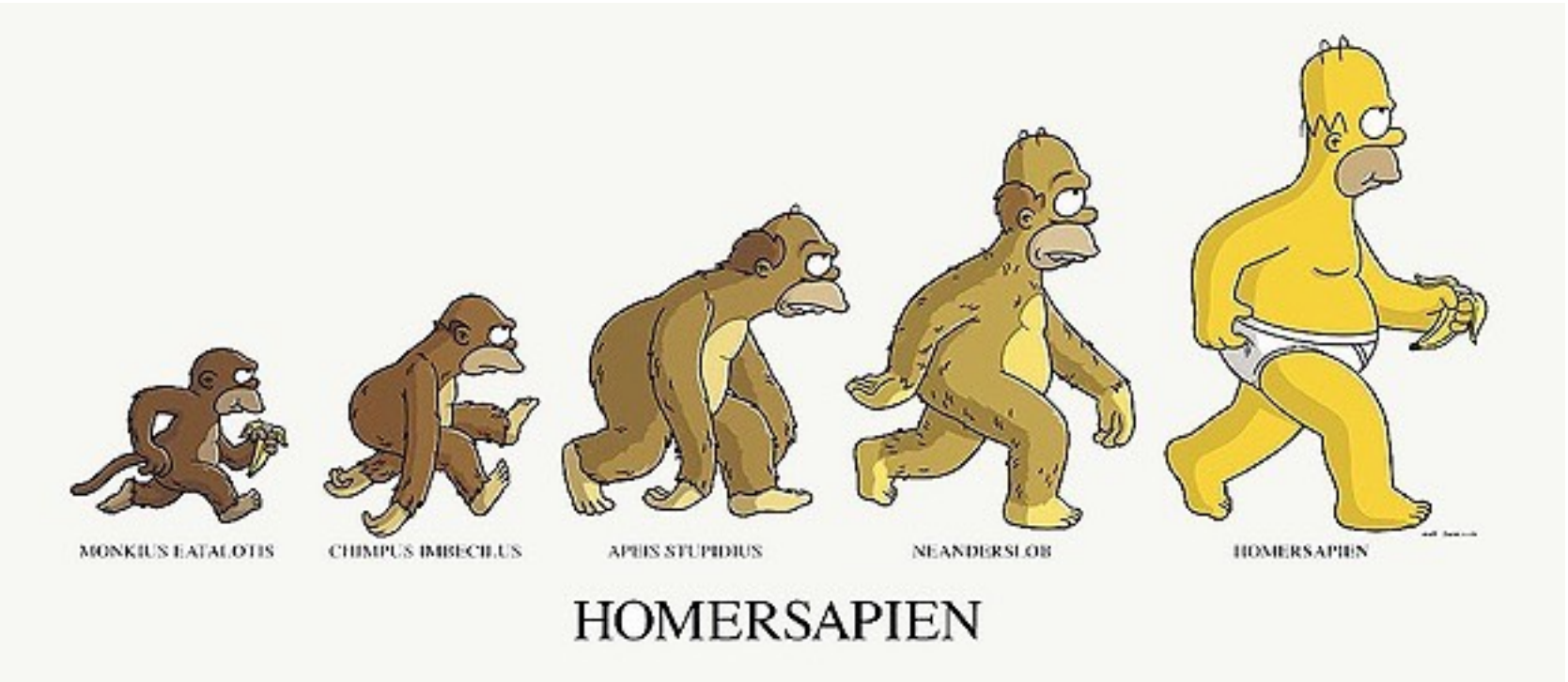

CASHO
ELECTRONIC CALCULATOR
CS-270
MC
MR
M−
M+
OFF
ON
CE
C


disk II
disk II

“Computers” in early 20th century

Examples: Nature (?)

Evolution



Computer Science:
The science that studies *computation*.

The computational lens

Computational physics

Computational biology

Computational chemistry

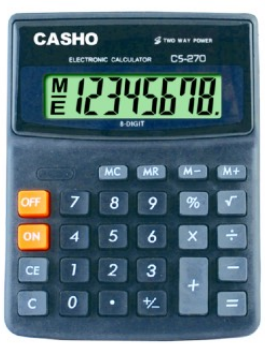
Computational neuroscience

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.

Computer:

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*.

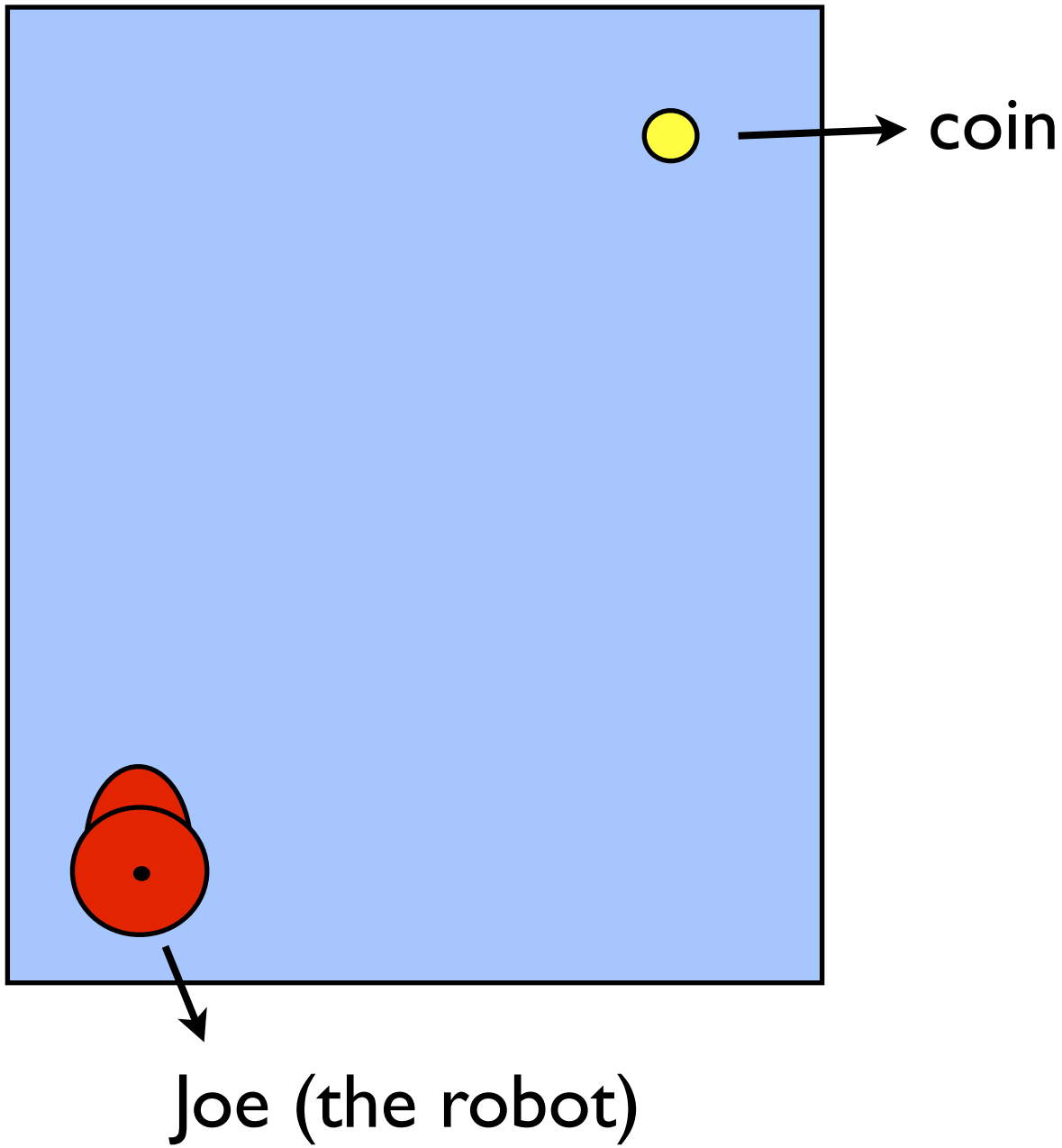
What is a computer program ?

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

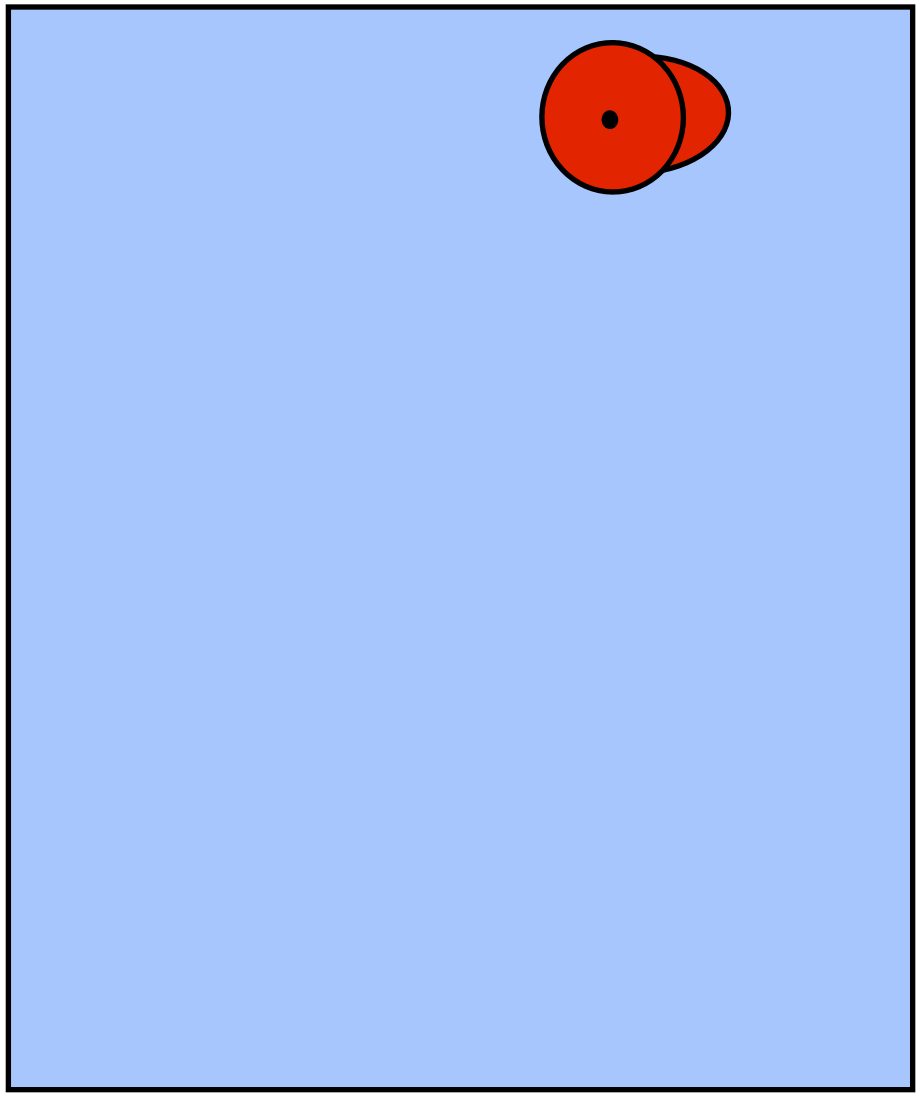
Who is a computer programmer ?

The person who writes the set of instructions.

Example of a program



Example of a program



Move 1 step forward

Move 1 step forward

Move 1 step forward

Move 1 step forward

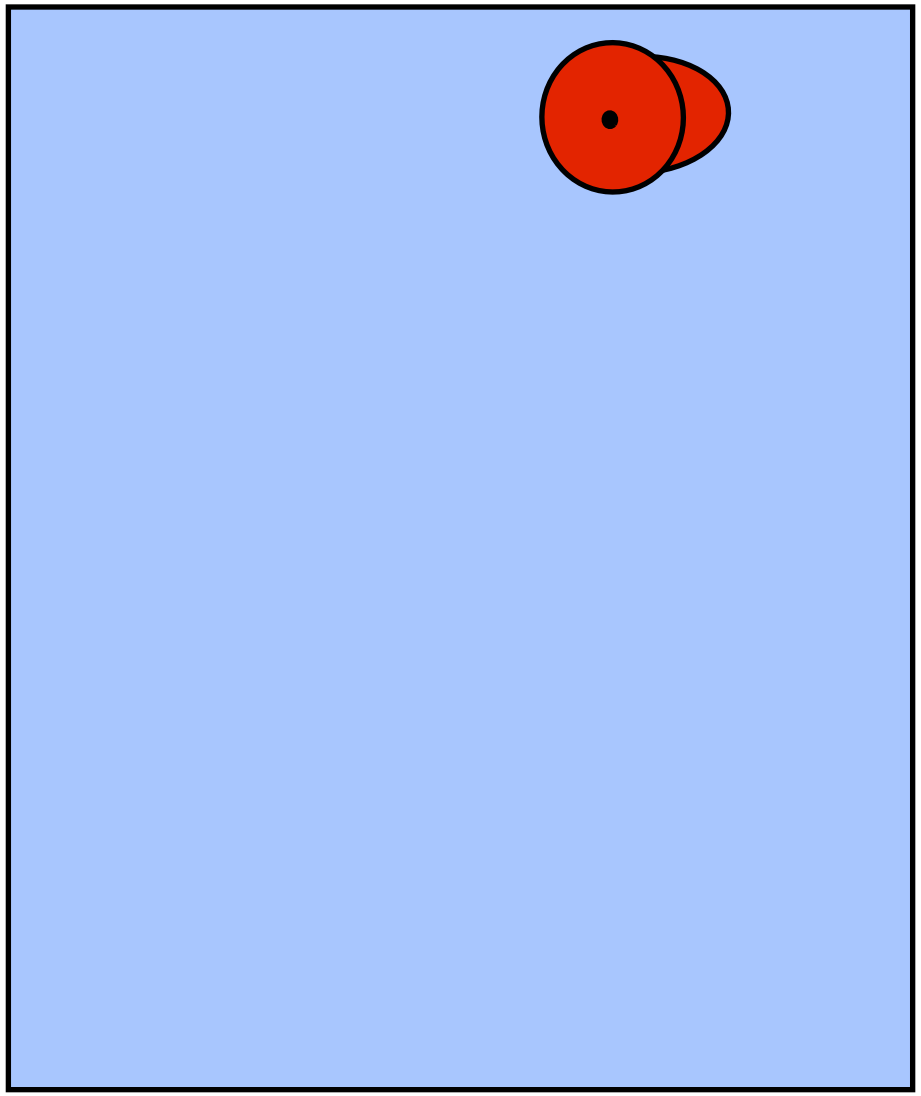
Turn right

Move 1 step forward

Move 1 step forward

Pick up coin

Example of a program



Repeat 4 times:

    Move 1 step forward

Turn right

Repeat 2 times:

    Move 1 step forward

Pick up coin

Another example: a recipe

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.

More appropriate to call this an **algorithm**.

In this course

Learn to write programs for:




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

Examples of Programs

Operating Systems	Applications	Web Sites
Windows	*Internet Explorer*	*Facebook*
MacOS	*iTunes*	*Twitter*
Unix	*Warcraft*	*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

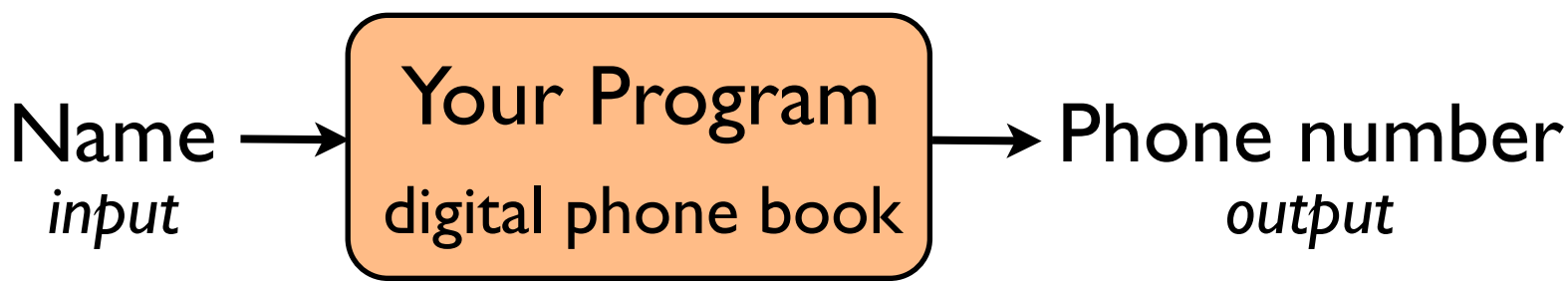
1. How to think like a computer scientist.

Solving problems.

- use instructions a machine can understand.
- divide the problem into smaller manageable parts.

Finding an efficient (preferably most efficient) solution.

EXAMPLE:



- *How do you solve it using instructions the computer can understand? (Can't just say "find phone number")*

- *How do you solve the problem efficiently?*

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.

Most important properties of a program:

- Does your program work correctly?
- Is it efficient?

Other 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)?

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.
e.g. English, Spanish, French, Japanese, etc.

Similarly, there are many **programming** languages.

- Mix of math and English.
- Lots of similarities between different languages, but also important differences.

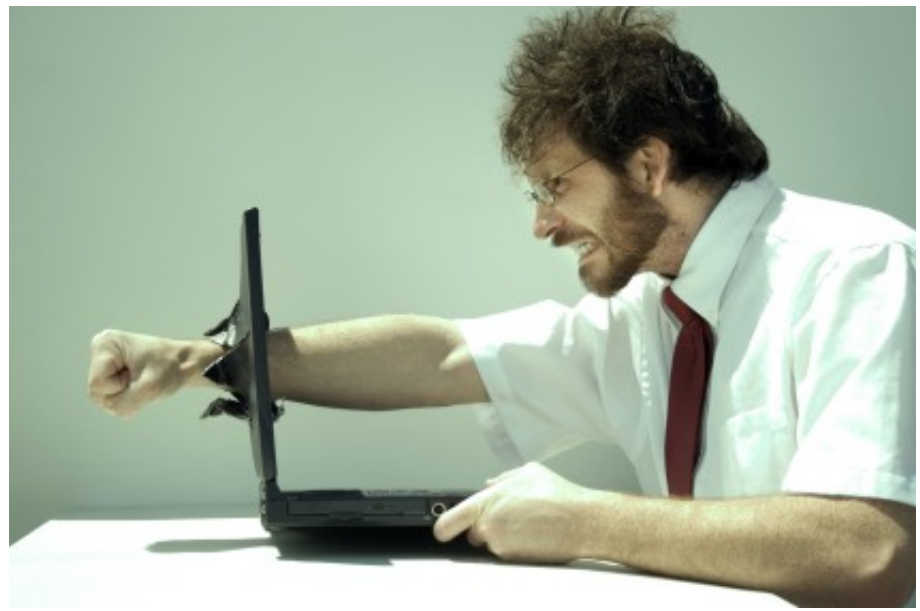
Programming is Awesome!

Sky is the limit.

Combines technical skill and creativity.

When your program does what it is supposed to do:

When it doesn't:



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.

Get to know your TAs. They are awesome.

Keys to success in this course

Most importantly:

Have fun!

Course Webpage

http://www.cs.cmu.edu/~aada/courses/SAMS17/

Video

http://www.youtube.com/watch?v=nK1u9yen5nc

Let’s start.

How do you create and run Python programs?

1. Install Python: *www.python.org/download*
   (version 3.6.x)

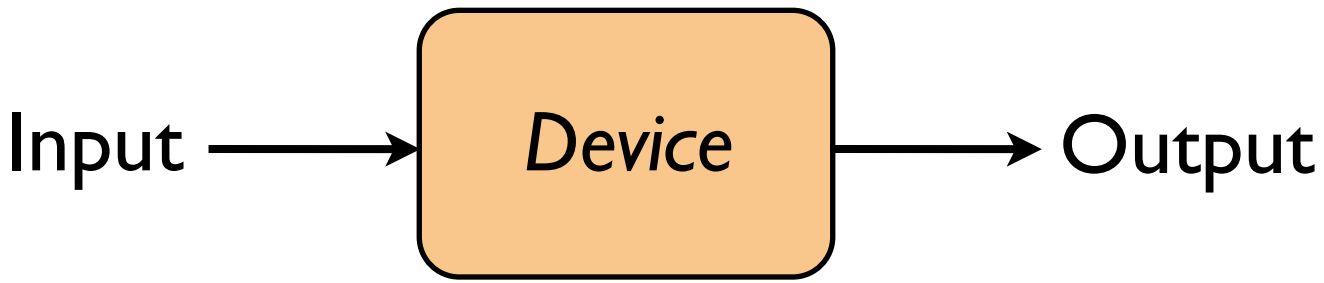
2. To type your code and run it, you need an IDE:

   e.g. Pyzo, Sublime, IDLE

What we know so far:

What is a computer?

A programmable device that manipulates data/information



What is a computer program?

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

This Lecture (and next, and next, and next…)

How do these instructions look like?

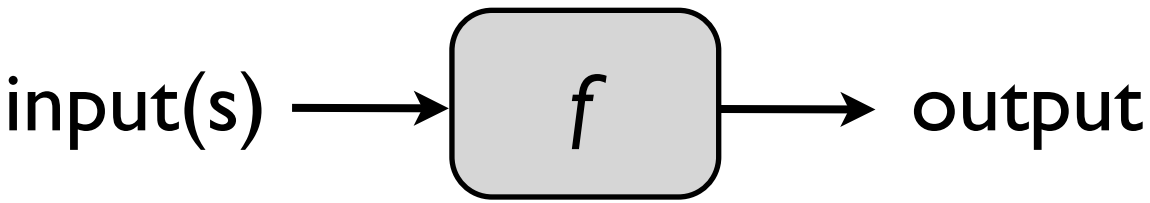
(What kind of instructions are allowed?)

How can I use these instructions to write programs?

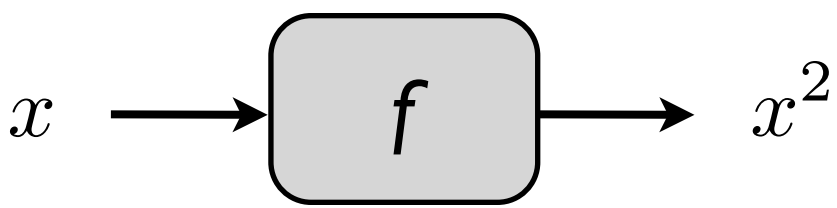
(How do I approach programming, where do I start?)

Calculation as computation

Can express calculation as a math function:



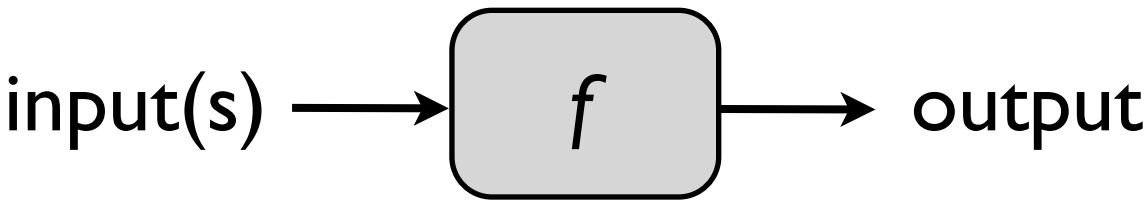
$f(x) = x^2$



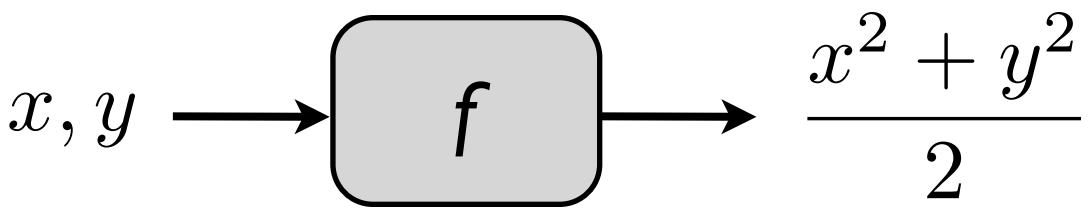
$f(2) + f(5)$ evaluates to 29

Calculation as computation

Can express calculation as a math function:



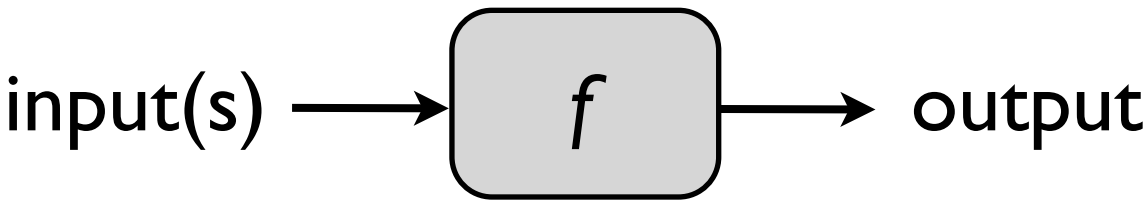
$$f(x, y) = \frac{x^2 + y^2}{2}$$



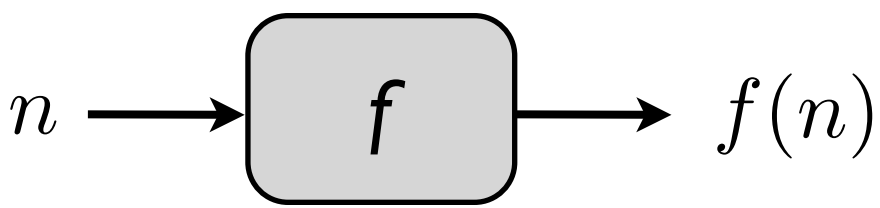
$f(2, 4) + 5$ evaluates to 15

Calculation as computation

Can express calculation as a math function:



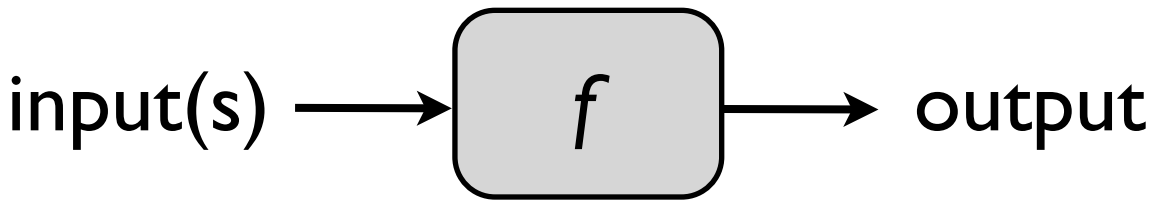
$f(n) = \text{n'th prime number}$



Often, there is no formula for the output.

Calculation as computation

Can express calculation as a math function:



Most important part of calculation/computation:

specifying how to go from the input to the output.

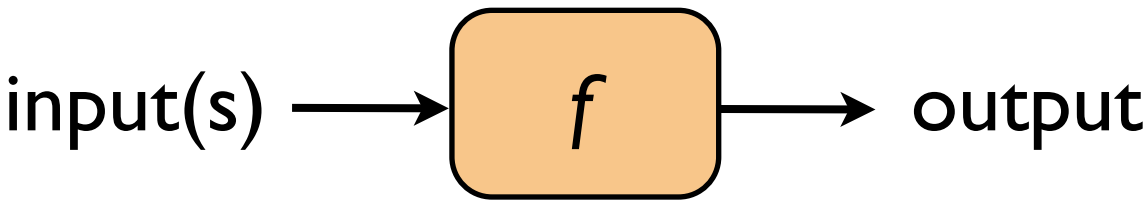
This specification/description is called:

> algorithm, if a human can follow it;

> computer program (or code), if a computer can follow it.

Computation using Python

Can express **computation** as a **Python function**:



Now, inputs and output can be any **type** of data.

Examples of defining math functions in Python:

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

```
def f(x, y):
    z = (x**2 + y**2)/2
    return z
```

```
def nthPrime(n):
    …
```

more complicated.

Computation using Python

Your program will be a collection of functions.

Basic Building Blocks

Statements

Tells the computer to do something. An instruction.

Data Types

Data is divided into different types.

Variables

Allows you to store data and access stored data.

Operators

Allows you to manipulate data.

Functions

Programs are structured using functions.

Conditional Statements

Executes statements if a condition is satisfied.

Loops

Execute a block of code multiple times.

Basic Building Blocks

Statements

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

In Python3, this is technically a function.

```
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.
```

Basic Building Blocks

Assignment Statements and Variables

variable-name = value

```
x = 5
y = "Hello World"
print(x)
print(y)
```

```
x = 3.14
y = x
x = 0
print(y)
```

In an **assignment statement**:

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

Basic Building Blocks

Data/value types

```
x = 5
y = "Hello World"
print(x)
print(y)
```

integer

string

```
x = 3.14
y = x
x = 0
print(y)
```

float

Data Types

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

●●●

Basic Building Blocks

Operators

x = 3 + 5	x stores 8
print("Hello" + " World")	Hello World
print(1.5 + 1.5)	3.0
x = 2 * x + 2**3	x stores 24
print(x > 25)	False
print((x < 25) and (x >= 0))	True
x = "Hi!" * 2	x stores "Hi!Hi!"

What an operator does depends on the types of data it's acting on.

Expression: - a valid combination of data and operators
- evaluates to a value

Expressions are evaluated first!

Basic Building Blocks

Functions

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

function definition

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

Basic Building Blocks

Functions

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

print(square(5))
```

function body (must be indented)

Basic Building Blocks

Functions

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

print(square(5))
```

parameter

Basic Building Blocks

Functions

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

print(square(5))
```

function call

Basic Building Blocks

Functions

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

print(square(5))
```

argument

Basic Building Blocks

Functions

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

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

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

Functions can have multiple inputs

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

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

Basic Building Blocks

Functions

```
def greetUser(name):
    print("Hello", name)

greetUser("Ty")
```

Hello Ty

Does this function return anything?

It actually returns None. Same as:

```
def greetUser(name):
    print("Hello", name)
    return None

print(greetUser("Ty"))
```

Hello Ty
None

Basic Building Blocks

Functions

Functions don't have to take any input

```
def greetEveryone():
    print("Hello everyone!")

greetEveryone()
```

Hello everyone!

```
greetEveryone("Ty")
```

ERROR

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

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

Basic Building Blocks

Functions

There are various built-in functions:

```
print(abs(-5))
print(max(2, 3))
print(min(2, 3))
print(pow(2, 3))
print(round(-3.14))

print(type(5))
print(type("hello"))
print(type(True))

print(int(2.8))
```

Basic Building Blocks

Statements

Tells the computer to do something. An instruction.

Data Types

Data is divided into different types.

Variables

Allows you to store data and access stored data.

Operators

Allows you to manipulate data.

Functions

Programs are structured using functions.

Conditional Statements

Executes statements if a condition is satisfied.

Loops

Execute a block of code multiple times.

Basic Building Blocks

Conditional Statements

```
def absoluteValue(n):
    if (n < 0):
        n = -n
    return n

print(absoluteValue(-5))        5
print(absoluteValue(3))         3
```

Basic Building Blocks

Conditional Statements

```
def absoluteValue(n):
    if (n < 0):
        return -n
    return n

print(absoluteValue(-5))        5
print(absoluteValue(3))         3
```

Basic Building Blocks

Loops

```
for i in range(5):
    print("Hello!")
```

```
Hello!
Hello!
Hello!
Hello!
Hello!
```

Basic Building Blocks

Loops

```
def printHello(n):
    for i in range(n):
        print("Hello!")
```

```
printHello(7)
```

```
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
```

Basic Building Blocks

Loops

```
def printHello(n):
    i = 0
    while (i < n):
        print("Hello!")
        i = i + 1

printHello(7)
```

Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!

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.