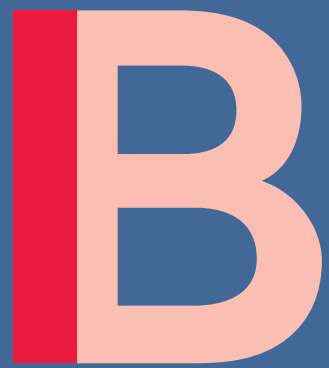
Data Structures and Algorithms

Lecture 1

Heikki Peura

h.peura@imperial.ac.uk



US & WORLD

Prisons turn to computer algorithms for deciding who to parole

18 COMMENTS

By Jacob Kastrenakes on October 14, 2013 10:06 am Email @jake_k

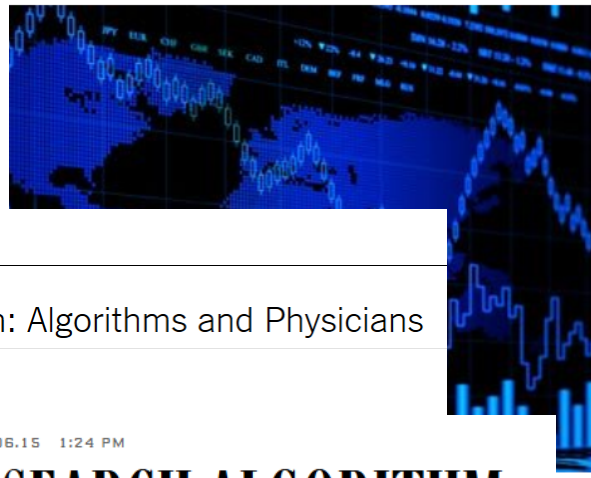
How algorithms rule the world

The NSA revelations highlight the role sophisticated algorithms play in sifting through masses of data. But more surprising is their widespread use in our everyday lives. So should we be more wary of their power?

TECH

At UPS, the Algorithm Is the Driver

Turn right, turn left, turn right: inside Orion, the 10-year effort to squeeze every penny from delivery routes



Music in the age of the algorithm

We now have instant access to almost any song. Could our tastes be narrowing as a result?

TheUpshot

Your New Medical Team: Algorithms and Physicians

Austin Frakt
THE NEW HEALTH CARE DEC. 7, 2015

Your Fish Sticks May Have Been Sliced by This Algorithm

7:09 PM BST
June 20, 2016

Cutting fish is a tough job... for humans. In this installment of Hello World, Bloomberg's Ashlee Vance visits a factory in Iceland to learn how "The Flexicut" is changing the fish slicing game forever. (Source: Bloomberg)

ADAM ROGERS SCIENCE 08.06.15 1:24 PM

GOOGLE'S SEARCH ALGORITHM COULD STEAL THE PRESIDENCY

FELIX SALMON AND JON STOKES MAGAZINE 12.27.10 12:00 PM

ALGORITHMS TAKE CONTROL OF WALL STREET

A-HED

What's Hot in the Art World? Algorithms

Admirers hold on to computerized formulas; paying $2,500 for a 'qrpff' necktie

Hedge funds + Follow

Investment: Rise of the DIY algo traders

Can tech-savvy amateurs beat the market — and the hedge funds?

BIG DATA

Using Algorithms to Determine Character

By QUENTIN HARDY JULY 26, 2015 5:30 AM 82

News Sport Weather iPlayer TV

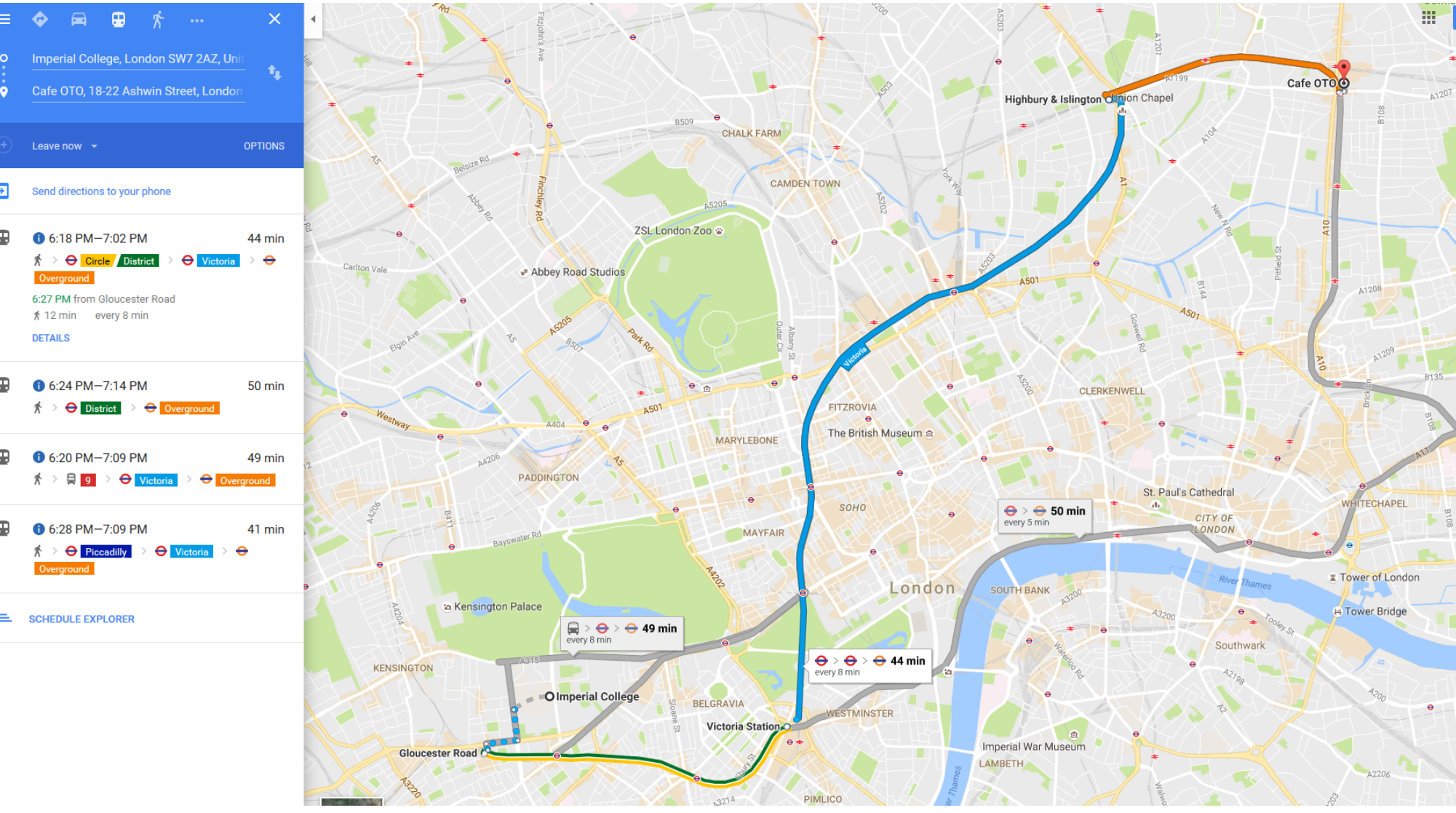
NEWS

Home UK World Business Politics Tech Science Health Education Ent

Technology

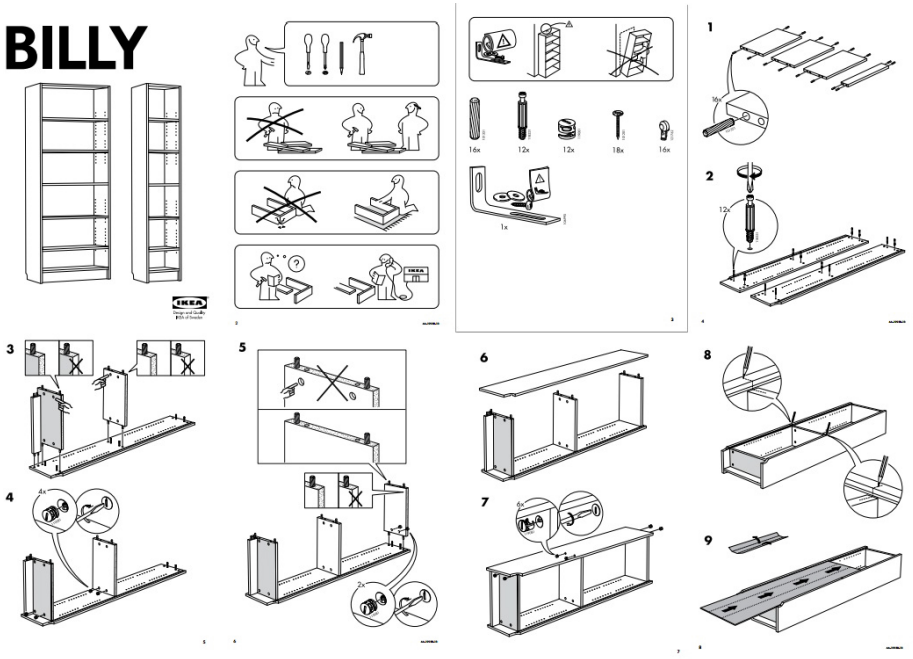
When algorithms control the world

By Jane Wakefield
Technology reporter

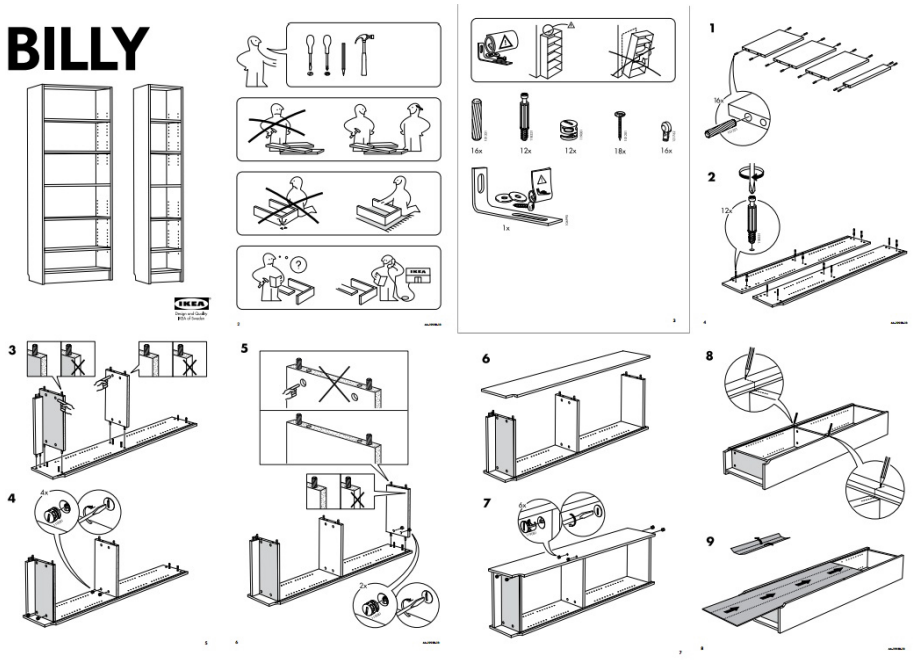


Getting from Imperial to Café OTO?

An algorithm is a recipe



An algorithm is a recipe



Algorithm:

- Step-by-step instructions
- Takes input (data) and produces output (data)

Pics: Hungry Gals, IKEA.

Solving computational problems

Data = digitised information

Data structures describe ways to organise data

Algorithms describe how we process data:

- Step-by-step instructions
- Take input data and produce output data

We write algorithms into **programs** (eg in Python)

Computers interpret and execute programs

Data Structures and Algorithms

Goals:

- Develop computational approaches for solving problems
- Understand what computers can and cannot do efficiently
- Become proficient in making a computer do these things

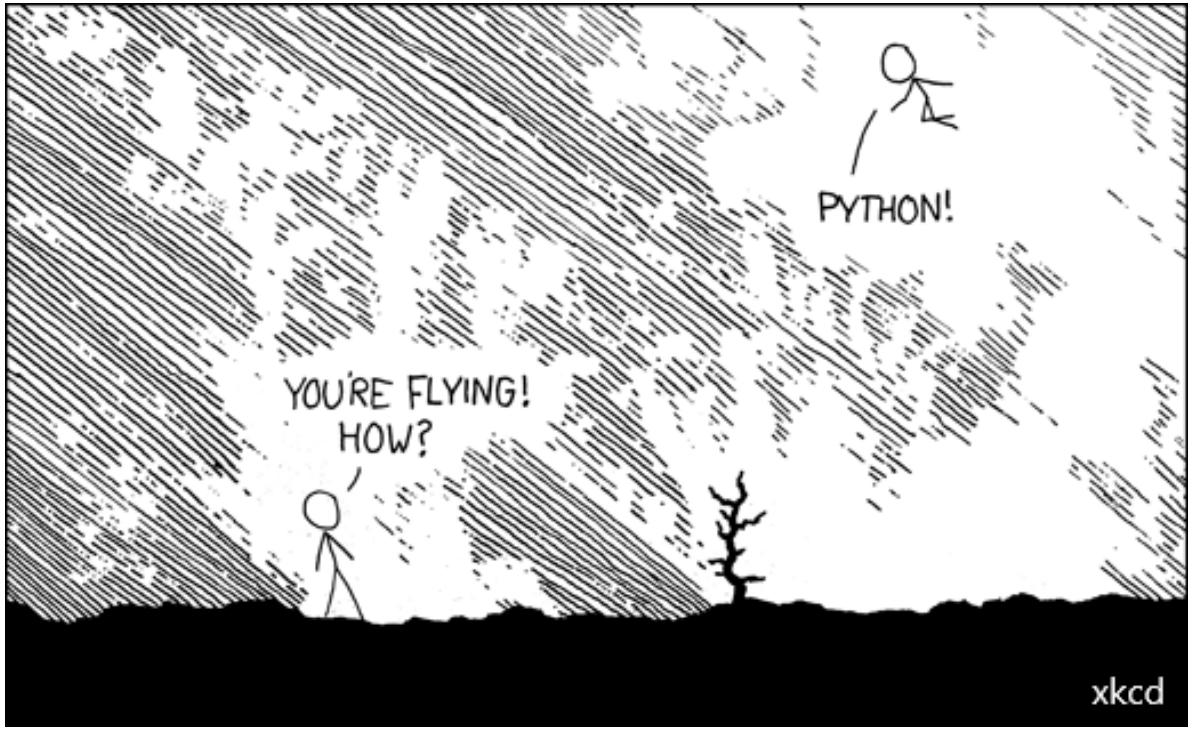
Data Structures and Algorithms

Goals:

- Develop computational approaches for solving problems
- Understand what computers can and cannot do efficiently
- Become proficient in making a computer do these things

By learning:

- Algorithms — recipes for problem solving
- Data Structures — methods for organizing data
- Python —


PYTHON!
YOU'RE FLYING!
HOW?
xkcd

Introductions

Heikki Peura, h.peura@imperial.ac.uk, ICBS room 394.

- Assistant Professor of Operations and Analytics
- PhD Management Science and Operations, London Business School
- MSc Engineering Physics and Mathematics, Aalto University, Finland

Research interests:

- Game theory and dynamic programming
- Sustainable energy, risk management, pricing

You?

You?

- 82 % have done at least some programming
- 54 % have done programming in the past year
- 20 % have taken an Algorithms class

You?

- 82 % have done at least some programming
- 54 % have done programming in the past year
- 20 % have taken an Algorithms class

Programming languages (N = 81): Python 42, R 37, SQL 28, Java 24, Matlab 23, C++ 15, Visual Basic 13, C 8, Javascript 7, Assembly 3, C# 3, Bash/Shell 2, PHP 2, Ruby 1, Swift 1, Scala 1, Julia 1

- Max: (at least) 9 languages

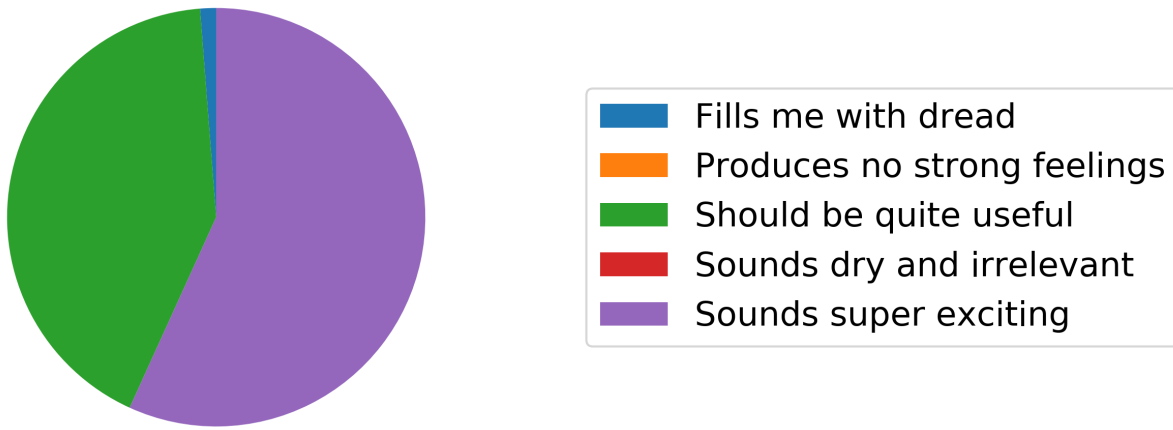
You?

- 82 % have done at least some programming
- 54 % have done programming in the past year
- 20 % have taken an Algorithms class

Programming languages (N = 81): Python 42, R 37, SQL 28, Java 24, Matlab 23, C++ 15, Visual Basic 13, C 8, Javascript 7, Assembly 3, C# 3, Bash/Shell 2, PHP 2, Ruby 1, Swift 1, Scala 1, Julia 1

- Max: (at least) 9 languages

The prospect of programming algorithms...



Practical matters: lectures

1. Each lecture has a workshop part

- Lecture material in syllabus
- Workshops: solve problems using Python
- **Bring your laptop!**

Outline (details in syllabus)

- Introduction to algorithms
- How to (and why) analyse algorithms
- Searching and sorting
- Data structures and object-oriented programming
- Graph algorithms
- Hard problems and greedy algorithms

Practical matters: tutorials

2. Tutorials focus on Python

- Materials provided on the Hub
- Core Python
- Data and scientific Python

Practical matters: tutorials

2. Tutorials focus on Python

- Materials provided on the Hub
- Core Python
- Data and scientific Python

3. Resources

- Work together (but write your own homework code)
- Ask us: **Tutorials and workshops**, email, office hours (TBD!)
- Guttag (2016): Introduction to Computation and Programming Using Python, MIT Press
- Other resources in syllabus
- Google / StackOverflow / ...

Practical matters: assessment

Grade consists of

- 50%: individual exam (Python in computer lab)
- 35%: two individual homework assignments
- 15%: tutorials and workshops

Practical matters: assessment

Grade consists of

- 50%: individual exam (Python in computer lab)
- 35%: two individual homework assignments
- 15%: tutorials and workshops

Note! UK grade expectations:

- 85% $\rightarrow$ A+, 70% $\rightarrow$ A, 60% $\rightarrow$ B, 50% $\rightarrow$ C (pass), 40% $\rightarrow$ pass with a ☹
- Average exam grade is likely to be B
- **Anything above 70% in the exam is outstanding**

Today

Functions and abstraction

Implementing a first algorithm

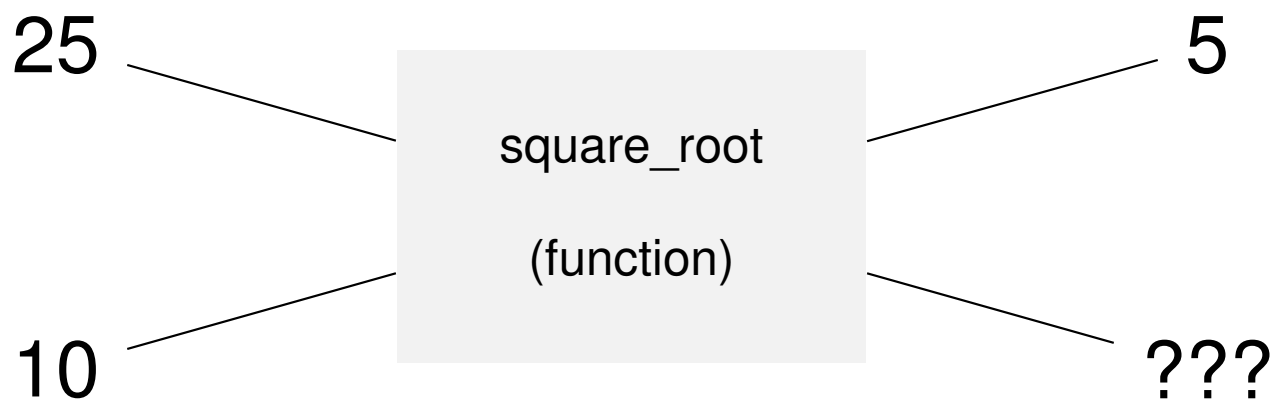
Workshop: functions and iteration

How do you calculate a square root?

The square root of a number x is a number y such that $y * y = x$

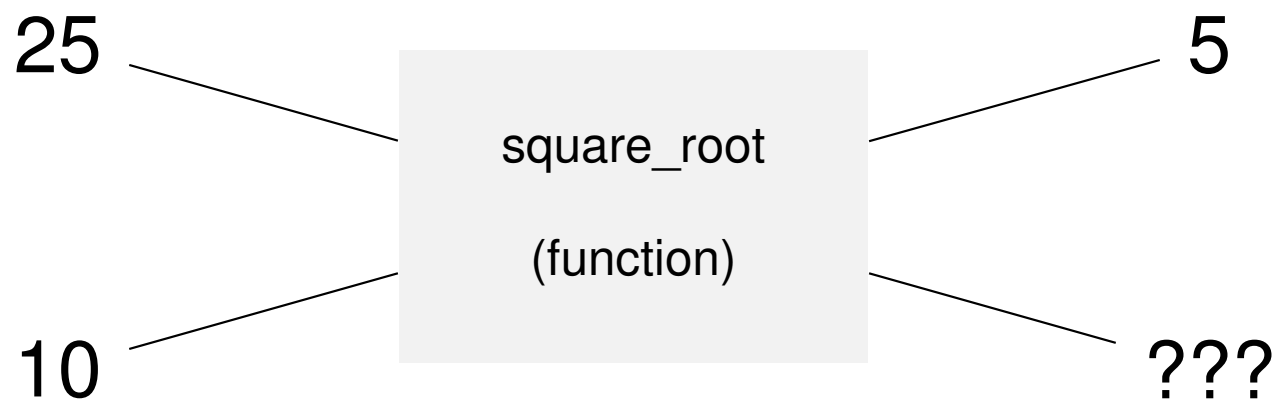
How do you calculate a square root?

The square root of a number x is a number y such that $y * y = x$



How do you calculate a square root?

The square root of a number x is a number y such that $y * y = x$



A function is like a factory

- In comes number, out comes square root
- Inside the factory, there's an **algorithm**

Square-root algorithm

The square root of x is y such that $y * y = x$

Algorithm (Heron of Alexandria, first century AD):

- Make a guess, for example $x/2$

Square-root algorithm

The square root of x is y such that $y * y = x$

Algorithm (Heron of Alexandria, first century AD):

- Make a guess, for example $x/2$
- Divide original number by the guess

Square-root algorithm

The square root of x is y such that $y * y = x$

Algorithm (Heron of Alexandria, first century AD):

- Make a guess, for example $x/2$
- Divide original number by the guess
- Find the average of these numbers

Square-root algorithm

The square root of x is y such that $y * y = x$

Algorithm (Heron of Alexandria, first century AD):

- Make a guess, for example $x/2$
- Divide original number by the guess
- Find the average of these numbers
- Use this average as next guess
- Repeat the above steps three times

Let's use Python

We use functions to organise tasks

A function is a named group of statements to perform a specific task.

- Input data $\rightarrow$ function $\rightarrow$ output data

We use functions to organise tasks

A function is a named group of statements to perform a specific task.

- Input data $\rightarrow$ function $\rightarrow$ output data

```
1  def abs_value(a):
2      if a < 0:
3          return -a # Keyword return stops function execution, outputs -a
4      else:
5          return a
6
7  # This function call runs the code block inside abs_value for a=-3
8  y = abs_value(-3)
```

We use functions to organise tasks

A function is a named group of statements to perform a specific task.

- Input data $\rightarrow$ function $\rightarrow$ output data

```
1  def abs_value(a):
2      if a < 0:
3          return -a # Keyword return stops function execution, outputs -a
4      else:
5          return a
6
7  # This function call runs the code block inside abs_value for a=-3
8  y = abs_value(-3)
```

Why functions?

1. Abstraction: user does not need to know what happens inside
2. Make **code easily re-usable** and modular
3. We often end up changing our code, so don't want to copy same code in many places

We use iteration to repeat actions

```
1 counter = 5
2 while counter != 0: # repeat indented block while condition is True
3     print("Countdown: " + str(counter) + "!")
4     counter = counter - 1
5 print("BOOM!")
```

Square-root function

```
1  def square_root(x):
2      guess = x/2
3      eps = 0.01
4      while abs(guess*guess-x) >= eps:
5          guess = (guess + x/guess)/2
6      return guess
7
8  z = 20
9  y = square_root(z)
```

Square-root function

```
1  def square_root(x):
2      guess = x/2
3      eps = 0.01
4      while abs(guess*guess-x) >= eps:
5          guess = (guess + x/guess)/2
6      return guess
7
8  z = 20
9  y = square_root(z)
```

- Takes input x and outputs its square root
- Note: uses another function inside it: built-in function `abs`
- Abstraction, reusability, reliability

Review

Algorithms are recipes for solving problems

We divide programs into named functions:

- Reusable code
- User can “just use” the function

Workshop after the break

- Looping
- Functions

Workshop

Workshop zip file on the Hub

- HTML instructions
- At some point, you'll need the `.py`-file with skeleton code (open in Spyder)