Data Structures and Algorithms with Python

Heikki Peura

h.peura@imperial.ac.uk

Lecture 1

Data Structures and Algorithms with Python

Goals:

- Develop computational approaches for solving problems (with lots of data)
- 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 — our weapon of choice

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

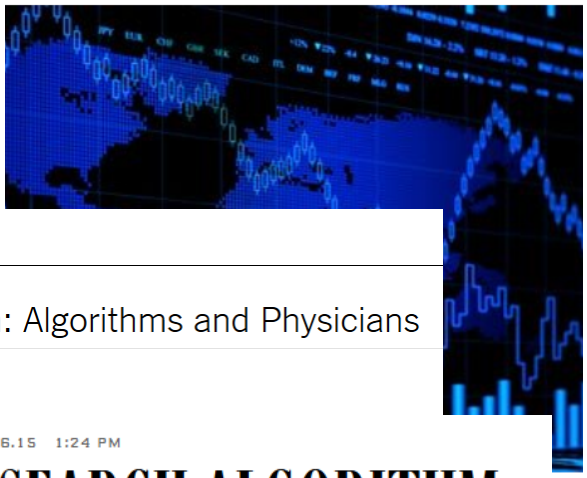
How algorithms rule the world

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

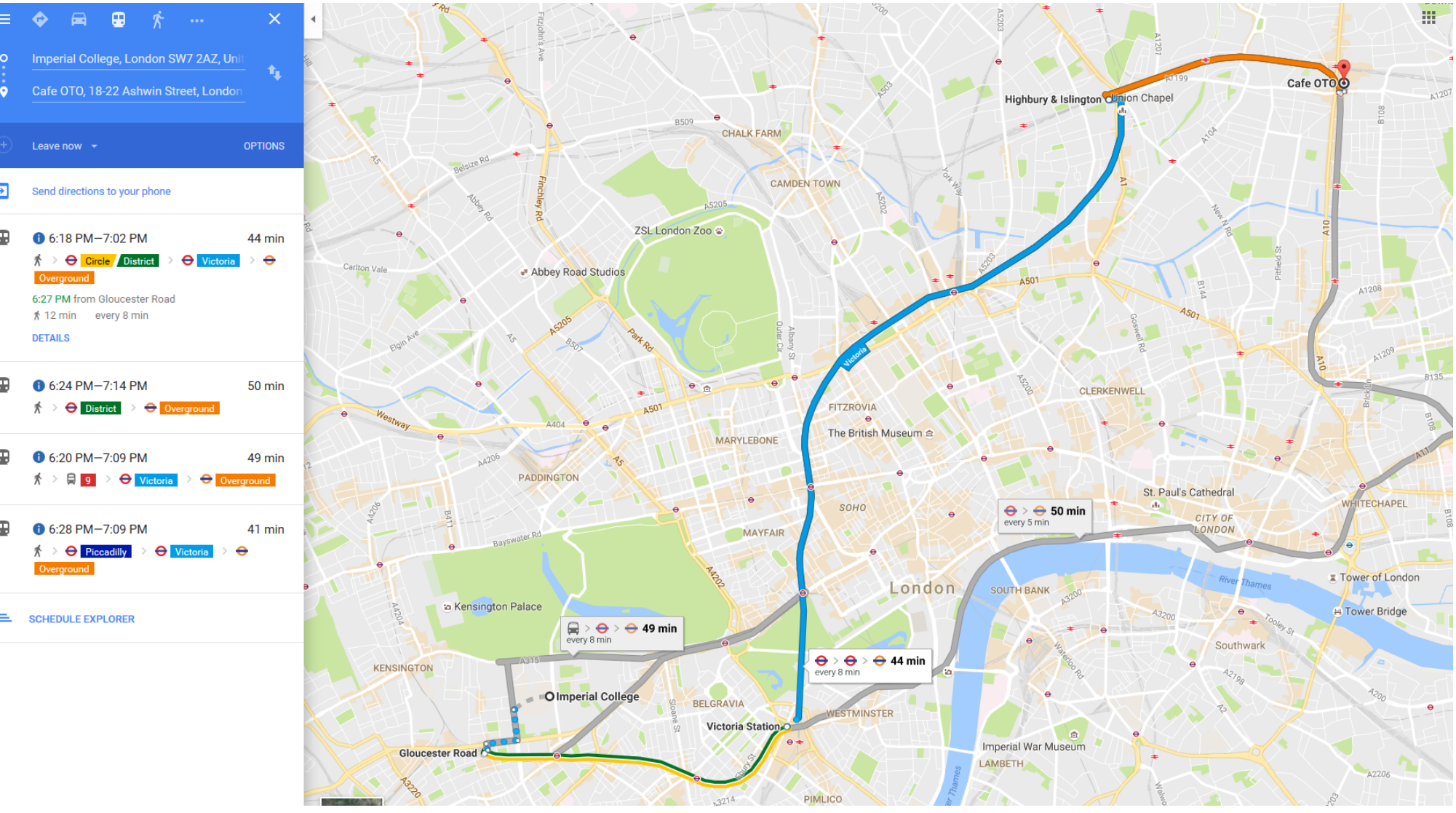
NEWS

Home UK World Business Politics Tech Science Health Education Ent

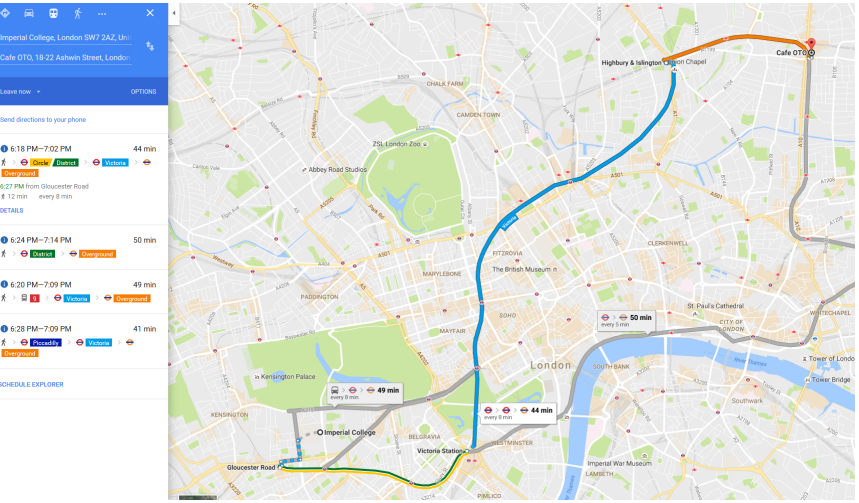
Technology

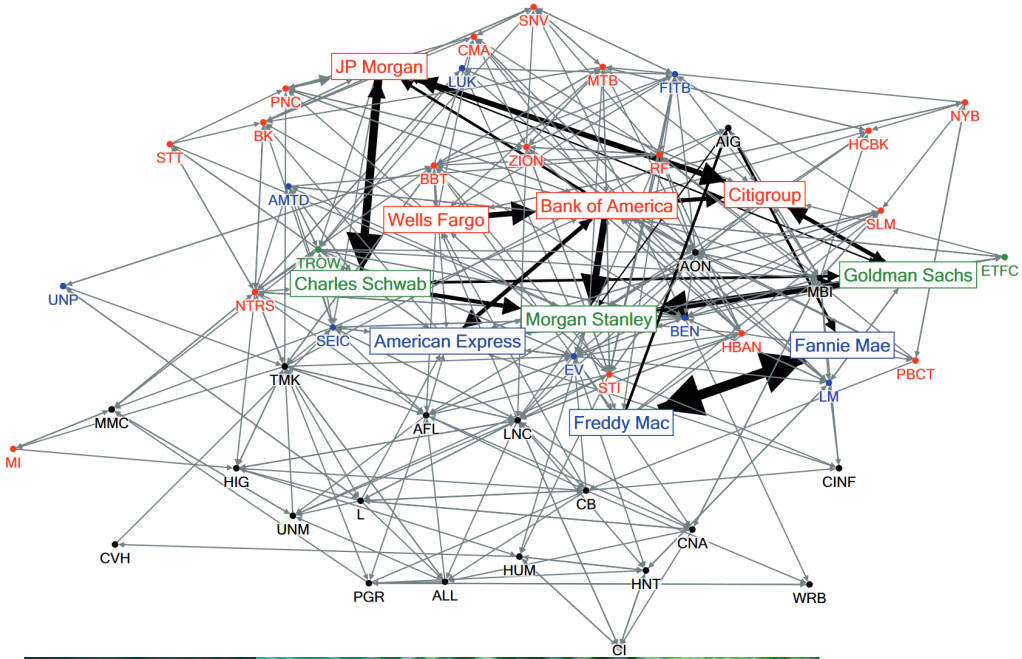
When algorithms control the world

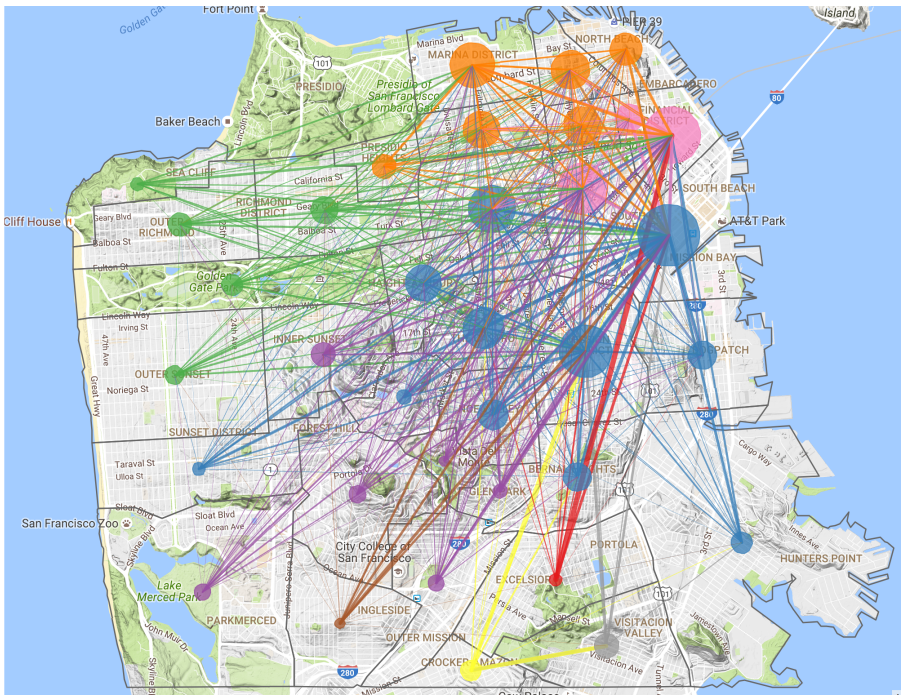
By Jane Wakefield
Technology reporter



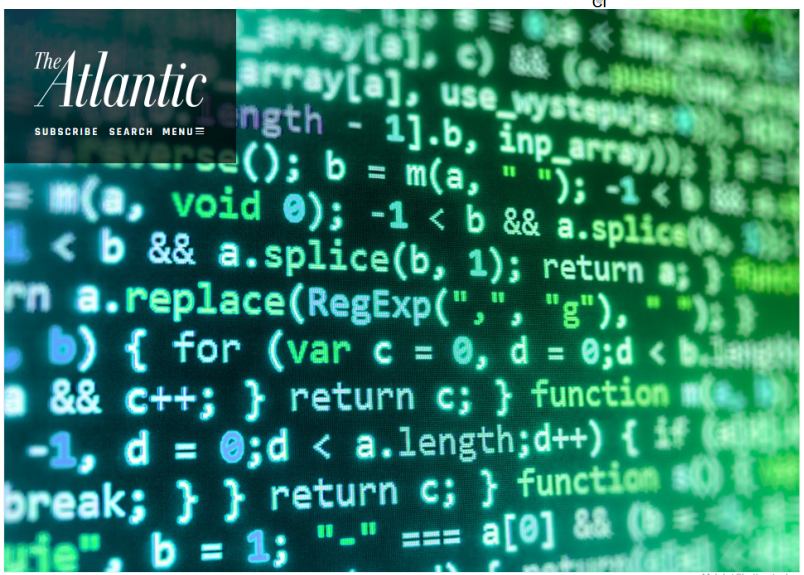
Getting from Imperial to Café OTO?







Pics: Gmaps, Uber, Hautsch et al.



Mclek / Shutterstock

Not Even the People Who Write Algorithms Really Know How They Work

The web's information filters are making assumptions about you based on details that you might not even notice yourself.

2.5k

TEXT SIZE

Introductions

You?

Practical matters

1. Classes generally comprise of a lecture and a workshop

- Materials distributed as we go along
- **Bring your laptop!**
- Workshops: solve problems using Python

Outline

- Introduction to computation (today!)
- How to analyse algorithms
- Searching and sorting
- Data structures and object-oriented programming
- Graphs, graph search, and shortest paths
- Greedy algorithms and hard problems (if there's time: dynamic programming)

Practical matters

2. Tutorials focus on developing a working knowledge of Python

- Materials provided on the Hub

- Introduction to Python — done!
- Files, Python data types, testing and debugging
- Scientific Python packages
- Algorithms practice

3. Resources

- Ask us! **Tutorials**, email, office hours by appointment
- Guttag (2013): Introduction to Computation and Programming Using Python, MIT Press **[textbook]**
- Dasgupta, Papadimitrou, Vasmirani (2006): Algorithms **[more advanced, language agnostic]**
- Others: Cormen et al: Introduction to Algorithms; Sedgewick and Wayne: Algorithms; ...
- Google / StackExchange / ...

Assessment

100 % of grade from coursework

- 30%: two individual programming assignments
- 30%: a group project
- 30%: **two short quizzes in tutorials 3 and 5 -> be there!**
- 10%: two short online questionnaires (lecture preparation and feedback)
- More details later

Group project: max 5 people per group

- Send me your group by 16 Sep (Friday next week)

What are computers good at?

Computers excel at: performing well defined calculations
Remembering the results

Solving a computational problem

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

How to find $\sqrt{x}$?

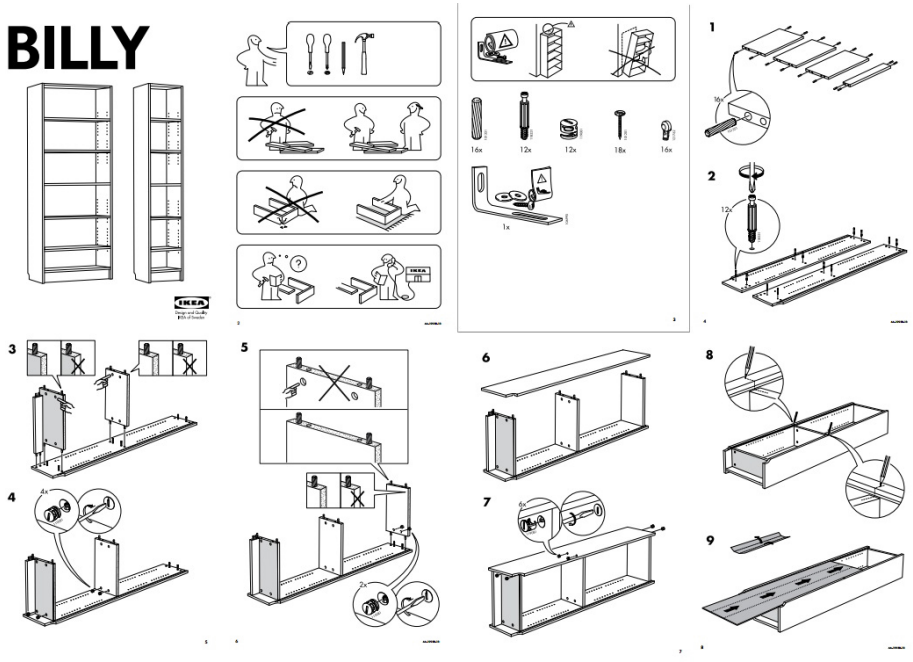
Algorithm (Heron of Alexandria, first century AD):

- Start with a guess g
- If $g * g$ is close to x, stop and return g as the answer
- Otherwise make new guess as the average of g and x/g
- Repeat process using new guess until close enough

An algorithm is a recipe

Algorithm:

- Step-by-step instructions — “pseudocode”
- Takes input (data) and produces output (data)



Pics: Hungry Gals, IKEA.

Solving computational problems

Data = digitized information

Data structures allow us to organize data

Algorithms describe how we process data:

- Step-by-step instructions — "pseudocode"
- Take input data and produce output data

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

Computers interpret and execute programs

Programming

We write algorithms into programs

- Translate recipe/pseudocode (English + maths) -> program (Python)

Goal: understand algorithms / data structures on **"recipe level"**

- Design algorithms without a specific programming language in mind
- Convenient for testing ideas and analysing algorithm efficiency
- Abstraction: organizing algorithms for humans

Goal: lots of **programming practice** in Python

- Write code
- Read and use others' code

Turning a recipe into computer instructions

We write algorithms into programs

- Computer processors use predefined **primitive** ("low-level") **instructions**
  - Arithmetic, logic, simple tests, moving data
- **Alan Turing** showed that **six operations are enough to compute anything** (that is computable) —> all languages can perform same calculations
- Modern ("high-level") programming languages like Python allow us to use **abstract methods** that are converted back to primitives
- **It is very easy to get useful work done quickly in Python**

You already know some Python

```
1  print("Hello World")
2
3  print(3+2)
4  print('3+2') # A string
5
6  print(type(3)) # int
7  print(type(3.0)) # float
8
9  pi = 3.14159 # variable assignment
10 diameter = 12
11 area = pi*(diameter**2) # arithmetic operations
12
13 print("The area is " + str(area)) # changing types and combining strings
```

You know how to use conditionals

```
temp = 20
humidity = 80

print(temp > 30) # Condition: boolean

# We use keywords, booleans, colon and whitespace to control program flow
if temp > 30:
    print("Too hot!")
elif temp < 15:
    print("Too cold!")
else:
    print("Too boring!")

if temp > 30 and humidity > 80:
    print("That's it, I'm out of here!")
```

You know how to use iteration and lists

```
x = 5
counter = x
while counter != 0: # perform while condition is True
    print("Countdown: " + str(counter) + "!")
    counter = counter - 1
print("BOOM!")
```

```
diameterList = [1,2,3,4,5] # a list
pi = 3.141519

for diameter in diameterList: # go through each item in list
    area = pi*(diameter**2)
    print(area)

elements = [] # empty list

for i in range(4): # go through integers using range
    elements.append(i) # add new item to list

print(elements[1:2]) # get second and third items from list
```

Translating Heron's recipe to Python

How to find the square root of x:

- Start with a guess g
- If $g * g$ is close to x, stop and return g as the answer
- Otherwise make new guess as the average of g and x/g
- Repeat process using new guess

Algorithm Find square root of x

0: **input**: x, starting guess g, acceptable error ϵ
1: **while** $|g^2 - x| \geq \epsilon$ **do**
2: $\quad g = (g + x/g)/2$
3: **end while**

```
1  eps = 0.01
2  x = 50
3  guess = x/2
4  while abs(guess**2-x) >= eps:
5      guess = (guess + x/guess)/2
6  print("The square root of ", x, " is approximately:", guess)
```

You know how to define functions

Function `squareRoot`(x) for repeatedly calculating square roots

- Takes input x and outputs its square root
- Abstraction: user does not need to see what happens inside

The `squareRoot` function calls built-in function `abs` as a subroutine:

```
1  def squareRoot(x,eps=0.01): # Default value eps = 0.01 if not specified
2      guess = x/2
3      while abs(guess**2-x) >= eps:
4          guess = (guess + x/guess)/2
5      return guess
6
7  z = 20
8  y = squareRoot(z) # Use default value for eps
```

Why functions?

1. Make **code easily re-usable** and modular
2. Duplicate code is bad practice — you **will** end up changing your code

Recursive programs

Idea: A recursive function solves problems by breaking them down to smaller subproblems, usually calling the function itself

Example: calculating factorials. "Inductive" definition:

$$1! = 1 \qquad \text{Base case}$$
$$n! = n \cdot (n-1)! \qquad \text{General case}$$

In Python, **a function can call itself**:

```
1 def factorial(n):
2     """Assumes that n is an int > 0
3     returns n!"""
4     if n == 1: # Base case
5         return n
6     return n*factorial(n-1) # General case moves towards base case
```

Workshop: recursion is useful but **dangerous**...

Workshop: programs, fast and slow

After the break...

Practice your coding skills:

- Fibonacci numbers and recursion
- Palindromes, anagrams and string manipulation