Introduction to Computer Code
Computer Code for Beginners
Week 1

Matt Luckcuck

21/09/2017

Housekeeping

Housekeeping

- Toilets
- Fire Alarm
- Additional Support
- **Please** ask questions if you're unsure!

Housekeeping

Each Week...

- $\sim$ 40 minutes of lecture
- $\sim$ 15 minutes break
- $\sim$ 65 minutes of practical

This Week

Outline

- What is Programming?
- Programming Languages
- Programming Process
- Python
  - Introduction
  - Detail
- Overview of Practical
- Overview of Course

What Is Programming?

What is Programming?

Has anyone done any programming before?

What is Programming?

Has anyone done any programming before?

What about...

- VCR?

What is Programming?

Has anyone done any programming before?

What about...

- VCR?
- Sky+ (or similar)?

What is Programming?

Has anyone done any programming before?

What about...

- VCR?
- Sky+ (or similar)?
- Washing Machine?

What is Programming?

Programming is the process of giving instructions to a computer.

What is Programming?

Programming is the process of *designing and writing* instructions for a computer.

What is Programming?

Computers...

- Computers are **Stupid**!
  - But good with numbers and repetition
- We need to tell them *exactly* what to do
- This means giving precise instructions...
  - ... in a language the computer can '*understand*'

What is Programming?

Building a Program

- Program: sequence of algorithms to perform behaviour
- Algorithm: sequence of instructions to solve a problem, e.g. . .
  - Recipe, or
  - Directions
- Design of a part of the program

What is Programming?

Building a Program

- Program: sequence of algorithms to perform behaviour
- Algorithm: sequence of *instructions* to solve a problem, e.g. . .
  - Recipe, or
  - Directions
- Design of a part of the program

Programming Languages

Programming Languages

Scratch

C

C++

Java

Python

Javascript

PHP

Programming Languages

Scratch

C

C++

Java

Python

Javascript

PHP

Programming Languages

Programming Languages

- Provide a set of instructions to tell computer what to do...
  - Expressions
  - Variables
  - Control Structures
  - Keywords
- That we combine according to grammar rules to make a program

'Understanding'

- Computers only *understand* binary (0010)
- Programming languages are human-readable then translated into machine-readable

Programming Process

Programming Process

Approaches to Building a Program

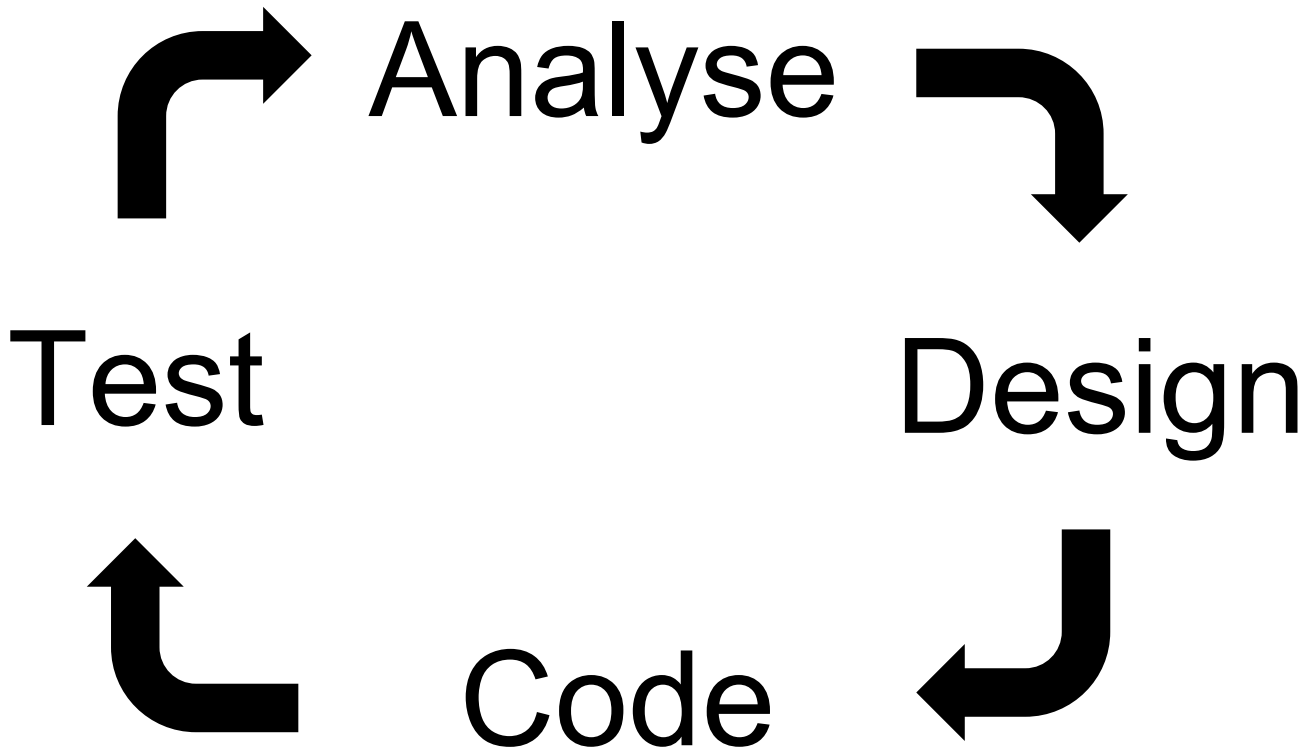
- Decomposition
  - Breaking a problem down into smaller parts
  - Solving a basic version of the problem, then add details
- Patterns
  - Identify similarities and sequences
  - Different specific problems are often the same general shape
- Abstraction
  - Focus on the important parts of a problem
  - Think of a bus route map

Programming Process

Approaches to Building a Program

- Algorithms
  - Step-by-step instructions
  - Recipe
- Debugging
  - Cycle of trial and error
  - Requires resilience

Programming Process



Programming Process

Designing an Algorithm

- Problem: We need to make some tea!
- Algorithm: ...

Tea Anyone? – Basic Algorithm

1. Boil water
2. Add tea to the pot
3. Add boiled water to the pot
4. Brew tea
5. Pour tea into cup
6. Stir tea
7. Enjoy!

Tea Anyone? – A Little More Detail...

```
1  Boil water
2  Add tea to the pot
3  Add boiled water to the pot
4  Brew Tea
5  if sugar = true then
6  |   Add sugar
7  Pour tea into cup
8  if milk = true then
9  |   Add milk
10 Stir tea
11 Enjoy!
```

Tea Anyone? – Who Wants Tea?

```
1  Boil water
2  foreach person do
3  |   Add 1 spoon of tea to the pot
4  Add boiled water to pot
5  Brew tea
6  foreach person do
7  |   if sugar = true then
8  |   |   Add sugar
9  |   Pour tea into cup
10 |   if milk = true then
11 |   |   Add milk
12 |   Stir tea
13 Enjoy!
```

Tea Anyone?

What does this show us?

- Processes:
  - Start small and add detail
  - Abstracting away from the details of boiling, brewing, etc
- Building Blocks:
  - Sequence, Branching, and Loops
  - Variables for data that changes
  - Expressions

Python

Programming Tool Kit

Basic Building Blocks of Programs

- Introduce:
  - Modules and Files
  - Functions
  - Variables
  - Control Structures
- Much more detail over the coming weeks

Python Program Example

Hello World

Basic introductory programs

```
1 print("Hello World!")
```

Modules

Modules

- A *module* is a collection of code that performs related behaviour
- In Python, each file is a module
- Design decision. . .
  - A simple program is likely to be one module
  - A more complex program is best split up into separate modules

Functions

Functions

- A block of code, wrapped up for us to use when we need it
- Lots of built-in functions (like `print()`)
- We can write our own
- Can take parameters (like `print("Hello World")`)
- Can return values
- Proper introduction to these next week

Variables

Variable

- Data that our program uses
- Box in the computer's memory with a value inside
- Label to remember what's inside
- So naming variables well is important!
- `name = value`
  - Assigning a `value` to the `name`

Variables

Variable Types

- Whole Numbers — Integers (eg 1 or 10)
- Decimal Numbers — Floating Point Numbers (eg 3.14)
- Boolean (`True` or `False`)
- String of characters (Text)
- Others...

```
1 score = 10
2 name = "Matt"
```

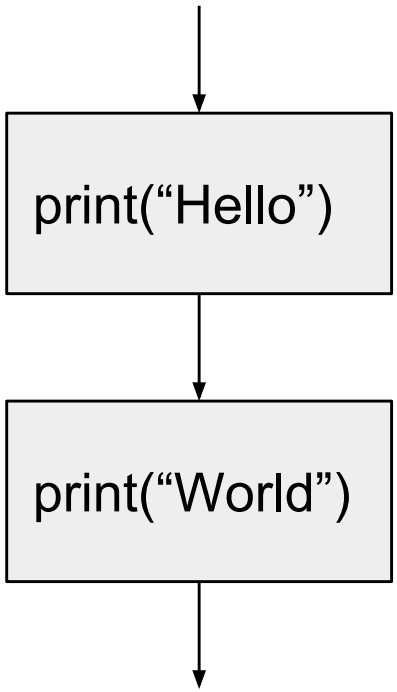
Sequences, Branches, and Loops

Sequential Instructions

- A program is a sequence of instructions...
  - Unless we tell it otherwise
- Sequential code is a basic building block
  - But often too simple
- Sequential instructions: back to Hello World...

Sequences, Branches, and Loops

```
1 print("Hello ")
2 print("World!")
```

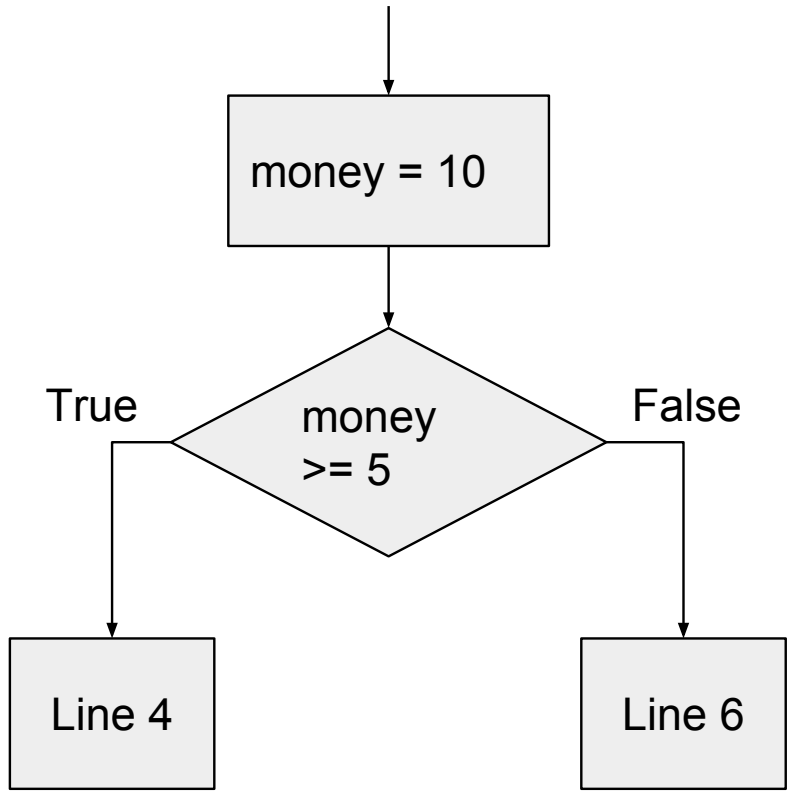


Branching Control Structure

- Simplest way of breaking up sequential code
- Choice between one branch or another branch
  - Based on a boolean condition
- Using the keywords `if` and `else`
  - Two blocks that are executed *conditionally*
  - Blocks must be indented
- Example: Sale of an item costing £5 . . .

Sequences, Branches, and Loops

```
1 money = 10
2
3 if money >= 5:
4   print("You have enough")
5 else:
6   print("Not enough")
```

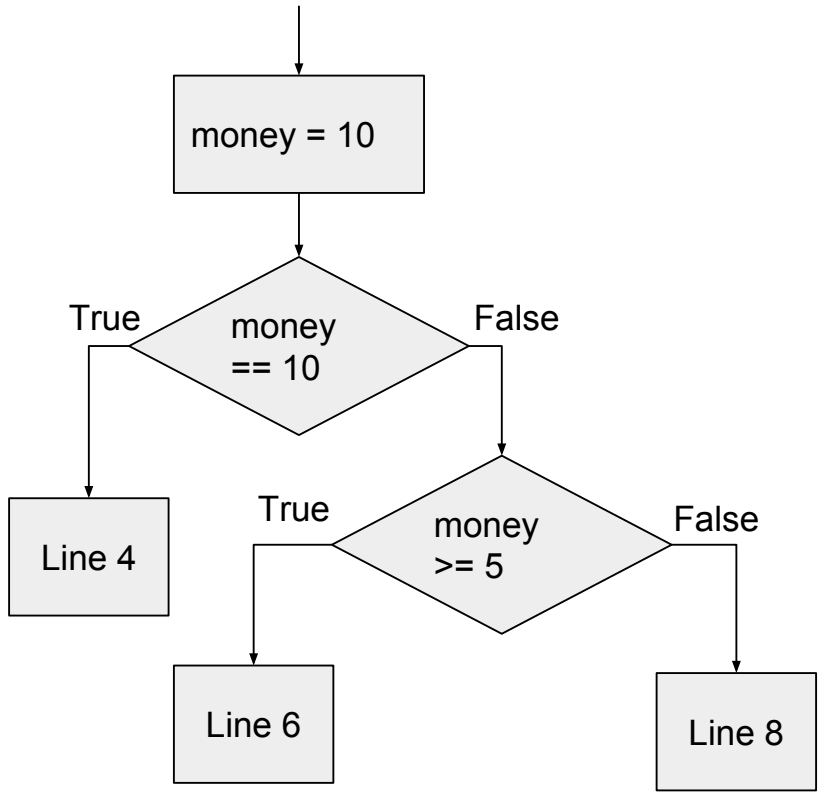


Multiple Branches

- What if two branches aren't enough?
- Extend the `if` structure
  - Uses the `elif` (else if) keyword
  - Adds another conditional block
  - Again must be indented
- This is useful for complex conditions
- Back to the previous example...

Sequences, Branches, and Loops

```
1 money = 10
2
3 if money == 10:
4   print("Buy two!")
5 elif money >= 5:
6   print("You have enough")
7 else:
8   print("Not enough")
```



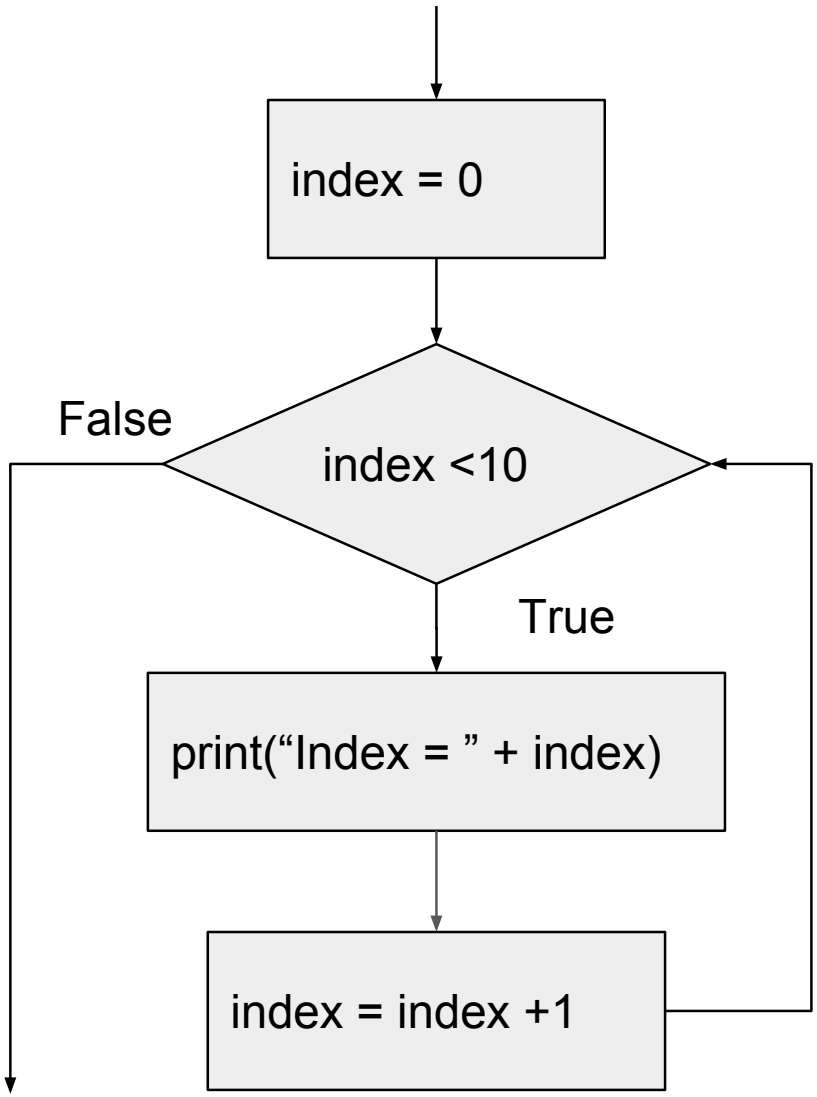
Sequences, Branches, and Loops

Looping Instructions

- Allows us to repeat a block of code
  - Iteration
- Basic form uses the `while` keyword
  - Checks the condition at the being of each iteration
  - Executes the body of the loop *while* that condition is true
  - Usually update the condition in the body of the loop to exit
  - Loop body must be indented
- Need to be careful of infinite loops!
- We'll cover loops in more detail next week
- Example: printing numbers...

Sequences, Branches, and Loops

```
1 index = 0
2
3 while index <10:
4   print("Index = " + index)
5   index = index +1
```



Python Detail

Operators

Arithmetic Operators

- Add: `x + y`
- Minus: `x - y`
- Multiply: `x * y`
- Divide: `x / y`

Operators

Assignments and Comparisons

- Assignment: `x = 10`
  - Increment: `x += 1`
  - Decrement: `x -= 1`
- Equal To: `x == 10`
- Not Equal To: `x != 10`
- Greater Then (Or Equal to): `x > 10` (`x >= 10`)
- Less Than (Or Equal to): `x < 10` (`x <= 10`)

Operators

Assignments and Comparisons

- Assignment: `x = 10`
  - Increment: `x += 1` (`x = x + 1`)
  - Decrement: `x -= 1` (`x = x - 1`)
- Equal To: `x == 10`
- Not Equal To: `x != 10`
- Greater Then (Or Equal to): `x > 10` (`x >= 10`)
- Less Than (Or Equal to): `x < 10` (`x <= 10`)

Operators

Boolean Operators

- `not x` : Negates (toggles) the value
  - `not True` = `False`
  - `not False` = `True`
- `x and y` : `True` if both values are `True`
  - `True and True` = `True`
  - `False and False` = `False`
  - (`True and False`) = (`False and True`) = `False`
- `x or y` : `True` if at least one value is `True`
  - `True or True` = `True`
  - `False or False` = `False`
  - (`True or False`) = (`False or True`) = `True`

Variables

Type Conversions

We can try to convert between types

- String : `str(x)`
  - `str(3)` : `"3"`
- Integer : `int(x)`
  - `int(3.14)` : `3`
- Float : `float(x)`
  - `float(3)` : `3.0`

Variables

Strings

- String is a sequence of characters
  - Either `"Hello World"` or `'Hello World'`
  - Join (concatenate) `"Hello" + "World"`
- A character is represented 'internally' by a unique code
  - ASCII — American Standard Code for Information Interchange
  - UniCode
- We can convert between characters and their code
- `ord('a')` – 97
- `chr(97)` – 'a'

Variables

String Escape Characters

- Some characters we can't type
- Use (\) to escape normal typing and produce special characters:
  - `\n` — New Line
  - `\'` or `\"` — single or double quote mark

User Input

Getting User Input

- We often need some user input to make our programs useful...
  - Which branch to take?
  - When to stop looping?

User Input

Python User Input

- In Python we can use the `input()` function
  - `result = input("Type Something Please")`
- Always returns a string
- If we want an integer: `result = int(input("..."))`
- Fails if the input isn't an integer...
  - Looking at that later
  - And come back to it in future weeks

Python Programming Style

Keywords

- Python has certain keywords that are *reserved*
  - Built-in and mean something specific
  - We can't use them for anything else
- We've already seen some
  - `True`
  - `False`
  - `if`
  - `else`
  - `while`

Python Programming Style

Programming Style

- Python groups blocks of code by how indented they are
  - Can be tabs *or* spaces...
  - Pick one and stick to it
- Blank lines to separate large blocks of code
- Good practice to aid readability

Summary

Summary

Summary

- Computers are **stupid**
- Programming: giving a computer instructions
- Key Skills:
  - Decomposition of problems
  - Modelling and solving problems in abstract terms
  - Trial and Error (Trying, Failing, Fixing)
- Introduced Basic Programming Building Blocks
- Introduced Basic Python

Summary

Practicals

- Each week's practicals
  - Examples how how to use the basic tools of programming
  - Recognise the pattern and solve similar problems
- Course Website: `mluckcuck.github.io/python/`
- Manual: `docs.python.org/3/library/`
  - Make sure you use **Version 3**!

Summary

Practicals This Week

- Introduction to
  - Variables
  - Branching and Looping

Summary

Course Topics

1. Introduce Programming and Python
2. More Loops, Basic Sequences, and Functions
3. More Complex Data Types, Handling Errors, File Handling
4. Algorithms using Graphs
5. Two Week Project
   - Design
   - Program