

# Supporting the transition from block to text based programming languages

By Andrew Csizmadia, Mark Dorling & Steve Bunce

CONSTRUIT 2017  
Warwick University



 @APCsizmadia, @MarkDorling

Slide

# What do we have to teach?

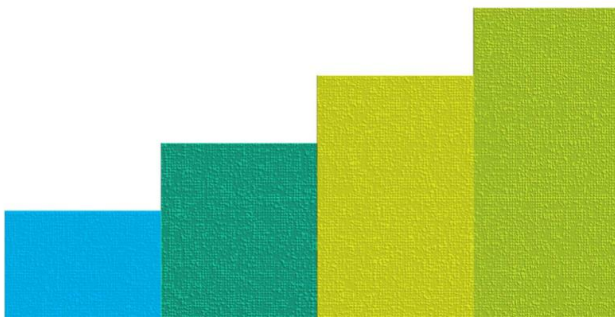
# Programmes of Study (Concepts and Key constructs)

Computing Progression Pathways						
Pupil Progression	Algorithms	Programming & Development	Data & Data Representation	Hardware & Processing	Communication & Networks	Information Technology
↓	<ul style="list-style-type: none"> <li>Understands what an algorithm is and is able to express simple linear flow branching algorithms symbolically. (AL)</li> <li>Understands that computers read precise instructions. (AL)</li> <li>Demonstrates care and precision to avoid errors. (AL)</li> </ul>	<ul style="list-style-type: none"> <li>Knows that users can develop their own programs, and can demonstrate this by creating a simple program in an environment that does not rely on text e.g. programmable robots etc.</li> <li>Executes, checks and changes programs. (AL)</li> <li>Understands that programs execute by following precise instructions. (AL)</li> </ul>	<ul style="list-style-type: none"> <li>Recognises that digital content can be represented in many forms. (AB) (GE)</li> <li>Distinguishes between some of these forms and can explain the differences that they communicate information. (AB)</li> </ul>	<ul style="list-style-type: none"> <li>Recognises that computers have no intelligence and that computers can do nothing unless a program is executed. (AB) (GE)</li> <li>Recognises that all software executed on digital devices is programmed. (AL) (AB) (GE)</li> </ul>	<ul style="list-style-type: none"> <li>Obtains content from the world wide web using a web browser. (AL)</li> <li>Understands the importance of communicating safely respectfully online, and the need for keeping personal information private. (EV)</li> <li>Knows what to do when concerned about content or being contacted. (AL) (EV)</li> </ul>	<ul style="list-style-type: none"> <li>Uses software under the control of the teacher to create, using a web browser. (AL)</li> <li>Understands that content using appropriate file and folder names. (AB) (GE) (DE)</li> <li>Understands that content using appropriate file and folder names. (AB) (GE) (DE)</li> <li>Shares their use of technology in school.</li> <li>Knows common use of information technology beyond the classroom. (GE)</li> <li>Talks about their work and makes changes to improve it. (EV)</li> </ul>
↓	<ul style="list-style-type: none"> <li>Understands that algorithms are implemented on digital devices as programs. (AL)</li> <li>Designs simple algorithms using loops, and selection i.e. if statements. (AL)</li> <li>Uses logical reasoning to predict outputs. (AL)</li> <li>Detects and corrects errors i.e. debugging, in algorithms. (AL)</li> </ul>	<ul style="list-style-type: none"> <li>Uses arithmetic operators, if statements, and loops, within programs. (AL)</li> <li>Uses logical reasoning to predict the behaviour of programs. (AL)</li> <li>Detects and corrects simple semantic errors i.e. debugging, in programs. (AL)</li> </ul>	<ul style="list-style-type: none"> <li>Recognises different types of data: text, number, images, sound. (AB) (GE)</li> <li>Appreciates that programs can work with different types of data. (EV)</li> <li>Recognises that data can be structured in tables to make it useful. (AB) (DE)</li> </ul>	<ul style="list-style-type: none"> <li>Recognises that a range of digital devices can be considered a computer. (AB) (GE)</li> <li>Recognises and can use a range of input and output devices.</li> <li>Understands how programs specify the function of a general purpose computer. (AB)</li> </ul>	<ul style="list-style-type: none"> <li>Navigates the web and can carry out simple web searches to collect digital content. (AL) (EV)</li> <li>Recognises and can use a range of internet services e.g. VODP. Recognises what is acceptable and unacceptable behaviour when using technologies and online services.</li> </ul>	<ul style="list-style-type: none"> <li>Uses technology with increasing independence to purposefully engage digital content. (AB)</li> <li>Shows an awareness for the quality of digital content. (EV)</li> <li>Demonstrates use of computers safely and responsibly, knowing a range of ways to report unacceptable content and contact when online.</li> <li>Uses a variety of software to manipulate and present digital content e.g. spreadsheets. (AL)</li> <li>Shares their experience of technology in school and beyond the classroom. (GE) (EV)</li> <li>Talks about their work and makes improvements to solutions based on feedback received. (EV)</li> </ul>
↓	<ul style="list-style-type: none"> <li>Designs solutions (algorithms) that use repetition and two-way selection i.e. if, then and else. (AL)</li> <li>Uses diagrams to express solutions. (AB)</li> <li>Uses logical reasoning to predict outputs, showing an awareness of inputs. (AL)</li> </ul>	<ul style="list-style-type: none"> <li>Creates programs that implement algorithms to achieve given goals. (AL)</li> <li>Declares and assigns variables. (AB)</li> <li>Uses print tested loop e.g. 'until', and a sequence of selection statements in programs, including an if, then and else statement. (AL)</li> </ul>	<ul style="list-style-type: none"> <li>Understands that the difference between data and information. (AB)</li> <li>Knows why sorting data in a flat file can improve search for information. (EV)</li> <li>Uses filters or can perform simple criteria searches for information. (AL)</li> </ul>	<ul style="list-style-type: none"> <li>Knows that computers collect data from various input devices, including sensors and application software. (AB)</li> <li>Understands the difference between hardware and application software, and their roles within a computer system. (AB)</li> </ul>	<ul style="list-style-type: none"> <li>Understands the difference between the internet and internet of, and can use a range of internet services e.g. VODP. Recognises what is acceptable and unacceptable behaviour when using technologies and online services.</li> </ul>	<ul style="list-style-type: none"> <li>Collects, organizes and presents data and information in digital content. (AB)</li> <li>Creates digital content to achieve a given goal through combining software packages and internet services to communicate with a wider audience e.g. blogging. (AL)</li> <li>Makes appropriate improvements to solutions based on feedback received, and can comment on the success of the solution. (EV)</li> </ul>
↓	<ul style="list-style-type: none"> <li>Shows an awareness of tasks best completed by humans or computers. (EV)</li> <li>Designs solutions by decomposing a problem and creates a sub-solution for each of these parts. (DE) (AL) (AB)</li> <li>Recognises that different solutions exist for the same problem. (AL) (AB)</li> </ul>	<ul style="list-style-type: none"> <li>Understands the difference between, and appropriately uses if and if, then and else statements. (AL)</li> <li>Uses a variable and relational operators within a loop to govern termination. (AL) (GE)</li> <li>Designs, writes and debugs modular programs using procedures. (AL) (DE) (AB) (GE)</li> <li>Knows that a procedure can be used to hide the detail with sub-solution. (AL) (DE) (AB) (GE)</li> </ul>	<ul style="list-style-type: none"> <li>Performs more complex searches for information e.g. using Boolean and relational operators. (AL) (GE) (EV)</li> <li>Analyses and evaluates data and information, and recognises that poor quality data leads to unreliable results, and inaccurate conclusions. (AL) (EV)</li> </ul>	<ul style="list-style-type: none"> <li>Understands why and when computers are used. (EV)</li> <li>Understands the main functions of the operating system. (DE) (AB)</li> <li>Knows the difference between physical, wireless and mobile networks. (AB)</li> </ul>	<ul style="list-style-type: none"> <li>Understands how to effectively use search engines, and knows how search results are selected, including that search engines use 'web crawler programs'. (AB) (GE) (EV)</li> <li>Selects, combines and uses internet services. (EV)</li> <li>Demonstrates responsible use of technologies and online services, and knows a range of ways to report concerns.</li> </ul>	<ul style="list-style-type: none"> <li>Makes judgements about digital content when evaluating and repurposing it for a given audience. (EV) (GE)</li> <li>Recognises the audience when designing and creating digital content. (EV)</li> <li>Understands the potential of information technology for collaboration when computers are networked. (GE)</li> <li>Uses criteria to evaluate the quality of solutions, can identify improvements and make some refinements to the solution, and future solutions. (EV)</li> </ul>
↓	<ul style="list-style-type: none"> <li>Understands that iteration is the repetition of a process such as a loop. (AL)</li> <li>Recognises that different algorithms exist for the same problem. (AL) (GE)</li> <li>Represents solutions using a structured notation. (AL) (AB)</li> <li>Can identify similarities and differences in situations and can use these to solve problems (pattern recognition). (GE)</li> </ul>	<ul style="list-style-type: none"> <li>Understands that programming bridges the gap between algorithmic solutions and computers. (AB)</li> <li>Has practical experience of a high-level textual language, including using standard libraries when programming. (AB) (AL)</li> <li>Uses a range of operators and expressions e.g. Boolean, and applies them in the context of program control. (AL)</li> <li>Selects the appropriate data types. (AL) (AB)</li> </ul>	<ul style="list-style-type: none"> <li>Knows that digital computers use binary to represent all data. (AB)</li> <li>Understands how bit patterns represent numbers and images. (AB)</li> <li>Knows that computers transfer data in binary. (AB)</li> <li>Understands the relationship between binary and file size (uncompressed). (AB)</li> <li>Defines data types: real numbers and Boolean. (AB)</li> <li>Queries data on one table using a typical query language. (AB)</li> </ul>	<ul style="list-style-type: none"> <li>Recognises and understands the function of the main internal parts of basic computer architecture. (AB)</li> <li>Understands the concepts behind the fetch-execute cycle. (AB) (AL)</li> <li>Knows that there is a range of operating systems and application software for the same hardware. (AB)</li> </ul>	<ul style="list-style-type: none"> <li>Understands how search engines rank search results. (AL)</li> <li>Understands how to construct static web pages using HTML and CSS. (AL) (AB)</li> <li>Understands data interchange between digital computers over networks, including the internet i.e. IP addresses and packet switching. (AL) (AB)</li> </ul>	<ul style="list-style-type: none"> <li>Evaluates the appropriateness of digital devices, internet services and application software to achieve given goals. (EV)</li> <li>Recognises ethical issues surrounding the application of information technology beyond school.</li> <li>Designs criteria to critically evaluate the quality of solutions, uses the criteria to identify improvements and can make appropriate refinements to the solution. (EV)</li> </ul>
↓	<ul style="list-style-type: none"> <li>Understands a recursive solution to a problem repeatedly applies the same solution to smaller instances of the problem. (AL) (GE)</li> <li>Recognises that some problems share the same characteristics and use the same algorithm to solve both. (AL) (GE)</li> <li>Understands the notion of performance for algorithms and appreciates that some algorithms have different performance characteristics for the same task. (AL) (EV)</li> </ul>	<ul style="list-style-type: none"> <li>Uses nested selection statements. (AL)</li> <li>Appreciates the need for, and writes, custom functions including use of parameters. (AL) (AB)</li> <li>Knows the difference between, and uses appropriately, procedures and functions. (AL) (AB)</li> <li>Understands and uses negation with operators. (AL)</li> <li>Uses and manipulates one dimensional data structures. (AB)</li> <li>Detects and corrects syntactical errors. (AL)</li> </ul>	<ul style="list-style-type: none"> <li>Understands how numbers, images, sounds and character sets use the same bit patterns. (AB) (GE)</li> <li>Performs simple operations using bit patterns e.g. binary addition. (AB) (AL)</li> <li>Understands the relationship between resolution and colour depth, including the effect on file size. (AB)</li> <li>Distinguishes between data used in a simple program (variable) and the storage structure for that data. (AB)</li> </ul>	<ul style="list-style-type: none"> <li>Knows the von Neumann architecture in relation to the fetch-execute cycle, including how data is stored in memory. (AB) (GE)</li> <li>Understands the basic function and operation of location addressable memory. (AB)</li> </ul>	<ul style="list-style-type: none"> <li>Knows the names of hardware e.g. hubs, routers, switches, and internet services and protocols e.g. SMTP, IMAP, POP, FTP, TCP/IP associated with networking computer systems. (AB)</li> <li>Uses technologies and online services securely, and knows how to identify and report inappropriate contact. (AL)</li> </ul>	<ul style="list-style-type: none"> <li>Justifies the choice of and independently combines and uses multiple digital devices, internet services and application software to achieve given goals. (EV)</li> <li>Evaluates the appropriateness of digital content and considers the usability of visual design features when designing and creating digital artefacts for a known audience. (EV)</li> <li>Identifies and explains how the use of technology can impact on society.</li> <li>Designs criteria for users to evaluate the quality of solutions, uses the feedback from the users to identify improvements and can make appropriate refinements to the solution. (EV)</li> </ul>
↓	<ul style="list-style-type: none"> <li>Recognises that the design of an algorithm is distinct from its expression in a programming language (which will depend on the programming constructs available). (AL) (AB)</li> <li>Recognises the effectiveness of different models for similar problems. (AL) (AB) (GE)</li> <li>Recognises where information can be filtered out in generalizing problem solutions. (AL) (AB) (GE)</li> <li>Uses logical reasoning to explain how an algorithm works. (AL) (AB) (GE)</li> <li>Represents algorithms using structured language. (AL) (DE) (AB)</li> </ul>	<ul style="list-style-type: none"> <li>Appreciates the effect of the scope of a variable e.g. a local variable can't be accessed from another function. (AB) (AL)</li> <li>Understands and applies parameter passing. (AB) (GE) (DE)</li> <li>Understands the difference between, and uses, both pre-tested e.g. 'while', and post-tested e.g. 'until' loops. (AL)</li> <li>Applies a modular approach to error detection and correction. (AB) (DE) (GE)</li> </ul>	<ul style="list-style-type: none"> <li>Knows the relationship between data representation and data quality. (AB)</li> <li>Understands the similarity between binary and electrical circuits, including Boolean logic. (AB)</li> <li>Understands how and why values are typed in many different languages when manipulated within programs. (AB)</li> </ul>	<ul style="list-style-type: none"> <li>Knows that processors have instruction sets and that these relate to low-level operations that are carried out by a computer. (AB) (AL) (AB) (DE)</li> </ul>	<ul style="list-style-type: none"> <li>Knows the purpose of the hardware and protocols associated with networking computer systems, including MAC addresses. (AB) (AL) (DE) (GE)</li> <li>Understands the client-server model including how dynamic web pages use server-side scripting and that web servers are accessed and why visited and data typed in by users. (AL) (AB) (DE)</li> <li>Recognises that persistence of data on the internet requires careful protection of online identity and privacy.</li> </ul>	<ul style="list-style-type: none"> <li>Understands creative projects that collect, analyse, and evaluate data to meet the needs of a known user group. (AL) (DE) (EV)</li> <li>Effectively designs and creates digital artefacts for a range of digital artefacts. (AB)</li> <li>Considers the properties of media when importing them into digital artefacts. (AB)</li> <li>Documents user feedback, the improvements identified and the refinements made to the solution. (AB)</li> <li>Explains and justifies how the use of technology impacts on society, from the perspective of social, economical, political, legal, ethical and moral issues. (EV)</li> </ul>
↓	<ul style="list-style-type: none"> <li>Designs a solution to a problem that depends on solutions to smaller instances of the same problem recursively. (AL) (DE) (AB) (GE)</li> <li>Understands that some problems cannot be solved computationally. (AB) (GE)</li> </ul>	<ul style="list-style-type: none"> <li>Designs and writes nested modular programs that enforce reusability utilising sub-routines wherever possible. (AL) (AB) (GE) (DE)</li> <li>Understands the difference between 'while' loop and 'for' loop, which uses a loop counter. (AL) (AB)</li> <li>Understands and uses two dimensional data structures. (AB) (DE)</li> </ul>	<ul style="list-style-type: none"> <li>Performs operations using bit patterns e.g. conversion between binary and hexadecimal, binary addition etc. (AL) (GE)</li> <li>Understands and can explain the need for data compression, and performs simple compression methods. (AL) (AB)</li> <li>Knows what a relational database is, and understands the benefits of storing data in multiple tables. (AB) (DE) (GE)</li> </ul>	<ul style="list-style-type: none"> <li>Has practical experience of a small (hypothetical) low level programming language. (AB) (AL) (DE)</li> <li>Understands and can explain Moore's Law. (GE)</li> <li>Understands and can explain multitasking by computers. (AB) (AL) (DE)</li> </ul>	<ul style="list-style-type: none"> <li>Understands the hardware associated with networking computer systems, including WANs and LANs, understands their purpose and how they work, including MAC addresses. (AB) (AL) (DE) (GE)</li> </ul>	<ul style="list-style-type: none"> <li>Understands the ethical issues surrounding the application of information technology, and the existence of legal frameworks governing its use e.g. Data Protection Act, Computer Misuse Act, Copyright etc. (EV)</li> </ul>


Computational Thinking Concepts: AB = Abstraction; DE = Decomposition; AL = Algorithmic Thinking; EV = Evaluation; GE = Generalisation  
 Note: Each of the Progression Pathway statements is underpinned by one or more learning outcomes (due for publication in 2014), providing greater detail of what should be taught to achieve each Progression Pathway statement and National Curriculum point of study. © 2014 Mark Dorling and Matthew Walker. Reviewed by Simon Humphreys and Sue Serence of Computing At School, CAS Master Teachers, and by teachers and academics from the wider CAS community. Computational Thinking mapping undertaken by Mark Dorling, Cyndee Selby and John Woodard.

## [INTERIM] CSTA K-12 COMPUTER SCIENCE STANDARDS

REVISED 2016  
CSTA STANDARDS TASK FORCE



**CSTEACHERS.ORG**  
COMPUTER SCIENCE TEACHERS ASSOCIATION



**COMPUTING AT SCHOOL**  
EDUCATE. ENGAGE. EMPOWER.  
In collaboration with BCS, The Chartered Institute for IT

[Document source](#)

[Document source](#)



@APCsizmadia, @MarkDorling

Slide: <#>



# Supporting the transition from: Block to Text



# IDLE: Script mode

- Interactive mode is great but its not designed to create programs that you can save and run later.
- The Script mode enables you to write, save, open and edit programs.
- File > New File

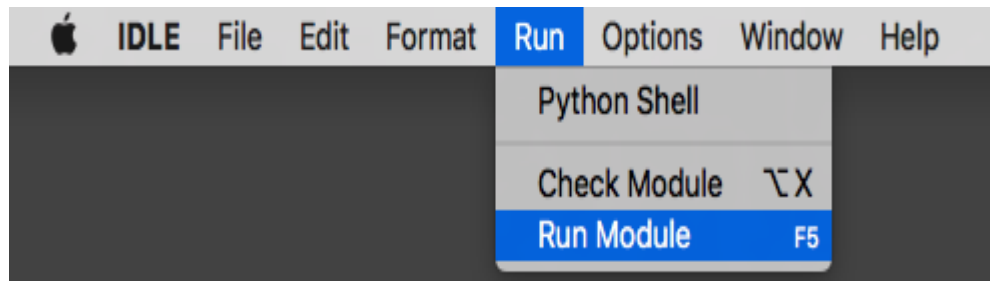


The image shows two windows from the Python IDLE environment. The left window is titled '\*Python 3.4.0 Shell\*' and displays the Python 3.4.0 startup information, including the version, date, and platform. The right window is titled '\*Python 3.4.0: Untitled\*' and contains a single line of Python code: `print('Hello world!')`. The status bars at the bottom of the windows indicate the current line and column numbers.



# IDLE: Script mode

- Run > Run Module



- Save the file.



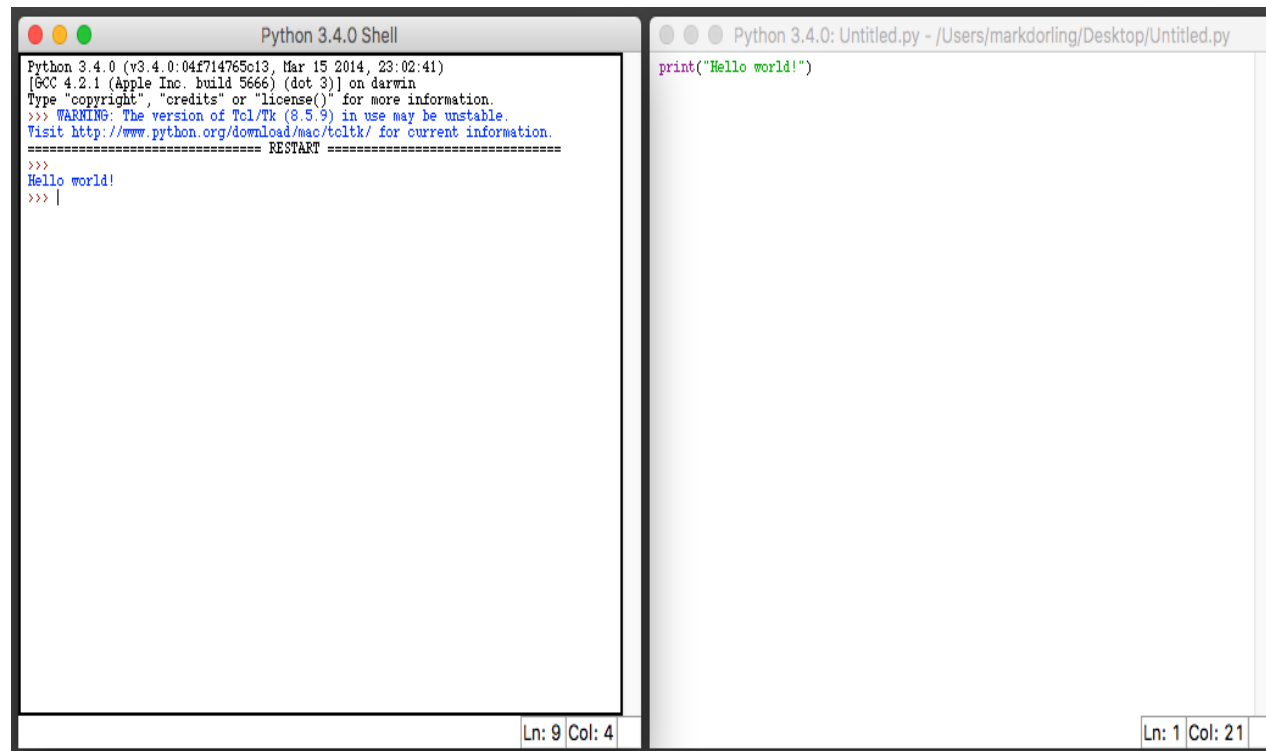
Start the program

Kill the program

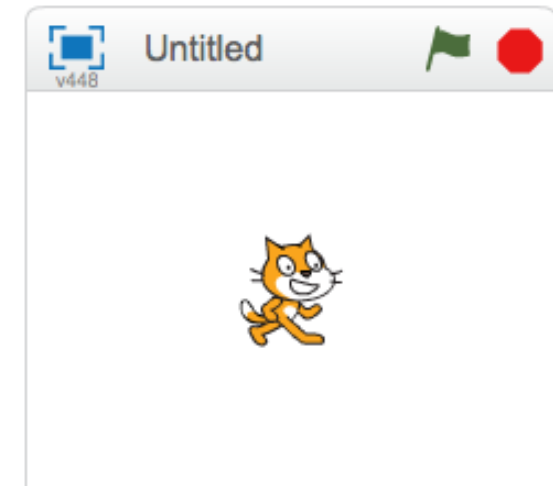


# IDLE: Script mode

- See the results!



The image shows two side-by-side windows from the Python IDLE environment. The left window is titled "Python 3.4.0 Shell" and displays the Python interpreter's startup output, including version information and a warning about the Tcl/Tk version. The right window is titled "Python 3.4.0: Untitled.py" and shows a single line of Python code: `print("Hello world!")`. The shell window shows the output of this code: `Hello world!`. The status bars at the bottom of the windows indicate the current line and column numbers: "Ln: 9 Col: 4" for the shell and "Ln: 1 Col: 21" for the editor.

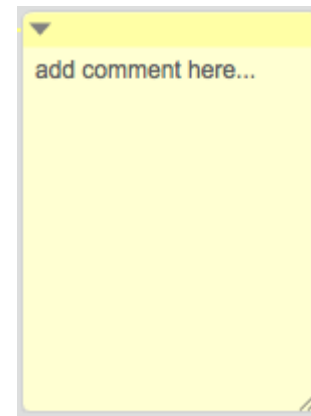




# Using comments

- It is good practice to use comments at the of a program to give:
  - Explanation of what it does
  - The author
  - The version:
  - The date:
  - Any copyright information

```
#####  
#  
# Explanation of what the program does  
#  
# Author :  
# Version :  
# Date:  
#####
```



# Wait in Python

- There is no print for 'n' seconds in Python like in Scratch and Snap!



- How do we get around this?
- **Python's time module has a handy function called sleep()**

SCRATCH λ Snap!

# Wait (sleep) in Python

- The syntax is this: `time.sleep(secs)`
- We need to remember to import the time module.

```
import time

for i in range (1,10):
    print("Hello World!")
    time.sleep(1)
```

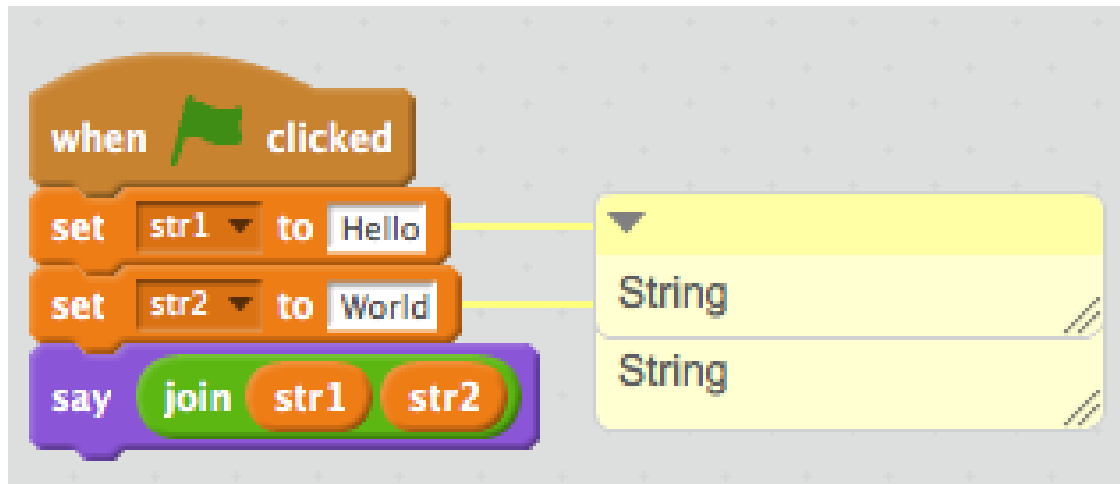


SCRATCH



λSnap!

# Introducing: Data Types



```
str1 = str("Hello")  
str2 = str("World!")  
  
print(str1 + str2)
```



# Introducing: Concatenation

```
print "red" + 3
```



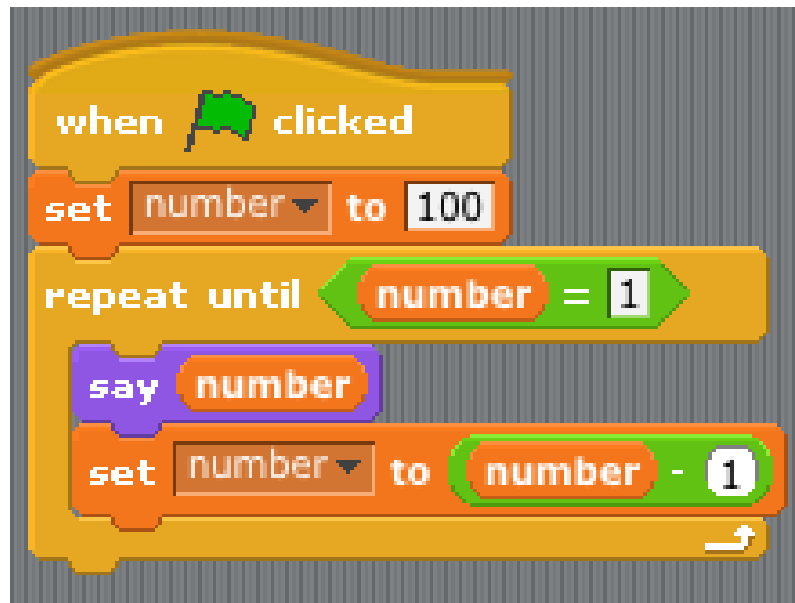
Traceback (most recent call last): File "", line 1, in  
TypeError: cannot concatenate 'str' and 'int' objects

- Python doesn't know how to add a word and a number, so it says "cannot concatenate 'str' and 'int' objects."
- You can't add a str and an int together. But you can turn an integer into a string if you use the str() function.

```
print "red" + str(3)
```



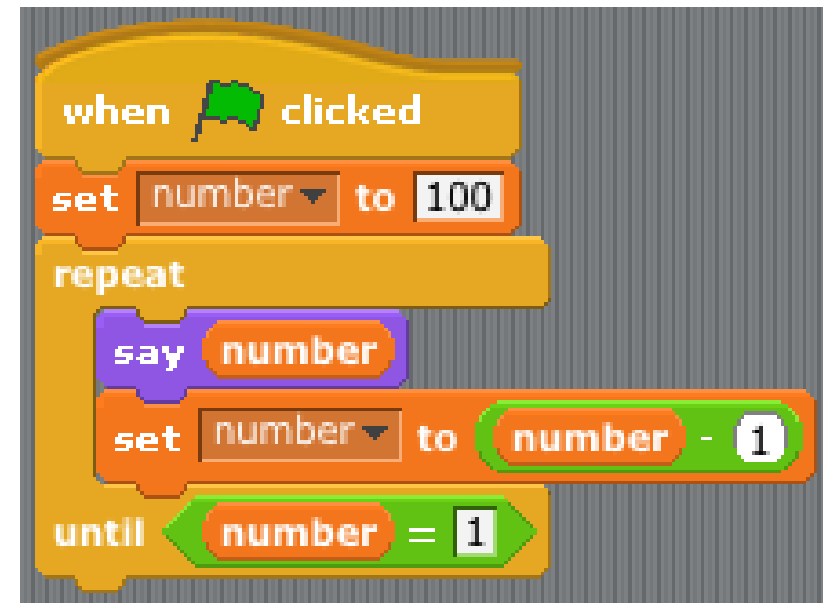
# Bottom/post-tested loops



```
when green flag clicked
  set number to 100
  repeat until number = 1
    say number
    set number to number - 1
```

The Scratch code block shows a 'when green flag clicked' event trigger. It sets a variable 'number' to 100. A 'repeat until' loop is used, with the condition 'number = 1'. Inside the loop, the variable 'number' is displayed on the stage and then decremented by 1.

SCRATCH



```
when green flag clicked
  set number to 100
  repeat
    say number
    set number to number - 1
  until number = 1
```

The Snap! code block shows a 'when green flag clicked' event trigger. It sets a variable 'number' to 100. A 'repeat' loop is used, with the condition 'until number = 1'. Inside the loop, the variable 'number' is displayed on the stage and then decremented by 1.

λSnap!

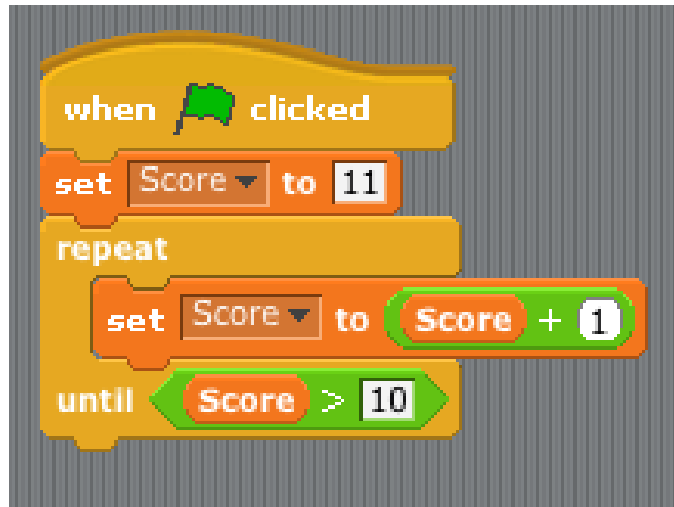
What if the count variable is set to 1 at the start of the program and not 100?

# Introducing: Top & bottom tested loops

Repeat

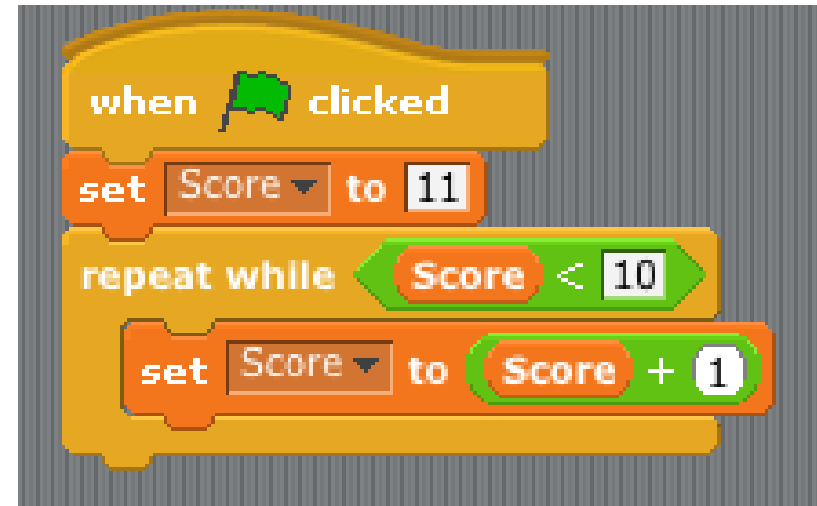
```
print("Enter name of item")
input(item)
print("Another item?")
input(response)
```

Until response = 'N' or "No"



Repeat While response = 'N' or "No"

```
print("Enter name of item")
input(item)
print("Another item?")
input(response)
```





# Range of languages



The image shows a Scratch script starting with a 'when clicked' event. The code blocks are: 'ask What number? and wait', 'set VarA to answer', 'set VarB to pick random 1 to 10', 'set VarC to VarA + VarB', and 'say VarC'. Below the code, three variable monitors are shown: VarA with value 5, VarB with value 2, and VarC with value 7. A speech bubble containing the number 7 is positioned above the Scratch cat character.

```
Python 3.3.3 Shell
Python 3.3.3 (v3.3.3:c3896275c0f6,
Nov 16 2013, 23:39:35)
[GCC 4.2.1 (Apple Inc. build 5666)
(dot 3)] on darwin
Type "copyright", "credits" or "lic
ense()" for more information.
>>> WARNING: The version of Tcl/Tk
(8.5.9) in use may be unstable.
Visit http://www.python.org/downloa
d/mac/tcltk/ for current informatio
n.

>>> import math
>>> import random
>>> print("what number?")
what number?
>>> VarA = int(input())
8
>>> VarB = random.randint (1, 10)
>>> VarC = VarA + VarB
>>> print (VarC)
14
>>> |
```

Ln: 17 Col: 4

# Range of problems

- Range of recursive and iterative activities
  - 10 green bottles
  - One man went to Mow
  - Fibonacci and Lucas series
- Rock Paper Scissors
- Hang man style games
- Searching Algorithms
  - Linear
  - Binary
- Sorting algorithms
  - Bubble
  - Insert
  - Merge
  - Selection

# Transition...

KS3/4



KS3



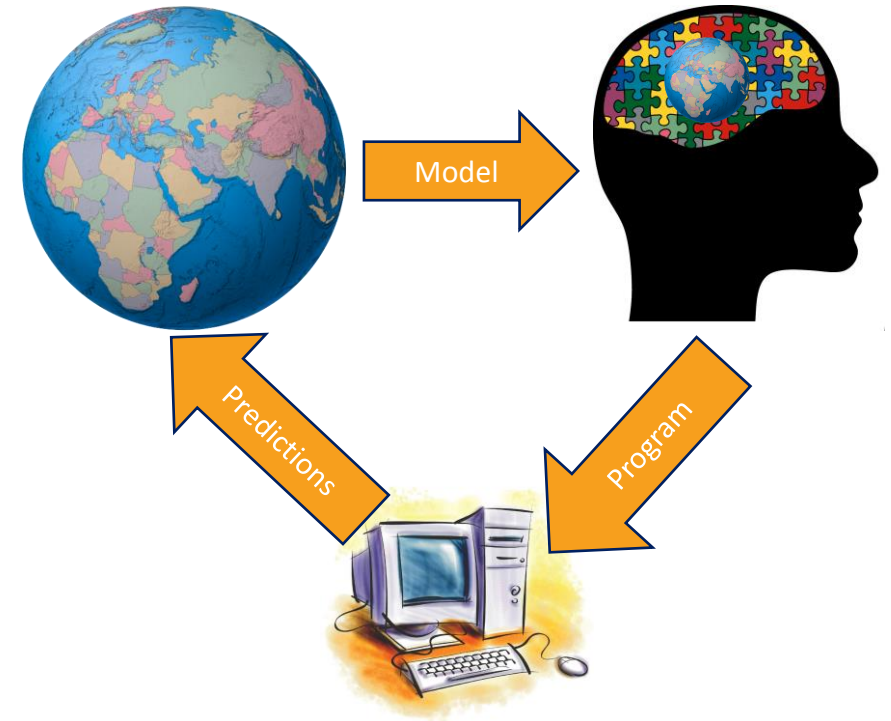
KS2



# Exploring programming pedagogy

# Why programming?

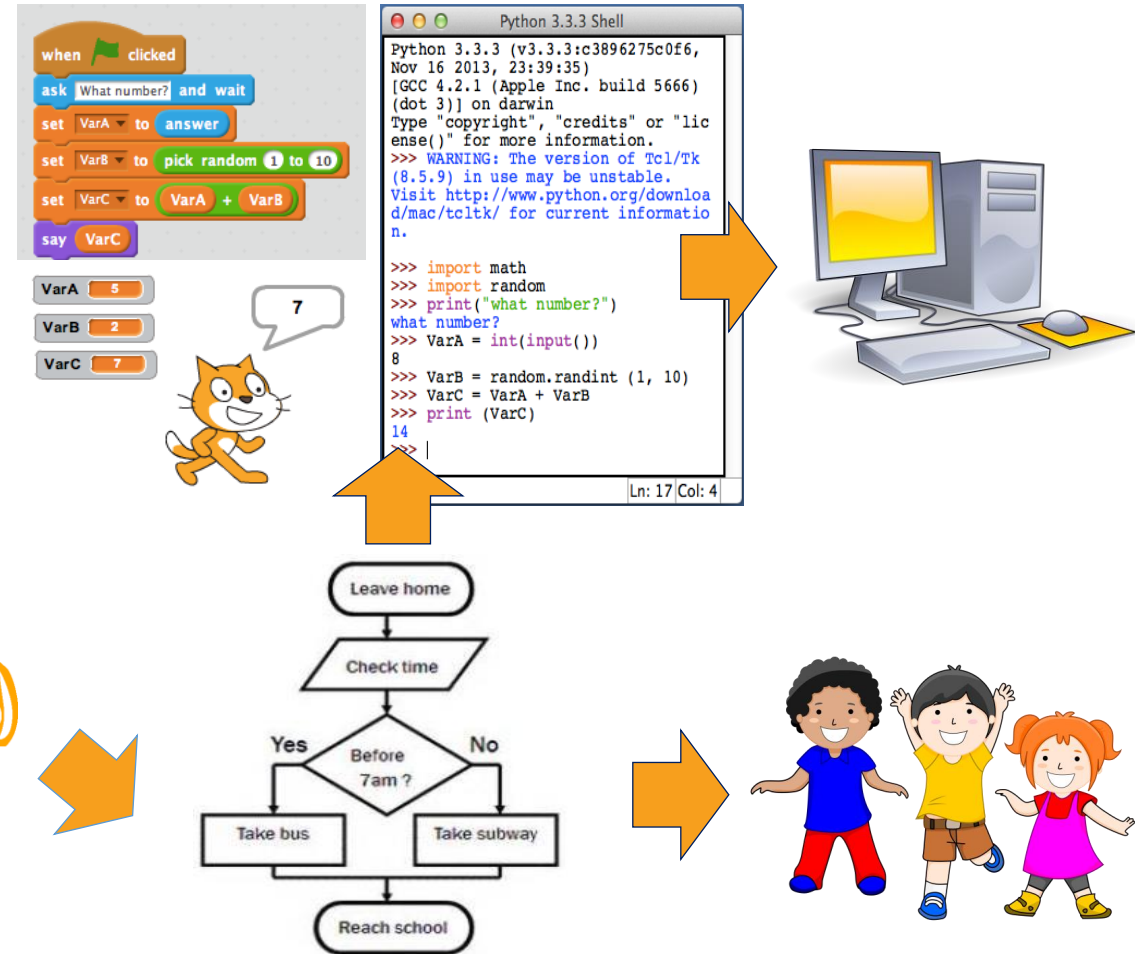
- We want to make models of the world to:
  - Understand it
  - Ask “what if” questions and predict the way it will change
- How do we make models?
  - Solving problems
    - Asking good questions
    - By characterising a problem
      - Looking for similar problems you already know how to solve
      - Think about what makes a problem similar to another



(Professor Greg Michaelson, Heriot-Watt University)

# Abstractions...

- How do we turn models into programs?
  - Writing programs based on algorithms
  - Programming bridges gaps between thinking and computers?
  - Choosing appropriate technology



# Our aim for learners is to...

- The McCracken group focused on problem solving in programming and suggested a 5-step process that students should learn:
  1. Abstract the problem from its description (**Abstraction**)
  2. Generate sub-problems (**Decomposition**)
  3. Transform sub-problems into sub-solutions (**Generalisation and Algorithmic Thinking**)
  4. Re-compose, and
  5. Evaluate and iterate (**Evaluation**)

**McCracken et al. (2001)**



# Challenges for novice programmers

- Students might lack skills that are a precursor to problem-solving.

**Lister et al. (2004)**

- Being able to read and trace code is really important pre-cursor to the problem-solving needed to write code.

**Lister et al. (2008)**

- Novice programmers need to be able to trace code with greater than 50% accuracy before they can independently begin to write programs.

**Lister (2011)**

# Essential Programming Skills

<b>Create</b>		<b>Design:</b> Devise a solution structure <b>Apply:</b> Use solution as a component in a problem	<b>Model:</b> Illustrate or implement an abstraction of a problem	<b>Refactor:</b> Redesign a solution for optimization
<b>Apply</b>	<b>Implement:</b> put a completed design into code	<b>Adapt:</b> modify a solution for other domains <b>Translate:</b>	<b>Debug:</b> Both detect and correct flaws in design	
-	<b>Recognise:</b> Base knowledge and vocabulary for the domain	<b>Trace:</b> Desk-check a solution	<b>Present:</b> Explain a solution to others <b>Analyse:</b> probe the [time] complexity of a solution	<b>Relate:</b> Understand solution in context of others
	<b>Know</b>	<b>Understand</b>	<b>Analyse</b>	<b>Evaluate</b>

**Fuller et al. (2007)**

# Use - Modify - Create

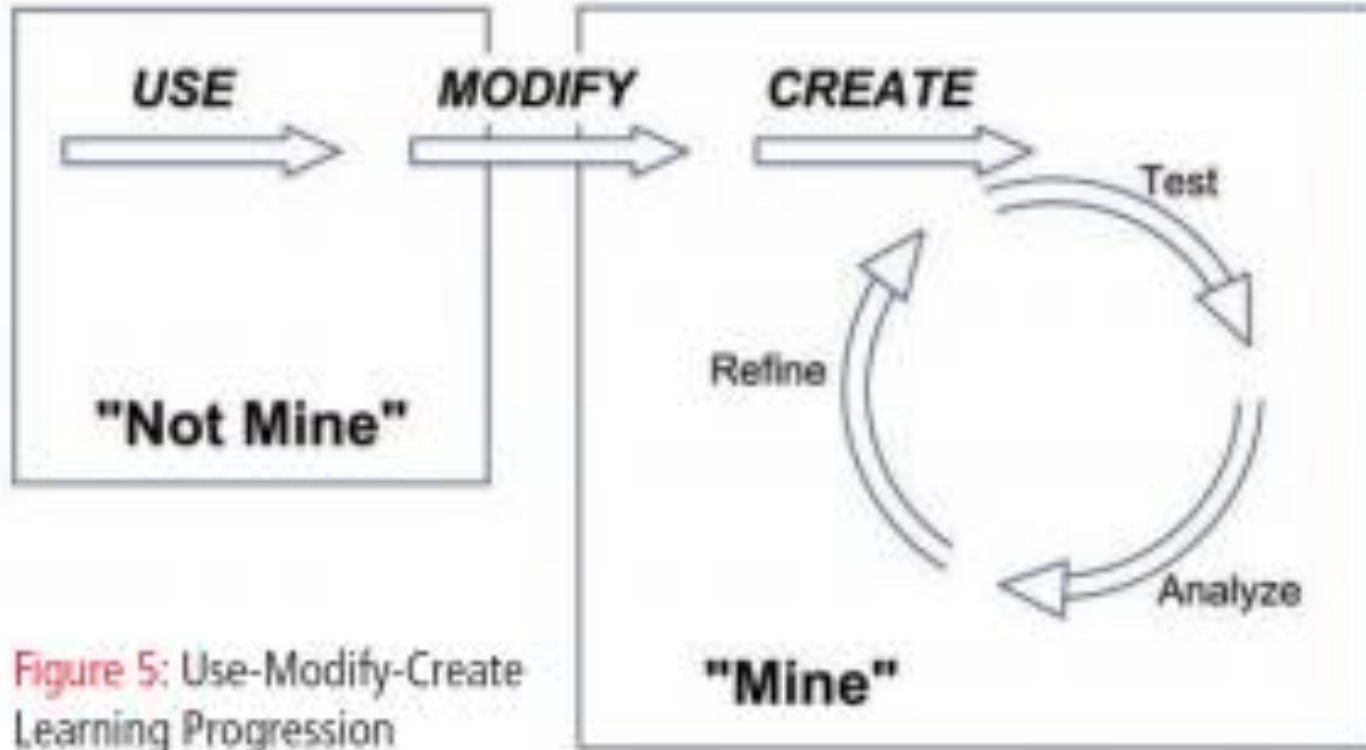
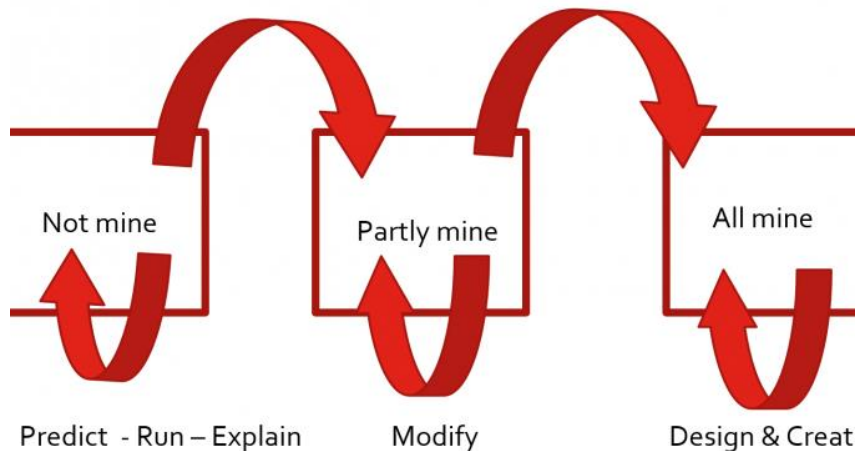


Figure 5: Use-Modify-Create Learning Progression

Lee et al. (2011)

# Use - Modify - Create



- **Predict** – given a working program, what do you think it will do? (at a high level of abstraction)
- **Run** – run it and test your prediction
- **Explain/Articulate** – What does each line of code mean? (low level of abstraction). I'm not sure that explain is quite the right term.
- **Modify** – edit the program to make it do different things (high and low levels of abstraction)
- **Create/Design** – design/create is a key computational thinking skill.

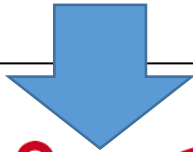
**Sentence. (2017)**

# Pulling this together

	Lee et al. (2011)	Fuller et al. (2007)	Sentence (2017)
<b>CONSTRUCT</b>	Novice coders constructing an understanding by using and running code provided	Construct knowledge: new vocabulary and knowledge about construct, reading and tracing, explaining, analysing and evaluating code	The code is 'not theirs'
<b>CHANGE</b>	Novice coders begin modifying and adapting code provided	Applying knowledge: Implementing code from a given design, adapting code for a different purpose, finding and correcting errors in code (debugging).	Making the code 'partly theirs'
<b>CREATE</b>	Novice coders begin designing and writing their own code	reusing solutions in bigger solutions, designing and coding solutions, fixing smelly code (commenting and optimising code).	The code is 'fully theirs'

# Historical: Choice of pedagogy

**Constructivism:** The learner is not a passive recipient of knowledge but that knowledge is 'constructed' by the learner.



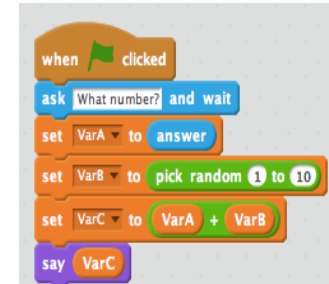
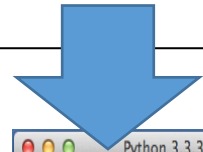
The Magic of Computer Science  
[www.cs4fn.org/magic/](http://www.cs4fn.org/magic/)



**Social Constructivism:** Groups construct knowledge for one another, collaboratively creating a smaller culture of shared artifacts with shared meaning.



**Constructionism:** The idea that learners' learn best through building things that are tangible and sharable with the public



```
Python 3.3.3 Shell
Python 3.3.3 (v3.3.3:c3896275c0f6,
Nov 16 2013, 23:39:35)
[GCC 4.2.1 (Apple Inc. build 5666)
(dot 3)] on darwin
Type "copyright", "credits" or "lic
ense()" for more information.
>>> WARNING: The version of Tcl/Tk
(8.5.9) in use may be unstable.
Visit http://www.python.org/downloa
d/mac/tcltk/ for current informatio
n.

>>> import math
>>> import random
>>> print("what number?")
what number?
>>> VarA = int(input())
8
>>> VarB = random.randint(1, 10)
>>> VarC = VarA + VarB
>>> print(VarC)
14
>>> |
```

# Example pedagogies

- **Code walkthroughs:** Learners step through code predicting outputs
- **Collaborating on solutions:** Writing algorithms and code in groups
- **Scaffolding:** Insert comments in into existing code
- **Code debugging:** Finding errors in given code e.g. spot the difference
- **Flipped learning:** Class time to collaborate and compare solutions

**Van Gorp and Grissom (2001)**



# Supporting constructivist

- Using examples that are **relevant to students' own experiences** e.g. relating to real-world experiences
- **Active learning experiences** e.g. unplugged, kinesthetic activities
- **Learning by exploration** e.g. exploring programming environments and open-ended tasks
- **Learning by solving problems** e.g. self-directed projects and problem-solving
- **Open-ended discussion and working in groups** e.g. paired and group problem-solving.

# Supporting social constructivist

- Software developers generally spend:
  - **30%** of their time working alone
  - **50%** of their time working with one other person
  - **20%** of their time working with two or more people

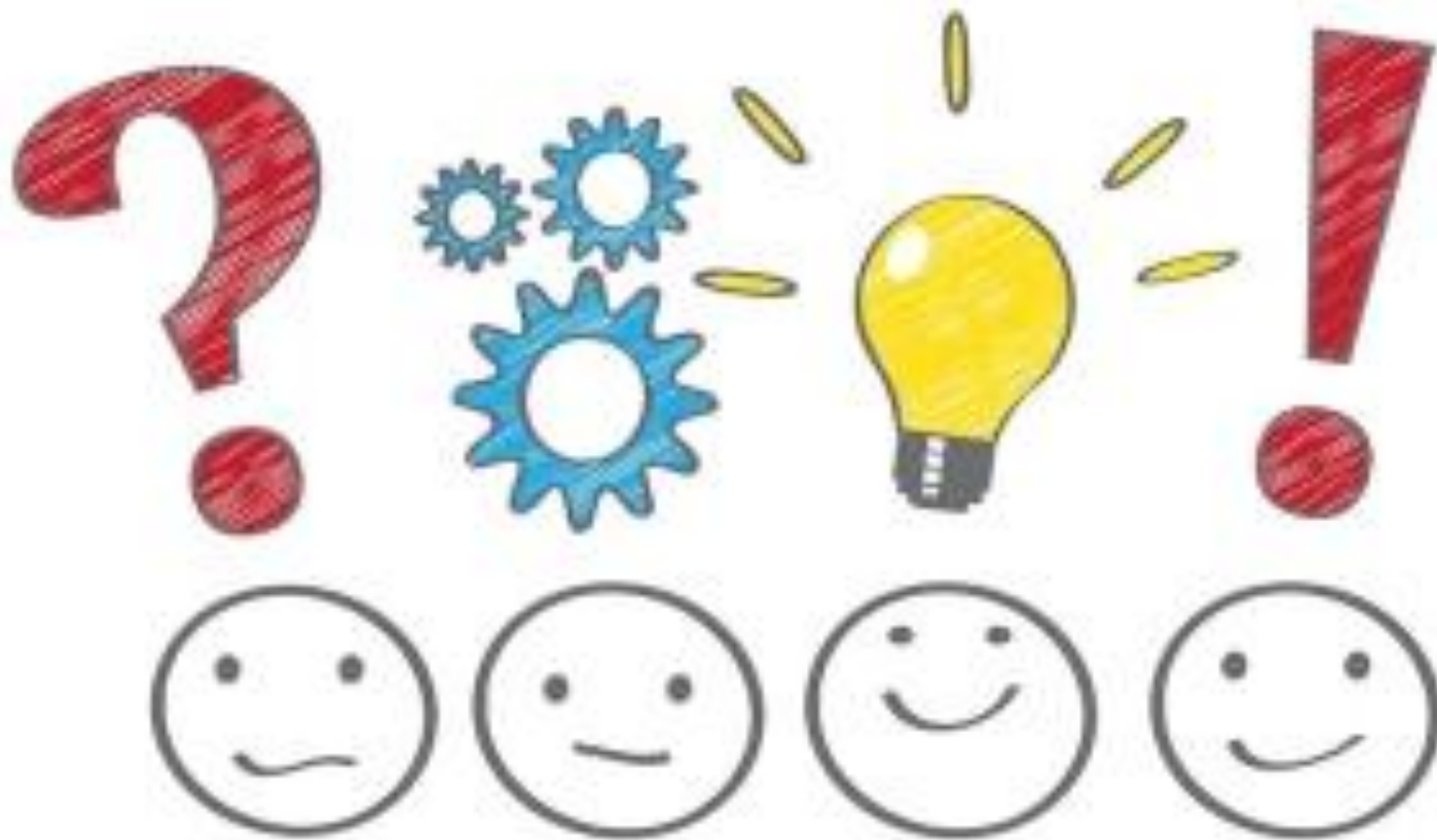
**DeMarco and Lister (1987)**

Three forms of peer-based interaction in the classroom:

1. Tutoring, where the less capable are guided by the more capable;
2. Co-operation, where learners work on different parts of the task;
3. Collaboration, where learners work jointly on almost all parts of the task.

**Jehng (1997)**

# Thanks for listening!



# References

- AQA GCSE Computer Science:  
<http://www.aqa.org.uk/subjects/computer-science-and-it/gcse/computer-science-8520>
- Cambridge iGCSE Computer Science:  
<http://www.cie.org.uk/programmes-and-qualifications/cambridge-igcse-computer-science-0478/>.
- Csizmadia, A., Cuzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C. and Woollard, J. (2015). *Computational thinking: A guide for teachers*:  
<https://community.computingatschool.org.uk/resources/2324>
- Computing At School (2015). *Lesson observation form with prompts*:  
<http://www.quickstartcomputing.org/secondary/section4.html>
- Department for Education (2014). *Computing Programmes of Study*: <https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study>

# References

- DeMarco, T., and Lister, T. (1987). *Peopleware*. New York: Dorset House Publishers.
- Dorling, M. & Stephens, S. (2016). *Problem solving and Computational Thinking rubric*: <http://community.computingschool.org.uk/resources/4793>
- Dorling, M. & Walker, M. (2014), *Computing At School Progression Pathways*: <https://community.computingschool.org.uk/resources/1692>
- Edexcel GCSE Computer Science: <http://qualifications.pearson.com/en/qualifications/edexcel-gcses/computer-science-2016.html>
- Fuller et. Al. (2007). *Developing a computer science-specific learning taxonomy*: <http://dl.acm.org/citation.cfm?id=1345438>

# References

- Jehng, J. C. J. (1997). The psycho-social processes and cognitive effects of peer-based collaborative interactions with computers. *Journal of Educational Computing Research*, 17, 19-46.
- Hazzan, O., Lapidot, T. and Ragonis, N., (2015). *Guide to teaching computer science: An activity-based approach*. London: Springer.
- Lister, R. (2011). *Concrete and other neo-piagetian forms of reasoning in the novice programmer*.
- Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., . . . Thomas, L. (2004). *A multi-national study of reading and tracing skills in novice programmers*.
- Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., . . . Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads*, 2, 32-37.

# References

- Lopez, M., Whalley, J., Robbins, P., & Lister, R. (2008). *Relationships between reading, tracing and writing skills in introductory programming*.
- McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagen, D., Kolikant, Y., Laxer, C., Thomas, L., Utting, I., and Wilusz, T. (2001). A Multi-National, Multi-Institutional Study of Assessment of Programming Skills of First-year CS Students. SIGCSE Bull., 33(4). pp 125-140.
- OCR GCSE Computer Science:  
<http://www.ocr.org.uk/qualifications/gcse-computer-science-j276-from-2016/>.
- Roth, W.-M. (1993). Construction sites: Science labs and classrooms. In K. Tobin (Ed.), *The practice of constructivism in science education*, (pp. 145-170). Hillsdale, NJ: Erlbaum.

# References

- Sentance, S. and Csizmadia, A., (2015). Teachers' perspectives on successful strategies for teaching Computing in school
- Van Gorp, M. J., and Grissom, S. (2001). An empirical evaluation of using constructive classroom activities to teach introductory programming.
- Williams, Wiebe, Yang, Ferzli, and Miller, "In Support of Pair Programming in the Introductory Computer Science Course," Computer Science Education, vol. 12, pp. 197-212, 2002.
- WJEC GCSE Computer Science:  
<http://wjec.co.uk/qualifications/computer-science/computer-science-gcse/>