Why use Python to teach programming? Read by 297 members

« back to Secondary Education

Yus Kaz
FE Teacher
Teacher Trainer
IT Professional

May 03 2016, 11:17

Can anyone shed light on why Python is the preferred programming for GCSE Computer Science? Everyone seems to be gearing up to use it. But I can't see the logic in it. Is it just a case of jumping on the bandwagon? I would have thought using Visual basic of C# would have been the logical choice. Not only is the programming environment more user friendly and intuitive, but the programming syntax relates better to the theory questions form the exam boards. I have found Python to the difficult, very user unfriendly and worst of all cuts corners on fundamental programming concepts. I can understand experts using it but as a teaching tool, for me is a bad choice. Enough to put people off programming forever! Just to mention a couple of issues I have come across. As far as I can gather Python doesn't use Arrays in the traditional way as other languages do. Also, standard string functions are not part of Python. These are theory concept on the specification and are frequently found in exam questions. Why teach a programming language whose syntax is different to the syntax used on the exams? Isn't that just asking for trouble?

9 people like this. Like Not helpful

Stuart Davison
Master Teacher
HE Academic
Teacher Trainer

May 03 2016, 11:41

I like Python and have used it successfully for the current GCSE. I chose it for the low barrier to getting started, the simplicity of the syntax, the forced indentation, etc. That said I constantly found myself saying "you need to know about xyz but we can't do that in Python" which wasn't ideal. I especially didn't like pretending that lists were arrays!

However, I found that as the requirements of the subject became more specific, Python perhaps wasn't the best choice of language. Things like not being able to pass by reference, limited use of arrays, inability to define constants, no DO WHILE loop, and so on, became quite limiting.

If I were still teaching now I would be tempted to use Python at KS3, VB at KS4 (closer to the pseudocode in the exam questions), and something like C# at KS5.

6 people like this. Like Not helpful

Glen Thomas
Teacher (11-18)
glen.thomas@physics.org

May 03 2016, 13:25

I know arrays are not one of the base types, but they are a simple module install and 'import numpy as np' away - these arrays make Python powerful enough for scientists and professional data analysts to use it routinely. The lack of Do While syntax does not stop you using other syntax to do the exact same thing (infinite loops with explicit exits). And constants are easy to enact - just don't redefine them.

@Yuz: " Also, standard string functions are not part of Python. "

String functions are mostly invoked as methods, but I haven't come across any absences in my day to day use of Python. Which string functions have you found missing?

What I DO like about Python for teaching is the interactive shell, the clean looking syntax and the speed with which you can move from having an algorithm in your head to a working piece of code.

2 people like this. Like Not helpful

Stuart Davison
Master Teacher
HE Academic
Teacher Trainer

May 03 2016, 15:02

You can work around a lot of this but, as Yus alluded to, other languages fit the specifications more closely. This makes it easier for pupils to grasp the key concepts we are trying to teach without adding another layer of complexity on top that obscures the true meaning.

Not sure you can really get around the lack of constants though (a variable you trust someone not to change is not a constant) or the pass by reference issue.

2 people like this. Like Not helpful

Owen Waller
HE Academic
Teacher Trainer
IT Professional

May 03 2016, 15:36

Interestingly, I am currently preparing a CAS blog post on a very similar topic!

Watch this space :)

Owen

2 people like this. Like Not helpful

Glen Thomas
Teacher (11-18)
glen.thomas@physics.org

May 03 2016, 16:40

Hi Stuart. Yes, I somewhat agree with you on all your points: there are trade offs no matter which language you use to teach. It all depends on personal overall judgements of pros vs cons. And there are cons for all possible choices.

For me, I'm happy to work around the departures from a more spec-focussed language for the reasons I gave. I would rather compromise on having different syntax than lose those features that allow for rapid prototyping and error feedback, but I can see the advantages of other options.

(I also really do like the interactive shell in the classroom for quick demonstrations short algorithm or syntax demos. Compiling and linking are simple, but can get in the way also, and are not part of the computational thinking objectives I want from most programming tasks.)

Like Not helpful

May 03 2016, 17:35

I was strongly anti-Python as a teaching language, and still have major misgivings. Much (most?) of the literature is gloriously one-sided and IMHO deceptive in pretending that Python is good at many things it sucks at.

Adam Martin
Teacher (11-18)
IT Professional
CS Grad

Significant whitespace is a cruelty we shouldn't be inflicting upon anyone. Especially when most Ss will have done years of HTML by the time they start Python, and already discovered that non-significant whitespace is the norm in programming.

Overall, I believe that if you learn programming primarily via Python you'll be a weaker programmer.

BUT … it has some huge benefits, IME:

1. The lack of syntax reduces the amount of "weird symbols" you have to explain to flummoxed Y8 / Y9's
2. Ditto them explaining to each other. Ditto them debugging line-by-line.
3. It fulfils many of the essential features of "My first programming language": case sensitivity, functions, traditional control structures (if/for), etc.
4. Basic input-handling is simpler than in any non-scripting language out there. This lets students rapidly start writing basic input code.
5. The small amount of syntax means students can more rapidly type in code by "read it and copy it".
6. The IDE ought to be irrelevant. Yes, IDLE sucks several ways … so don't use IDLE :). The major IDE vendors generally support Python directly by now, so anything (insert language of choice here) does, also exists for Python.

(I've heard a lot of schools say they use VB mostly because it has an IDE. This is disappointing - the same IDE's exist for all the languages)

4 people like this. Like Not helpful

Adam Martin
Teacher (11-18)
IT Professional
CS Grad

May 03 2016, 17:42

…my ideal plan is something like:

1. Use Scratch/blockly/etc in Y7 upwards to get everyone on level playing field, until we reach a point where Primary schools are consistently providing us with Ss who've done similar minimum levels of Scratch.
2. Use Python as the soft intro to textual programming languages - it's great to get started with. This is a Y8/Y9 sampler, to get them keen on choosing Computing at GCSE.
3. Re-teach everything in Y9 (or Y10) with more rigour, deeper understanding, more complex problems … in a more suitable language. Possibly a scripting language - e.g. Java (beanshell), Ruby (maybe?), Javascript - or possibly a mainstream compiled language - e.g. C#, Java.
4. At Y12/13, ideally switch to a different language, partly so that non-GCSE-takers aren't so far behind (partly because it's excellent background for AS/A-level to be able to answer questions w.r.t. the multiple languages they know … and should help them write their CA/project-work/etc (more experience/knowledge))

…but I'm constantly reappraising that. e.g. I've no idea how long we'll keep using microbits - will they be an all-class activity in Y8? Y9? … or will the students tire of them, outgrow them?

3 people like this. Like Not helpful

Stuart Davison
Master Teacher
HE Academic
Teacher Trainer

May 03 2016, 19:27

@Glen: I think with the specifications changing there are too many nuances to overcome. More so with the focus shifting to examination and there is less room for ambiguity. It's a shame because Python works well for the reasons you mentioned.

@Adam: I had a similar plan at my previous school. It works well shifting to a new language every Key Stage together with a shift of paradigm, i.e. Structured -> Procedural -> OOP.

2 people like this. Like Not helpful

Paul Powell
Teacher (11-16)
IT Professional
Curriculum Lead

May 03 2016, 19:58

Everyone uses Python because, well, everyone uses Python. The ecosystem of resources and CPD seems to have coalesced around Python and so that is what people go with.

I'm ditching Python for next year's year 10.

Like Not helpful

Phil Gardner
Master Teacher
Hub Leader

May 03 2016, 20:04

The name helped. It sounded cool - same as when Java was introduced. Bit trendy. Schools unsure of what to use tended to go with what they heard other schools were adopting.

Also popular as it is free and can be used on PC, Mac or Pi.

Highly visible on Pi with snakey-icon so people knew it was there and more likely to try it.

Very easy to write a simple first program that makes something happen on screen, turtle drawing or perform simple calculations using numbers, so caught on with people that had not done much teaching of programming in the classroom. Getting further into GCSE work has proved more difficult for some after initial intro.

1 person likes this. Like Not helpful

Glen Thomas
Teacher (11-18)
glen.thomas@physics.org

May 03 2016, 21:16

Python is widely used by those who are not developers and just want to get the job done, and by developers who want rapid prototyping or easy to develop and maintain infrastructure Google/Youtube use Python widely, while the Ricardo Engineering multinational near me use Python for scripting and glueing applications together.

CERN physicists, data analysts, modellers, mathematicians, engineers, chemists: all widely use Python professionally. It is a developer's bias to denigrate a popular and widely used language as a "My first programming language" or to suggest that its use is just a fashion that limits the progress of students.

I am not training developers in school - that is for universities or companies. More of my students will be going into non-developer careers where they will need to be able to quickly generate code to solve problems. Python does that well, with SciPy, NumPy and Matplotlib and other high class libraries. Other languages may be fasster to run, but not faster to write or modify, and time is valuable to many professionals.

Mostly, the choice of language is best decided based on the abilities and teaching styles of their teachers - it is ultimately irrelevant as long as the objectives are principles over syntax.

6 people like this. Like Not helpful

Miles Berry
HE Academic
Teacher Trainer

May 03 2016, 21:43

@Glen

All right, but apart from the sanitation, the medicine, education, wine, public order, irrigation, roads, the fresh-water system, and public health, what has Python ever done for us?

7 people like this. Like Not helpful

Paul Revell
Teacher (11-18)
School Governor
paul.revell@lakes.cumbria.sch.uk

May 03 2016, 21:56

I don't focus on Phyton, but we dabble and use it in Visual Studio. Normally we use VB.net

I think the [and] are better than the (and) for lists/arrays because they are easy to type. But I don't like the colons, likewise I don't like the semicolon terminators in other langs. Students sometimes forget them.

I think the indents can be a fag at times, give me an End If any day. Students see that better.

I haven't decided if I like dictionaries for students. Likewise lists of lists. But it's nice to have them.

I like the way Phyton handles very very very big numbers.

Having said all that, the person up-thread had it, when they said it's all about people looking for the most convenient way in. If you feel a bit lacking in confidence always go with the majority.

I think using it as the 'taster' for GCSE because it's the lowest bar is a bit of a red herring. The real taster IMO is the algorithms and students do all that with Scratch in Y8 and Y9. Text based work comes after the options process has started. They can always opt out if they get freaked. If they like the applied maths of figuring out algorithms and then implementing them in Scratch, you know they will enjoy the GCSE programming. Shame the gov made the boards test that in writing though. That's not Type 1 Fun.

Like Not helpful

Bruce Nightingale
Teacher (11-18)
bpnightingale@gmail.com

May 03 2016, 22:32

Python is now the most popular introductory teaching language at top U.S. Universities.

Source: http://m.cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-us-universities/fulltext

Extract:

"Top-ranked CS departments at MIT and UC Berkeley recently switched their introductory courses to Python. The largest three MOOC providers (edX, Coursera, and Udacity) all offer introductory programming courses in Python. And professors in diverse subfields (e.g., Lorena Barba) are now advocating for teaching Python to novices."

2 people like this. Like Not helpful

Mike Davis
Teacher (11-18)
mike@mike-davis.net

May 03 2016, 23:30

The disconnect between exam specs and Pythonicism (Pythonicness? Pythonicity?) worries me, as per posts upthread. VB.Net has proved a pretty good fit with the GCSE so far. It's all a bit swings and roundabouts, really. Loads of support is quite compelling.

Like Not helpful

Adam Martin
Teacher (11-18)
IT Professional
CS Grad

May 04 2016, 07:47

To add: my experience from industry is that few people are using it compared to the amount that Python evangelists claim.

When I dig deeper, it's almost always "some giant corporates *who use everything* are using it" and "the academic departments, the ex-academics, the research divisions … prefer python". The bias is strong. One academic friend suggested this is because it's become the language of choice for published code in research papers.

For my students, I wanted something non-academic to aim for. The Civilization games (Civ 3 onwards) fit - Python traditionally has too low performance for games, but these made an exception (I can't comment on performance today. IDLE is very slow but that's probably just IDLE).

Right now I'm working my way through different depts at the publisher, trying to secure permission for schools to install copies for free, so we can use them in lessons. I think there are great possibilities if they say yes.

3 people like this. Like Not helpful

Mark MacInnes
Teacher (11-18)
IT Professional

May 04 2016, 08:02

The major point for me is Python's lack of punctuation, which means students focus more on solving problems using a language rather than "why's my code not running" which I have had with other languages. I also quite like Python. It has everything that I need at Key Stages 4 & 5, and it is used professionally (which helps to motivate *some* children). Of course it has flaws, but then so do all languages so I don't care about that argument.

My ideal would be to start with Python/Ruby (did Ruby for a year and really liked it) in Key Stage 3, then introduce a different language at Key Stage 4. I haven't decided which, but I have a shortlist. The biggest problem I have with introducing another language is time. I need time to create good quality resources that fit our kids and our courses.

1 person likes this. Like Not helpful

Moya Shannon
Teacher (16-18)

May 04 2016, 08:25

This presupposes that all computer science departments in schools have specialist computer science teachers with any number of programming languages at their finger tips - which they don't. A thread like this is intimidating for those of us who are struggling to learn any programming language to a level adequate enough to deliver it at GCSE. I had a parent ask me why I was teaching Python and not PHP - did he then go to the languages department and ask them why they're teaching French? I heard of a school where the teachers at KS3 teach in their preferred language - because the HOD sees computational thinking as the focus and not the specific language. I introduced lists yesterday - to Year 8 - and got them to create a Shakespearean insult generator program - they enjoyed it and were engaged with the task - some stayed after the bell to finish off their programs - and the keen ones developed their programs further. Who knows one day they might get as far as A level and beyond. I'm just hoping that Python will be OK for GCSE - just in case Computer Science does - one day - become a GCSE option in our school.

14 people like this. Like Not helpful

Mark MacInnes
Teacher (11-18)
IT Professional

May 04 2016, 08:35

@Moya Python is fine for both GCSE and Alevel. As you rightly point out, the language is the medium to teach the theory. If the language works for you then it works for the kids.

3 people like this. Like Not helpful

Stuart Davison
Master Teacher
HE Academic
Teacher Trainer

May 04 2016, 08:58

@Moya: Great point! Python is an excellent way in to programming. I think what a lot above are trying to say is that it is not a panacea for all CS teaching in Secondary.

I look at it from the viewpoint of the pupils learning to program and especially at GCSE and A-level. Right or wrong, teaching at this level is limited by the qualifications delivered. I feel Python makes it slightly more difficult to teach concepts in the specifications that other languages are more suited to do so.

In a way, starting with a different language each time helps level the playing field and acts as a way of revising key concepts.

Once you have a handle on programming in general switching languages isn't as daunting as it seems. A bit like how learning to speak Italian is easier having learnt to speak French.

3 people like this. Like Not helpful

Mike Davis
Teacher (11-18)
mike@mike-davis.net

May 04 2016, 09:56

Moya makes a good point. Just because Python might not be ideal doesn't make it a bad choice. You're definitely far better off using what you know rather than something you don't but that fits the theory a bit better :)

1 person likes this. Like Not helpful

Phil Gardner
Master Teacher
Hub Leader

May 04 2016, 10:26

Glen - I don't think of Python as "My first programming language" - far from it. It is put to some excellent uses and many companies do just that.

People can start using it to make small scripts with success, very very simple to get into.

The sticking point for quite a few teachers (and we see it in the forums) is when they try to do something more challenging but they have little or no prior experience, such as a GCSE Controlled Assessment task.

There are plenty of teachers that have been thrown into a role of teaching GCSE Computing for the first time with little training or time to prepare. That's when quite a few people start to struggle.

It's a learning curve that rises quite steeply for many, not a straight-line gentle progression.

When schools are successful with Python, the next trap is the GCSE theory paper that covers things that the practical programming has not given their students direct exposure to. That might be arrays (although many make the connection with Python lists), compiling, the relationship between high-level source-code, assembly and machine-code instructions, data-types, or the use of an IDE. These can all be addressed in the classroom if you have time as you go along or if you have enough time to teach them - and unfortunately plenty of people don't have that time.

For anyone that doesn't enjoy IDLE for their Python, try Geany (Windows and Linux) or Notepad++. Don't ditch the language for want of a different way of writing code.

1 person likes this. Like Not helpful

Post by Simon Johnson marked as not helpful (expand)

Simon Johnson
Master Teacher
Teacher (11-18)
2 more

May 04 2016, 10:59

With regards to 'Why use Python?' From my experience, it is my opinion that, due to Python's extensive libraries, the majority of programming controlled assessment are written (although not stipulated) for Python!

The dilemma I find this causes is that by using Python the students get higher coursework marks as the built-in libraries make it easier to reach the higher mark bands however, a down-side to this is that the students aren't necessarily better programmers!

Like Not helpful

Adam Sullivan
Teacher (11-16)

May 04 2016, 11:00

What an awesome resource - Thanks Simon - Just shared with my Year 10 class as we are revising for an exam on Programming :)

1 person likes this. Like Not helpful

Paul Powell
Teacher (11-16)
IT Professional
Curriculum Lead

May 04 2016, 12:58

@Moya - teacher fluency is really important. This is the same when people decide to create programs - they use a language that is familiar to them.

There are some specific Python features which, whilst not shortcomings of the language, make it tricky to teach some common language features - the lack of a post-tested loop for example, or the lack of constants.

These things can be got round in Python, and in the short term that is what anyone who is struggling with their first language should do. On the other hand, I don't think learning a second language is anything like as tricky as people often imagine.

Like Not helpful

Simon Johnson
Master Teacher
Teacher (11-18)
2 more

May 04 2016, 14:02

Love the power of Social media Adam :) You may also like this: Highest paying programming languages (Source: http://blogs.perceptionsystem.com/infographic/highest-paying-programming-languages/)

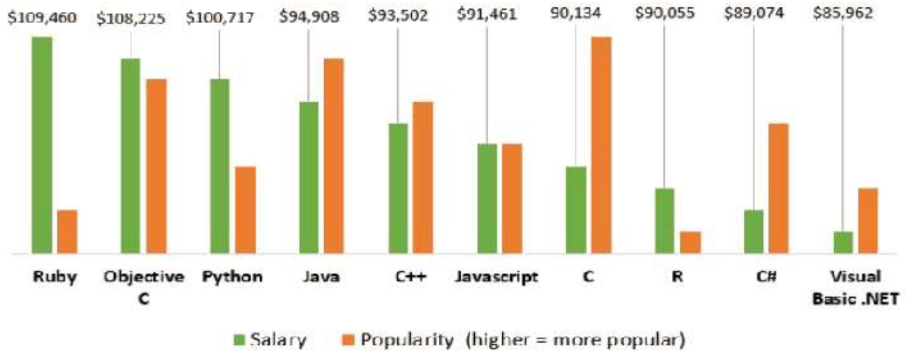
PERCEPTION SYSTEM
WE PERCEPT VISION

HIGHEST PAYING

PROGRAMMING LANGUAGES

Ever wonder which computer programming languages pay the most? According to Business Insider, the average salary for computer programmers just hit an all-time high as it approached $100,000.

Programming Langauges 10 Highest Salaryes and Popularity



JAVA

One of the biggest reasons Java is so popular today is because it's a favorable option for client-server web applications. There are approximately 9 million Java developers - many of them can expect to earn nearly $100,000.

Python

This widely used high-level programming language is perhaps as popular as it is because it allows programmers to express concepts in fewer lines of code.

Ruby

Although it's not quite as popular as your Java & C languages of the world, Ruby has been emerging lately as one of the top choices for tech startups in the United States. Since it's fully-open sourced, & has a large community culture, Ruby is likely to rise in popularity soon.

Objective C

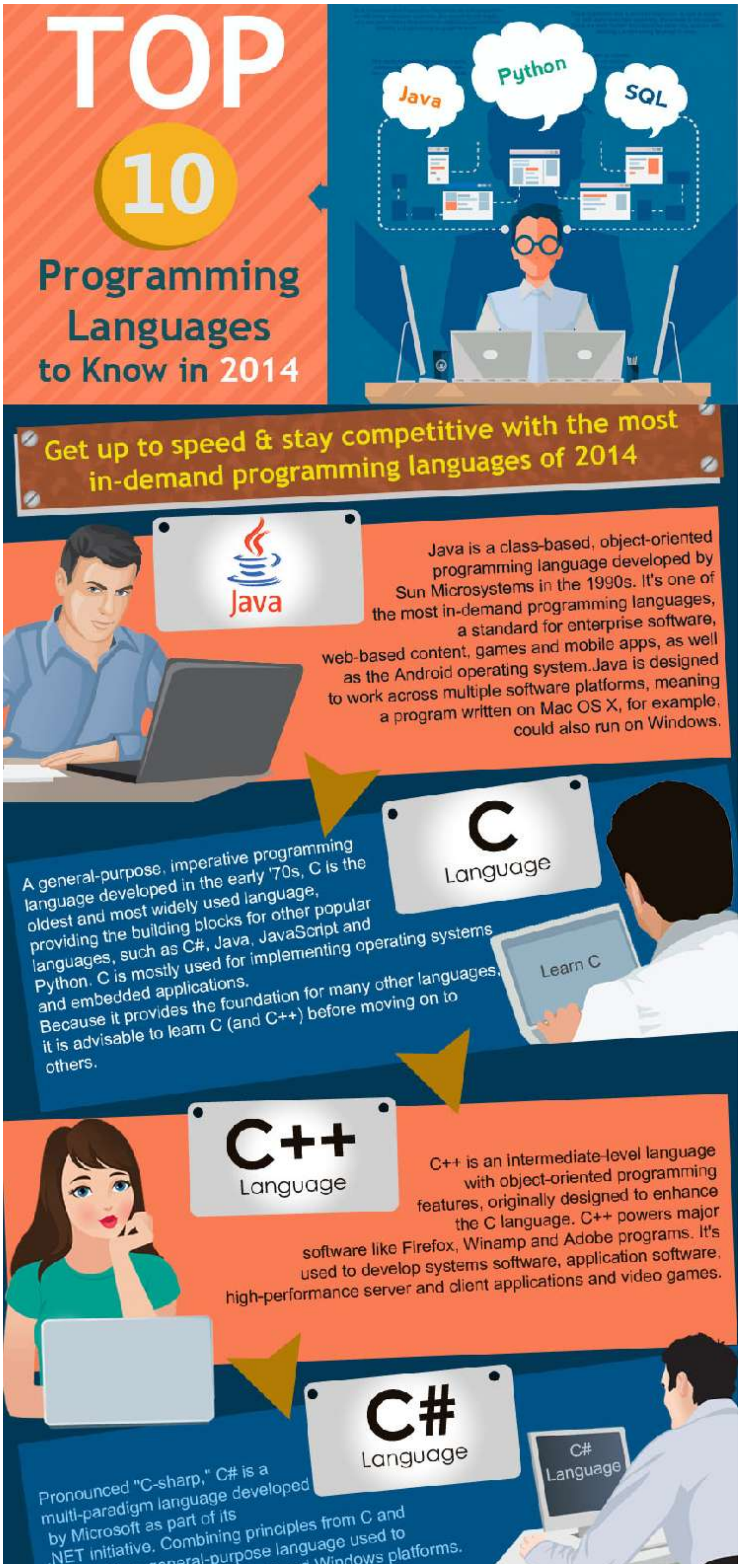
Used by some of the first iOS developers, it comes to no surprise that Objective C is most

Like Not helpful

Simon Johnson
Master Teacher
Teacher (11-18)
2 more

May 04 2016, 14:09

And this: Top 10 programming languages (Source: http://www.visualistan.com/2014/11/top-10-programming-languages-to-know-in-2014.html)


TOP 10 Programming Languages to Know in 2014
Java
Python
SQL
Get up to speed & stay competitive with the most in-demand programming languages of 2014
Java
C Language
Learn C
C++ Language
C# Language
C# Language

Java is a class-based, object-oriented programming language developed by Sun Microsystems in the 1990s. It's one of the most in-demand programming languages, a standard for enterprise software, web-based content, games and mobile apps, as well as the Android operating system.Java is designed to work across multiple software platforms, meaning a program written on Mac OS X, for example, could also run on Windows.

A general-purpose, imperative programming language developed in the early '70s, C is the oldest and most widely used language, providing the building blocks for other popular languages, such as C#, Java, JavaScript and Python. C is mostly used for implementing operating systems and embedded applications. Because it provides the foundation for many other languages, it is advisable to learn C (and C++) before moving on to others.

C++ is an intermediate-level language with object-oriented programming features, originally designed to enhance the C language. C++ powers major software like Firefox, Winamp and Adobe programs. It's used to develop systems software, application software, high-performance server and client applications and video games.

Pronounced "C-sharp," C# is a multi-paradigm language developed by Microsoft as part of its .NET initiative. Combining principles from C and [illegible] general-purpose language used to [illegible] Windows platforms.

2 people found this unhelpful. Like Not helpful

Peter Dickman
IT Professional
ex-Academic

May 04 2016, 15:15

Great posters/infographics. But to avoid any confusion (especially given the title of he thread)….

Please please please don't confuse:

(a) information about the current preferred skill set sought by employers of **professional** software engineers (most of whom are graduate or post-graduate level educated with work experience)

with

(b) what's the right thing to do in your classroom for 5-16 year olds.

And I recommend being extraordinarily careful about allowing this to influence your decisions w.r.t 16-18 year olds, whether or not they are going on to University.

Pedagogy first please, influenced somewhat by the next stage (or two) of their educational progression. Don't try to second-guess industry trends w.r.t something that's 5-10+ years away for your class.

Declaration of Interest: I work for Google

14 people like this. Like Not helpful

Adam Martin
Teacher (11-18)
IT Professional
CS Grad

May 04 2016, 15:24

@Simon - I detest some of those infographics, especially the first one. It is fantastically well laid-out and beautifully designed - and yet filled with grossly incorrect statements. It's a gilded lie :(.

There's almost more falsehoods on there than truths. When I first saw it, I wondered if it was meant to be a very clever parody, that everything was intended to be wrong, and I was merely missing some of the inaccuracies.

I wish this graphical talent were deployed on technically correct information :(.

Equally, the salaries sheet is simply false across the board. Useful perhaps for a lesson on why/how you should assume anything on the web is false until you have substantial evidence otherwise? :)

5 people like this. Like Not helpful

Stephen Richards
Master Teacher
Teacher (11-16)

May 04 2016, 15:52

I have heard the only bad first language was the BASIC of old.

Not visual basic.NET but the BASIC that went...

```
10 PRINT "HELLO. WHAT IS YOUR NAME?"
20 INPUT X
30 IF X <> "BOB" THEN PRINT "GO AWAY I DO NOT LIKE YOU" ELSE GOTO 50
40 GOTO 10
50 PRINT "I LIKE BOB"
```

So as long as it doesn't have a GOTO then it's fine (whatever it is) :)

Like Not helpful

Peter Dickman
IT Professional
ex-Academic

May 04 2016, 16:17

@Adam's right that the posters are full of lies; but useful one perhaps? I especially enjoyed the incorrect, weirdly over-precise and fake numbers for salaries and vacancies. And some of the phrasings are odd (C is the oldest of the languages considered perhaps, but is far from the oldest language). But I still think they have a place, to get people talking while highlighting that there *are* different languages to learn.

As an alternative/complement, how about using the old O'Reilly "History of Programming Languages" poster, since it shows change and development?

Declaration of Interest: I work for Google

1 person likes this. Like Not helpful

Miles Berry
HE Academic
Teacher Trainer

May 04 2016, 16:34

This is fun, particularly if you haven't seen it already: A Brief, Incomplete, and Mostly Wrong History of Programming Languages.

5 people like this. Like Not helpful

Simon Humphreys
Teacher (11-18)
National Coordinator Computing At School

May 04 2016, 17:20

> And I recommend being extraordinarily careful about allowing this to influence your decisions w.r.t 16-18 year olds, whether or not they are going on to University.

This is SO important though often a challenge with parents as Moya attests. We've all been there :(

Like Not helpful

Paul Powell
Teacher (11-16)

May 04 2016, 19:21

Nothing intrinsically wrong with using GOTO* - I think it is a much more intuitive way to learn to code for younger children. Maps well to flow charts too.

Loops and structured if ..then.. elif..else blocks are only really syntax that get turned into branch/jump instructions in the processor - i.e. a

IT Professional
Curriculum Lead

GOTO.

*Something of a parallel here with “guns don’t kill people…”

2 people like this. Like Not helpful

Jim Darby
IT Professional
STEMNet Ambassador

May 04 2016, 19:33

There is no goto in Java or Python. For a reason. Let’s take a trip back to 1968 and read the definitive article (not a grammar term).

“For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of go to statements in the programs they produce.” (opening sentence)

It hasn’t aged a day!

Like Not helpful

Adam Martin
Teacher (11-18)
IT Professional
CS Grad

May 04 2016, 19:37

I like your point about flowcharts. Although if I had to choose between making it easy to “go from flowcharts to code”, or to “write good code (see below)”, I’d probably lean towards the latter.

For what little it’s worth, I’ve heard two particularly convincing explanations for avoiding Goto:

1. Maths-heavy pure Computer Science: it’s provably bad, leads to inevitable drops in performance, and difficulty proving that programs are correct/incorrect. Great at degree level, overkill for everyone else
2. If you ban goto, new programmers are noticeably less inclined to write monolothic programs where everything is in a single function, thousands of lines long. Forcing them to use functions, and to plan their code better, is good for all.

Like Not helpful

Bruce Nightingale
Teacher (11-18)
bpnightingale@gmail.com

May 04 2016, 20:16

w.r.t. the original post, Mark Guzdial wrote the following several years ago ‘Predictions on future CS introductory languages’.

Your neither right nor wrong in your expressed view!

Like Not helpful

Paul Powell
Teacher (11-16)
IT Professional
Curriculum Lead

May 04 2016, 20:33

Dijkstra makes good points, but was disagreed with by Knuth, Kernighan, Ritchie and Torvalds. I don’t think it is an open and shut case and there are entirely legitimate uses in professional programming (depending on what you are coding.)

I don’t believe that GOTO inevitably leads to drops in performance. It does make things more difficult to prove correct - as do break, continue and return.

I want to play with it with some students and find out if it helps with them being able to program independently. If it does then it will be much easier to teach them to loop later on - and I suspect they’ll understand the loops a lot better (plus they’ll be grateful for them!)

1 person likes this. Like Not helpful

Simon Johnson
Master Teacher
Teacher (11-18)
2 more

May 04 2016, 21:41

I’m going to be even more controversial now ;D Based on my experience of teaching Computing at GCSE:

If you want better grades, teach your students Python - The reason for this is that all the exam boards accept built-in libraries for solutions and do not penalise for using them!

If you want better programmers, teach them another language other than Python!

Not wishing to start a flame war, but it will probably happen anyway, it doesn’t really matter what language we teach our students - no one knows what language will be the preferred language when our current year 10s finally get a job! What matters is we give them a head start, something I didn’t get when I was at school! I didn’t learn programming until college and that was COBOL - and look where that got me :D. What we must do is teach the fundamentals and teach them well! To be honest, the language should be determined by the teacher (whichever they are most confident in teaching) and if that is Python, then so be it!

In answer to the first question, the main benefit of Python is the wealth of resources on CAS and elsewhere to help those teachers with no prior programming experience - plus with extensive support for Raspberry Pi (epsecially Minecraft Pi) and micro Python on the BBC micro:bit, there are even more excuses to use Python.

Runs away now :D

7 people like this. Like Not helpful

Jim Darby
IT Professional
STEMNet Ambassador

May 04 2016, 21:53

goto can confuse optimisers but more importantly it can destroy meaningful structure. These two are *not* unrelated (guess what my degree project involved).

I use break, continue and return a lot; but they have far more restricted and clear actions. I should add catch, throw and finally into that as well.

If I’m reviewing code for security issues then as soon as I see a goto I know that someone is going to have to spend a lot of time fixing their mistakes. This is because there is a *far* higher chance of error and it’ll be hideous to fix it. They don’t tend to make the same mistake twice.

Which leads me on to the big cost in software being *maintenance*. If you write some vile spaghetti abomination you *will* be caught out eventually and it’s not going to end well. This seems to be never mentioned at GCSE/A-level yet it’s critical.

All of which is rather off-topic. To answer the original question we *do* use Python (most people do) but as a scripting language, often for running tests, setting things up and general supporting features. It’s not feature rich, powerful, fast or good at writing large software in for writing large industry-grade projects in (your mileage may vary). What it is *excellent* for is writing quick test programs in or test concepts. For that I believe it's great for teaching, others seem to as well.

I also agree with Simon's post. With one *huge* caveat: if you want to learn to program you're going to need to get a degree and learn an industrial grade language such as Java or C++. For GCSE or A-level Python gives you enough of a taster without all the stuff you need for million line, hundred programmer projects but with all the support you need.

4 people like this. Like Not helpful

Post by Howard Bennett marked as not helpful (expand)

Paul Powell
Teacher (11-16)
IT Professional
Curriculum Lead

May 05 2016, 01:10

@Jim, I agree for the most part. All constructs need to be well used.

I would argue that a loop that is several pages long with six different breaks, a couple of continues, plus a condition at the loop top is going to cause as many issues as a few GOTOs. In most cases I would be shocked to see a goto in serious code - especially without some pretty heavy commenting. On the other hand I understand that the Linux kernel contains about 100,000 of the things.

That's an argument about production code, but this is about teaching and understanding how programming languages work. The chief criticism I would lay at Python's door for teaching at KS3 and GCSE is that the knowledge needed to understand how Python works is way above that needed for GCSE and doesn't match the spec. Even 'simple' things such as for loops and assignments have quite convoluted explanations.

In science we teach atoms, followed by electrons, neutrons and protons for KS3 and GCSE. Simpler ideas for sure than the underlying reality. Is there any merit in starting with a simpler model of programming? I just don't know. I'll post when I find out!

1 person likes this. Like Not helpful

Liam Edgeley
Teacher (11-18)

May 05 2016, 06:33

Python isn't my first programming language*, but it is the one we chose to teach when we started delivering GCSE Computing back in 2012. It wasn't a bandwagon thing - for me, Python is this generation's Pascal (i.e. an excellent teaching language), but is also incredibly useful for solving real-world problems (I wrote a script a few months ago that uses an online API to retag my MP3 library. A few hours work in Python thanks to the huge range of libraries available - $DEITY knows how long it would have taken me in one of my other languages).

Python's syntax is overwhelmingly a clean one - much fewer "funny symbols" than the C family. This means that my pupils can focus on learning how to solve problems when coding without an awkward an unforgiving syntax getting in the way.

Strong/Dynamic typing is a bit of a double-edged sword, but for programmers just starting out means there is one less thing for them to think about - they can focus on problem solving. I encourage my pupils to put together a variable glossary for their solutions to CA tasks to show that they have thought about typing.

Semantic whitespace certainly takes some getting used to, but forces pupils to indent their code correctly. Have you ever tried debugging code written in one of the C families that has inconsistent indentation?

Similarly, Python's lists are so powerful that they let you concentrate on the algorithms rather than the implementation when teaching searching and sorting. They make for some beautiful and elegant code too.

There are features that I sometimes wish Python supported if only to make teaching some concepts easier: DO…WHILE loops, proper arrays (although numpy is a thing), constants, passing-by-reference and passing-by-value etc. However, when I need to teach these, I find its helpful to dip into another language (C is typical for me) to explain the concepts, and then have a discussion about **why** Python doesn't have that feature. In a lot of cases it's because its omission makes code cleaner and easier to understand.

*Commodore 16 BASIC! Followed by VB, BBC Basic V (for my own A Level back in the Acorn days), ARM Assembly (ditto), C/C++ (first couple of years at uni), PHP (for a Databases course), Java (decided it would be fun to learn it for my third-year project), C# (for giggles), JavaScript (which I loathe) and most-recently, Python. All that said, my own internal pseudocode is pretty close to C and I've been told I "write python like a C programmer" - not a compliment, I suspect!

4 people like this. Like Not helpful

Stephen Richards
Master Teacher
Teacher (11-16)

May 05 2016, 07:54

@Liam, I love the concept of students making variable glossaries for CA. It's an excellent idea for a simple step to take to help them write well documented code.

Like Not helpful

Simon Johnson
Master Teacher
Teacher (11-18)
2 more

May 05 2016, 08:10

I think we're starting to stray from the original question again here!

The question, and please tell me if I'm wrong, is "Why is Python the preferred language for teaching at GCSE".

The answer to this is simple.

1. GCSE Results - As teachers, we always want to do the best for our students and give them the best start in life (which is why many teachers, delivering Computing for the first time, worry about if they're teaching the right language) However, whether we like it or not, we all know that the priorities from SLT/Government is different - to them it's all about the grades and that's, at the end of the day, our performance is measured on. Python, at least in the current spec, gives you better grades - period!! The reason for this is because, as stated in the original question, Python cuts corners. This, along with the number of libraries makes it easy to complete more difficult tasks. For example, take the speed task from the OCR controlled assessment a couple of years back, many found this difficult but in Python all you needed to use was the Regular Expression library to compare licence plates and the csv library made it easy to handle csv files - in fact you could do task 3 (the hardest task) on about 10 lines of code. While this doesn't get you better programmers, it does get you better grades and, like all of the teachers here, I believe Computing IS for everyone and there's nothing wrong with a D grade, especially if that student has worked hard at it - just try having that conversation with you line manager/slt. I suppose it also depends on your school's ethos.
2. A lot of teachers are teaching Computing with no Computing experience (Some are former D&T teachers, Science teachers and even Music teachers) With no prior programming experience, Python is desirable for two things. 1. It's natural syntax makes it looks less intimidating than other languages & 2. The plethora of Python resources available online written specifically for KS3 & GCSE meaning that, not that I condone this, a teacher can simply give their students an Introduction to Python booklet and let them get on with it!

So there you are, in answer to the original question, this is why Python is the preferred language for teaching at GCSE.

3 people like this. Like Not helpful

Karl Davis
Teacher (11-18)

May 05 2016, 09:16

Spot on Simon. Python is fantastic in showing them the constructs. Once you get to A-level, then you can start molding them into better programmers. If they really are keen programmers and want to take it on further, you can always give them the resources to go away and fiddle with other languages (if they haven't done so already).

Moya has it nailed. It is so accessible to teachers and kids alike that I think its a question of why WOULDN'T you take it up at GCSE? Too much focus on getting them to be professional programmers rather than getting them a GCSE will give you headaches.

5 people like this. Like Not helpful

Simon Johnson
Master Teacher
Teacher (11-18)
2 more

May 05 2016, 09:47

Well said Karl, totally agree!

Like Not helpful

Bryan Owen
Teacher (11-18)

May 05 2016, 10:10

Look, I didn't graduate as a Computer Scientist. I did some VERY simple BASIC programming in the 80's but didn't look at it again until OCR came up with GCSE Computing and I could see the writing on the wall for ICT.

I was on the one of the first CAS courses for non-specialists, taught through Python. I like using Python, it is easy to understand and get a grasp on. If it was easy for me, it would be easy for my GCSE kids - ergo I stayed with it.

You will find that most teachers now teaching CS will have followed a similar path, and no longer have the time (in between marking, paperwork, planning, teaching, and having a bit of a life) to devote 3 years to a CS degree to be able to offer Java / any of the C languages / etc.

Most of us are now thinking about an A Level offering, and whilst we respect the fact most of you here have done CS degrees, there seems to be a hint of snobbery going on here on those without; an element of being talked down to.

I like Python. I get my head around it. It has lead me to learn JavaScript, which seems to following the same style and structure, and was not too bad to learn. Not sure what I will do for A Level (still waiting for more info on what is a suitable project), but can we stop the intellectual snobbery bit please?

9 people like this. 4 people found this unhelpful. Like Not helpful

Chris Charles
Master Teacher
Teacher (11-18)

May 05 2016, 10:41

I don't like python as a beginners language because of the lack of strong typing - the way I teach I want my students to have to think about what variables they are going to use and what type of data is going to be stored.

Also the whole white-space having meaning is an awful idea in my opinion. The change of syntax (although small) between Python 2 and 3 is again awful when students are looking for example code. The lack of standard arrays doesn't fit well with the requirements of the spec I deliver.

However, python is easy to read, there are tons of useful libraries, it's free and available on many Operating Systems. The number of resources, blog posts etc is again very useful, especially for people who don't have a programming background.

I don't think that there is a perfect language. Maybe a fork of python with strict-typing, no use of white-space for meaning (I like END IF etc!) and a way of creating GUIs as easily as Visual Studio (C/VB) would make me happy!

Overall though I don't really think it matters what language you offer, we aren't (shouldn't be?) trying to enable students to leave after GCSE and become programmers but make them able to understand the concepts and problem solve.

(As an aside I started programming in ZX Spectrum BASIC and still managed to become a salaried programmer before becoming a teacher.)

2 people like this. Like Not helpful

Adam Martin
Teacher (11-18)
IT Professional
CS Grad

May 05 2016, 11:30

> "Have you ever tried debugging code written in one of the C families that has inconsistent indentation?"

Yep :) - it's almost trivial! Entirely seriously:

Click the Source menu, click "reformat entire project to be correct" (or whatever your IDE calls it), hit Save-all. That's assuming the IDE didn't autocorrect it at the moment it opened the files in the first place…

…I'm commenting to make the point: whatever languages we use, we should be using decent IDEs. Features like "correct indentation" (which are trivial for a computer to do automatically) have been standard in the free IDEs for well over a decade.

Trying to work with wierdly-indented code is like choosing to use a 12" CRT monitor that only has 2 colours, and no mouse, when you have a 17" LCD and Windows sitting next to you.

4 people like this. Like Not helpful

Phil Gardner
Master Teacher
Hub Leader

May 05 2016, 12:17

Indentation? Well whatever language you choose, as a teacher you can emphasise to students the importance of setting it out neatly/clearly to make it easier to understand. Text editors can also tidy tabs/spaces for you, although it's presumably better if you can remember to lay things out neatly rather than relying on the computer to do it for you.

Much gets written about C-like languages (on CAS, infographics or elsewhere) that is not strictly true and certainly not the whole picture - symbols, difficulty when sorting. They are not actually a problem in a simple classroom setting. All languages have some sort of symbols that you need to get used to.

Nearly all languages have mega-massive books with 1000+ pages in them that will teach your to program, possibly in 24 hours(!) but how much of that is actually needed for GCSE? A small part (thankfully). Unfortunately there aren't that many books specifically aimed at teaching or classroom learning at KS3/KS4 - big programming books are often aimed at undergraduate level or professionals.

At school level, rather then undergraduate level the simple features that are laid out in the GCSE Specifications don't ask for object-polymorphism etc. They want constants, variables, well-known simple data-types, simple arrays etc. These are so similar in many languages.

If you can write some JavaScript, you can write C++. At school-level, so many of the languages are just covering the same ground. C at undergraduate level (the source of quite a few anecdotes on CAS) is not the same as C++ in the classroom. Many people wouldn't find it very easy to work with "null-terminated arrays of characters without error detection" (old-style C strings) but they can easily create a "string" variable (string is a data-type that you can use, as is bool, char, int, float).

Whatever language you choose, the trick is keeping things simple and how you put the bits together to solve problems. The latter seems to be what teachers and students have difficulty with - not the language, the application to a particular problem. Students can write good or bad code in any language.

Many similarities between languages and experience with one can usually be transferred to another.

4 people like this. Like Not helpful

Dr Dave Wild
Teacher (16-18)
HE Academic
2 more

May 05 2016, 13:31

I am Head of Computer Science in waiting and I am certainly thinking of changing the existing language of choice in the Departmnet to Python (from Java)….. here're my thoughts:

1) Python I think is easier to follow for most students. Even a hello world in Java can baffle some people - exposing them to **args, etc.

2) I would prefer Pascal because it is type-safe. I know students will use myVariable = 12/43 and wonder why it yields a result of 0. I think typesafety is something students can learn from and become better programmers because of it.

3) Python doesn't have so many options for native GUIs… I know there's QT and other things about but they are disjoint, unlike some of the Java GUI developments. I don't want to have students thinking it is just a console programming language.

4) We have a lot of existing resources to help prepare lessons in Java, including answers to prior examination papers in that language. I would need to convert all of our resources into Python. Some of these resources have nice GUI elements with the existing HoD has created. I'd have tore-create those in QT. Not sure my wife to be would like me during the summer holidays!

5) I still… show more

Like Not helpful

Peter Kemp
HE Academic
Teacher Trainer

May 05 2016, 16:33

Dear All,

It's been a few years since this was last updated, but for arguments on the benefits and drawbacks of each language, please see here. And it's a wiki, so please jump in and edit!

3dami, wikibooks and kemputing

Like Not helpful

Keith Wyles
Teacher (11-18)
Retired
wylesk@shtc.org.uk

May 05 2016, 18:32

Is arguing about which language to choose that sensible? never results in agreement.

4 people like this. Like Not helpful

Michael Sparks
IT Professional

May 06 2016, 01:42

"Is arguing about which language to choose that sensible? never results in agreement."

It depends on whether you're interested in consensus, or interested in the arguments for/against different languages. My impression from this thread (and other threads where this comes up) is that there's less emphasis on "winning the argument" and greater emphasis on discussing benefits and merits.

As someone who uses one of the languages in preference to others on a regular basis, but some of the other languages on an as needed basis, I think this is an interesting discussion. (Especially the parts where I see clear misunderstandings of various languages under discussion - not because that's good, but because it's useful to see the effect on teaching)

As someone who has an interest in implementing programming languages as well, seeing these discussions can also be particularly helpful.

2 people like this. Like Not helpful

Karl Davis
Teacher (11-18)

May 06 2016, 07:51

> I'm commenting to make the point: whatever languages we use, we should be using decent IDEs. Features like "correct indentation" (which are trivial for a computer to do automatically) have been standard in the free IDEs for well over a decade.

I agree however if the kids just use the tools of the IDE then (and this isn't against the IDE, this is more exam board mark schemes) how are they going to show development and fixing errors they have created?

I have PyCharm installed on our computers however GCSE use IDLE because of the LACK of tools. Fixing mistakes is how kids learn, no matter how many times they bang their head against the proverbial brick wall and how frustrated they get.

We switch to C# at A-Level in order to give the students a much more grounded experience. I have had no trouble so far in getting them on board with the different syntax and structures.

1 person likes this. Like Not helpful

Mike Davis

May 06 2016, 08:28

IDEs sort out syntax, but not logic. Which is IMHO the more important thing for them to think about.

1 person likes this. Like Not helpful

Teacher (11-18)
mike@mike-davis.net

Andrew Morbey
Teacher (11-18)

May 06 2016, 09:08

I'll chuck my 2c in.

- Python is strongly typed. I guess what people mean is it is dynamically typed but these are different things.
- Discussing/Arguing about what language to teach is missing the point. A pupil can get their head round any syntax with a good teacher. The language is irrelevant to solving problems. How many of you have had a pupil say I could do problem x so much easier in language y and then still fail to solve it.

I see what I am aiming to do as teaching how to use a programming language to solve problems and how to understand how different constructs, statements, data structures, data type can be used to achieve a goal. The language is irrelevant really so:

- Pick a language you are comfortable with, meets the specification requirements that you are teaching, suits your pupils well, works on your school's systems/network, ticks accessibility needs (free? cross platform?) and is well documented for pupils to get help easily.

Out of interest we use:

Y7 scratch/byob Y8 small basic y9 processing/sonic pi(ruby) Y10/11 Visual basic but may be switching to python in line with the new spec Y12 c# Y13 whatever they please but dabble in c++ to see pointers and chuck in Haskell for next year.

These suit our needs, kids and exam requirements very nicely.

5 people like this. Like Not helpful

Chris Charles
Master Teacher
Teacher (11-18)

May 06 2016, 12:04

Regarding static/dynamic/weak/strong typing definitions - People do use the terms differently, as is often the case with Computer Science there is no one true definition.

For example some of the Cornell Computer Science materials describe strong typing as "the type of every variable and every expression is a syntactic property —known by looking at the program, without resorting to executing it" - I'd argue this isn't true of Python.

Like Not helpful

Paul Powell
Teacher (11-16)
IT Professional
Curriculum Lead

May 06 2016, 14:46

Python is strongly typed - but it is the values that are typed rather than the variable (or name) that points to it.

Hence:

```
a=5
a="abc"
```

is fine - the type of the value never changes

```
a=5 + "abc"
```

fails because there is no form of the + operator that takes an integer and a string.

```
s=str(5) + "abc"
```

is fine because the string constructor is called that takes an integer and returns a string object which can then be used with the str + str form of the + operator.

My preference is for strong static types. This means that certain logic errors turn into syntax errors (well, compile time…)

I saw this one today:

```
choice=input("Enter your menu choice")
if choice==1:
    print("this is menu option 1")
```

The logic error in the above program would be picked up by many languages, but not so in Python.

1 person likes this. Like Not helpful

Andrew Brixey
Teacher (4-11)
Teacher Trainer
IT Professional

May 07 2016, 07:44

I'm a primary school teacher so my perspective is a little different.

Children are starting to use Scratch at a much younger age now, many schools introduce it in year 3 and though my experience with code clubs have found many children starting to use it much earlier at home, particularly if they have older siblings.

Code Club (which is growing at an amazing rate) provides excellent resources for introducing students to Scratch, HTML / CSS and Python. It's target audience is Primary Schools.

So increasingly students should be starting secondary school with some experience of Python already under their belts. I am sure that more non-specialist primary school teachers will dabble with python using trinket.io after experiencing it through code club.

Personally I like Sniff as a follow on language from Scratch. Not as a first programming language, but as a transition between Scratch and Python.

http://www.sniff.org.uk/p/getting-started-with-sniff.html

Although I personally dislike Python, the children I have taught have found it very accessible and thanks for GPIO on the Raspberry Pi, etc, a great deal of fun.

Most importantly it is supported by a wealth of teaching materials and generally speaking there are also plenty of experts out around who are willing to help if I (as a non specialist teacher) get stuck.

Looking at the GCSE and onto the Alevel I wonder if C should have a more prominent role to play … I like to think of it as a coding equivalent to learning Latin and definitely benefited from learning it before moving onto OOP.

1 person likes this. Like Not helpful

Adam Martin
Teacher (11-18)
IT Professional
CS Grad

May 11 2016, 14:00

> if the kids just use the tools of the IDE then (and this isn't against the IDE, this is more exam board mark schemes) how are they going to show development and fixing errors they have created?

IDE's (apart from TouchDevelop, which is why I hate it) don't write code for you. They don't create algorithms, they don't implement them.

The good ones *do* spot common typos in your code and suggest corrections. Unlike Microsoft Word, and Apple's iPhone, they are extremely cautious: the cost of wrongly changing some source code is many times greater than the cost of wrongly changing a word from British to American spelling (in most cases).

But this does not materially reduce the amount of comprehension and debugging students do in the classroom.

If anything, it may increase it. Great IDE's prompt you with a question: did you mean to do X instead, or Y?

…leaving the reader forced to stop and think "did I?" … or even "no, neither, leave my code as-is, thank you".

In my experience, this helps develop an additional important skill: defending your reasoning for the code you've written. Not defending the code - you're open to being shown a better way - but being able to explain WHAT you were attempting to achieve.

Like Not helpful

Reply

- First Level Heading
- Second Level Heading
- Heading 3
- ---------------
- Bold
- Italic
- ---------------
- Bulleted List
- Numeric List
- ---------------
- Picture
- Link
- ---------------
- Quotes
- Code Block / Code

You can use Markdown and/or HTML (<a href="...">, <b>, etc)
Please read the posting guidelines and terms of use.
Post reply

« back to Secondary Education