Suggestions on starting a child programming [closed]

Asked 16 years, 4 months ago Modified 14 years ago Viewed 9k times

45

votes

As it currently stands, this question is not a good fit for our Q&A format. We expect answers to be supported by facts, references, or expertise, but this question will likely solicit debate, arguments, polling, or extended discussion. If you feel that this question can be improved and possibly reopened, visit the help center for guidance.

Closed 13 years ago.

Locked. This question and its answers are locked because the question is off-topic but has historical significance. It is not currently accepting new answers or interactions.

What languages and tools do you consider a youngster starting out in programming should use in the modern era?

Lots of us started with proprietary Basics and they didn't do all of us long term harm :) but given the experiences you have had since then and your knowledge of the domain now are there better options?

There are related queries to this one such as "Best ways to teach a beginner to program?" and "One piece of advice" about starting adults programming both of which I submitted answers to but children might require a different tool.

Disclosure: it's bloody hard choosing a 'correct' answer to a question like this so who ever has the best score in a few days will get the 'best answer' mark from me based on the communities choice.

language-agnostic children

Share

edited May 23, 2017 at 12:13

community wiki
3 revs, 2 users 87%
sparkes

Comments disabled on deleted / locked posts / reviews

46 Answers

Sorted by: Highest score (default)

1 2 Next

43
votes

I would suggest LEGO Mindstorm, it provides an intuitive drag and drop interface for programming and because it comes with hardware it provides something tangible for a

child to grasp. Also, because it is "LEGO" they might think of it as more of a game then a programming exercise.

Share

answered Aug 21, 2008 at 14:43

community wiki
ejack

+1: This how I started when I was 10. – Callum Rogers Aug 15, 2009 at 16:24

Those things also get you started with hardware - you're not just printing something on the screen... you're moving things around in physical space, which is great! – Utkarsh Sinha Nov 17, 2010 at 14:02

24

votes

My day job is in a school, and over the past few years I've seen or taught (or attempted to teach) various children, in various numbers, programming lessons.

Children are all different - some are quick learners, some aren't. In particular, some have better literacy skills than others, and that definitely makes a difference to the speed at which they'll pick up programming. I bet that most of us here, as professional computer programmers and the kind of people who read and post to forums for fun, learnt to read at a pretty young age. For those kinds of children, and if it's your own child who you can teach one-on-one, you could do worse than JavaScript - it has the advantage that you can do real stuff with it right away, and the edit-test

cycle is simply hitting "refresh" in the browser. It gets confusing when you start to run in to how JavaScript does everything asynchronously, and is tricky to debug, but for a bright child under close tuition these problems can be overcome.

LEGO Mindstorms is definitely up there at the top of the list. Most schools now super-glue the bricks together to create pre-made models that can't have bits nicked off of them, but this shouldn't be a problem at home. Over on the Times Educational Supplement site (website forum for the UK's weekly teaching newspaper), the "what programming language is best for children?" topic comes up pretty regularly. Lots of recommendations over there for Scratch as an alternative to Mindstorms - bit more freedom than Mindstorms, again probably better for the brighter student who could also be given a soldering iron.

I've found that slower pupils can still have problems with Mindstorms, even though the programming environment is "graphical" - there's still a lot going on on screen, and there's a fair bit to remember (this was an older version, mind - haven't tried the snazzy new one yet). In my experience, the best all-round introduction to programming is probably still LOGO - actually a considerably more powerful language than most people give it credit for. The original Mindstorms book by Seymour Papert (nothing to do with LEGO - they nicked the title of the book for their product), one of the originators of LOGO, is the canonical reference for teaching programming to children as a

"thinking skill" and for the concept of Constructionism in learning.

We've had classes of 7 or 8 year-olds programming LOGO. Note that we aren't aiming to make them "software developers", that's a career path they can decide on at some point post-16. At a young age we're trying to get them to think of "computer programming" as just another tool - how to set out a problem to be solved by a computer, in the same way they might use a mind map to help them organise and remember stuff for an exam. No poor child should be sat down and drilled in the minutia and use of a particular language, they should be left to explore and figure stuff out as they like.

Share

edited Jan 17, 2009 at 13:40

community wiki
2 revs, 2 users 94%
David Hicks

13

votes

I'll second Geoff's suggestions of Phrogram (used to be KPL), and Alice.

My only other suggestion is Lego Mindstorms NXT. The NXT's programming language is drag-and-drop, is very easy to use, and can do some very complicated tasks once you learn it. Also young boys usually like seeing things move. :)

I've used Alice and NXTs with some young kids, and they've taken to it very well.

Share

answered Aug 21, 2008 at 14:33

community wiki
Eric Haskins

10

votes

Two possibilities are:

Scratch - developed at MIT - http://scratch.mit.edu/

and

EToys from the One Laptop per Child fame - http://wiki.laptop.org/go/Squeak

Share

answered Aug 21, 2008 at 15:19

community wiki
cpuguru

> Just to add that we now have two whole year groups using Scratch and it's working out really well - they're getting the hang of loops, subroutines, the full works. – David Hicks May 22, 2009 at 14:14

10

votes

Full disclosure: I'm one of the guys who invented Kid's Programming Language, which is now http://www.Phrogram.com, which others have recommended here. Let me add some programmer-oriented info about it.

It's a code IDE, rather than drag-and-drop, or designer-based. This was intentional on our part - we wanted to make it easy and fun to do real text-based programming, particularly programming games and graphics. This is a fundamental difference between us and Alice and Scratch. Which you pick is a matter of the kid, their age and aptitudes, your goals. Using them serially with the same beginner might be a great way to go - if you do that, I would recommend Scratch, Alice, Phrogram as the order. Phrogram has worked best for 12 years and up, but I know dads with 6 year olds who have taught their kids with it, and I know 10 year olds who have taught themselves with it.

The language is as much like English as we could make it, and is as minimal as we could make it. The secret sauce is in the class-based object heirarchy, which is again as simple, intuitive and English-like as we could make it. The object heirarchy is optimized for games and graphics. 3D models are available, and 2D sprites. Absolute movement using screen coordinates is supported, or relative movement ala LOGO turtles - Forward(x), TurnLeft(y).

The IDE comes with over 100 examples, some language examples (loops), some learning examples (arrays), some

fully-functional games and sims (Pong, Missile Command, Game of Life).

To give you a sense of how highly leveraged we made the language and the IDE: with 27 instructions you can fly a 3D spaceship model around a 3D skybox, using your keyboard. The same with a 2D sprite is 12 to 15 instructions.

We are working on a Blade-compatible release of Phrogram that will allow programs to run on the XBox 360. Yeah, the XBox, on your big TV. Nice motivator for getting a kid started? :)

Phrogram includes support for class-based programming, with methods and properties - but that's only encapsulation, not inheritance or polymorphism.

A tutorial and user guide is available,

My own ebook is available at Amazon and other places online, "Learn to Program with Phrogram!," and gets a beginner started by programming the classic Pong.

Phrogram Programming for the Absolute Beginner, by Jerry Lee Ford, Jr., is also available, as a paperback, at Amazon and elsewhere.

answered Sep 16, 2008 at 16:53

community wiki

TheSchwartz

8 votes

For a child, I would go with Alice. Any kid is going to like the drag-and-drop interaction that Alice uses better than trying to remember how to spell and punctuate any programming language. He/She will learn the basic programming structures (conditionals, loops, etc.) and will experience the fun of building an animated program they can show off to other family or friends.

A beginner CS class at the local community college actually uses Alice to teach programming in a language-independent way. It provides a good foundation for moving into programming in a particular language (or a few languages) down the road.

Share

answered Aug 21, 2008 at 14:28

community wiki
Justin Bennett

7 votes

I recently saw a presentation about GreenFoot (a java based learning environment for children). It looked awesome. If I would have kids, I would give it a try

Link to the presentation

It is a very playful environment, where you could start with very basic methods. The kids learn thinking in an object

oriented way (you cannot instantiate an animal, but you can instantiate a cat). And the better they get, the more of Java you can uncover for/with them.

Share

answered Aug 21, 2008 at 14:25

community wiki
Mo.

7

votes

I'd go with Scratch, some points regarding it.

- It's a graphical programming language. It isn't text based (this might be positive or negative). It does make it more intuitive and easy for kids (7 and up).
- It's actually highly object. The objects you write these graphical scripts have the code attached to them and can be reused and moved around.
- **Very Important**: quick and impressive results. Kids need to get going fast and get results in order to get hooked.

I'd like to note that although many of us started programing at a young age in basic or logo and because programmer later in life doesn't mean those are good languages to start with. I think that kids today have much better options, like scratch or Alice. Text based languages (python, ruby, basic, c# or even c) are dependent on external libraries and tools (editors, compilers) while something like Alice or scratch is all inclusive and will teach kids (not aimed at teens)

programming concepts. Later they can move on and expand their learning.

Share

edited Sep 1, 2008 at 11:35

community wiki
2 revs
Rotem

6

votes

Check out Phrogram (formerly KPL) and Alice

Share

answered Aug 21, 2008 at 14:27

community wiki
Geoff

6

votes

I'd say: give the kid a real C64, because that's how I got started. But, today... I'd say Ruby, but Ruby is a bit too chaotic. BASIC would be better in the long run. Processing is easy to learn, and it's basically Java.

The reason I recommend a C64 is because it's BASIC, but you still have to learn certain computer-related things, like the memory model, pixels, characters, character maps, newlines, etc. etc, if you want to do more advanced stuff. Also, if your kid finds it boring, you know his heart really isn't into coding.

Share

answered Aug 21, 2008 at 15:26

community wiki
GhassanPL

> There's at least one school in the UK that's done this - gone and got a bunch of C64s from a carboot sale and used them for programming classes. The trusty old BBC micros are still in use in several places, too. – David Hicks May 22, 2009 at 14:16

6 votes

I would pitch LOGO. It was something that was taught in my elementary school. It gives nearly immediate feedback, and will teach really basic programming concepts. Moving that little turtle around can be a lot of fun.

Share

answered Aug 21, 2008 at 17:22

community wiki
Nate Smith

5 votes

> For a child, I would go with Alice.

Here is another vote for Alice. My 4 kids have had a ton of fun working with it and learning the basic concepts of programming. Of course to them it's all about socializing

with fairies and ogres, but heck the darn legacy system I work on could use some faries and ogres too.

Share

answered Aug 21, 2008 at 14:32

community wiki
Dan Blair

---

5 votes

I'd recommend python, because it's so terse and expressive. Seems less likely to frustrate when getting started, but offers plenty of room to learn more advanced concepts as well.

Share

edited Jan 17, 2009 at 16:29

community wiki
2 revs, 2 users 67%
AlexCuse

---

4 votes

Game Maker might be another approach. You can start simple with easy drag and drop development, and then introduce more advanced programming as you go. The book The Game Maker's Apprentice: Game Development for Beginners has a number of sample games and takes you through the steps required to make them.

Share

answered Aug 21, 2008 at 15:56

community wiki
James Sutherland

---

3 votes

I think python is a good alternative; it is a very powerful language also you can easily do a lot of things (not boring at all).

Share

answered Aug 21, 2008 at 14:37

community wiki
jl23x

---

3 votes

Checkout Squeak developed by Alan Kay who think programming should be taught at early ages.

Share

answered Aug 21, 2008 at 14:42

community wiki
epatel

---

3 votes

How old? Lots of us stared with BASIC at some point, but before then, I learned the concepts of stringing commands together, variables, and looping with LOGO. Figuring out how to draw a circle with a triangle that can only go in a

straight line and turn was my very first programming accomplishment.

Edit: This question & its answers make me feel old.

Share

edited Aug 21, 2008 at 14:44

community wiki
2 revs
Greg Hurlman

2

votes

Though _why hasn't given it much love in the past year or so, for a while I was really excited about Hackety Hack. I think the key for most new programmers, especially children who are more than apt to losing interest in things, is instantaneous feedback. That was the really wonderful thing about Hackety Hack: a few lines of code, and suddenly you have something in front of you that does something. There are a few similar applications aimed at things like drawing graphics (one of which, I briefly assisted Nathan Weizenbaum on, Scribble!). Kids simply need positive feedback that they're doing something correct on a regular basis, else there's nothing to keep them interested in the task at hand. What I think the future is for teaching children to program is some sort of DSL built on top of a language with friendly syntax (these would include, arguably, Ruby, Python, and Scheme) whose purpose is to provide an intuitive environment for constructing simple games (say, Tic-Tac Toe, or Hangman).

Share

answered Aug 21, 2008 at 14:30

community wiki
wfarr

---

2 votes

I think you should start them off in C. The sooner they can get the hang of pointers the better.

See Understanding Pointers and Should I learn C.

Share

edited May 23, 2017 at 11:48

community wiki
2 revs
David

---

2 votes

I think the first question is: what sort of program would it be interesting to create? One of the things that got me started with programming as a kid (in BBC basic and then QBasic) was the ease of writing graphical programs. I could write a couple of lines of code and see my program draw a line on the screen straight away.

The closest I've seen to that sort of simplicity recently are the pygame library for python and Processing, a set of java libraries with an IDE.

I imagine that hacking on web pages would be another good way to get started: that would entail HTML, Javascript (using a library like jQuery), perhaps PHP or something along those lines.

Whatever tools you provide, the crucial thing is for it to be easy to get started straight away. If you have to write twenty lines of correct code and figure out how to invoke the compiler before you see any tangible results, progress is going to be slow.

Share

answered Aug 21, 2008 at 15:01

community wiki
Ned

2

votes

There are many good suggestions here already. I really agree with Kronikarz. Get a retro computer (or emulator) that you are interested in and teach with that. Why a retro computer? Basic is built in. Making sounds and primitive graphics is a trivial task. The real deal might be better than an emulator because it will be a bit more fascinating to a child who is used to seeing only modern devices.

Share

answered Aug 21, 2008 at 15:47

community wiki
Daniel Auger

2

votes

As I said here, I'd go for Squeakland and the famous Drive a Car example (powered by Squeak).

Smalltalk syntax is simple, which is great for children.

And later as the child evolves, he can learn more complex and even very advanced concepts that are also in Squeak (eg. programing statefull webapps with automated refactoring and automated unit tests!).

And like @cpuguru and @Rotem said, Scratch (also Squeak based) is great too.

Share

edited May 23, 2017 at 11:48

community wiki
2 revs
Sébastien RoccaSerra

1 vote

I think Java might be a good choice simply because you can make GUIs easily, and see "cool things" happening. For the same reason, maybe any of the .NET languages. I've also heard good things about scripting languages (Ruby and Python, especially) for getting kids to learn how to program.

Share

answered Aug 21, 2008 at 14:26

community wiki
Thomas Owens

1 vote

Well, if they're young and haven't learnt their ABC's you could try them on BF - non of those pesky letters and numbers to deal with.

I'll get me' coat.

Skizz

Share

answered Aug 21, 2008 at 14:42

community wiki
Skizz

care to explain what is the BF? – Axarydax Jan 14, 2010 at 12:50

It's a programming language that uses the eight symbols "[]+-> <,." - no letters or numbers. OK, it's name isn't particularly child friendly - keep it as BF. – Skizz Jan 18, 2010 at 10:27

1 vote

I would go with what I wish I had known first: a simple MS-DOS box and the integrated assembler (debug). It is great to really learn and understand the basics of talking to a computer.

If that does not scare away a child, then I would go the "next level up" and introduce C. This shouldn't be hard given that the basic concept of pointers, registers and instructions in general are well-understood by then.

However, I am not entirely sure, where to go next. Take the big jump to Lisp, Haskell or similarly abstracted languages or should there be some simple object oriented languages (maybe even C++) be thrown in or would that more hurt than help?

Share

answered Aug 21, 2008 at 14:58

community wiki
HS.

1

vote

Looking at Alice, I see it is "designed for high school and college students". There appears to be another language/version called Story Telling Alice that is "designed for middle-school students"

Alice Download Page

Share

answered Aug 21, 2008 at 17:40

community wiki
Keith Sirmons

1

vote

I think Context Free Art might be a good choice, with output of graphics, it makes it a lot of fun learning about context-free grammar.

Share

answered Sep 3, 2008 at 3:25

community wiki
Matt Nelson

1

vote

Try [Guido van Robot][1]. It's an excellent introduction to robotics, and it's a great way to introduce kids to the programming side of things (vs the "building the robots" side).

Share

answered Oct 23, 2008 at 18:01

community wiki
JPLemme

0

votes

Wasn't Smalltalk designed for such a purpose? I think Ruby would be a good choice, as a descendant of Smalltalk.

Share

answered Aug 21, 2008 at 15:34

community wiki
wvdschel

0

votes

I know in the first few years of high school we were 'taught' Logo, and strangely, HTML. After that, the progression went to macros in MS Office, followed by basic VBA, followed by Visual Basic.

Share

answered Sep 1, 2008 at 14:03

community wiki
James Inman
