Why Python should be your first high school programming language.
Is the language available for the platform(s) at my school?

You will probably not have the resources to change platforms just to use a particular programming language, but you might be able to dual-boot to accommodate a certain language. (C# on OS X or Objective C on windows)
Is the language available for the platforms that my students are likely to have at home?

This is more problematic as you don’t have any control over it.
How much does the language cost?

This could have bearings on both school and home use.
Is the language proprietary?

A language that is only available from one vendor might be less likely to be relevant over a long period of time.
Is the language used in industry?

How widely? Prominent companies? This might not be a big concern in some situations.
Is it used to teach intro courses in higher education?

If no one in higher ed. uses it, is there a reason? If some use it, are there resources that they make available that you can use as well? Online lectures? Homework ideas? Textbooks?
Number of top 39 U.S. computer science departments that use each language to teach introductory courses.

Is the language likely to be relevant in the future?

This is pretty subjective, but here are some websites: www.langpop.com
www.tiobe.com
## Very Long Term History

To see the bigger picture, please find below the positions of the top 10 programming languages of many years back. Please note that these are *average* positions for a period of 12 months.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C++</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Python</td>
<td>4</td>
<td>7</td>
<td>5</td>
<td>10</td>
<td>29</td>
<td>21</td>
<td>-</td>
</tr>
<tr>
<td>Visual Basic .NET</td>
<td>5</td>
<td>11</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C#</td>
<td>6</td>
<td>5</td>
<td>7</td>
<td>7</td>
<td>27</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>JavaScript</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>18</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PHP</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL</td>
<td>9</td>
<td>-</td>
<td>-</td>
<td>6</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Objective-C</td>
<td>10</td>
<td>3</td>
<td>36</td>
<td>45</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>COBOL</td>
<td>25</td>
<td>19</td>
<td>15</td>
<td>11</td>
<td>3</td>
<td>9</td>
<td>13</td>
</tr>
<tr>
<td>Lisp</td>
<td>27</td>
<td>13</td>
<td>18</td>
<td>14</td>
<td>13</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Pascal</td>
<td>204</td>
<td>14</td>
<td>14</td>
<td>96</td>
<td>6</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>
Very Long Term History
To see the bigger picture, please find below the positions of the top 10 programming languages of many years back. Please note that these are average positions for a period of 12 months.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C++</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Python</td>
<td>4</td>
<td>7</td>
<td>5</td>
<td>10</td>
<td>29</td>
<td>21</td>
<td>-</td>
</tr>
<tr>
<td>Visual Basic .NET</td>
<td>5</td>
<td>11</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C#</td>
<td>6</td>
<td>5</td>
<td>7</td>
<td>7</td>
<td>27</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>JavaScript</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>18</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PHP</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SQL</td>
<td>9</td>
<td>-</td>
<td>-</td>
<td>6</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Objective-C</td>
<td>10</td>
<td>3</td>
<td>36</td>
<td>45</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>COBOL</td>
<td>25</td>
<td>19</td>
<td>15</td>
<td>11</td>
<td>3</td>
<td>9</td>
<td>13</td>
</tr>
<tr>
<td>Lisp</td>
<td>27</td>
<td>13</td>
<td>18</td>
<td>14</td>
<td>13</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Pascal</td>
<td>204</td>
<td>14</td>
<td>14</td>
<td>96</td>
<td>6</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>
Is the user community helpful?

This could also include customer support for proprietary languages. Are they a source for answers for you? For your students?
Are community code examples likely to teach good coding techniques?

When your advanced students look for more advanced things that they can do, how likely are they to find things that promote good programming?
Can you teach imperative programming?

Traditional control structures, subroutines (“normal” programming).
Can you teach functional programming?

Languages like Lisp, Scheme, OCaml, Haskell.
Can you teach object oriented programming?

Pure OO languages like Smalltalk and Ruby or languages that also support procedural programming like most other languages these days.
Can you teach "objects first", “objects early”, “objects late”?

Some languages force you to do one of these, others give you some flexibility to make decisions for pedagogical reasons.
The language is small enough for students to keep it in their head?

Can students write the programs that you want them to write without a reference book? This is true to a lesser extent if there is additional complexity that beginners can safely ignore.
<table>
<thead>
<tr>
<th>keyword</th>
<th>keyword</th>
<th>keyword</th>
<th>keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>elif</td>
<td>lambda</td>
<td></td>
</tr>
<tr>
<td>None</td>
<td>else</td>
<td>nonlocal</td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>except</td>
<td>not</td>
<td></td>
</tr>
<tr>
<td>and</td>
<td>finally</td>
<td>or</td>
<td></td>
</tr>
<tr>
<td>as</td>
<td>for</td>
<td>pass</td>
<td></td>
</tr>
<tr>
<td>assert</td>
<td>from</td>
<td>raise</td>
<td></td>
</tr>
<tr>
<td>break</td>
<td>global</td>
<td>return</td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>if</td>
<td>try</td>
<td></td>
</tr>
<tr>
<td>continue</td>
<td>import</td>
<td>while</td>
<td></td>
</tr>
<tr>
<td>def</td>
<td>in</td>
<td>with</td>
<td></td>
</tr>
<tr>
<td>del</td>
<td>is</td>
<td>yield</td>
<td></td>
</tr>
<tr>
<td>Keywords</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>elif</td>
<td>lambda</td>
<td></td>
</tr>
<tr>
<td>None</td>
<td>else</td>
<td>nonlocal</td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>except</td>
<td>not</td>
<td></td>
</tr>
<tr>
<td>and</td>
<td>finally</td>
<td>or</td>
<td></td>
</tr>
<tr>
<td>as</td>
<td>for</td>
<td>pass</td>
<td></td>
</tr>
<tr>
<td>assert</td>
<td>from</td>
<td>raise</td>
<td></td>
</tr>
<tr>
<td>break</td>
<td>global</td>
<td>return</td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>if</td>
<td>try</td>
<td></td>
</tr>
<tr>
<td>continue</td>
<td>import</td>
<td>while</td>
<td></td>
</tr>
<tr>
<td>def</td>
<td>in</td>
<td>with</td>
<td></td>
</tr>
<tr>
<td>del</td>
<td>is</td>
<td>yield</td>
<td></td>
</tr>
</tbody>
</table>
How much "hand waving" is required at the start?

How much do you have to explain to show a first program? How often do you say, “We will learn that later”?
class HelloWorld {
    public static void main(String args[])
    {
        System.out.println("Hello World");
    }
}
What do “Hello World” and “Sum of two numbers” look like?

Let’s look at two very small programs and see how hard they are to explain to beginning programmers.
class HelloWorld
{
    public static void main(String args[])
    {
        System.out.println("Hello World");
    }
}
```cpp
#include <iostream>

int main()
{
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```
print "Hello World!"
say Hello World!
```java
import java.util.Scanner;

public class addTwoNumbers {
    private static Scanner sc;

    public static void main(String[] args) {
        int Number1, Number2, Sum;
        sc = new Scanner(System.in);

        System.out.println("\n Please Enter the First integer Value: ");
        Number1 = sc.nextInt();

        System.out.println("\n Please Enter the Second integer Value: ");
        Number2 = sc.nextInt();

        Sum = Number1 + Number2;
        System.out.println("\n Sum of the two integer values is = " + Sum);
    }
}
<html>
<head><title>Sum of Two Numbers</title></head>
<body>
Number 1: <input type="text" id="n1">
<br />
Number 2: <input type="text" id="n2">
<br />
<input type="button" value="Add"
    onclick = "num1 = parseFloat(document.getElementById('n1').value);
            num2 = parseFloat(document.getElementById('n2').value);
            sum = num1 + num2;
            document.getElementById('ans').value = sum; " />
<br />
Answer: <input type="answer" id="ans">
</body>
</html>
# Python program to Add Two Numbers

number1 = input(" Please Enter the First Number: ")
number2 = input(" Please Enter the second number: ")

# Using arithmetic + Operator to add two numbers
sum = float(number1) + float(number2)
print('The sum of {0} and {1} is {2}'.format(number1, number2, sum))
<html>
<head>
<title>Addition of Numbers Using Forms</title>
</head>
<?php
    error_reporting(0);

$value1 = trim($_REQUEST['val_1']);
$value2 = trim($_REQUEST['val_2']);

// Clear the text field.
if ($_REQUEST['submit2']){
    $value1 = "";
    $value2 = "";
}
?>

<h2><center>Sum of Two Numbers </center></h2>

<form method="post" action="">
    <label>Enter First Value</label><br />
    <input type="text" name="val_1" value="<?php echo $value1;?>"><br />
    <label>Enter Second Value</label><br />
    <input type="text" name="val_2" value="<?php echo $value2;?>"><br />

    <br>
    <input type="submit" name="submit" value="Add">
    <input type="submit" name="submit2" value="Clear">
</form>

<?php
    if ($_REQUEST['submit']){
        $add = ($value1 + $value2);
        echo "<h3>The sum of "$value1." and "$value2." is " "$add";";
    }
when clicked

set a to 0
set b to 0
set sum to 0
ask Enter the First NUM and wait
set a to answer
ask Enter the Second NUM and wait
set b to answer
set sum to (a + b)
say The answer is for 2 secs
say Hello!
```cpp
#include <iostream>

using namespace std;

int main(int argc, char** argv) {
    int a;
    int b;
    int sum;
    cout << "Insert the first number to add: ";
    cin >> a;
    cout << "Insert the second number to add: ";
    cin >> b;
    sum = (a + b);
    cout << "The sum is ";
    cout << sum;
    cout << endl;
    exit(EXIT_SUCCESS);
}
```
Is the language applicable to your grade level?

Logo might not be a good choice for your Jr/Sr level programming course. Or maybe it is.
Are there book available?

Textbooks? Any free textbooks? How many? What quality?
AUTOMATE
THE BORING STUFF
WITH PYTHON

PRACTICAL PROGRAMMING
FOR TOTAL BEGINNERS

AL SWEIGART

NO STARCH PRESS
Dive Into Python

All you need to know to get off the ground and start to fly with Python programming

Mark Pilgrim
Learn Python 3 the Hard Way
A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code

Zed A. Shaw
How easy is it to cover topics like:

iteration (loops of all varieties),
conditionals (of all varieties), recursion,
arrays, multidimensional arrays,
pointers, algorithmic complexity,
memory management. Which topics are important will depend on your curriculum.
Does the language invite code reuse?

Is it easy for beginning programmers to reuse code in other programs? Are there mechanisms in place for importing procedures from external files?
Are error messages useful for beginners debugging?

Can the error messages be used to pinpoint the most likely cause of the problem? Do they help explain the thing that the programmer did that the computer didn’t expect? Are they approachable by beginners?
Are programs "readable"?

How hard is it for a beginning programmer to read through a program and tell approximately what is happening? What is the ratio of English to Symbols? “Perl is executable line noise”, “Python is executable pseudocode” Most languages fall somewhere in between and usually based more on the programmer than the language.
Is documentation well supported?

Is commenting code easy? Are there any other mechanisms that support well documented code?
Concerning IDEs:

Are there IDEs available? Are there free IDEs available? Do you NEED an IDE? There is a school of thought that an IDE just gets in the way of a beginning programmer. It is just another thing to learn at the start.
Are there debuggers available?

Can you step through a program? Can you set breakpoints? Is there a way to examine variable values?
Are there profilers available?

Are there tools included or available to see how efficient your code is?
Are there useful libraries available?

Can you parse HTML? connect to Twitter or the Google search API? Is there a GUI that is easy to implement?
Are there high level data structures built in?

This may or may not be a plus for you depending on what you’re teaching. (stacks, queues, priority queues, heaps, trees, hash tables...)}
When can students write programs they can "feel good about"?

How long will it take to write a program that students consider non-trivial? Maybe that’s defined as one that by writing a program they saved time over doing the same task by hand?
When can students write programs that would impress their friends?

This is probably harder to do. Can you write a program that they would recognize as an “actual” program? Is there some other final product that this language offers instead?
Interpreted or compiled?

Some people feel that one is better than the other for teaching? Do you learn more about how a computer works by compiling? Is the quick feedback nature of an interpreted language better?
Static or dynamically bound variables?

Python is dynamically bound, but I don’t know that that matters in a first language. If you like static bound variables, Python has a typing module that lets you specify the type of variables.
Strong or weakly typed variables?

Python is strongly typed, but I don’t know that that matters in a first language.
How easy will it be to learn other common languages?

Are there things about this language that will make other languages harder to learn? Little idiosyncrasies that students might not understand are not common across other languages?
What did we miss?

What topics did I intentionally avoid so that I didn’t have to tell you reasons not to use Python?
Python in Education
Teach, Learn, Program
Nicholas H. Tollervey