How good are you at programming?

A CEFR-like approach to measure programming proficiency

Raphaela kena Poss
July 2014

The table

<table>
<thead>
<tr>
<th>Programming skills self-assessment matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A1</strong></td>
</tr>
<tr>
<td>Base Level</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>Writing code</td>
</tr>
<tr>
<td>Refactoring</td>
</tr>
<tr>
<td>Debugging a larger system</td>
</tr>
<tr>
<td>Reading code</td>
</tr>
<tr>
<td>Understanding and evaluating code</td>
</tr>
<tr>
<td>Debugging, testing, and writing code</td>
</tr>
<tr>
<td>Mastery of the environment</td>
</tr>
<tr>
<td>Troubleshooting</td>
</tr>
</tbody>
</table>

The online test

A web application is available to assess your own skills according to this table for one or more programming languages.
How to use this table

The table characterizes the proficiency level (columns) of programmers of a particular programming language in the context of different programming activities (rows).

This table is inspired by the CEFR table of the same name, for assessing proficiency in natural languages. Like the CEFR, this table divides learners into three broad level divisions: Basic user (A), Independent user (B) and Proficient user (C). The broad divisions are each further divided in two levels (A1, A2, B1, B2, C1, C2) that correspond to testable milestones in language acquisition.

This table can be used in different ways, for example:

- row by row, to assess one’s own level per activity (different skill levels for different activities are possible);
- column by column from left to right, to determine one’s own minimum level for a programming language (the rightmost level where all requirements in the column and all columns to the left are matched);
- column by column from right to left, to determine one’s most developed skill (the rightmost level where any requirement in the column is matched);
- language per language, to assess one’s own relative proficiency in different programming languages.

Possible applications

- to track one’s own progress while learning how to program;
  for example: this year, I transitioned from A2 to B2 in C++. For Java, I am B1 for understanding but still A2 for writing.
- to advertise the educational goals of a programming course;
  for example: this Java introductory course will bring you to level A1 or A2 for all activities.
- to advertise one’s own skillset to peers or prospective employers;
  for example: I am C1 in Python, B2 in O’Caml and A2 in Haskell.
- to set basic level requirements for courses or professional activities:
  for example: This course requires A2 proficiency with a language in the C family.
- to select a programming course that best matches one’s skill level;
  for example: My current level is A2 but this course requires B1, so I will need some extra work before starting.
- to coordinate the teaching objectives of successive programming courses in a curriculum;
  for example: Our introductory course brings students to A2 in Java, but our follow-up program requires B1 or B2, so we need to propose a supplementary course for that level.
Design methodology

The table was designed following the CEFR methodology:

First, recognize the different modalities (production, reception, interaction, mediation). Then group and abstract activities in each modality by clusters that share similar motivations and actors. The resulting set becomes the rows in the matrix.

The requirements for A levels are then phrased to identify users that can perform language acts under supervision or under dedicated/personalized guidance from peers.

The requirements for B levels are then phrased to identify users that can perform language acts without supervision or guidance from peers, or with minimal effort from peers.

The requirements for C levels are then phrased to identify users who are fully independent and who can demonstrate skills corresponding to a high expertise level according to the majority of other users of the language.

References


Acknowledgements

The initial idea of an assessment matrix appeared during discussions with Merijn Verstraaten about average programmer skills in online fora. The concrete form as presented here was further developed thanks to inspiration and suggestion from Philip Hölzenspies, Jan Kuper, Cosmin Dumitru, Taddeüs Kroes, and many others.

Copyright and licensing

Copyright © 2014, Raphael kena Poss. Permission is granted to distribute, reuse and modify this document according to the terms of the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/4.0/.

SC fingerprint: fp:GynRnlqPoH3C-D5wQOqRi0om0wsgHms47b5Vz0xDhfN7XA