Highlights

- Collected advice from students who were about to finish their first course in computer science (similar to our CSCI 203).
- Data from three universities of varying size and mission.
- Most advice were general (63%), followed by attitudinal (34%), followed by programming-specific (23%). These don’t add up to 100% which will be explained.

Data Collection and Result

- 73 students, 63 students, and 84 students in each of the three institutes participated.
- The advice from these students can be categorized into three groups:
  - General study
  - Programming specific
  - Attitudinal

General Study Advice

- Analogies: Making analogies to other domains such as foreign languages or music.
- Attend Class
- Consider Alternatives: Ask “What if?” questions, take different approaches.
- Group Work: Work/study/learn with others.
- Learn from Mistakes: General, no reference to process, compiler, etc.
- Learning Approaches: e.g., learn basics first, build a strong foundation.
- Listen/Pay Attention/Concentrate.
- Pay Attention to Feedback.
- Practice: General advice about practicing, trying on your own.
- Questions/Help: Ask for help in person or by email.
- Resources: Use resources such as Google, Java API, reference books.
- Stay on Top of Things: Keep up with the work, programming is cumulative.
- Take a Break.
- Textbook: Any reference to the textbook.
- Understand Expectations: Know what the professor expects.

Programming Specific Advice

- Conceptual Understanding: Getting a big picture, forming a mental model, etc.
- Details Are Important: Pay attention to syntax, keywords, punctuation, etc.
- Learning Approaches: Learn syntax first, etc.
- Learn from Mistakes/Errors: Bugs and compiler messages are learning opportunities.
- Practice: Experiment/tinker, introduce bugs intentionally, etc.
- Problem Solving -- Other: Look at the big picture, program from examples, etc.
- Problem Solving -- Work Incrementally: Approaches that specifically mention programming in small chunks or steps.
- Programming Style: Use good identifiers, indentation, comments, etc.
- Testing/Debugging/Fixing Errors: Tracing, diagnostic statements, test cases, etc.
- Use Tools: Use the compiler, debugger, IDE, or other tools.

Attitudinal Advice

- Avoidance: Don't take the class in the first place.
- Be Confident.
- Don't be Frustrated; Be Patient.
- Don't be Overconfident.
- Don't be Scared/Intimidated.
- Don't Panic: Stay calm.
- Focus Long Term: Keep a long term perspective.
- General Encouragement: Keep a good attitude, focus on learning.
- Have Fun/Enjoy Learning.
- It's OK to be a Novice: Understand you're a beginner, learn at your own pace.
- It's OK to be Confused: Confusion is a natural part of learning to program.
- Persevere: Don't give up; there is always a solution.
Method of Study

- Authors (in each of the three institute) evaluate student responses independently, came up with three general categories.
- Each student response is then evaluated to see if it includes each of the three advice type. A student response can contain multiple types of advice.
- Authors independently decides if a response contains any of the three types. See the following table for results.

Coded Advice Distribution

<table>
<thead>
<tr>
<th>Number of Coders</th>
<th>General</th>
<th>Program Specific</th>
<th>Attitudinal</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>63.4%</td>
<td>22.6%</td>
<td>33.5%</td>
<td>39.8%</td>
</tr>
<tr>
<td>2</td>
<td>9.7%</td>
<td>9.8%</td>
<td>5.5%</td>
<td>8.3%</td>
</tr>
<tr>
<td>1</td>
<td>9.2%</td>
<td>10.4%</td>
<td>5.5%</td>
<td>8.3%</td>
</tr>
<tr>
<td>0</td>
<td>17.7%</td>
<td>57.3%</td>
<td>55.5%</td>
<td>43.5%</td>
</tr>
<tr>
<td>0 and 3</td>
<td>81.1%</td>
<td>79.9%</td>
<td>89.0%</td>
<td>83.3%</td>
</tr>
</tbody>
</table>

Examples of General Advice

- Practice their skills with a large variety of different programs that would require them to use a lot of different forms of the program.
- Use examples of other programs that experienced programmers have done and try to do one similar.
- Keep on top of the notes and chapters is important because the next lecture keeps building from what you just learnt so don’t fall behind.
- If you fall behind in programming, it’s very hard to catch up.

Examples of Programming-Specific Advice

- I highly recommend trying things. “What if I do ____”, then test it. Often I have to repeat this 5 -- 6 times ...
- Copy code given in lectures and alter it, change it to a subject you have an interest in.
- Make deliberate mistakes to see output.
- You'd better do the programs step by step. It means don't write the whole program and run it then try to debug it, ...
- Also, breaking the tasks/programs into smaller parts or steps ... make it easier to deal with.
- Thus, you are not always overcome with the size of the program you are dealing with.

Examples of Programming-Specific Advice (2)

- I often notice my colleagues grow frustrated with the compiler during labs ... as they expect the computer [to] “think” like a person, and be able to understand what they want without explicit instructions. Those new to programming should realize that computers have no form of derived intelligence ...
Attitudinal Advice

• Also, realize that [THIS COURSE] is an introductory class. If ever you feel frustrated, just know that everyone else has been there too.

• The main thing I would suggest is to make sure you ask questions in lectures and not to worry about looking stupid for not knowing. Everyone is at different levels, there are bound to be other people that know less or the same as you.