Please fill in the information below and upload the proposal form (in PDF format) at the secure submission website for consideration for presentation at the iSchool Partnerships and Practices track at the 2019 iConference in College Park, Maryland, USA. Please keep to the advised length or the proposal will not be considered for review.

Please consider also the key review criteria for selection:

- Transferability to other institutions
- Grade of innovation
- Pedagogical dimension
- Degree of knowledge transfer

Questions about the iSchool Partnerships and Practices track should be directed to the chairs of the track:

**iSchool Best Practices Chairs**

- [Elke Greifeneder](#), Humboldt-Universität zu Berlin
- [Sean McGann](#), University of Washington
- [Timothy Summers](#), University of Maryland, College Park

For general questions about the iConference, please contact iConference Coordinator [Clark Heideger](#).

<table>
<thead>
<tr>
<th><strong>Name(s) of Author(s):</strong></th>
<th>Jennifer Proctor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Title of submission:</strong></td>
<td>Teaching Functional Coding Skills: Designing assignments that challenge, inspire, and support</td>
</tr>
<tr>
<td><strong>Area (please check the applicable area description with an</strong></td>
<td></td>
</tr>
</tbody>
</table>
Submission abstract (max 150 words):
Teaching programming-aided subjects in an iSchool where coding proficiency is, by design, not a prerequisite, is difficult. Challenges include:
- classes with students of varied experience with coding in general and the chosen language in particular,
- large class sizes complicating 1-on-1 support and troubleshooting,
- students struggling to focus on the classic, boring Code-Along,
- and problems guiding students in the transition from tutorials to writing original code.

This presentation will answer these difficulties by equipping attendees with pedagogical techniques for engaging students through active learning, explaining design principles for challenging students at all ability levels, and exploring methods of helping students develop the skill sets needed for programming independence.

Teachers and aspiring teachers at all professional levels will be able to benefit. The example exercise will focus on an undergraduate data science class and use Python, but lessons are applicable to subjects and levels across iSchool programs.

Submission description (max 2,350 words):

Programming is not easy to teach. Professors know it. Students know it. Everyone who’s ever sat in a classroom while a teacher typed on a screen and expected their students to learn by mimicry knows it. To teach coding in an iSchool - to truly equip our students with the functional skills necessary to generalize and apply lessons in code to new contexts - a creative approach is needed.

The Case in Focus

This discussion will focus on a lesson on web scraping using Python as a means to explore the challenges and opportunities inherent in teaching coding skills in an iSchool. It will address lesson design to incorporate trends towards active and student-centered learning and attempt to push students towards higher order thinking. And, it will elucidate a hierarchy of coding skills to aid in adapting the principles to other levels and lessons that I hope will be valuable to teachers across iSchools, programs, and subjects.

This lesson was designed for a blended format course, half online asynchronous assignments and lectures with the Professor and half in-person
lab activities led by a teaching assistant. This upper-level undergraduate Data Science class is one of several courses for students majoring in Information Science and specializing in Data Science. The only prerequisite is a course in basic statistics also taught in the iSchool.

**Coding as a Means, not a Learning Outcome**

This Data Science course, like many iSchool technical subjects, did not directly require coding and unlike Computer Science courses, where the coding skills are core to the program and its courses, proficiency in the instructor’s chosen language was not a prerequisite. iSchool courses have much to offer even non-coding students and so do - and should continue to - accept students at all levels of coding ability. The implications of this are two fold. First, it means that the core content of the course must be understandable regardless of a student’s ability or lack of ability at coding in a particular language. Second, it means that you may get a much wider range of coding abilities in the same class than in coding classes.

**Beginners and Professionals in the Same Class**

The class had 50 students, and from questions I asked them on Day 1, I knew most were interested in Data Science as a marketable tool to incorporate into future careers in areas like health, business, and sports. About a third rated Python their preferred coding language. Several had worked in jobs that required them to code professionally. On the other hand, about half identified themselves as having little or no experience with Python and while all were technologically savvy, some had little experience with coding at all.

**Large Class, Small Classroom**

Teaching coding in a classroom setting is hard enough when you can help students 1-on-1 if they run into trouble. With a 50-person class in a 55-person classroom, I couldn’t get to most of my students when everyone was seated and, even if I could have, in two 75 minute lessons, there wasn’t a lot of time to provide individualized support.

**Pedagogy**

While each of the challenges was difficult, they were also opportunities for creativity. Inspired by my University’s push for active learning methods and student-centered teaching [1], I was strongly against simply teaching web scraping by Code-Along, but I wasn’t sure what to do instead. Inspiration came from drawing together an array of bits and pieces of advice I have picked up.

Two pieces of advice were given to me during Teaching Assistant Orientation:

1) to improve student writing in a particular discipline, model good writing in that discipline through critical reading assignments

2) build assignments that offer more for advanced students to do than
beginners to keep them actively learning and get all students finishing at the same time.

This led me to the three level model and gave me the idea of modeling the Beginner’s assignment on critical reading worksheets I had already implemented to prepare my students for the Communicate Results component of the Data Science process.

But while my critical reading worksheets worked by drawing attention to the presentation of key pieces of information in different formats, how could this be adapted to coding? Would it be like drawing attention to different pieces of syntax at the beginner level and forms of structure at the intermediate? That sounded a lot like a classic coding tutorial.

**What are ‘functional coding skills?’**

But anyone who’s had the experience completing a lengthy tutorial in a new coding language only to finish the last module, step away, and immediately realize that they still have no idea how to adapt what they’ve just finished for anything useful knows there is a difference between finishing some assignments and learning coding.

So I thought about my anthropology background, my library science education, my experiences as a student learning to code, and the student-centered teaching model, and I shifted my focusing from the coding language to the student. What would it look like when a student could really code at a beginner, intermediate, or advanced level? This is what I came up with:

<table>
<thead>
<tr>
<th>Stages of Coding Competence</th>
<th>Intermediate Stages of Learning</th>
<th>Attributes, Abilities, and Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null</td>
<td></td>
<td>Has no frame of reference for coding - does not understand what code is, what it does, or how it works</td>
</tr>
<tr>
<td>Zero</td>
<td></td>
<td>Knows what coding is and how it works in principal but has no familiarity with any coding languages</td>
</tr>
<tr>
<td>One</td>
<td></td>
<td>Has developed familiarity with the basic syntax of the coding language in question, either through a good tutorial or a well-written syntax guide such that they can recognize the principles when seen</td>
</tr>
<tr>
<td>Two</td>
<td>A</td>
<td>Can read and understand individual lines of basic code (but not necessarily relate them to each other) but still needs help understanding non-syntax rules and functions (like python's .text)</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Can relate lines of code to one another and recognize patterns within them, complete basic statements by applying syntax rules, can name and correctly apply variables within an existing structure of code, provided they are guided through the process of locating and constructing the statements that fill the variables</td>
</tr>
<tr>
<td>Three</td>
<td>A</td>
<td>Can troubleshoot existing code that is 90% or more correct and contains only basic to moderate difficulty</td>
</tr>
<tr>
<td>A</td>
<td>Can read and understand error messages with or without the help of web searching, but needs help fixing errors</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>B</td>
<td>Can read language and package documentation to identify the function of particular statements and the capabilities and methods of functions of interest</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Can do all of the above, plus create basic code from line by line instructions, but must be given moderate or complex portions of the code directly</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Can insert troubleshooting code such as print commands in strategic places to identify the location and nature of problems</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Can google their way out of problems by recognizing relevance of results, analyzing and evaluating proposed solutions, and testing via trial and error until a solution is found</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Can make reasoned choices about how to structure code to solve a problem given basic structural guidance</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Outlines basic structure of code before writing [2]</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Can recognize a real world problem that can be solved with code and devise code of any complexity to solve it</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Outlines complex structures of code before writing and can reformulate to address problems as they arise [2]</td>
<td></td>
</tr>
</tbody>
</table>

From what functional coding skills would look like, I thought about how they could be taught. My own teaching philosophy is heavily committed to developing students’ higher order skills, so I remembered Bloom’s Taxonomy.

Image 1: Bloom’s Taxonomy [3]
Code-Alongs, I realized, focused on Remembering and, hopefully, Understanding. Tutorials added Applying. I wanted to do better. I wanted even my Beginners to be engaged in each of these, plus Analyzing by actively examining and discovering. I wanted my Intermediate group to need to engage in Evaluating by interpretation. And I knew my Advanced group was capable of Creating, if the assignment was designed to solicit it.

The Assignment(s)

Drawing together all of these influences, I designed the web scraping lesson thusly. To ensure I captured the interest of my students, undergraduates approaching graduation and therefore invested in finding internships and jobs, I decided to have them scrape job ads. Some research identified Indeed.com as the best combination of quality ads and low risk of banning.

For the advanced students, I provided only the url and a prompt describing the types of information to collect and how I wanted them formatted. Then I completed the task myself, creating a Teacher’s Key on which I based the other assignments. In between the two class sessions set aside for the lesson, I also initiated a discussion asking them to share visual or textual representations of the structure of their solution, whether they did it as an organizer before starting coding or retroactively when asked to think about it. The goal of this was to draw their attention to the role of pre-planning the structure of a coding task which is a key functional coding skill for experienced coding students to help them ‘level up’.

For my intermediate students I gave detailed instructions, line by line, for
most of the task, as well as a few lines of code where the idiosyncrasies of the
Indeed job ads made collecting salary information particularly tricky. I then
asked this group too to retroactively draw the flow of information through
each stage of the script to get them to start thinking about structures.

My beginners were given a heavily commented fill-in-the-blank script file with
strategically inserted errors. I talked them through the script line by line,
‘reading’ the script to them in plain language so they could understand what
each line was doing. I guided them to discover for themselves how to call
each piece of information from within the html and how to compare portions
of the code and apply what they saw to later blanks. And I taught them the
process of troubleshooting, using the errors as a way to review Python syntax,
teach the most common errors, and practice methods of resolving problems
when coding including reading documentation and searching Stack Exchange
and web-published tutorials for possible solutions.

At each level, I took great care to design elements to build specific coding
skills and help students at each stage “level up”. Additionally, for all students,
I began the lesson talking about the ethical and legal debates around web
scraping and its importance as a research method in Information Science. I
also provided a list of points to consider before decided whether to scrape and
how.

Applying and Adapting

The design principles here can be useful in other circumstances. This Data
Science course, like many iSchool technical subjects, did not directly require
coding, but students would certainly benefit from using programming in it.
Library technology, database design, data management, digital curation, and
data visualization are just a few examples of other courses in the same
position.

Works Cited

1. University of Maryland - College Park, Teaching and Learning Transformation Center: Fearless
2. Robins, A., Rountree, J., Rountree, N.: Learning and teaching programming: A review and
3. Writing Measurable Course Objectives | The Center for Teaching and Learning | UNC
   Charlotte, https://teaching.uncc.edu/teaching-guides/course-design/writing-measurable-
course-objectives