The primary thrust of my work is about Big Data Science as a context for introductory programming, but in order to use this as the context, it must be facilitated through certain new technologies. I've already had preliminary success in providing data sources through python libraries with a caching layer, but this is only part of the problem that beginners have during their introductory learning experience. It is unreasonable to expect that a novel context alone will have meaningful impact on students learning experience – the context does not live in a vacuum, but is part of a system with complex interactions. The context is a crucial part of a students introductory learning experience, but I think it will prove unfruitful to study it in isolation from other components; therefore, I'm positing my existing research in the broader problem of "introductory computing experiences". The proposed experience for a learner has a context of Big Data Science, facilitated through a Block-based Programming Environment that's tuned for Big Data Science exploration.

This Block-based programming environment (named "Kennel" for now) features high-level support for my data sources. I direct your attention to figure 1, which demonstrates the current version of Kennel that has been used in the Computational Thinking course. You can see the integration of the data sources through the Data blocks in the lower left.

Using these data sources becomes relatively trivial, and adds important motivation to the context. However, this in and of itself is insufficient for getting students to succeed. Think of my existing work as half the battle, the part where we get students to want to succeed (and indeed, I have survey data that suggests that we have motivated students); now we need to be able to guide them to actually succeed. Success in this case is defined as the students creating algorithmic solutions to computational problems (whether self-directed or after being given a problem) in a serious programming environment such as Spyder, Eclipse, PyCharm, etc.. In order to get them there, I'm proposing several pieces of technology that build on the state-of-the-art. Although this may seem diverged from what I've been doing before, it's all part of a slightly bigger picture of getting introductory students to success. I've proven that this context motivates them - now I need to show that, given the right facilitations, instructors can use to motivate them and get them to perform to some specified level of performance. So, what is this block-based environment that I want to build?

First, let me stop referring to it as the block-based programming environment, and instead refer to it as a Data Science Environment for beginners. It has a large amount of technology wrapped up in it, and the Block-based component is the least interesting from a research perspective (the Snap! people have already proven that BBP is effective). So let me make a list of the different technologies and why they are crucial to it:

Big Data Sources This is most important component – the same libraries that work in Python/-Java/Racket need to work in this environment, to really make it so students can explore a useful and empowered context. I'll be leveraging Jonathon's technology to make it so that users can connect to arbitrary data sources, but there'll also be a lot of built-in ones. The same caching mechanisms and lessons that I've learned from my existing work all apply to this.

Browser-based This environment needs to be readily available to instructors, with no trips to the server that are unnecessary. I've sewn together Blockly and Skulpt to do this heavy lifting – everything happens quickly on the client, and we aren't tied to a single LMS.

Visualization You can't really do much with data if you're not visualizing it – seriously, you can only calculate the average temperature in a city so many times before it gets old. By having rich visualizations, you open a whole other dimension of problems. This is part of my research's ongoing conversation about "what kinds of problems and questions can we ask students using Big Data Science?"

Big Data Exploration Students struggle with getting to know the data, and there are things that the interface will be able to do – Spyder (a Python programming environment) is able to break down the current value of complex variables after code is run, in a series of nested windows (e.g., a list of dictionaries of integer values can be shown in multiple windows). This kind of visualization can be explored to see how it support students' understanding.

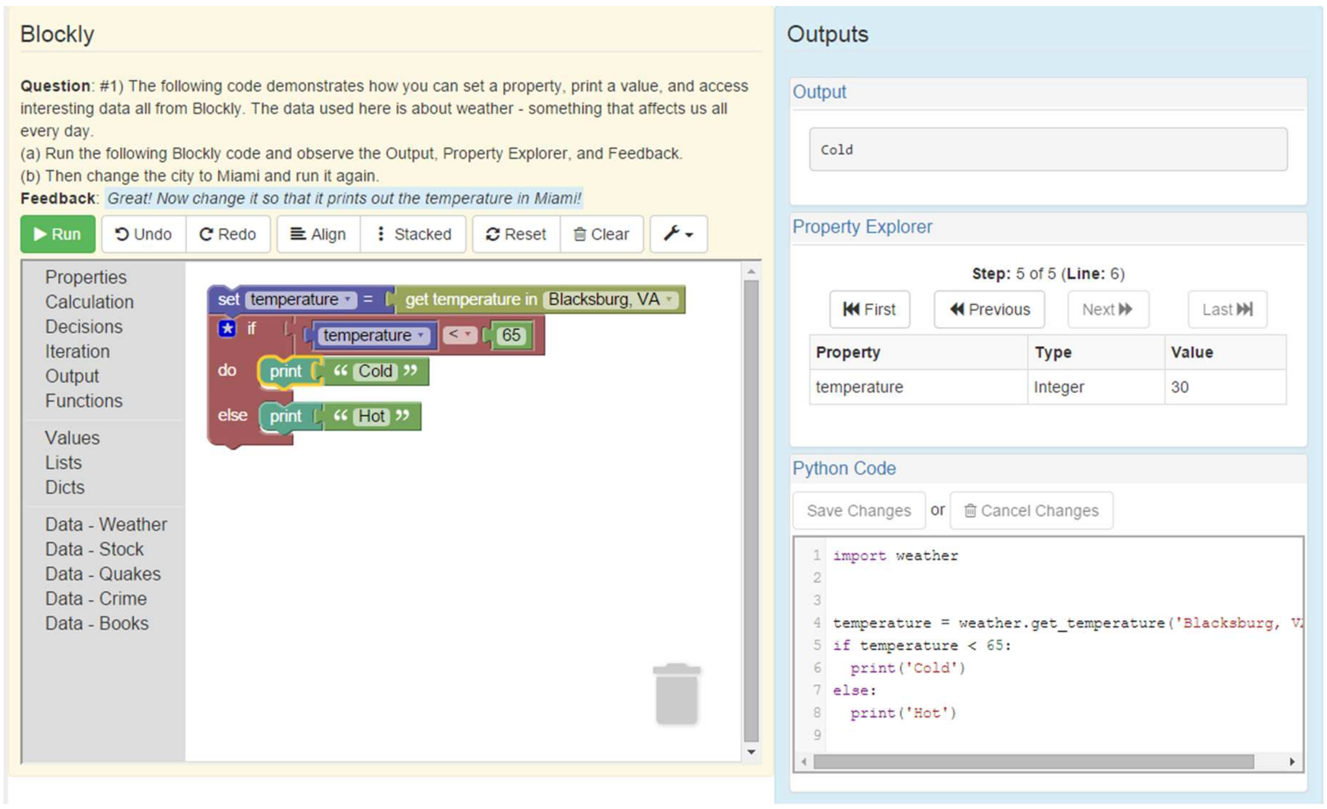
Mutual-Language Transformation [?] published a paper on this at SIGCSE this year, and its one of the more interesting technical problems. The goal is to get students to be writing code in a real environment, not connecting blocks. I can show data from the Computational Thinking class that shows that students *want* to outgrow the blocks and start doing real-stuff (not all students, but a good number). MLT lets them get to that point in a seamless way. At the same time, we had many students unaware of how to take advantage of this, so there's a lot of research potential in figuring out how to get students to use it gracefully. And from the systems side, its actually really hard to create a semantic block-based mapping to text for a dynamic programming language like Python - tons of interesting little technical challenges. I already have a working version of this.

Inline Code Blocks [?] presented a poster in the SRC on this at SIGCSE this year, and its also very interesting. Much like the MLT, this is meant to let students transition to writing real code. It may be more effective – it's an open-ended problem.

Problem Support Students make a lot of mistakes, and most of these can be picked up automatically – for example, not knowing to use iteration, or trying to iterate over an empty list. Human instructors are in short demand, so this environment should be able to provide some level of guidance that can help alleviate the bigger mistakes that students have with a given question. This is a mixture of static and dynamic program analysis.

Student Tracking The system needs to be able to keep track of all the students and stuff, so the staff can know who's struggling. This means logging student work.

Most of these things are technical features, but they all support the overarching research question of how we can motivate and guide students to explore Computing via Big Data Science - they're facilitations. I don't think I can do much interesting work without doing them. Now, they may not all feature in my prelim, but I think it's a convincing story to fit them all together into the same environment in order to guide students. A big question is how to know what is helping and what isn't – the answer is user studies, log analysis, surveys, and instructor reports. That'll be how I collect data on the different components.



Question: #1) The following code demonstrates how you can set a property, print a value, and access interesting data all from Blockly. The data used here is about weather - something that affects us all every day.
(a) Run the following Blockly code and observe the Output, Property Explorer, and Feedback.
(b) Then change the city to Miami and run it again.
Feedback: *Great! Now change it so that it prints out the temperature in Miami!*

```
Cold
```

Step: 5 of 5 (**Line:** 6)

Property	Type	Value
temperature	Integer	30

```
import weather


temperature = weather.get_temperature('Blacksburg, V
if temperature < 65:
  print('Cold')
else:
  print('Hot')

```

Figure 1. Current Version of Kennel

References

[1] Y. Matsuzawa, T. Ohata, M. Sugiura, and S. Sakai. Language migration in non-cs introductory programming through mutual language translation environment. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, pages 185–190, New York, NY, USA, 2015. ACM.

[2] D. Weintrop. Minding the gap between blocks-based and text-based programming (abstract only). In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, pages 720–720, New York, NY, USA, 2015. ACM.