Position Paper: From Interest to Usefulness with BlockPy, a Block-based, Educational Environment

Austin Cory Bart, Eli Tilevich, Clifford A. Shaffer, Dennis Kafura
Computer Science
Virginia Tech
Blacksburg, VA 24060
{acbart, tilevich, shaffer, kafura}@vt.edu

***Abstract*—As block-based environments are used for more mature audiences, the environments must mature themselves. Based on holistic theories of academic motivation, this means making the environment present itself as both interesting *and* useful, without sacrificing pedagogical power and scaffolding. We present Data Science as a potential context that satisfies all of these constraints, and describe our new block-based programming environment for education that supports data science from day one: BlockPy, available at `http://think.cs.vt.edu/blockpy/`. BlockPy features a number of powerful, authentic features meant to promote transfer for students to conventional environments as they progress. This includes mutual language translation and interactive feedback, but also powerful tools for getting real-world data and visualizing it. As we have developed the tool, we have identified a number of major research questions that should be answered in order to determine the validity of our hypothesis and the potential of our approach: in particular, how can this environment and context support educators and diverse learners as they progress into conventional environments.**

I. Problem

How do we bring introductory computing to mature, domain-identified undergraduates, who have concerns for both their own self-efficacy and for the value in learning computing? Many universities are now defining core credit hours in subjects such as “Computational Thinking”, introductory computer science classes meant for solving interdisciplinary problems using some degree of programming. This means that universities now have students of every different discipline and background taking a programming course. Students with a clearly domain-identified interest (i.e. their major) may view non-major courses with doubt and suspicion – what does this have to offer them, and why should they engage?

Our position is that, in order to fully engage all undergraduate students, an introductory programming environment should be both *interesting* and *useful*, while still promoting *success*. Critically, this means that the environment should enable working with a context that students relate to, enjoy, and helps them solve useful problems. Further, they should feel that the material that they’re learning will help them to transition to more authentic, serious problem-solving of real programming environments. These changes cannot come at the cost of the pedagogically valuable scaffolding, but instead should provide new opportunities for students’ learning.

A. Existing Solutions

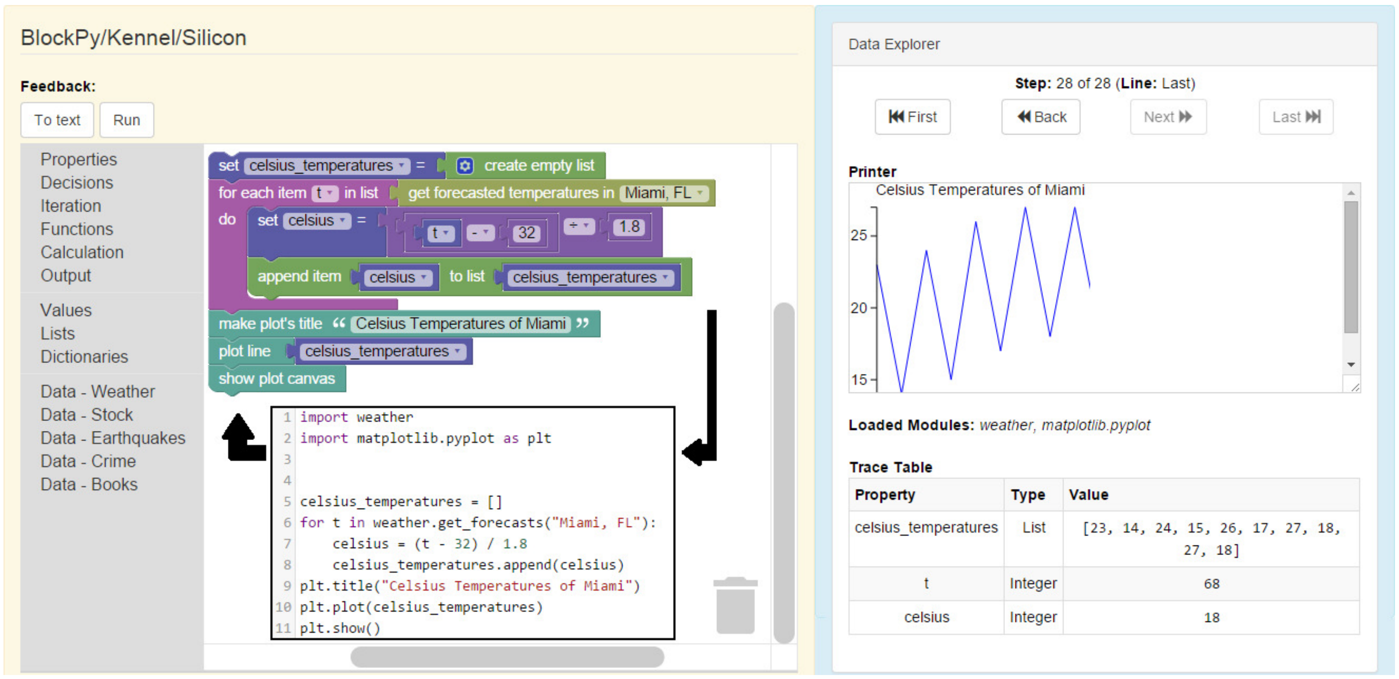
Although Block-based environments like Scratch and Snap! have already been successful with high-school students, they may not be suitable for more mature but diverse populations. In addition to side-stepping syntax headaches, Snap! and Scratch environments make it easier to get working with their motivating educational context, such as game and animation design, robots, or media computation, because they expose the range of actions afforded by the context (e.g., a “draw a circle” block for media computation) and support the experiences directly (e.g., a drawing canvas embedded within the environment). These contexts can be popular with certain students: young male children usually enjoy creating games, for example, so game and animation design is a compelling hook. However, many students at the undergraduate level may doubt the usefulness of the environment, as they consider it a toy rather than a useful tool, not only because of the puzzle-piece metaphor they utilize, but also for the evironments’ contexts.

B. A New Solution: Data Science

We submit *Data Science Exploration* as an educational context that block-based environments should support as a priority, similar to how Snap! and Scratch support game design. Data Science for Introductory Computing is a growing movement, with many instructors recognizing the inherent value [1], [2] Data science provides an authentic, useful context for every kind of student, since exploring data-oriented problems is something that almost all fields are beginning to find relevant [3], [4]. Additionally, it is readily possible to find data sources that connect to the world around the student and their past experiences, establishing a sense of personalized interest. When students inevitably ask, “What am I going to be using this for?”, it is possible to point to well-defined data problems in their field requiring computation. This does not mean that we are creating an end-user programming environment for data science, however, but an educational environment that allows students to learn computing principles through the context of data science (similar to how Scratch is meant to learn programming, rather than to learn game design).

C. Academic Motivation and Blocks

The prior research on block-based environments in undergraduate settings sheds some light on the limitations of environments for those populations. Mishra conducted a two-week intervention in an introductory course where students worked with Scratch before they began working with Java, reporting positive outcomes in both learning and engagement [5]. However, although the sample population was large (N=450), the students do not represent the typical body of a university:


BlockPy/Kennel/Silicon
Feedback:
To text
Run
Properties
Decisions
Iteration
Functions
Calculation
Output
Values
Lists
Dictionaries
Data - Weather
Data - Stock
Data - Earthquakes
Data - Crime
Data - Books
set celsius_temperatures = create empty list
for each item t in list get forecasted temperatures in Miami, FL
do set celsius = t - 32 ÷ 1.8
append item celsius to list celsius_temperatures
make plot's title “ Celsius Temperatures of Miami ”
plot line celsius_temperatures
show plot canvas
Data Explorer
Step: 28 of 28 (Line: Last)
First
Back
Next
Last
Printer
Celsius Temperatures of Miami
25
20
15
Loaded Modules: weather, matplotlib.pyplot
Trace Table

```python
import weather
import matplotlib.pyplot as plt


celsius_temperatures = []
for t in weather.get_forecasts("Miami, FL"):
    celsius = (t - 32) / 1.8
    celsius_temperatures.append(celsius)
plt.title("Celsius Temperatures of Miami")
plt.plot(celsius_temperatures)
plt.show()
```

Property	Type	Value
celsius_temperatures	List	[23, 14, 24, 15, 26, 17, 27, 18, 27, 18]
t	Integer	68
celsius	Integer	18

Fig. 1. A complete representation of BlockPy (`http://think.cs.vt.edu/blockpy/`)

they were engineering majors, 88% male, and “highest ranked” in “mathematics, physics, and chemistry”. The students did vary greatly on prior programming performance, but their other demographics suggest that they have uniform motivational concerns, compared to the general undergraduate population. In particular, many students cited the game design context afforded by Scratch as a major motivating factor: “[I am] thrilled to be able to code complex games” and “[coding] games helped increase my interest, [...], there was lot of room for experimentation.” These students valued the game design component because it was interesting to them, but not because they saw it as useful to their careers. It is unclear how more diverse students would react to this environment.

In our research project, we use the MUSIC Model of Academic Motivation [6] to explain the different ways students become engaged. Specifically, this model differentiates between five components of motivation: **eMpowerment**, the control that a student feels that they have over their learning experience; **Usefulness**, the student’s expectation that the material will be valuable to their short- and long-term goals; **Success**, the student’s self-efficacy; **Interest**, the situational and dispostional value the student’s feels; and **Caring**, the student’s perception of their professor’s and classmates’ attitudes toward them. We apply this theory to describe existing game-based programming environments as providing Interest to certain populations, but limited opportunities for Usefulness. We suggest that more students can be better served with a context that supports all five dimensions and, in particular, both Interest and Usefulness. To that end, we have created a new block-based environment with this in mind.

II. BlockPy

In this section, we concretely describe our work on a new web-based, dual text/block environment: **BlockPy**, a beginner-friendly programming environment that scaffolds the learner into a more mature environment while supporting a sense of Usefulness up-front. Internally, BlockPy uses a modified version of the open-source Blockly library to provide a block editor, a modified version of the open-source Skulpt library to execute Python code client-side, and an unmodified version of the open-source CodeMirror library to provide a text editor. Figure 1 demonstrates the interface: the problem presentation and feedback on the top-left, the dual program representations in the bottom left (via Mutual Language Translation), and the data science dashboard on the right (giving students powerful insight into their programs execution).

A. *Data Science as a First-Class Feature*

The code represented in the figure demonstrates the data science API exposed to the student, including blocks and functions to access real-world data sources and to create visualizations. These data sources include weather forecasts, earthquake reports, and stock feeds. Data returned from their interface is extremely simple – usually either primitive (numbers and text) or minimally structured (maps and lists), ensuring that students can begin working with Big Data blocks at the earliest possible points in the course. In addition to obtaining data, we support the popular MatPlotlib library to provide a set of visualization functions create simple line plots and histograms. By basing everything around the MatPlotLib API and relying on the Blocks interface for scaffolding, BlockPy seeks to maintain complete compatibility with conventional Python APIs so that all code written is authentic, as opposed to the use of simplified toy APIs in environments like CodeSkulptor.

B. *Guided Practice*

BlockPy is not just a code-authoring environment but also a system for guided practice. Instructors can create problems by writing introductory text and then using an assessment API to define interactive feedback. Specifically, the instructor can define rules based on students' current code, output, and program state, and gives automatic feedback to the student. This just-in-time feedback is meant to guide students to success. Of course, the environment also supports free-form coding experiences, as you would find in traditional programming environments; as the students progress through their introductory experience, short-term feedback can decrease and then fade away.

C. Transfer to Textual Languages

Although some research is working towards creating end-user block-based environments, we view block-based languages as "Training Wheels", meant to be faded away. Work by Weintrop on the transition from Snap to Python analyzes this transition and offers a number of ways to mediate the transfer through programming tools. One of the largest findings is that being able to write inline code inside a Block-based language is extremely helpful to students' learning [7]. Another approach we support is Mutual Language Translation, devised by Matsuzawa [8], that creates an isomorphic view of students' code as both text and blocks. These features are meant to transfer students away from blocks towards text.

D. An Example Scenario

Consider a lesson for students on Iteration. The instructor could create a problem asking students to find the average temperature in their local city for this week, using the assessment API to construct rules demanding they use iteration blocks. Students would drag in a `Get Temperatures for [city]` block. If they attempt to run their code, they will be reminded that averaging requires iteration. As they continue, they switch between the block and text view as they feel comfortable. They also use the data explorer to step through their code and watch its state change. Finally, when they successfully print the answer, they are given positive feedback.

III. Research Questions

Our new programming environment offers a number of affordances to educators, but much of its promise is still unproven. Beyond just usability testing, we wish to explore questions relating to the nature of using data science in a block environment. One of the major values of a context is being relatable – it should be a metaphor for students, helping to build on their prior knowledge. Will the entire undergraduate population find data science to be sufficiently relatable? For instance, some students may possess weak math skills or have low self-efficacy with math. Will they find the necessary mathematics (e.g., finding the average of a list) too confusing?

Along similar lines, how do we quickly introduce students to a given dataset, and make them comfortable manipulating and understanding the data it contains? What interaction can the students have with the blocks in order to aid this experience? In the datasets currently supported by the environment, some are "easier" than the others: our students had no trouble working with weather data, for instance, but struggled when confronted by stock trading data. Are some datasets inherently more suitable for introductory experiences? And just how crucial are students' perceptions of interest and usefulness?

Of course, our environments' affordances also raise questions. In our experiences with using a block-based environment to scaffold learners into a conventional environment, the transfer can be rocky. Some students are eager to start using the text-based environment and do not need to pushed to move away from the blocks. However, some students may be wary about losing their training wheels and, if left to their own devices, may choose to delay trying out the text-based code. How do we gracefully transition students to coding text, based on the students' abilities, motivation, and the course's time table? How does Mutual Language Translation support and hinder this process? And how do we provide accurate block-based representations of a dynamic language like Python – consider the difficulties involved in inferring whether a variable block has the appropriate type to be connected to another block.

IV. Conclusions

In this paper, we have introduced our new environment, "BlockPy", that promotes Data Science through a block-based interface. We make a case that by relying on a more generally Useful context, rather than Interest, we can appeal to a wider range of mature learners. We describe a number of features we seek to support in our environment. Finally, we discussed the research that we are now exploring through this environment.

V. Acknowledgements

This material is based upon work supported by The National Science Foundation, grant TUES-1140318 and The National Science Foundation Graduate Research Fellowship, Grant No. DGE 0822220

References

[1] R. E. Anderson, M. D. Ernst, R. Ordóñez, P. Pham, and S. A. Wolfman, "Introductory programming meets the real world: using real problems and data in CS1," in *Proceedings of the 45th ACM technical symposium on Computer science education*. ACM, 2014, pp. 465–466.

[2] D. G. Sullivan, "A data-centric introduction to computer science for non-majors," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '13. New York, NY, USA: ACM, 2013, pp. 71–76.

[3] L. Layman, L. Williams, and K. Slaten, "Note to self: Make assignments meaningful," in *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '07. New York, NY, USA: ACM, 2007, pp. 459–463.

[4] M. Goldweber, J. Barr, T. Clear, R. Davoli, S. Mann, E. Patitsas, and S. Portnoff, "A framework for enhancing the social good in computing education: A values approach," *ACM Inroads*, vol. 4, no. 1, 2013.

[5] S. Mishra, S. Balan, S. Iyer, and S. Murthy, "Effect of a 2-week scratch intervention in CS1 on learners with varying prior knowledge," in *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, ser. ITiCSE '14. New York, NY, USA: ACM, 2014, pp. 45–50.

[6] B. D. Jones, "Motivating students to engage in learning: The MUSIC model of academic motivation," *International Journal of Teaching and Learning in Higher Education*, vol. 21, no. 2, pp. 272–285, 2009.

[7] 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*, ser. SIGCSE '15. New York, NY, USA: ACM, 2015, pp. 720–720.

[8] 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*, ser. SIGCSE '15. New York, NY, USA: ACM, 2015, pp. 185–190.