Data Science in Snap!: A Block-Based Approach to Data Science Education

Isaac Merritt

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2022-43
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-43.html

May 9, 2022

Data Science in Snap!: A Block-Based Approach to Data Science Education

Isaac Merritt

May 13th, 2022

As Data Science increases in popularity, demand is increasing for accessible educational Data Science content. Though it is common to address this demand with educational content in text-based programming languages like Python, we have identified a different approach that has proven successful in other fields of computing in the past. Course content known as BJC (The Beauty and Joy of Computing) offers students the experience of learning how to program in the block-based language Snap*!*. BJC content is taught all over the country at the high school level, so our goal was to take a similar approach with Data Science content.

By adapting Data Science educational content into Snap*!* in the form of DASIS (Data Science in Snap*!*) and BJDS (The Beauty and Joy of Data Science), we have created new opportunities for high-school-age Data Science students. DASIS is a new programming library in Snap*!* that provides basic and advanced Data Science tools to Snap*!* users, and BJDS is a series of online modules that covers introductory Data Science Principles using the DASIS library. We have made Data Science more accessible not only by adapting it to be taught in a high school setting, but also by making it publicly available on the Internet without the need for any additional software. This means that any high school teacher or student can access introductory Data Science material with only an Internet connection.

Introduction

Data Science as a discipline is in demand more than ever before [10]. As data begins to drive more and more important decisions in the world, having the ability to process and interpret data becomes a crucial asset. The interdisciplinary nature of Data Science's principles offers a student interested in almost any other field the opportunity to enhance their skill set. Data literacy is important for navigating difficult choices in life, where making data-driven decisions can avoid common misconceptions. This is why all high school students can benefit from a strong Data Science background.

It is therefore very important to ensure that those interested in learning about Data Science can do so without having to wait until they reach the university level of education. There is little to no existing standard infrastructure for Data Science content in high schools, so the opportunity to standardize learning objectives and create a beginner-friendly environment in which students can learn is integral to the development of the field as a whole. Making Data Science accessible to any and all interested students is an important step in making

data-driven problem solving a more common skill.

This report is organized into four main sections.

- **Section 1** discusses why DASIS is written in Snap*!*, and how it differs from existing libraries like UC Berkeley's `datascience` library and Python's `pandas`.
- **Section 2** details the development of DASIS from inception to design to implementation.
- **Section 3** outlines DASIS functionality, covering different groups of blocks and what Data Science principles they facilitate.
- **Section 4** details how BJDS builds upon existing BJC infrastructure and creates an iteration of block-based programming education centered around Data Science.

1 Block-Based Education

1.1 Snap!

Snap*!* is a block-based programming language that extends the implementation of MIT's Scratch [2]. University-level courses like UC Berkeley's CS10 [3] use Snap*!* as its primary programming language, and hundreds of high-school-level courses like BJC [2] use it as well. Its visual style makes it more intuitive for students with no coding background to learn programming principles without having to use text editors or IDEs. Snap*!*'s interface is publicly available online for anyone to use [11]; there are no downloads or installs required to run Snap*!* on any machine with an Internet connection.

This immediately helps make Snap*!* more accessible, as students do not have to worry about troubleshooting software installs or downloading potential malware from the Internet. Students can run Snap*!* just as easily as they can use Google.

1.2 Data Science in Text-based Languages

There are many text-based tools that can be used to teach Data Science principles. Some of these alternatives include Python libraries like Berkeley's `datascience` and `pandas`. The `datascience` library is used in UC Berkeley's introductory Data Science course, Data 8 [7], and `pandas` is used in the following Data Science course, Data 100 [6]. Both of these libraries have more advanced features than DASIS, but the additional functionality does not necessarily enhance a beginner's educational experience. DASIS provides students with all the functionality required to learn Data Science principles, whereas `datascience`

and `pandas` may be better suited for a transition towards more advanced topics in Data Science.

These tools also require Python to be installed on the student's machine and the libraries themselves must be downloaded from the Internet as well. This necessary step in the process can cause students to have to spend time troubleshooting software downloads and installation, which can discourage otherwise determined students who feel intimidated by the complications.

Even if download and installation do not present a problem to students, the increased functionality comes at the cost of interactive complexity. Unlike Snap*!*, Python typically entails using a text editor and the command line or Jupyter Notebooks. These are additional sources of potential troubleshooting for students, where Snap*!*'s drag, drop, and click-to-run model is streamlined and more intuitive for new programmers. All the tools available to a student learning how to use their Data Science toolkits are visibly present in the Snap*!* toolbar, whereas when using a language like Python, the student has to remember the names and arguments for every function or method they learn as they type.

Above all else, these Python tools require students to have existing Python literacy. They already need to know how Python works, because `datascience` and `pandas` are tools within the Python framework. Learning a text-based programming language can take a significantly longer amount of time and energy than learning a block-based language, so to get Data Science principles into students' hands faster, the opportunity to explore a block-based approach was a natural development step.

Overall, because students would need to download software from the Internet to use Python, install the libraries from the Internet, install a text editor or Jupyter Notebooks, and know how to use Python before being able to learn anything about Data Science, there is an opportunity to simplify the process and provide students with a more accessible introduction to Data Science. This is where Snap*!*'s block-based approach offers itself as a means to reduce the student's technological burden.

2 DASIS

As stated earlier in this report, there was strong motivation to create block-based Data Science content, but unfortunately there was no way to facilitate this material in Snap*!*. The clear choice was going to be writing a new library that would allow the same functionality as other Data Science tools in other languages. This is where the idea for DASIS was born.

DASIS is the **Da**ta **S**cience **i**n **S**nap*!* programming library. It contains blocks that provide students with the tools to learn Data Science principles in a block-based environment. We designed the library from scratch, as the existing Data Science blocks in Snap*!* were a bit disorganized, difficult to use, and provided only minimal functionality.

2.1 Design

The design process began in the summer of 2021. Initial designs reflected SQL-style querying options. The end goal was functionality similar to SQL, a known tool for teaching, such as in UC Berkeley's CS 186 [5]. Where DASIS was intended to be different was in all the ways it could take advantage of Snap*!*'s block-based, visual style. We wanted queries in DASIS to be composed of blocks that do individual operations, different from SQL syntax that does multiple operations at once, so that students can better understand how each operation works.

Ultimately, DASIS was designed to be a suite of blocks created for individual querying and visualization tasks, where the outputs of blocks could be used as inputs to others. Unlike SQL's rigid query structure, DASIS was designed to be more flexible with complex queries that require more than one block. Students would be able to use querying blocks in any valid order they choose, which would allow them to be able to interact with important ideas without having to spend as much time learning language-specific syntax typical of a text-based language. The data visualization blocks were also designed to remove complicated syntax from the visualization process and redirect focus on the importance of good data visualization practices.

2.2 Development

Once the design stage was complete and there was a concrete vision for the new library, development of DASIS began in the fall of 2021. Each DASIS function, method, and helper block was written from scratch using Snap*!*'s default blocks. We used Snap*!*'s JavaScript capabilites to do atomic sort and group operations on the finest levels of granularity, which significantly improved runtimes in the `ORDER`, `GROUP`, and `PIVOT` blocks. All of the computational Javascript code is on the backend, invisible to students and never used directly in BJDS content.

This means that by importing or including DASIS blocks in a Snap*!* file, DASIS blocks work with the default Snap*!* environment. This is just another way to improve the student's experience by providing them an educational programming environment with the least amount

of potential troubleshooting necessary.

One of the best ways DASIS takes advantage of Snap!'s visual style is its data visualization functionality. DASIS supports 4 central data visualization techniques:

- Horizontal Bar Graphs
- Line Plots
- Scatter Plots
- Histograms

These visualization techniques take advantage of Snap!'s stage. The stage is an area in the upper right corner of Snap!'s interface that can be used to draw, move sprites, and more. For our purposes, it was the perfect integrated feature to facilitate conversations about data visualization.

DASIS now includes over 40 interactive blocks for student use and over 70 blocks overall. Currently, to use DASIS there is an `xml` file that includes all DASIS blocks available on the BJDS website [9] for download. By clicking the link and dragging the `xml` file into Snap!'s interface, DASIS blocks will be available for use. However, we are currently looking to include DASIS as one of Snap!'s default downloadable libraries, further simplifying the technological needs of Data Science students.

DASIS blocks are used like any other blocks in Snap!. Users provide arguments to DASIS blocks and click the block to see the output. Reporter blocks (See Figure 1) will give a return value when clicked, and command blocks (See Figure 2) will not return anything, but will make a change to the stage or to a Table object.

Figure 1: Reporter block in Snap!

Figure 2: Command block in Snap!

3 DASIS Functionality

In this section, we will provide an overview of different DASIS blocks. All the blocks in this section, as mentioned eariler in this report, were made from scratch.

3.1 Table Object

DASIS's fundamental data structure is the Table. The figure on the right displays the shape indicating a Table Object being used as an argument to another DASIS block. In DASIS, a Table is represented by a 4-item Snap! list:

Figure 3: Argument shape for a Table Object in DASIS.

Table 1: The DASIS Table object

Table Item	Use
Data	• Data contained by this Table • Type: list of lists, each row is an item of the columns list • Accessed using the "table data of `Table`" method
Rows	• Number of rows in this Table • Type: integer • Accessed using the "number of rows in `Table`" method
Columns	• Number of columns in this Table • Type: integer • Accessed using the "number of columns in `Table`" method
Column Names	• Column names in this Table • Type: list of strings • Accessed using the "column names in `Table`" method

This Table data structure is involved in nearly every single DASIS block. Reporter blocks involving Tables **will never** modify the original Table, and will return a copy of the Table with the requested query modifications. Orange command blocks involving Tables **will** modify the Table object passed in as an argument, so they should be used with more caution. As a precaution, command blocks with reporter counterparts contain `permanent` in their titles so as to warn students that their effect is permanent and difficult to undo in Snap*!*.

There are two ways to make a new Table object. Using either raw textual data (See Figure 4) or imported `csv` data (See Figure 5), there are DASIS blocks that turn data into a Table object. It is recommended to always use imported `csv` data, however if students just want to make a quick, small Table to run tests on, there is an option to create your own Table from scratch by typing the table data into an argument of a DASIS block.

Figure 4: Create Table with textual data block

Figure 5: Create Table with imported data block

3.2 *Viewing a Table*

Because of the Table data structure's abstracted design, viewing a Table is not as trivial as clicking the variable name that stores a Table. Using one of two DASIS methods, we can view a Table in a visually straightforward fashion. These blocks, the `VIEW Table` block (See Figure 6) and the `head of Table` block (See Figure 7), allow a student to be able to see a Table's column names and data combined in one return value. Students could see this information using a combination of other DASIS blocks, but these blocks are present to simplify the process of viewing a Table as people expect them to look.

Figure 6: View Table block

Figure 7: head of Table block

3.3 Querying Blocks

Table 2: DASIS Table Methods

Block Image	Use
	• Accessor method that returns the data contained in a Table object • Does not include column labels
	• Accessor method that returns the number of rows in a Table object
	• Accessor method that returns the number of columns in a Table object
	• Accessor method that returns the list of column names in a Table object
	• Returns the nth column of a Table object as a list, where n is the integer argument
	• Returns the name of the nth column in a Table object, where n is the integer argument
	• Returns the column of data from a Table object corresponding to the column name argument as a list
	• Returns the index of the column in a Table object corresponding to the column name argument
	• Returns True if the column name argument is a column label in the given Table object, False otherwise
	• Returns the nth row of a Table object as a list, where n is the integer argument • Since rows do not have labels in Table objects, the only way to access a specific row is by its index
	• Returns the nth item of a given row, where n is the integer argument • There is no Table argument required to use this block because no label is necessary to access a row
	• Returns the nth item of the column in a Table object corresponding to the column name argument, where n is the integer argument
	• Returns the item in the ith row, and the jth column of a Table object, where i and j are the first and second integer arguments
	• Returns the item of a Table object in the given row and in the column corresponding to the column name argument

Table 3: Querying Reporter Blocks in DASIS

Block Image	Use
	• Returns a new Table with a new row added to the original Table • Does not edit the original Table
	• Returns a new Table with a new column added to the original Table; name provided by the column name argument • Does not edit the original Table
	• Returns a list that represents all the values in the given column argument modified with the reporter argument
	• Returns a Table without the columns provided as drop arguments • Best used when only a few columns are to be omitted
	• Returns a Table with one row for each unique value in the grouping column argument • Each row is aggregated via argument using sum, mean, or count • The column names of the returned Table are the same as the original Table, but all rows except the grouping row have been appended with the name of the aggregator function
	• Returns a Table with a column for every unique value in the first pivoting column argument and a row for every unique value in the second pivoting column argument • Each value in the returned Table is the number of rows in the original with the combination of values at the intersection of its location
	• Returns a Table with only the top n rows, where n is the integer argument • Can be helpful for viewing a large Table
	• Orders Table data by the provided column argument • Ordering column can optionally be modified before ordering • Can order in ascending or descending order
	• Returns a Table with only the columns selected by the user • Best used when only a few columns are to be kept
	• Returns a Table with only rows with a value in the given column that satisfies the given boolean function • Boolean function is created by the boolean operator argument, comparing the row's value in that column to the value argument • Filtering column can optionally be modified before filtering

The querying blocks can be organized into 4 groups: Table methods, reporter blocks, command blocks, and data visualization blocks. Tables displaying all the blocks in these categories with their usages appear in the tables above and below.

Table 2 contains all blocks that give information about a Table without any modifications. These blocks help students learn about a data set before answering any questions or coming to conclusions about the data.

Table 3 contains all querying reporters. These blocks modify a Table in some way and return a modified copy to satisfy the query. None of these blocks edit the original Table object passed in as their arguments.

Once students have learned about how to investigate a data set without modifications using the blocks mentioned in the previous tables, they can use the following blocks to calculate answers to potential questions about the data.

Table 4: Querying Command Blocks in DASIS

Block Image	Use
	• Adds a column to the Table permanently • Column is added to the end (right) of the Table
	• Adds a row to the Table permanently • Row is added to the end (bottom) of the Table
	• Applies the reporter function argument to the items of the column corresponding to the first column name argument and adds the result as a new column in the Table with the name provided by the second column name argument
	• Deletes the column corresponding to the column name argument from a Table
	• Deletes all columns corresponding to each column name in the list argument from a Table
	• Deletes the column at the index provided as the integer argument from a Table
	• Deletes the row at the index provided as the integer argument from a Table
	• Relabels an existing column name, provided as the first column name argument, and changes it to the name provided as the second column name argument in a Table

Table 4 contains query command blocks. These blocks execute functions homologous to their reporter counterparts, except that they edit a Table directly and permanently. These blocks all have functionality represented by another DASIS reporter, but the calculations or changes they make do not happen to a copy of the original Table; their changes are difficult to undo and should be used with caution.

Table 5: Data Visualization Blocks in DASIS

Block Image	Use
horizontal bar graph for Table ➤ on column ( ) (optionally) grouped by column ( ) with title ( )	• Generates a horizontal bar plot of a categorical variable • Can optionally group data before plotting
histogram of Table ➤ , grouped by column ( ) into -1 bins , with title ( ) x-axis label ( ) and y-axis label ( )	• Generates a histogram of a numerical variable • Can choose number of bins, or let DASIS choose by default
line plot for Table ➤ with time column ( ) (optionally) grouped by column ( ) with title ( )	• Generates a line plot of a numerical variable over time • Can optionally group data before plotting
scatterplot for Table ➤ on columns ( ) and ( ) (optionally) grouped on column ( ) with title ( )	• Generates a scatter plot of two numerical variables • Can optionally group data before plotting

Table 5 contains data visualization blocks. These blocks execute functions that visually represent data on the Snap! stage. They have a purple color to indicate that they are not querying blocks and that they draw on the stage.

These data visualization blocks create visual plots in Snap!, examples of which can be found in the margins as Figures 8-11. They each run very quickly on large data sets; they have been tested on data sets on the order of 7000 rows.

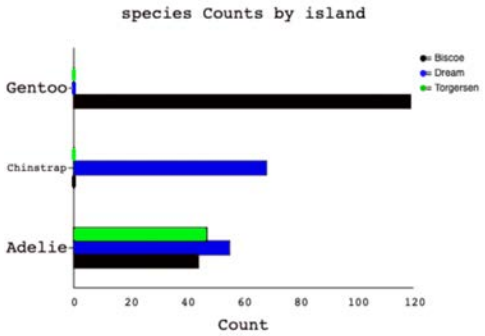

Figure 8: A DASIS Horiontal Bar Plot

4 BJC and BJDS

The Beauty and Joy of Data Science (BJDS) is an introductory Data Science curriculum in Snap!. BJDS uses DASIS to facilitate its learning objectives, and it currently consists of 4 interactive modules available online [8]. These modules are hosted on the same website as BJC (The Beauty and Joy of Computing), and they can be seen as an extension of BJC-style material.

It is not a requirement that students have previous experience with Snap*!* in order to engage with BJDS material, but a background with Snap*!* helps students establish comfort with Data Science principles because they have established comfort in the programming language first. For this reason, students who have already taken BJC at their high school are in a great position to learn with BJDS.

4.1 UC Berkeley's CS10

CS10 is the university-level implementation of BJC content. Each semester, between 150 and 350 [1] new students at UC Berkeley take this course, many of whom have little to no previous programming experience before enrolling in the course. As a former CS10 student and a former CS10 staff member (Academic Intern, Reader, and eventually TA), I can speak from experience that the opportunity to learn programming for the very first time at the university level can feel like a daunting task. However, Snap*!* does not feel as intimidating to students as they learn how to code, and the course's structure allows students to follow a natural progression through various topics in computing.

CS10 spends a little over $\frac{2}{3}$ of its curriculum teaching exclusively in Snap*!*, and spends the rest of the semester introducing students to Python [3]. There is a very natural transition between the two languages, where students are able to make comparisons between their functionalities. Snap*!* has many functions that students can find in Python, so when making the switch from a block-based language to a text-based language, there is already an existing knowledge base that supports students. The task of learning a new programming language comes down to understanding its syntax and translating knowledge from your known language to the new one. This takes a lot of the pressure off students as they venture into Python.

Additionally, CS10 provides coding opportunities for students of generally under-represented communities in the field of programming (women and ethnic minorities) [4]. At UC Berkeley, female enrollment routinely surpasses 50% of the class, and this fact has generated national exposure from several major news outlets including the New York Times, KQED, NPR's All Things Considered, USA Today, San Jose Mercury News, San Francisco Chronicle, and many others [4].

All these qualities are desirable starting points for any computing curriculum. This is why the style of BJDS was so heavily influenced by BJC and CS10. Creating a Data Science addition to existing infrastructure like this was a high priority for us, and the BJDS that exists today reflects that desire.

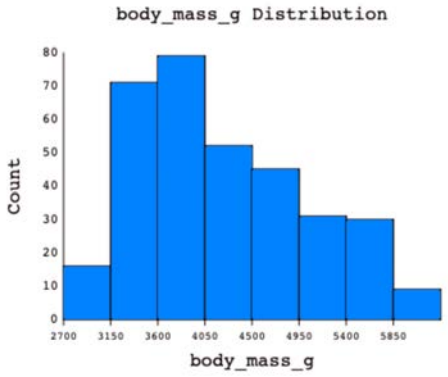

Figure 9: A DASIS Histogram

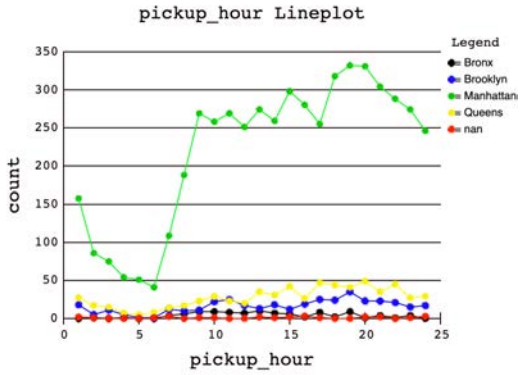

Figure 10: A DASIS Line Plot

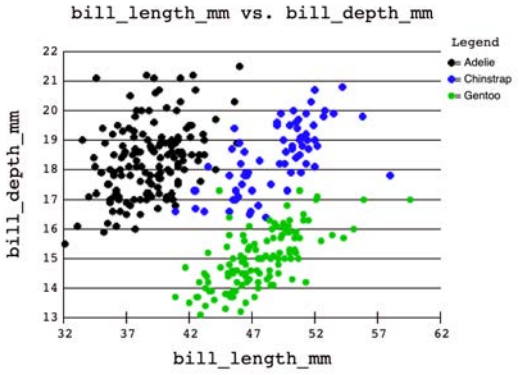

Figure 11: A DASIS Scatter Plot

4.2 BJDS Learning Objectives

As mentioned earlier in this report, BJDS is a Data Science curriculum implemented with online modules. The learning objectives of the curriculum aim to provide students with introductory Data Science skills that can be built upon by future Data Science education. There are three main goals that students are encouraged to remember as they make their way through the curriculum:

1. Explore Data Science using Snap*!*
2. Explore Tables and how we can use Snap*!* to manipulate them
3. Understand how Tables can be used to answer questions about data

The technical skills and activities they will engage with during their time with BJDS all contribute to these goals. While fluency with Snap*!* is important, that knowledge is only a stepping stone towards better understanding Data Science principles.

4.3 BJDS Module Content

Table 6: BJDS Modules

Module	**Covered Content/Activities**
Assignment 1: Tables and Data	• Familiarizing students with DASIS • Creating a Table from scratch using data that is not imported • Accessing data from a Table • Accessing information about a Table • Importing a Table from a file • Exploring introductory calculation using Table data
Assignment 2: Querying a Table	• Exploring new Table methods in DASIS • Using those Table methods to answer more complex questions about data
Assignment 3: Group and Pivot	• Explore the Group and Pivot table methods in Snap*!* • Use those Table methods to answer more complicated questions about data • Understand how Group and Pivot allow us to more easily visualize data
Assignment 4: Data Visualization	• Learn about statistical variable types: numerical (continuous/discrete) and categorical (ordinal/nominal) • Learn how to properly visualize variables using DASIS blocks • Detect patterns and draw conclusions about data using visualization

BJDS is a series of 4 modules that develop students' technical programming knowledge and provide the skills required to explore

the field of Data Science. These 4 modules build off each other, as the skills learned in a previous module are necessary for students in the next module.

4.4 Next Steps for BJDS Students

By the time students have completed these 4 modules, they will be ready to take their next steps in their Data Science education. In most cases, that will mean a transition to a text-based language, similar to the way BJC and CS10 transition to Python. There are many languages students will be able to learn after completing BJDS, like Python's `datascience` and `pandas`, R, or SQL, but no matter which path they take with their Data Science future, BJDS will have prepared them with the skills to properly analyze, query, and draw meaningful conclusions about data.

Students at this point are free to explore new languages on their own or even continue on to university-level Data Science. A course like Data 8 at UC Berkeley would be an example of a Python transition, but many other universities also offer students opportunities to grow their skills and interest in Data Science.

5 Conclusion

Overall, DASIS and BJDS aim to help expose younger students to the field of Data Science in a beginner-friendly programming environment. By making them both accessible to the public, we open the door to any high school teachers in America being able to teach our material to their students, and we also allow any interested students to engage with the material on their own.

5.1 Next Steps

In the future, we would be happy to see BJDS expand. We would be excited to see BJDS in more high schools in the coming years, as right now the curriculum is being piloted in a Bay Area high school by teacher Sean Morris and will be taught as a part of a new Data Science class in the Fall of 2022 by teacher Parisa Safa. Once it has been taught in a live environment, we hope to see it spread to other high schools where more and more high school students will be able to learn about Data Science. The more students that are able to interact with BJDS and explore Data Science, the better.

We can also see a next step where another group builds on the content we have created to improve the quality of the curriculum. This could entail creating new activities with relevant data sets or creating projects for students to work on and test their new skills.

Overall, we want DASIS and BJDS to facilitate the growth of Data Science as a field to motivate the next generation to make data-driven conclusions. The best way to do that is ensure that interested students can begin learning at a younger age and develop their skills earlier than college.

Acknowledgements

This project would not have been possible without the help of many others who supported me and guided me during my time at Berkeley. I would like to thank:

- Josh Hug, my advisor, without whom I would not have been able to conduct my research or been able to learn so much about the educational process.
- John Denero, my second reader, who graciously agreed to help me revise this report and provide me valuable feedback.
- Dan Garcia, my first computer science professor and the educator whose enthusiasm and passion for the field of computer science showed me that anyone can code (even those who don't start learning until college!).
- Suraj Rampure, Niki Zarkub, Maxson Yang, Murtaza Ali, Kathleen Gao, all my fellow CS10 staff members, and countless others who I have had the pleasure of teaching alongside during my time at Berkeley.
- My friends and family whose support and compassion motivated me to be the best version of myself as a student at UC Berkeley.

References

[1] *BerkeleyTime | CS10 Enrollment Statistics*. URL: https://berkeleytime.com/enrollment/2-2315-spring-2021-332192&3-2315-fall-2020-325182 (visited on 04/15/2022).

[2] *BJC | The Beauty and Joy of Computing*. URL: https://bjc.berkeley.edu/ (visited on 04/12/2022).

[3] *CS10 | Course Webpage*. URL: https://cs10.org (visited on 12/04/2022).

[4] *CS10 Spring 2022*. URL: https://cs10.org/sp20/.

[5] *CS186 | Course Webpage*. URL: https://cs186berkeley.net/ (visited on 05/02/2022).

[6] *Data 100 | Course Webpage*. URL: https://ds100.org/ (visited on 12/04/2022).

[7] *Data 8 | Course Webpage*. URL: http://data8.org/ (visited on 12/04/2022).

[8] *Data Science in Snap! | BJDS Home*. URL: https://cs10.org/bjc-r/course/berkeley_bjds.html (visited on 04/14/2022).

[9] *Data Science in Snap! | Download DASIS*. URL: https://cs10.org/bjc-r/cur/programming/data_science/lab01/03-download.html?topic=berkeley_bjds%2Ftables%2F1-tables.topic&course=berkeley_bjds.html&novideo&noreading&noassignment (visited on 04/13/2022).

[10] *Salaries And Job Opportunities For Data Scientists Continue To Rise*. URL: https://www.forbes.com/sites/gilpress/2021/06/27/salaries-and-job-opportunities-for-data-scientists-continue-to-rise/?sh=675eb0104276 (visited on 04/12/2022).

[11] *Snap! User Interface*. URL: https://snap.berkeley.edu/snap/snap.html (visited on 12/04/2022).