Starting to Program

Students begin to program in Pyret, learning about basic datatypes, operations, and value definitions.

Prerequisites	None
Relevant Standards CSTA	*Select one or more standards from the menu on the left (⌘-click on Mac, Ctrl-click elsewhere).*
Lesson Goals	Students will be able to… • Explain the difference between several data types: Numbers, Strings, Images and Booleans • Identify a data type for a given value • Write Numbers, Strings, and Booleans in the Interactions Area • Define values, and evaluate simple expressions that use defined values
Student-facing Lesson Goals	• Let's explore programming in Pyret and learn about data types.
Materials	• Lesson Slides (Google Slides) • Computer for each student (or pair), with access to the internet • Student workbook, and something to write with
Preparation	• Make sure all materials have been gathered • Decide how students will be grouped in pairs • Make sure student computers can access the Pyret IDE (CPO) • All students will need acces to code.pyret.org, also known as "CPO". They should be able to log in using a Google Classroom, Google, or YouTube login.
Supplemental Resources	
Language Table	Students are not expected to have any familiarity with the Pyret programming for this lesson.

Glossary

data row :: a structured piece of data in a dataset that typically reports all the information gathered about a given individual

definitions area :: the left-most text box in the Editor where definitions for values and functions are written

editor :: software in which you can write and evaluate code

header :: the titles of each column of a table, usually shown at the top

identifier column :: a column of unique values which identify all the individual rows (e.g. - student IDs, SSNs, etc)

interactions area :: the right-most text box in the Editor, where expressions are entered to evaluate

---

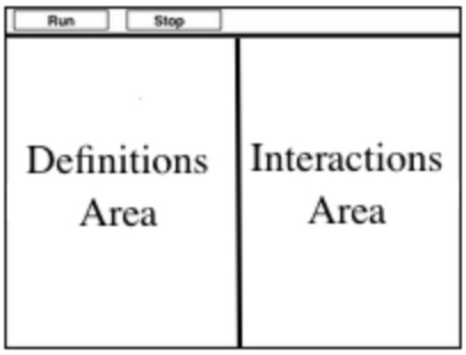
Introducing Pyret 10 minutes

Overview

Students open up the Pyret environment (code.pyret.org, or "CPO") and see how tables look in Pyret.

Students open up the Pyret environment (code.pyret.org, or "CPO") and see how tables look in Pyret.

Launch



Open up the Animals Starter File in a new tab. Click "Connect to Google Drive" to sign into your Google account. This will allow you to save Pyret files into your Google Drive. Next, click the "File" menu and select "Save a Copy". This will save a copy of the file into your own account, so that you can make changes and retrieve them later.

This screen is called the ***Editor***, and it looks something like the diagram you see here. There are a few buttons at the top, but most of the screen is taken up by two large boxes: the ***Definitions Area*** on the left and the ***Interactions Area*** on the right.

The ***Definitions Area*** is where programmers define values and functions that they want to keep, while the ***Interactions Area*** allows them to experiment with those values and functions. This is like writing function definitions on a blackboard, and having students use those functions to compute answers on scrap paper.

For now, we will only be writing programs in the Interactions Area.

The first few lines in the Definitions Area tell Pyret to `import` files from elsewhere, which contain tools we'll want to use for this course. We're importing a file called Bootstrap:Data Science, as well as files for working with Google Sheets, tables, and images:

```
include shared-gdrive("Bootstrap-DataScience-...")
include gdrive-sheets
include tables
include image
```

After that, we see a line of code that *defines* `shelter-sheet` to be a spreadsheet. This table is loaded from Google Drive, so now Pyret can see the same spreadsheet you do. (Notice the funny scramble of letters and numbers in that line of code? If you open up the Google Sheet, you'll find that same scramble in the address bar! That scramble is how the Pyret editor knows which spreadsheet to load.) After that, we see the following code:

```
# load the 'pets' sheet as a table called animals-table
animals-table = load-table: name, species, age, fixed, legs
  source: pets-sheet.sheet-by-name("pets", true)
end
```

The first line (starting with `#`) is called a *Comment* . Comments are notes for humans, which the computer ignores. The next line defines a new table called `animals-table` , which is loaded from the `shelter-sheet` defined above. We also create names for the columns: `name` , `species` , `sex` , `age` , `fixed` , `legs` , `pounds` and `weeks` . We could use any names we want for these columns, but it's always a good idea to pick names that make sense!

Even if your spreadsheet already has column headers, Pyret requires that you name them in the program itself.

Click "Run", and type `animals-table` into the Interactions Area to see what the table looks like in Pyret. Is it the same table you saw in Google Sheets? What is the same? What is different?

In Data Science, every table is composed of cells, which are arranged in a grid of rows and columns. Most of the cells contain data, but *the first row and first column* are special. The first row is called the ***header row***, which gives a unique name to each variable (or "column") in the table. The first column in the table is the ***identifier column***, which contains a unique ID for each row. Often, this will be the name of each individual in the table, or sometimes just an ID number.

Below is an example of a table with one header row and two data rows:

name	**species**	**sex**	**age**	**fixed**	**legs**	**pounds**	**weeks**
"Sasha"	"cat"	"female"	1	false	4	6.5	3
"Mittens"	"cat"	"female"	2	true	4	7.4	1

Investigate

- How many variables are listed in the header row for the Animals Dataset? What are they called? What is being used for the identifier column in this dataset?
- Try changing the name of one of the columns, and click "Run". What happens when you print out the table back in the Interactions Area?
- What happens if you remove a column from the list? Or add an extra one?

After the header, Pyret tables can have any number of *data rows*. Each data row has values for every column variable (nothing can be left empty!). A table can have any number of data rows, including *zero*, as in the table below:

`name`	`species`	`sex`	`age`	`fixed`	`legs`	`pounds`	`weeks`

Numbers, Strings and Booleans

25 minutes

Overview

This lesson starts them programming, showing students how to make Pyret do simple math, work with text, and create simple computer graphics. It also draws attention to error messages, which are helpful when diagnosing mistakes.

Launch

Pyret lets us use many different kinds of data. In the animals table, for example, there are Numbers (the number of legs each animal has), Strings (the species of the animal), and Booleans (whether it is true or false that an animal is fixed). Pyret has the usual arithmetic operators: addition (`+`), subtraction (`-`), multiplication (`*`), and division (`/`).
To identify if an animal is male, we need to know if the value in the `sex` column is *equal* to the string `"male"` . To sort the table by age, we need to know if one animal's age is *less than* another's and should come before it. To filter the table to show only young animals, we might want to know if an animal's age is *less than* 2. Pyret has Boolean operators, too: equals (==), less-than (<), greater-than (>), as well as greater-than-or-equal (>=) and less-than-or-equal (<=).

Investigate

In pairs, students complete Numbers and Strings (Page 7).

Discuss what students have learned about Pyret:

- Numbers and Strings evaluate to themselves.
- Anything in quotes is a String, even something like `"42"` .
- Strings *must* have quotation marks on both sides.
- Operators like `+` , `-` , `*` , and `/` need spaces around them.
- Any time there is more than one operator being used, Pyret requires that you use parentheses.
- Types matter! We can add two Numbers or two Strings to one another, but we can't add the Number `4` to the String `"hello"` .

Error messages are a way for Pyret to explain what went wrong, and are a really helpful way of finding mistakes. Emphasize how useful they can be, and why students should read those messages out loud before asking for help. Have students see the following errors:

- `6 / 0` . In this case, Pyret obeys the same rules as humans, and gives an error.
- A`(2 + 2`. An unclosed quotation mark is a problem, and so is an unmatched parentheses.

In pairs, students complete Booleans (Page 8).

Synthesize

Debrief student answers as a class.

Going Deeper

By using the `and` and `or` operators, we can *combine* boolean tests, as in: `(1 > 2) and ("a" == "b")`. This is handy for more complex programs! For example, we might want to ask if a character in a video game has run out of health points *and* if they have any more lives. We might want to know if someone's ZIP Code puts them in Texas or New Mexico. When you go out to eat at a restaurant, you might ask what items on the menu have meat and cheese. We'll use these Boolean operators in a lot of our Data Science work later on. See "Additional Exercises" if you'd like to have students get some practice with `and` and `or`.

Defining Values

20 minutes

Overview

Students learn how to define values in Pyret (note that these definitions work the way variable substitution does in math, as opposed to variable assignment you may have seen in other programming languages).

Launch

Pyret allows us to define names for values using the `=` sign. In math, you're probably used to seeing definitions like $x = 4$, which defines the name x to be the value 4. Pyret works the same way, and you've already seen two names defined in this file: `shelter-sheet` and `animals-table`. We generally write definitions on the left, in the Definitions Area. You can add your own definitions, for example:

```
my-name = "Maya"
sum = 2 + 2
kittens-are-cute = true
```

With your partner, take turns adding definitions to this file:

- Define a value with name `food`, whose value is a String representing your favorite food
- Define a value with name `year`, whose value is a Number representing the current year
- Define a value with name `likes-cats`, whose value is a Boolean that is `true` if you like cats and `false` if you don't

Synthesize

TODO

Additional Exercises:

- Boolean Operators