Working with Calvin: Your Online Lab Assistant

To help you complete the three acid-base experiments, the department has hired Calvin, your new online lab assistant. Your group will submit experiments to Calvin in a text file by e-mail. He will perform the experiments and upload his results to your group’s folder in Box. Unfortunately, Calvin knows very little chemistry, so you have to tell him exactly what you want him to do.

Getting ready

Since Calvin only understands text files,[[1]](#footnote-1) we highly suggest you write your commands to him in a text editor designed for programming, not a word processor. Please go to the link on Canvas or turn to the last page of this document for instructions on downloading and setting up a free text editor that works on your computer.

When writing to Calvin, add a .calvin extension to your filename to enable syntax coloring as shown below. The exact colors you see will depend on your text editor and how you set your preferences.In the example shown below, commands that Calvin understands are in blue, prompts are in brown, and literal names, which must be enclosed in quotes, are in red. (Literal names are often called “strings” in programming.) Notice that Fill\_Empty\_100ml\_Beaker\_with\_Water is black, which implies that Calvin does not understand this command. The command should read Fill\_Empty\_100ml\_Beaker\_with\_H2O. Comments, which Calvin ignores, always start with two slashes “//” and are colored in green below.



Naming Things for Calvin

To help Calvin keep track of your experiment, you need to name all of your glassware. In the commands above, the beakers have names beaker\_base, beaker\_acid, and so forth. These names must be surrounded by quotes. If you forget the quotes, the text editor will turn the name black (or whichever color your editor uses) to indicate that Calvin will not understand the name.

Calvin only accepts names that start with a letter and contain letters, numbers, or the underscore. Calvin *does not* allow spaces in names. Calvin thinks capitalization is irrelevant, but it is good practice to improve the readability of your experiments by using namesInCamelCase or names\_like\_this.

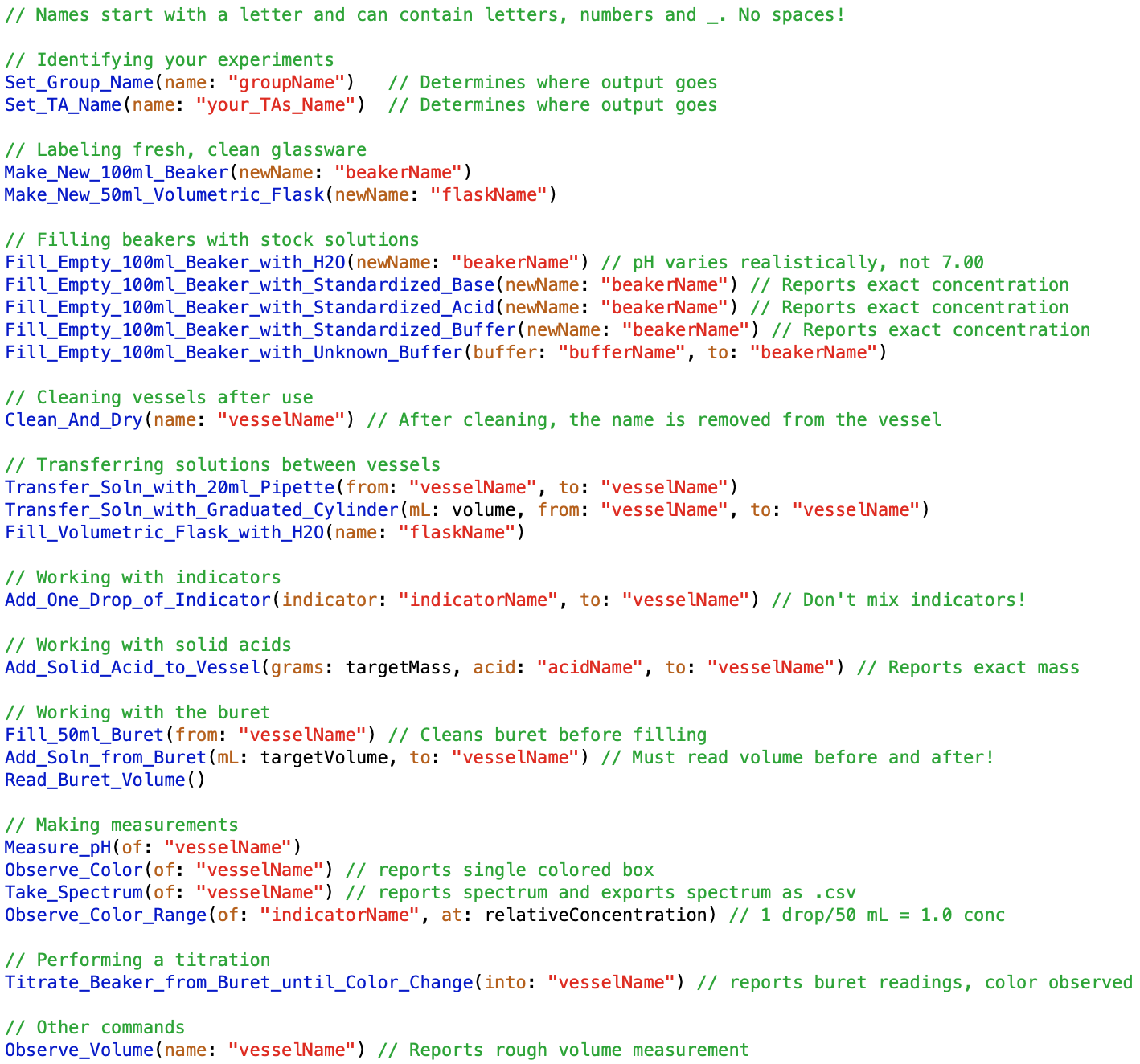
**Valid names:** myBeaker, Flask\_1, The\_splendiferous\_buffer

**Invalid names:** 1, 3\_Beaker, Beaker\_15.2, !#@&\*

Writing Experiments for Calvin

The easiest way to design experiments for Calvin is to have two files open at once: your experiment file and the file of Calvin command templates, Command\_File.calvin. This will allow you to simply copy individual commands and fill them out with your instructions.

The current list of commands that Calvin understands are:



Communicating with Calvin

First and most importantly every experiment file that you submit to Calvin must set your group name and your TA name using the commands Set\_Group\_Name and Set\_TA\_Name. I suggest making these the first two commands in every experiment. These names will be given to you by your TA. Second, give your experiment file an identifiable name, such as Experiment\_1.calvin.

**Important:** Do not submit any experiments to Calvin until your TA gives you a group name and grants your group access on Box. You will not be able to access your output until then.

When you are ready to submit your file to Calvin, e-mail your file as an attachment to:

Chem\_20.rpumm9z0d7ba31yz@u.box.com

The subject of the e-mail and any text in the body will be ignored. Only the attachment will be uploaded. You should receive an acknowledgment from Box within a few minutes of your submission.

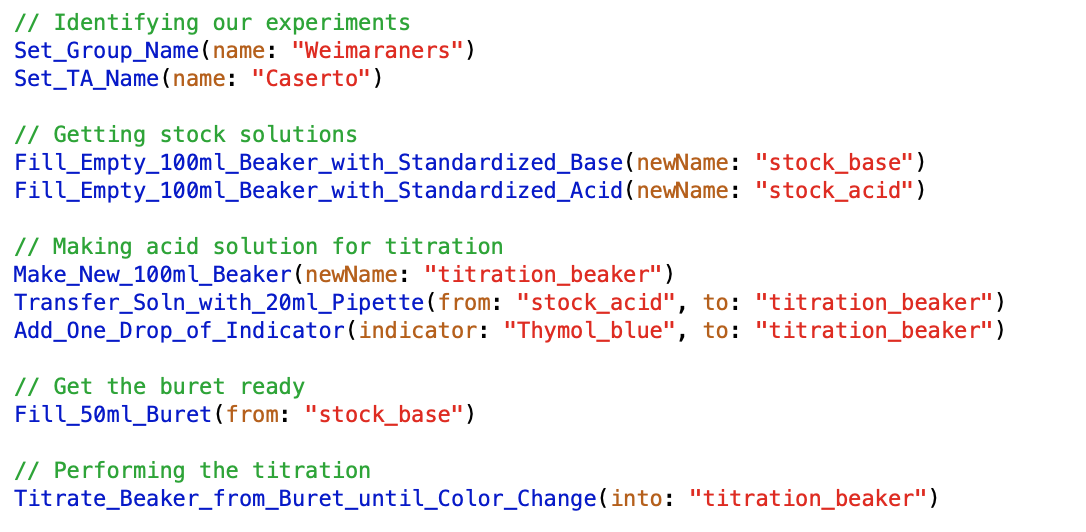
Once Calvin completes your experiment, he will put his notebook and any generated spectra in a folder. He will compress (*i.e.,* zip) the folder and upload it to your group’s folder on Box. For example, a typical output from Calvin would be named

Beagles\_Testing Buffers\_03-24\_195948.zip

This is the output generated for the Beagles group from their Testing Buffers.calvin experiment which was performed on March 24 at 19:59:48. You can copy this file from Box to your own computer for analysis. On most computers, double clicking a zip file will decompress it.

A Sample Experiment

The following is a simple experiment where Calvin titrates the stock acid solution with the stock base solution to make sure everything is working. Read through the experiment to get a feeling for Calvin operations.



Reading Calvin’s Experiment Log

After Calvin performs your experiment, he will save his log (*i.e.,* notebook) as a rtf (rich text format) file in your group’s folder on Box. **Do not try to read this file in a text editor, as you will not see Calvin’s graphics.** Open the file in a word processor.

For example, the complete output from the sample experiment above is shown on the next page.

**Calvin command file: Simple Titration.calvin**

**Output file: Weimaraners\_Simple Titration\_03-25\_223827**

// Identifying our experiments

Set\_Group\_Name(name: "Weimaraners")

Set\_TA\_Name(name: "Caserto")

// Getting stock solutions

Fill\_Empty\_100ml\_Beaker\_with\_Standardized\_Base(newName: "stock\_base")

A clean 100 ml beaker was named stock\_base.

The beaker stock\_base was filled with 100 ml of 0.1030 M standardized NaOH.

Fill\_Empty\_100ml\_Beaker\_with\_Standardized\_Acid(newName: "stock\_acid")

A clean 100 ml beaker was named stock\_acid.

The beaker stock\_acid was filled with 100 ml of 0.1003 M standardized HCl.

// Making acid solution for titration

Make\_New\_100ml\_Beaker(newName: "titration\_beaker")

A clean 100 ml beaker was named titration\_beaker.

Transfer\_Soln\_with\_20ml\_Pipette(from: "stock\_acid", to: "titration\_beaker")

Pipetted 20 ml of solution from stock\_acid to titration\_beaker.

Add\_One\_Drop\_of\_Indicator(indicator: "Thymol\_blue", to: "titration\_beaker")

Added one drop of Thymol\_blue to titration\_beaker.

// Get the buret ready

Fill\_50ml\_Buret(from: "stock\_base")

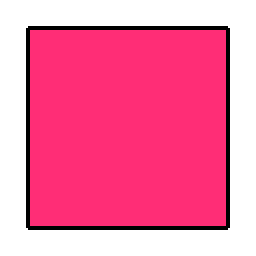
Added solution from stock\_base to your 50 ml buret.

The volume of solution in the buret is now 50.00 ml.

// Performing the titration

Titrate\_Beaker\_from\_Buret\_until\_Color\_Change(into: "titration\_beaker")

The color of the solution in titration\_beaker is:

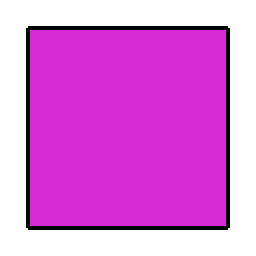


The volume of solution in the buret is now 50.00 ml.

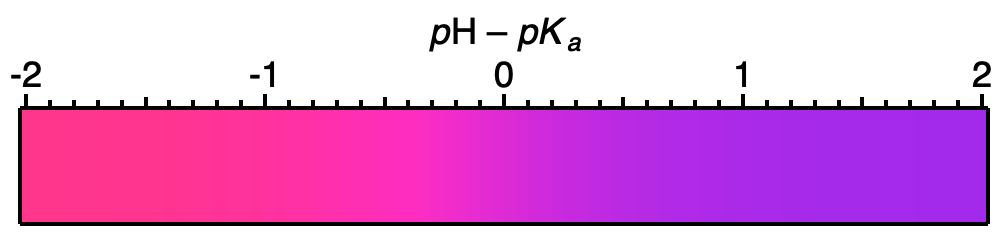
Performing titration now.

The volume of solution in the buret is now 30.50 ml.

The color of the solution in titration\_beaker is:



The color range of the indicator thymol\_blue is:



Good Programming Practice.

Calvin is the ultimate honey badger.[[2]](#footnote-2) He don’t care how you write your experiment or the names you choose.

You, on the other hand, are going to need to understand your experimental output, perhaps days or weeks after you write your experiment file. For this reason, we strongly suggest that you liberally comment your experiment, describing what you are trying to do. If you put a comment on its own line (*i.e.,* starting the line with //), Calvin will copy your comment in green into his log.

How Calvin Handles Errors

Calvin will try his best to follow your instructions, but he will write **red errors** in his log if your instructions lead to problems in the lab.

Calvin is a trooper. He will continue to perform your commands no matter how many errors are generated. This can be problematic if an error leads to a piece of glassware being cleaned and put away, as illustrated by the output from the unhappy experiment below. Notice that overfilling myFlask caused Calvin to clean and dry the spilled flask, which erased the name of the flask. As a result, myFlask did not exist when Calvin tried to measure its volume in the next step.

// Get a beaker of stock base solution

Fill\_Empty\_100ml\_Beaker\_with\_Standardized\_Base(newName: "base\_beaker")

A clean 100 ml beaker was named base\_beaker.

The beaker base\_beaker was filled with 100 ml of 0.1001 M standardized NaOH.

// Giving two beakers the same name is bad

Make\_New\_50ml\_Volumetric\_Flask(newName: "myFlask")

A clean 50 ml volumetric flask was named myFlask.

Make\_New\_50ml\_Volumetric\_Flask(newName: "myFlask")

I'm sorry. A vessel by the name of myFlask already exists.

// Adding a negative mass is bad

Add\_Solid\_Acid\_to\_Vessel(grams: -1.2, acid: "KHP", to: "myFlask")

You cannot transfer a negative mass!

// Overfilling a vessel makes a mess

Transfer\_Soln\_with\_Graduated\_Cylinder(mL: 75, from: "base\_beaker", to: "myFlask")

Used a 25 ml graduated cylinder to transfer 25 ml of solution from base\_beaker to myFlask.

Used a 25 ml graduated cylinder to transfer 25 ml of solution from base\_beaker to myFlask.

Oh, no! You put too much solution in myFlask. What a mess! Clean and dry myFlask.

// Check the volume of the overfilled flask

Observe\_Volume(name: "myFlask")

I'm sorry. The vessel myFlask does not exist.

There is one type of error that you may not expect. Calvin will not allow you to name a beaker or flask after common programming terms (*e.g.,* print, exists, variable). This will generate a regular error as well as a “Bad Command Ignored” error as illustrated by the following:

The name exists conflicts with Calvin. Please use a different name.

Bad Command Ignored: Fill\_Empty\_100ml\_Beaker\_with\_Standardized\_Base(newName: "exists")

Working with Colors and Spectra

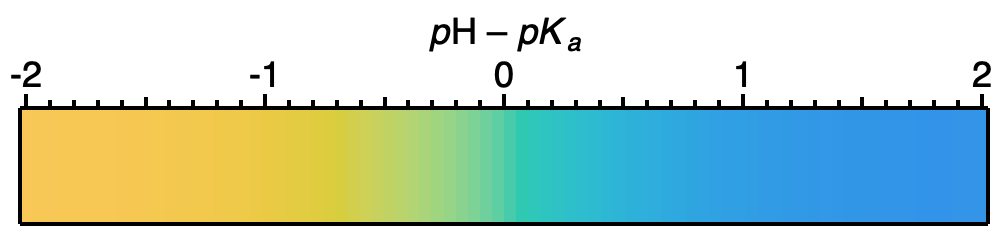
If your solution contains an indicator dye, Calvin can tell you what color your solution is, and he can acquire the solution’s visible spectrum. In doing this, the concentration of the indicator should be approximately 1 drop of dye per 50 ml, although this varies somewhat by dye. If the solution is too concentrated, the colors will be too saturated, and the spectrum may be clipped. Conversely, if the solution is too dilute, the colors will be faint and the spectrum noisy.

You can see the effect of concentration most easily using the Observe\_Color\_Range command as shown by the following experiment which examines bromocresol green at three concentrations: just right, too dilute, and too concentrated.

// Examine the color range at the optimum concentration of 1 drop/50 ml

Observe\_Color\_Range(of: "bromocresol\_green", at: 1.0)

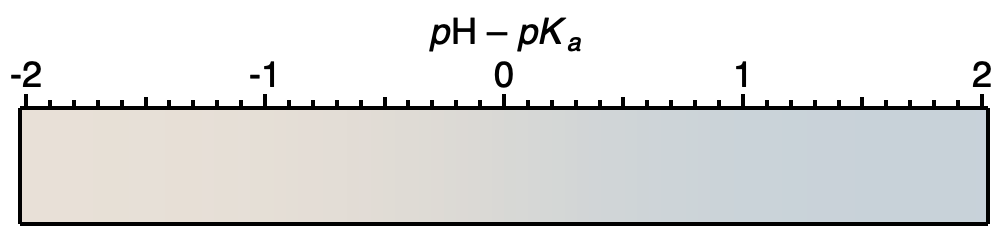
The color range of the indicator bromocresol\_green is:



// Examine the color range at 1/10 of optimum concentration

Observe\_Color\_Range(of: "bromocresol\_green", at: 0.1)

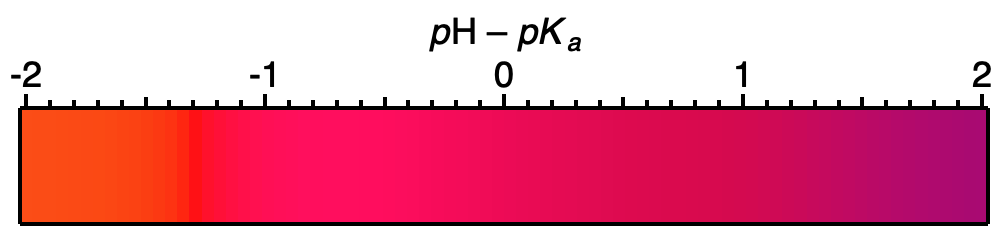
The color range of the indicator bromocresol\_green is:



// Examine the color range at 10 time optimum concentration

Observe\_Color\_Range(of: "bromocresol\_green", at: 10.0)

The color range of the indicator bromocresol\_green is:



Why does the solution look red at high concentration? Simple! The solution is essentially black at all other wavelengths, so the tiny amount of light that is transmitted is red.

Why does the solution look gray at low concentration? Equally simple. All colors are being transmitted (none absorbed), so the solution appears to be a neutral gray.

The output file on the next page illustrates the same problem, only this time with visible spectra instead of colors.

Fill\_Empty\_100ml\_Beaker\_with\_Standardized\_Base(newName: "stock\_base")

A clean 100 ml beaker was named stock\_base.

The beaker stock\_base was filled with 100 ml of 0.0976 M standardized NaOH.

// Make a solution at the optimum concentration

Make\_New\_100ml\_Beaker(newName: "optimum\_concentration")

A clean 100 ml beaker was named optimum\_concentration.

Transfer\_Soln\_with\_Graduated\_Cylinder(mL: 50.0, from: "stock\_base", to: "optimum\_concentration")

Used a 25 ml graduated cylinder to transfer 25 ml of solution from stock\_base to optimum\_concentration.

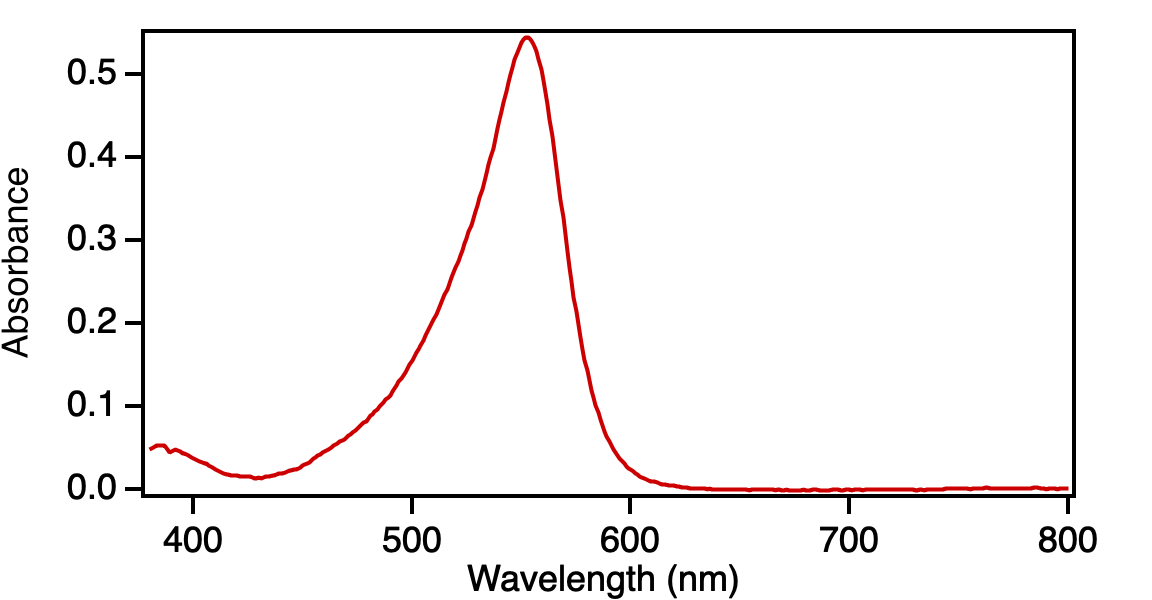
Used a 25 ml graduated cylinder to transfer 25 ml of solution from stock\_base to optimum\_concentration.

Add\_One\_Drop\_of\_Indicator(indicator: "Phenolphthalein", to: "optimum\_concentration")

Added one drop of Phenolphthalein to optimum\_concentration.

Take\_Spectrum(of: "optimum\_concentration")

The spectrum of the solution in optimum\_concentration is:



This spectrum has been saved as Spectrum\_1.csv

// Make a solution at 10 times the optimum concentration

Make\_New\_100ml\_Beaker(newName: "too\_concentrated")

A clean 100 ml beaker was named too\_concentrated.

Transfer\_Soln\_with\_Graduated\_Cylinder(mL: 5.0, from: "stock\_base", to: "too\_concentrated")

Used a 10 ml graduated cylinder to transfer 5 ml of solution from stock\_base to too\_concentrated.

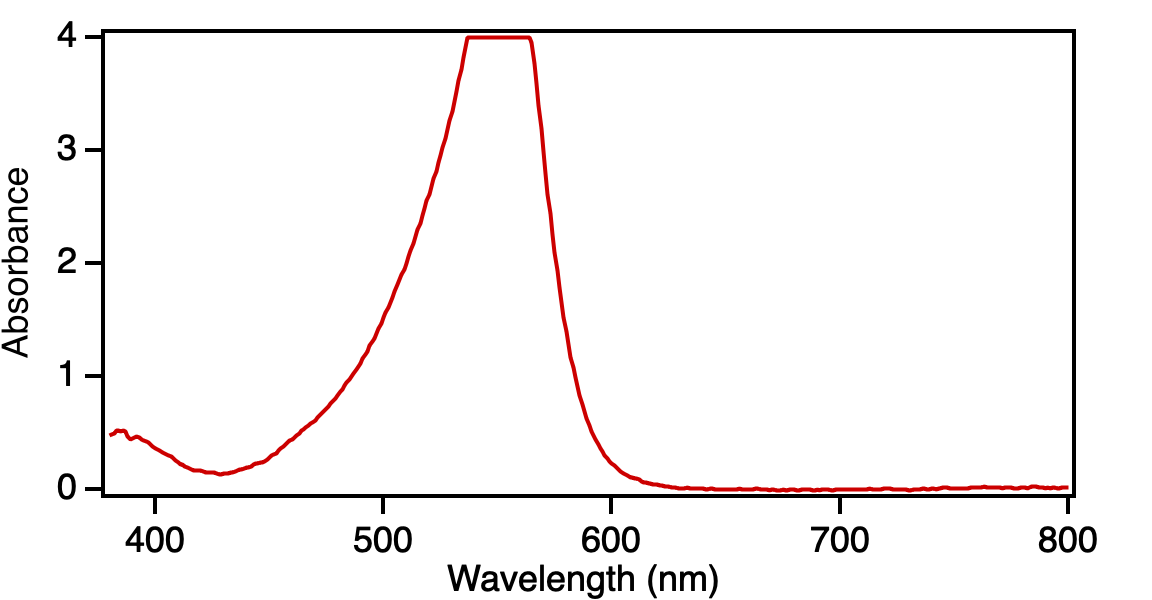
Add\_One\_Drop\_of\_Indicator(indicator: "Phenolphthalein", to: "too\_concentrated")

Added one drop of Phenolphthalein to too\_concentrated.

Take\_Spectrum(of: "too\_concentrated")

Your solution is too concentrated! The spectrum is being clipped at some wavelengths.

The spectrum of the solution in too\_concentrated is:



This spectrum has been saved as Spectrum\_2.csv

Asking Calvin to Use a Buret or Perform a Titration

Calvin can use a buret in two different ways. First, he can transfer a relatively precise amount of solution using a buret. Second, he can titrate a solution until a color change is observed, assuming of course that you have added an indicator to the solution. These tasks are illustrated by the output below, which we will examine in two parts. (Some of the output has been replaced by … for brevity.)

In the first part of the experiment, Calvin tries to add a very small amount of base to 20 ml of H2O, which generates some interesting results:

// Fill the buret with stock base solution and get some H2O

Fill\_Empty\_100ml\_Beaker\_with\_Standardized\_Base(newName: "stock\_base")

…

Fill\_50ml\_Buret(from: "stock\_base")

…

Fill\_Empty\_100ml\_Beaker\_with\_H2O(newName: "pure\_H2O")

…

// Fill a beaker with 20 ml H2O, measure pH, add tiny amount of base, measure pH

Make\_New\_100ml\_Beaker(newName: "dilute\_base")

A clean 100 ml beaker was named dilute\_base.

Transfer\_Soln\_with\_20ml\_Pipette(from: "pure\_H2O", to: "dilute\_base")

Pipetted 20 ml of solution from pure\_H2O to dilute\_base.

Measure\_pH(of: "dilute\_base")

The pH of dilute\_base is 6.14

Read\_Buret\_Volume()

The volume of solution in the buret is now 49.20 ml.

Add\_Soln\_from\_Buret(mL: 0.0001, to: "dilute\_base")

Read\_Buret\_Volume()

The volume of solution in the buret is now 49.15 ml.

Measure\_pH(of: "dilute\_base")

The pH of dilute\_base is 10.26

The first surprise in this experiment is the first *p*H measurement. Even though there was only “pure” H2O in the beaker, the *p*H was 6.14. This is typical of an actual lab, where the *p*H of pure H2O can range from ~6 due to dissolved CO2 which acidifies the H2O to ~7.5 from trace contaminants in the pipes.

The second surprise is that even though you asked Calvin to add 0.0001 ml of base, he actually added 0.05 ml. There are two problems. First, there is a limit as to how small of an amount you can add with a buret, typically ~1/20 ml (~one drop). The second problem is that 50 ml burets can only be read to an accuracy of 0.05 ml. When you ask Calvin to add a tiny amount from a buret, he will always add something. That change in volume may or may not be too small to read on the buret.

Finally, even though you added a tiny amount of base to the H2O, the *p*H skyrocketed by more than 4 *p*H units! This is to be expected, as a even a single drop of ~0.10 M base has a significant number of moles of OH–.

The experiment continues on the next page.

// Fill a beaker with 20 ml H2O, measure pH, add tiny amount of base, measure pH

Make\_New\_100ml\_Beaker(newName: "titration\_beaker")

A clean 100 ml beaker was named titration\_beaker.

Transfer\_Soln\_with\_20ml\_Pipette(from: "pure\_H2O", to: "titration\_beaker")

Pipetted 20 ml of solution from pure\_H2O to titration\_beaker.

Add\_One\_Drop\_of\_Indicator(indicator: "thymol\_blue", to: "titration\_beaker")

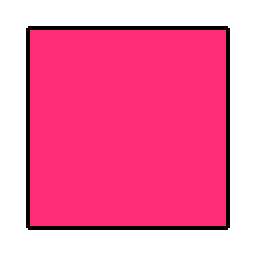
Added one drop of thymol\_blue to titration\_beaker.

Measure\_pH(of: "titration\_beaker")

The pH of titration\_beaker is 6.14

Titrate\_Beaker\_from\_Buret\_until\_Color\_Change(into: "titration\_beaker")

The color of the solution in titration\_beaker is:

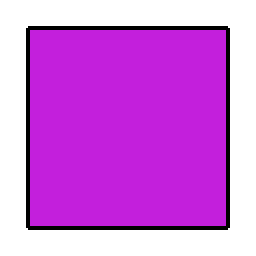


The volume of solution in the buret is now 49.15 ml.

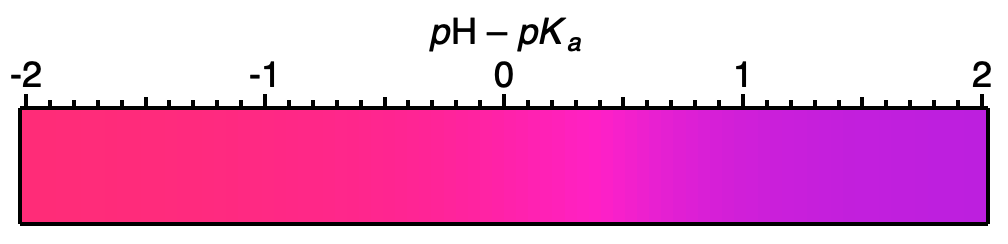
Performing titration now.

The volume of solution in the buret is now 49.10 ml.

The color of the solution in titration\_beaker is:



The color range of the indicator thymol\_blue is:



Measure\_pH(of: "titration\_beaker")

The pH of titration\_beaker is 10.58

In the second experiment, we added an indicator, thymol blue (*pKa* = 9.20), and had Calvin perform a titration. In principle, Calvin should have added just enough base to reach a *p*H of ~9.2, but you can see that he overshot the mark by over one *p*H unit and ended up at a *p*H of 10.58. Again, the problem wasn’t Calvin. He was working at the limit of his glassware and his eyesight. The problem is that the *p*H of a titration changes drastically when the titration is near the equivalence point.

Is Calvin a Good Lab Assistant?

Calvin almost assuredly performs your experiments more accurately than you could. Nevertheless, he is limited by the accuracy of his glassware and his equipment. Even the best of chemists would struggle to transfer one drop of solution accurately and reproducibly.

You should also realize that the *p*H of “pure H2O” — the stuff that comes out of real faucets — fluctuates with time. The *p*Hs of standardized solutions also change when new carboys are prepared. As a result, the precise *p*H values of H2O and the standardized solutions will typically change with each experiment file.

Calvin’s Glassware Supply

Calvin has an essentially endless supply of 50 ml volumetric flasks and 100 ml beakers. He also has a 20 ml pipette and both a 10 and 25 ml graduated cylinder. (The 10 ml graduate cylinder is more accurate than the 25 ml graduated cylinder, but Calvin will automatically use the larger cylinder for volumes exceeding 10 ml.) Calvin has one 50 ml buret.

Calvin washes and dries his measuring glassware and buret between uses, so you don’t need to tell him to do this. If you would like to refill a vessel with fresh solution, Clean\_and\_Dry the vessel, then make a new vessel with the old name.

Calvin’s Supply of Known and Unknown Chemicals

**Indicators:** Calvin has four known indicators with properties described in your textbook: Methyl\_orange, Thymol\_blue, Phenolphthalein, and Bromocresol\_green. Your TA will also assign you a named unknown indicator. Unknown indicators have names of the form Dye\_of\_*noun*.

**Solid acids:** Calvin has one known solid acid which can only be used for the Unknown Solid Acid experiment. That acid is named KHP, which is an abbreviation of potassium hydrogen phthalate. Your TA will also assign you a named unknown acid. Unknown acids have names of the form Acid\_of\_*DisneyCharacter.*

**Buffers:** Calvin has one known buffer which can only be used for the Unknown Buffer experiment. That buffer is accessed through the Fill\_Empty\_100ml\_Beaker\_with\_Standardized\_Buffer command. Calvin will inform you of the concentration of the standardized buffer when you run your experiment. Your TA will assign you a named unknown buffer. Unknown buffers have names of the form Buffer\_of\_*FictionalCharacter*.

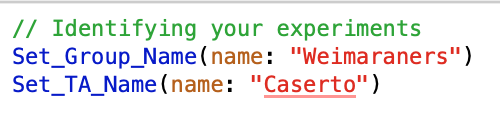
TA Names and Group Names

Your TA will tell you what form of their name to use. Some TAs will use their first name, others their last. Your TA will also assign you to a group. Group names are always a plural dog breed, such as the Weimaraners.

Problems with Calvin?

If your group name and/or TA name are set incorrectly or misspelled, your output will be sent to \_Lost & Found. Your TA may be able to find it, but it would be much faster for you to just spell things correctly and resubmit.

If you think that Calvin may be sick or sleeping, submit a two-line test file that sets your group name and TA name. For example:



If you are sure everything is OK with your group and TA names and Calvin has not responded within a reasonable period of time, please send an e-mail to Melissa.Hines@cornell.edu.

For the Cognoscenti

Calvin is a custom interpreted language implemented in *Igor Pro* and running on an iMac. All of the data used by Calvin are real and were collected in the undergraduate labs by Prof. Hines shortly before Cornell went on lock down. All of the colors generated by Calvin are calculated from actual spectra of indicators using the CIE 1931 color space. A general introduction to this conversion can be found at <https://www.fourmilab.ch/documents/specrend/>.

The name Calvin is an homage to Python, which is an interpreted language named after *Monty Python’s Flying Circus.* Python was written by engineer Guido von Rossum out of boredom while his research lab was closed over Christmas week in 1989. Boredom was not our primary motivator in writing Calvin, but we are nevertheless cooped up at home.

*Igor Pro* is an outstanding graphing and data analysis package for scientists and engineers. Igor is, of course, Dr. Frankenstein’s lab assistant and also Calvin’s supervisor. For more information about *Igor Pro,* visit WaveMetrics at <https://www.wavemetrics.com/>. Any piece of software that can enable a chemistry professor to write and deploy a new programming language around the world in a couple of weeks is wicked cool in my book.

— Melissa, the programmer, and Hobbes, the golden retriever

Copyright 2020, Melissa A. Hines

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Finding a Text Editor and Setting It Up for Calvin

**Mac users:** We suggest you download the *free version* of BBEdit from the link below. Do not purchase a license, as you do not need the extra bells and whistles.

<https://www.barebones.com/products/bbedit/download.html>

After downloading and installing BBEdit, go to BBEdit > Folders > Language Modules. This will open a folder in Finder. Put the file CalvinLanguageModule.plist in this folder, then restart BBEdit.

When writing to Calvin, you can either add a .calvin extension to your filename or set the popup menu at the bottom of your editing window to Calvin.

**Windows users:** We suggest you download Sublime Text from the link below. It is free for evaluation purposes, but will nag you about purchasing after some period of time.

<https://www.sublimetext.com/>

After downloading and installing Sublime Text, go to Preferences > Browse Packages… which will open a folder on your computer. Put the file calvin.sublime-syntax in the User folder, then restart Sublime Text.

You can change the color scheme using Preferences > Color Scheme… If you don’t care for the installed color schemes, go to Open Tools > Command Palette… Start typing Package Control: Install Package. When that option appears in the palette, click it. Wait for the available packages to show up, and then start typing Github Color Theme. When that option appears in the palette, click it to install the theme. Go to Preferences > Color Scheme… to select the scheme.

When writing to Calvin, add a .calvin extension to your filename to enable syntax coloring.

Sublime Text is also available for Mac and Linux.

**Ubuntu/Linux users:** We suggest you use Gedit, which is the standard GUI text editor on Ubuntu and comes pre-installed with most installations. You can find more information about Gedit as well as installation instructions at the following link:

<https://help.ubuntu.com/community/gedit>

You should then add calvin.lang to /usr/share/gtksourceview-3.0/language-specs.

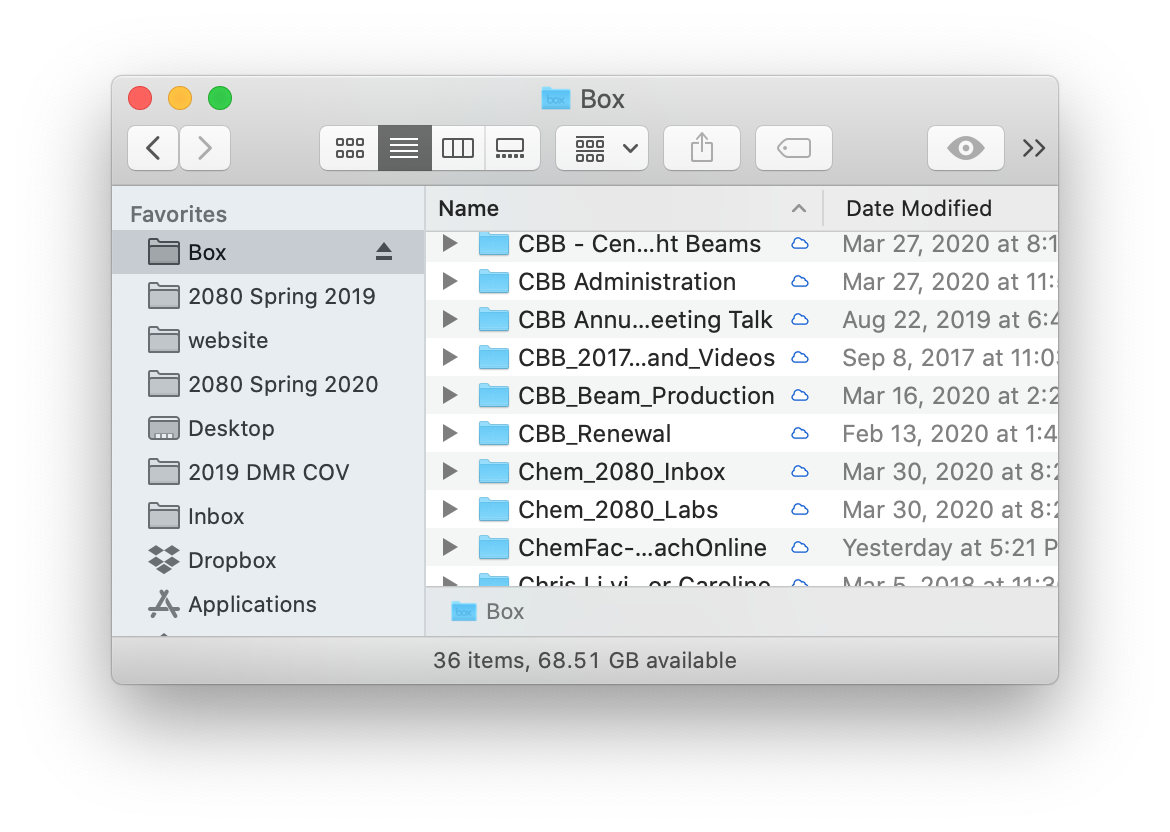
When writing to Calvin, add a .calvin extension to your filename to enable syntax coloring.

Using Box

Box is a cloud-based storage system. All Cornell faculty, staff, and students have an unlimited amount of storage on Box. Folders and files can be shared across multiple users. There are three ways of accessing your files on Box.

**Browser:** Go to <https://cornell.box.com/>

**Computer:** If you install Box Drive (not Box Sync, which is no longer recommended!), you can access your Box files like any other file on your computer. Go to <https://www.box.com/resources/downloads/drive> and download the installer for Mac or Windows. See the image below for an example.



**Box in China:** We think that Box is accessible in China at <https://app.boxcn.net/>, but we (obviously) have no way to test this. If you are having difficulty, this may be a good place to start:

<https://community.box.com/t5/Archive-Forum/Accessing-Box-from-China/td-p/56552>

1. Text files contain only plain text, no formatting or other information, and are stored as a sequence of characters. They can have many different file extensions, including .txt and .calvin. Word processors typically store their documents as binary files. Common binary file extensions are .doc and .docx. If you really want to use a word processor, you must save your file as plain text, which will typically lead to a .txt extension. [↑](#footnote-ref-1)
2. <https://www.youtube.com/watch?v=4r7wHMg5Yjg>, <https://www.youtube.com/watch?v=c36UNSoJenI> [↑](#footnote-ref-2)