LARSON—INFO 790–CLASSROOM WORKSHEET 01
Using CONJECTURING on CoCalc

1. Log in to CoCalc.

   (a) Start the Chrome browser.

   (b) Go to `https://cocalc.com`

   (c) Login (**your VCU email address** is probably your username).

   (d) You should see an existing Project for our class. Click on that.

   (e) Click "New", then type **790-c01** into the box, and click "Jupyter Notebook".

   (f) When your notebook opens look on the upper-right to mane sure the SageMath kernel is running.

Sage for Mathematics

The multiplication operator in Sage is "*". The most common error in Sage is forgetting to put in a "*" when multiplying.

2. Find $3 \cdot 4$.

3. Find $900(1 + .06(90/365))$.

4. Find 25^2 by evaluating either `25**2` (as in Python) or 25-caret-symbol-2 (like on a TI-calculator). Find 25^3.

`plot` is Sage's powerful and flexible command for plotting functions of a single variable.

5. Sketch the graph of x^3 on the interval $(-2, 2)$ by running the command: `plot(x**3,-2,2)`.

6. Sage was originally designed for number theorists. To test if the number 47 is prime, run `is_prime(47)`.

Setting up the Conjecturing Program

7. Follow the directions for setting up Conjecturing in CoCalc here: `https://nvcleemp.github.io/conjecturing/`

Conjecturing Invariant Bounds

We can use the `conjecturing` program to conjecture upper and lower bounds for an *invariant* of an mathematical object (number, matrix, graph, etc). An *invariant* in this context means any number associated with that object. So, for instance, the determinant of a matrix is a matrix-invariant.

Inequalities show up everywhere in mathematics; famous ones include the Cauchy-Schwartz inequality. Investigating bounds can be of enormous practical importance: bounds are useful when we want to reduce a *search space* where the answer to some question may be (for instance optimizing a discrete function).

8. Load “conjecturing.py” by running `load("conjecturing.py")`. (Your “790-c01.ipynb” must be in your root directory for this to work, and there should be an “expressions” file there.).

9. Try this first simple example. Interpret the conjectures. Are they true?

```
objects = [2,3,4]
invariants = [Integer.nbits, Integer.ndigits, Integer.sqrt]
invariant_of_interest  = invariants.index(Integer.nbits)
conjecture(objects, invariants, invariant_of_interest, upperBound = True)
```

   How does Conjecturing work?

   (a) What is the *Truth* heuristic?

   (b) What is the *Significance* heuristic?

   (c) Re-run the code to see some under-the-hood details using the *verbose* and *debug* options:

```
objects = [2,3,4]
invariants = [Integer.nbits, Integer.ndigits, Integer.sqrt]
invariant_of_interest  = invariants.index(Integer.nbits)
conjecture(objects, invariants, invariant_of_interest, upperBound=True,
     verbose=True, debug=True)
```