1. Log in to your Sage Cloud account.

   (a) Start the Chrome browser.
   (b) Go to http://cloud.sagemath.com and sign in.
   (c) You should see an existing Project for our class. Click on that.
   (d) Click “New”, call it c37, then click “Sage Worksheet”.

Mathematical objects not only have associated invariants but they also have properties, some condition $P$ that is either true or false for each member of the class. An example of a matrix property is being square: any given matrix is square or it is not. An example of a graph property is being Hamiltonian: any given graph is either Hamiltonian or it is not. An example of an integer property is being prime: any given integer is prime or it is not.

Properties define classes: the objects that have property $P$ define a class—the class of objects having that property. Similarly, classes define properties: the class $P$ defines the property of being a member of that class—any given object has the property of membership or non-membership in class $P$.

The conjecturing program can be used to generate conjectures about relationships between properties of objects (property-relation conjectures). The conjectures have the form $P \subseteq Q$, that is, “if a an object is in class $P$ then is in class $Q$”. Given a property of interest $P$, CONJECTURING can generate necessary conditions $Q$ for having property $P$ (these are also called upper bounds for $P$).

CONJECTURING can also generate sufficient condition conjectures: given a property $Q$ it will generate $P$’s that (might) imple membership in class $Q$ (these are also called lower bounds for having property $Q$).

2. We will first investigate property-relation conjectures for integers. is_prime, is_prime_power, is_square, is_squarefree, is_triangular_number are all built-in Sage integer properties. Test them all for the integers 2, 6, 8, 47, 1000. See if you can figure out what these properties mean.
3. Let's also define our own integer property; this will be a function that takes an integer as input, and returns a boolean. An integer is *perfect* if the sum of its proper divisors equals the number. Test this property on 2, 6, 8, 47, 1000.

``` python
def is_perfect(n):
    return sigma(n) == 2*n
```

4. Now let's generate lower bound conjectures for an integer to be squarefree. Investigate!

``` python
load("conjecturing.py")

def is_perfect(n):
    return sigma(n) == 2*n

objects = [2, 6, 8, 47, 1000]
properties = [is_prime, is_prime_power, is_perfect, is_square, is_squarefree, is_triangular_number, is_even, is_odd]
propertyBasedConjecture(objects, properties, properties.index(is_squarefree))
```

5. Now let's generate upper bound conjectures for an integer to be squarefree. Investigate!

``` python
load("conjecturing.py")

def is_perfect(n):
    return sigma(n) == 2*n

objects = [2, 6, 8, 47, 1000]
properties = [is_prime, is_prime_power, is_perfect, is_square, is_squarefree, is_triangular_number, is_even, is_odd]
propertyBasedConjecture(objects, properties, properties.index(is_squarefree), sufficient = False)