1. Log in to your Sage/Cocalc account.
   
   (a) Start the Chrome browser.
   (b) Go to http://cocalc.com and sign in.
   (c) You should see an existing Project for our class. Click on that.
   (d) Click “New”, call it c28, then click “Sage Worksheet”.

A **graph** is a mathematical object consisting of *dots* and *lines* (also called *vertices* and *edges*). A **tree** is a graph that contains no cycles.

Sage includes the **graphs** class which contains a number of **methods**. Some of these include constructors for making well-known graphs.

2. Try:

   ```python
   g=graphs.PetersenGraph()
   g.show()
   ```

   The **order** of a graph is the number of vertices it has. The **size** of a graph is the number of edges it has. How many vertices and edges does the Petersen graph have? Try `g.order()` and `g.size()`.

3. Try the following Sage Interact which shows some famous graphs and uses a **dictionary**:

   ```python
   @interact
   def i_graph(graph=selector(["icosahedron", "dodecahedron", "tetrahedron", "octahedron"], label="Select a graph", default="tetrahedron"):)
       dict={"icosahedron":graphs.IcosahedralGraph(),
             "dodecahedron":graphs.DodecahedralGraph(),
             "tetrahedron":graphs.TetrahedralGraph(),
             "octahedron":graphs.OctahedralGraph()}
       g=dict[graph]
       order = g.order()
       size=g.size()
       print "The {} has {} vertices and {} edges".format(graph,order,size)
       g.show()
   ```
4. Let's get acquainted with paths, cycles, stars, and complete graphs. Try:

```python
@interact
def i_graph(graph=selector(['path', 'cycle', 'star', 'complete'],
                            label='Select a graph', default='path'),
           order=slider(3, 20, 1, 3):
    dict = {'path': graphs.PathGraph(order),
            'cycle': graphs.CycleGraph(order),
            'star': graphs.StarGraph(order),
            'complete': graphs.CompleteGraph(order)}

g = dict[graph]
order = g.order()
size = g.size()
print "This graph has {} vertices and {} edges".format(graph, order, size)
g.show()
```

5. We can create our own graph using the `Graph()` constructor, and the `add_vertex()` and `add_edge()` methods. Let's make a cycle on 5 vertices. First initialize the graph and make the vertices:

```python
g = Graph()
for i in [1..5]:
    g.add_vertex()
g.show()
```

Notice that the vertex labels start at 0. Now make the edges:

```python
for i in [0..3]:
    g.add_edge(i, i+1)
g.show()
```

You're still missing an edge. So add that.

6. Now use `Graph()`, `add_vertex()` and `add_edge()` to make the bull:

![Bull Graph](image)

Start by letting `bull = Graph(5)`. Instead of using `add_vertex()`, you can start with `Graph(5)` to get a graph with 5 vertices and no edges. Now add the edges that you see in the diagram of the bull using `bull.add_edge()`. Remember that the layout of the graph doesn’t matter—only that it has the same edges.

7. Try the problem at [http://projecteuler.net/problem=8](http://projecteuler.net/problem=8). How will you get that number into your program??? Do not type it in by hand.