1. Log in to your Sage Cloud account.
   
   (a) Start the Chrome browser.
   (b) Go to \( \text{http://cloud.sagemath.com} \) and sign in.
   (c) You should see an existing Project for our class. Click on that.
   (d) Click “New”, call it \textbf{c13}, then click “Sage Worksheet”.

2. Try \( \text{plot\_step\_function([(x,x) for x in [3..9]])} \)

3. Try \( \text{plot\_step\_function([(i,sin(i)) for i in [5..20]])} \)

4. Try \( \text{plot\_step\_function([(i*.2,sin(i*.2)) for i in [5..100]])} \)

   Given a list \( L \) of pairs \((x,y)\) you can plot the \textit{step function} that holds \( y \) constant from one \( x \) to the next with \texttt{plot\_step\_function(L)}.

5. Try \( \text{scatter\_plot([(0,1),(2,4),(3.2,6)])} \)

6. Try \( \text{scatter\_plot([(x,x) for x in [5..20]])} \)

7. Try \( \text{scatter\_plot([(x,x**2) for x in [-5..5]])} \)

8. Try \( \text{scatter\_plot([(i*.2,sin(i*.2)) for i in [5..100]])} \)

9. Define a function \texttt{points(x)} that plots all the points \((1,2), (2,3), \ldots (x,x+1)\). Use \texttt{scatter\_plot().}
Recursion

A recursive function is a function that calls itself. It must always have a base case so that the recursion eventually stops.

10. Here is an example of a recursive definition of the factorial function. The base case here is the case where the input is 0 or 1.

```python
def factorial(n):
    if n==0 or n==1:
        return 1
    else:
        return n*factorial(n-1)
```

Now try `factorial(0)`, `factorial(1)`, `factorial(2)`, `factorial(3)`, and `factorial(10)`.

11. It is often intuitive to define a function recursively, but usually the same function can be defined without recursion. Here is a function `factorial2(n)` that does the same thing as `factorial(x)` but is not recursive. Test it to make sure it gives the same results.

```python
def factorial2(n):
    result=1
    if n==0:
        return result
    for i in range(1,n+1):
        result=result*i
    return result
```

12. The gcd of 2 non-negative integers is their greatest common divisor. The following recursive function calculates the gcd of integers \( a \) and \( b \) using the fact (which can be proved) that, if \( a \geq b \) then \( \text{gcd}(a,b) = \text{gcd}(a - b, b) \). It uses the fact that \( \text{gcd}(0,a) = \text{gcd}(a,0) = a \), for any non-negative integer \( a \), as the base case.

```python
def gcd(a,b):
    if a==0 or b==0:
        return max(a,b)
    else:
        return gcd(max(a,b)-min(a,b),min(a,b))
```

Try `gcd(0,5)`, `gcd(2,5)`, `gcd(5,5)`, `gcd(10,5)`, `gcd(50,51)`, `gcd(50,55)`, and `gcd(1234,5678)`.