Defining Functions in Python

Overview: This activity focuses on an important idea in programming: user-defined functions. This concept appears in practically every programming language, and in Python, there are some particular options available (default arguments, for example) that make it easy to define functions in flexible ways.

Motivation:

Suppose that in a PixelMath experiment, we wish to set up three copies of the Mona Lisa image, resize each window to a larger size and zoom in one level. The following sequence of commands would need to be done three times:

```
a = pmOpenImage(0, "mona-rgb.jpg")
pmPositionWindow(a, 0, 0, 600, 600)
pmZoom(a, 200, 200)
```

Rather than repeat all this three times, we could simple create a function once and then "call" it three times as follows:

```
def make_zoomed_mona():
  a = pmOpenImage(0, "mona-rgb.jpg")
  pmPositionWindow(a, 0, 0, 600, 600)
  pmZoom(a, 200, 200)

make_zoomed_mona()
make_zoomed_mona()
make_zoomed_mona()
```

Next let's consider defining a function that can do something to a value that can be different each time the function is called.

```
def cube(x):
   return x*x*x
```

Each call to cube can use a different value…

```
>>> print cube(2)
8
>>> print cube(5)
125
```

Write the body for the definition of a new function (below) that will create a square image of a size determined by a parameter (argument) to the function. The definition has been started for you. You just have to fill in the rest of it. The function should return the window number of the new image. (You should use the function pmNewComputedImage to do the work.)

```
def newSquareImage(size):
    # add your code here.
```

Next write functions to do each of the following tasks:

Create an image of size 64 by 64 whose pixels are colored using HSV(a, b, c) where a is a parameter to your function, b is 1, and c is 1.

Draw a horizontal line across the middle of an image by changing some of the pixels to red:

```
def horizontal_line(a, width, height):
```

Your function definition should make use of:

a. pmSetPixel(wn, x, y, r, g, b) to change the r, g, and b values of a pixel in an image.
b. looping with the for statement.

Finally, try to write a function that will accept as an argument the window number of an existing image and that will create a new image that is a mirror image of the old one.

```
def make_mirrored_copy(wn):
```