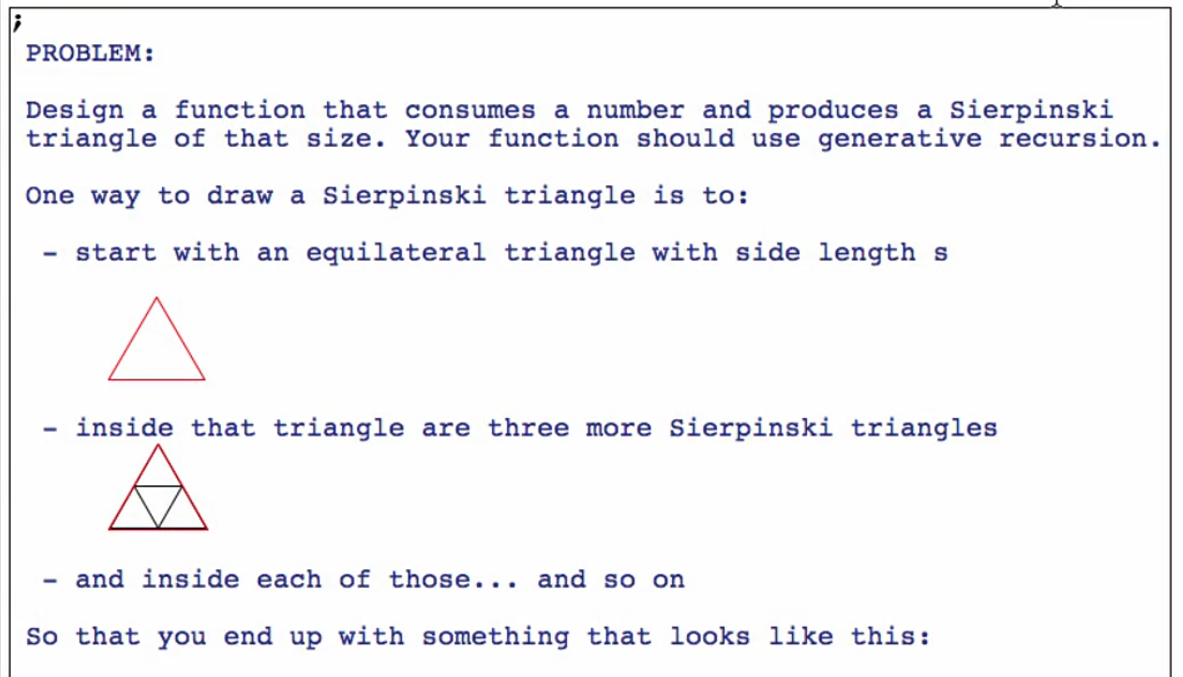
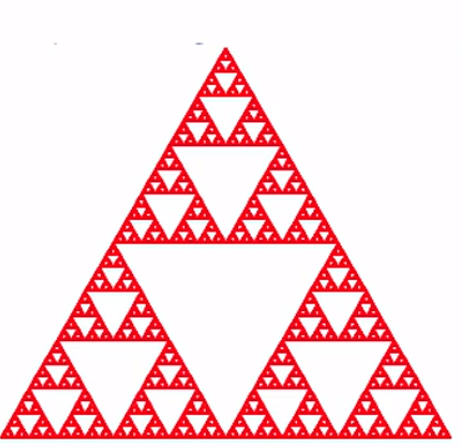
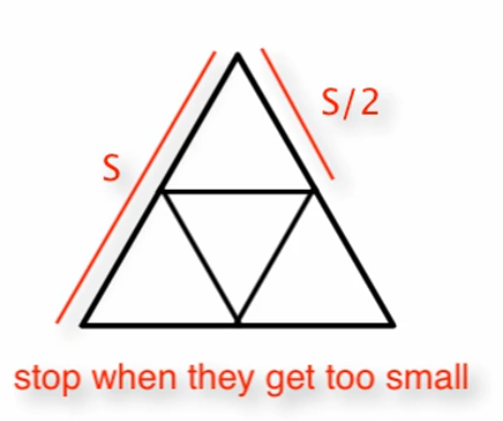
Fractals – where the image itself has a recursive structure



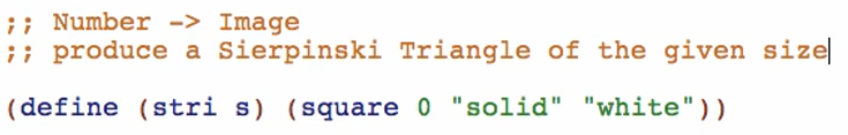


Small domain analysis



***Function definition***

**Signature, purpose and stub**



**Examples**

Always start with **base** case (or the trivial case)



Define the smallest you want as a constant

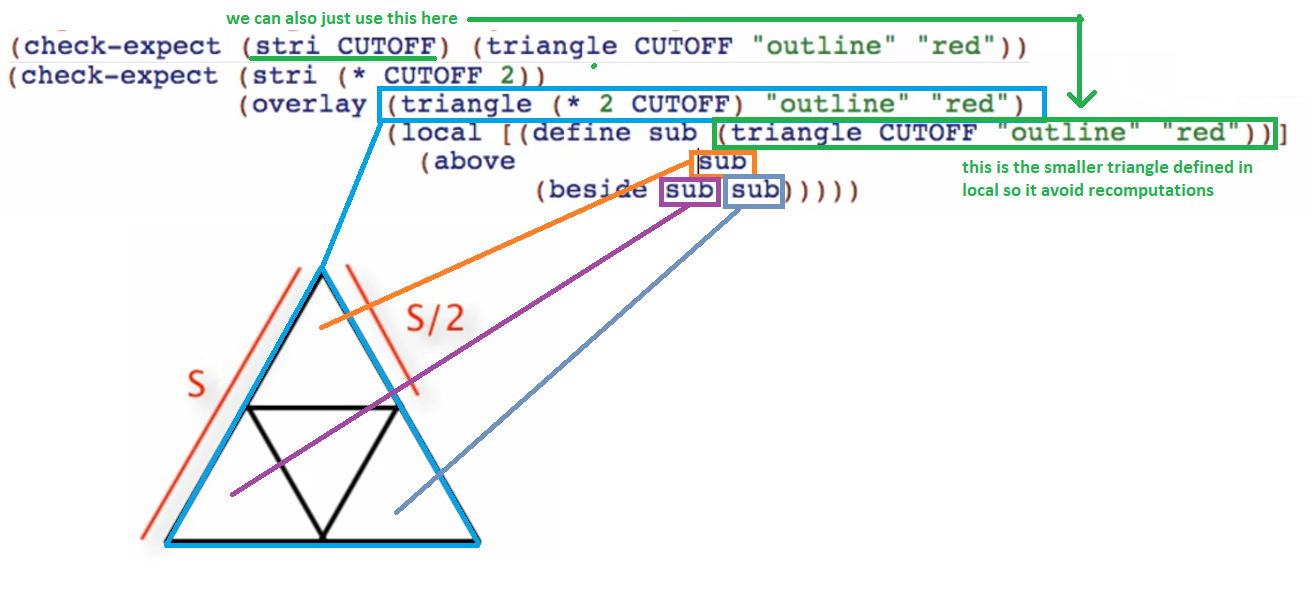


Check-expect



Then to the **NEXT SMALLEST** case (esp. in Generative Recursion)

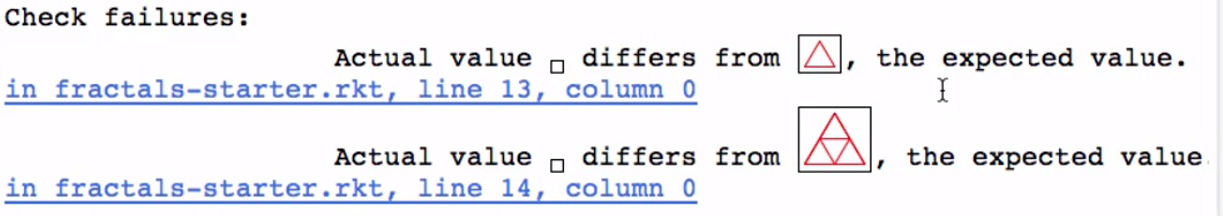
* This can really help us understand how the function works
* Avoid any recomputations: use local!



Run and see what it looks like and if it is well-formed!

You can temporarily increase the cutoff value for the smallest triangle so you can see a preview of your triangles on the results

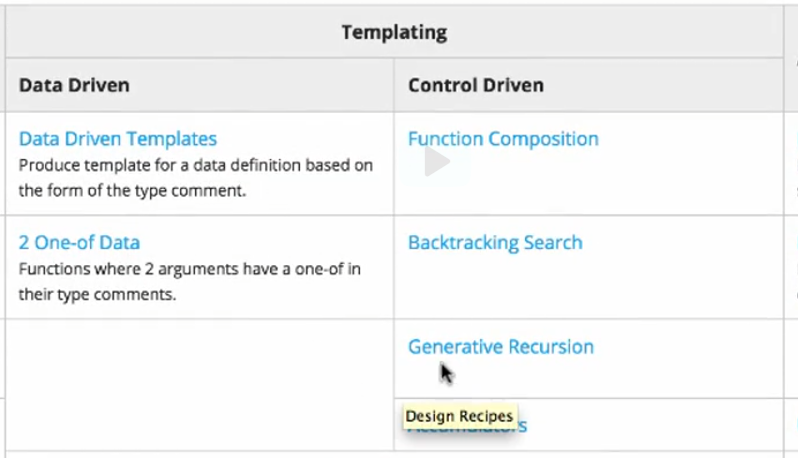


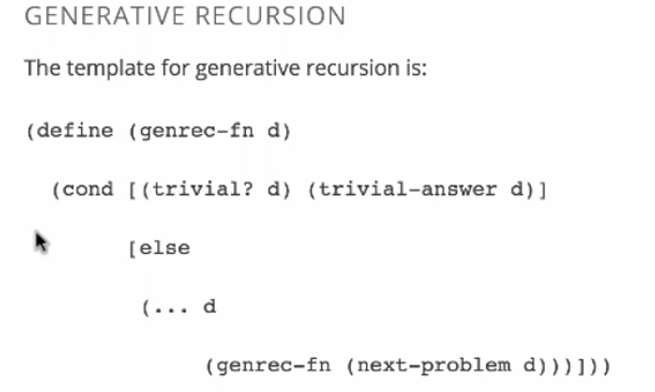


Don’t forget to put it back!

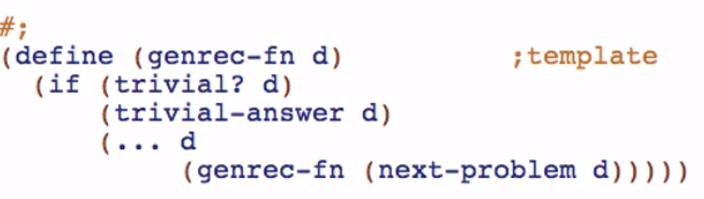


**Template**

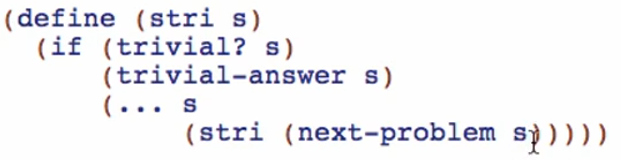




Copy and paste the template

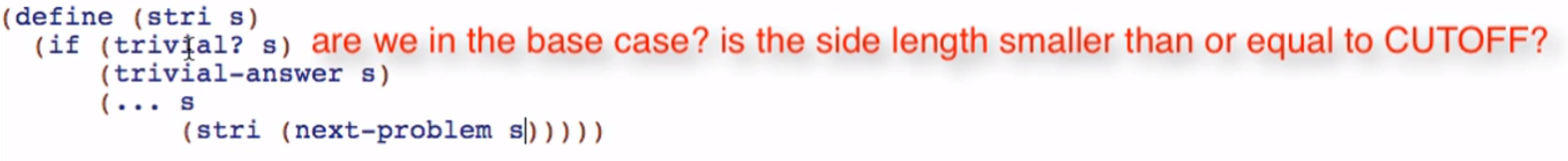


Rename the function names, natural recursions and parameters

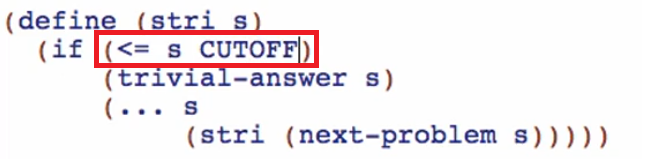


**Code body**

How to read the template?

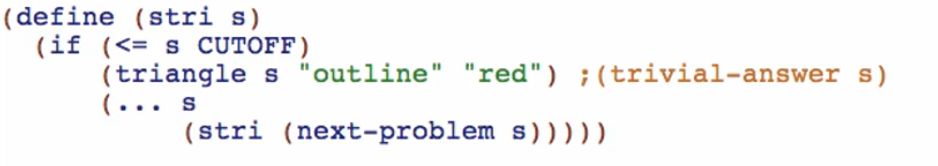


When is our serpenski triangle trivial or the smallest? It is when the side is CUTOFF



What you’ll do when there will be no more recursion?

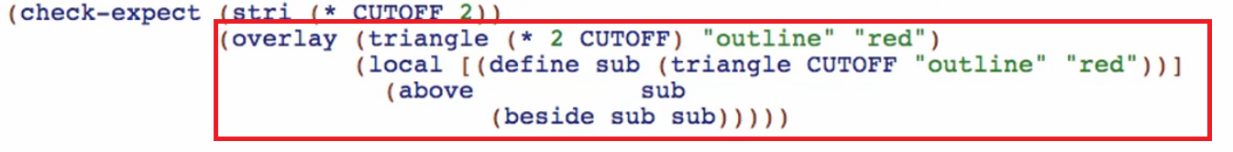






* So we can handle s that are less than the CUTOFF!

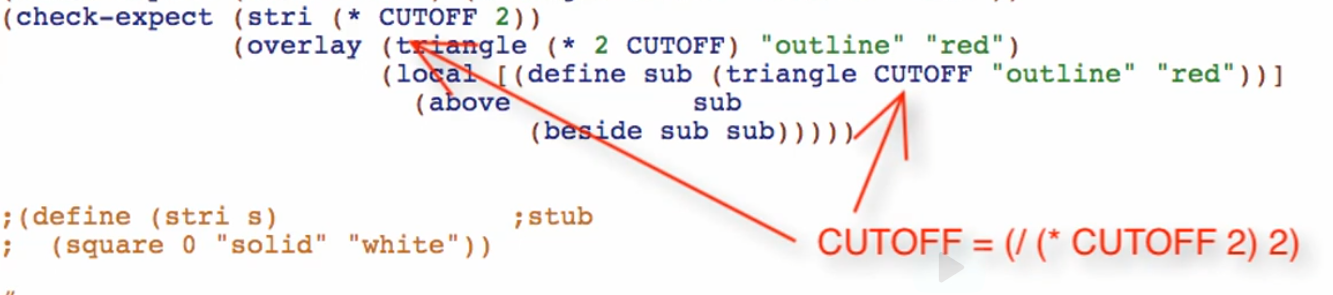
What you’ll do when we’re not in the base case?

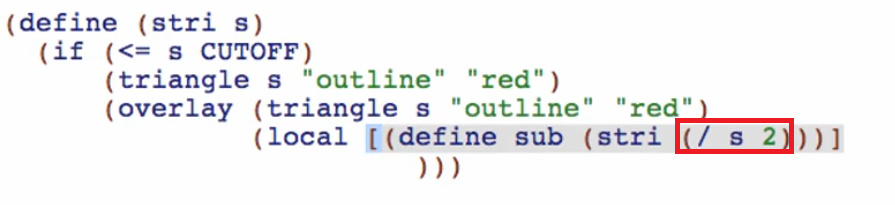




* If you will not reduce s, the recursion will not end!

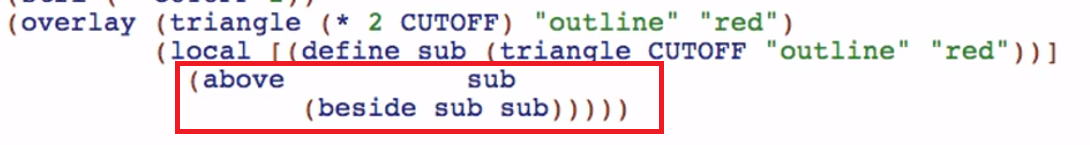
We make a triangle that is smaller than the previous one

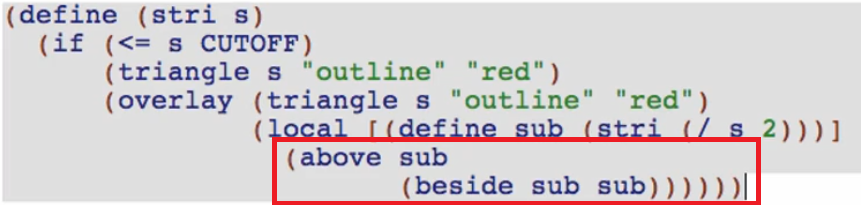




* we divide the current argument “s” to 2

Then we can now code this part of the check-expect





Run and test:

