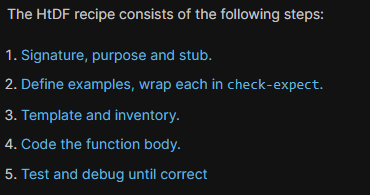
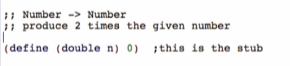
Recipe: 

**Signature, Purpose and Stub**



Signature

* Type of data consumes -> Type of data return/produces
* Type names are capitalized
* 
* Read: function consumes a Number and produces a Number
* Start each line with ;; to indicate that it is permanently commented out

Purpose

* 1 line description of what the function produces in terms of what it consumes
  + Needs to say more than the signature
  + 
* Start each line with ;; to indicate that it is permanently commented out

Stub

* A function definition that:
  + Has correct function name
  + Has correct number of parameters
  + Produces dummy result of correct type



* Comment out only with one semi-colon ; since we are going to delete it later while we progress

**Examples (wrapped in check-expect)**

* Help us understand what function must do
* Multiple examples to illustrate behavior
* Wrap in check-expect to also serve as unit tests



After Examples, we run the program to see if the examples are well-formed

* This is where stub is going to help us
  + Letting us make sure that the tests actually run (no signature violations and whatsoever)
  + This is better rather than building the whole thing before knowing the errors like syntaxes, signature errors, and such.
  + 

Note: every step of the recipe helps with the steps after it!

**Inventory – template & constants**

Template

* Right function name and parameter
* The body of the template is the outline of the function
  + For now, (…n) where n is the parameter of the function
  + 

Copy the template, remove the label and comment out the original template



**Code body**

* Use everything written before to know how to complete the function body
* Sometimes helps to elaborate examples to show how the expected value could have been produced: 
* 

**Test and debug**

** -> **