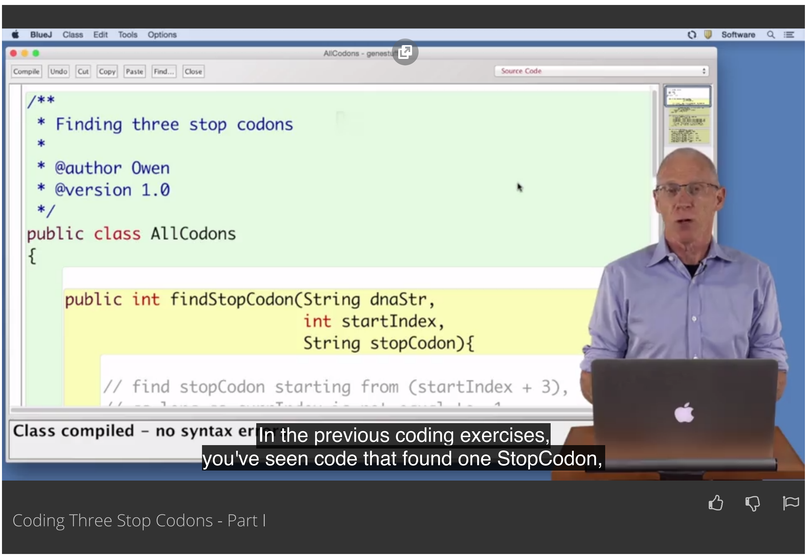

Coding Three Stop Codons - Part I

Have a question? Discuss this lecture in the week forums.

Interactive Transcript

Search Transcript

English

0:03
Hi, welcome to another version of Gene-Finding. In today's episode, we're going to look for three different stopCodons. In the previous coding exercises, you've seen code that found one StopCodon, making sure with a multiple three away from the StartCodon. In this version of the code that you've already seen described previously, we're going to look for three different StopCodons. As you can see here we have TAA, TAG, and TGA, and in order to find each of those we are going to write another method. A method called findStopCodon that we saw outlined before with our seven step method.

0:43
Up here I've got this method, findStopCodon, and it has three parameters, dnaStr, the string of DNA in which we're searching for a gene, startIndex, the location at which we're going to begin the search. And stopCodon, the specific codon we're going to be looking for. We've got the code from our seven step process. And what we're going to do here is, write the code that goes along with this. You can see here as we saw in the previous version of code, I'm going to create a variable currIndex and I'm going to make it the location of the first stopCodon, starting at startIndex + 3. So as we've seen before the way we do that is to create an integer variable currIndex

1:30
and we right currIndex gets the location in dnaStr of the stopCodon, starting at startIndex.

1:46
That's exactly what it says here. Find stopCodon starting from (startIndex + 3), so I'd better make sure I'm moving that correctly, and store that value in currIndex. Now as long as current index is not equal to -1, and as you already learned that's a while loop, so we're going to say while currIndex is not equal to -1. And when I type the right curly brace as you see here, I'm going to make sure I put in a left curly brace at the same time.

2:18
Indented the same way, and one thing I can do there to make sure that I've got it right is to compile my program, so I'm compiling it you can see I've got no syntax errors, so I know I'm on shape. Now, as long as currIndex is not equal to -1, so I've just coded that, check to see if the currIndex- startIndex is a multiple of 3. I'm going to store that difference. I'm going to say int diff gets currIndex- startIndex. Now you don't need a new variable for that. As you saw in the previous video, you can just say, if currIndex- startIndex mod 3 is equal to 0. But this sometimes easier conceptually to say if (diff % 3 == 0), that means, and you can see I've already got my curly braces in there. Return current index because it's my answer. So I'm just going to say return currIndex.

3:15
That's what it says here. Check if currIndex- startIndex is a multiple of three, and if so, currIndex is the answer return it. So I've got those two things taken care of.

3:26
And it says, if not, update currIndex.

3:31
So if not, that would be else. And if you think carefully, you might see that, although I don't need the else because I've already returned. But it's probably easier to write it that way in terms of thinking about it. It says, update currIndex, looking for stopCodon again, starting from currIndex plus one. So, you can see up there how I look for it. So, I'm going to do the same thing. CurrIndex gets dnastr.indexOf, I'm still looking for stopCodon and remember, that's a parameter to this method. And I'm starting, according to our comments at currIndex + 1. So, currIndex + 1 and if we exit the loop we didn't find the stopCodon, so we return dnastr.length. You can see here that the loop is going to continue while currIndex is not negative 1. If it is equal to negative 1 we've exhausted the string, the dna string in which we are searching. And so it says return dnaStr.length. As we'll see later when we write this code, that's to ensure that when we find the minimum value that our code works properly. So I'm going to make sure that my method looks right, find the stopCodon

4:55
and I started at index+3, so that's the last comment I have. I'm going to get rid of that comment now.

5:04
So I've stored in currIndex the location of stopCodon while that's not -1, if it's a multiple of 3 I'm done, and if it's not I need to keep searching.

5:15
My class compiles, but just compiling doesn't indicate at all that I'm done. I've written some test code down here, testFindStop. And you can see as you saw in the previous coding video. I've created a strand of DNA and I'm going to look for TAA. Which is one of the stop codons, and I've got it in two different places. When I run the method you'll see what it does, and then I'll describe how that testing works. So here's my workbench, I'm going to right click on this and make a new object.

5:49
It comes in the object work bench. I'm going to right click again and I'm going to run test find stop.

5:56
And you can see it printed tests finished. That means all the tests work correctly. Because if there was an error, my print statements would have indicated that. Lets look at the testing code so you can see how I did that.

6:09
In testing findStop, the first thing I did was look for the stopCodon TAA, starting at index 0. And you can see here, I've got the indices printed above the string. So TAA here is at index 9. 9 minus 0 Is in fact a multiple of three, so it should have returned 9, that it found TAA, and if it wasn't 9, I got an error. To see how it might work if it returned to wrong thing. Supposed I said of dex is not equal to ten, now that would be a mistake, because we know it returns nine and it's supposed to return nine. You'll see in this case when I run it, that it will print something. Right-click to create the object. Right-click to invoke the findStops. And you can see error on 9. Now, that wasn't an error because I knew I was supposed to get 9. All these tests print something if there was an error. And if there is no error, they don't print anything. My last print statement prints when the tests are finished. So what we have now is this method findStopCodon written and tested and we are ready to move on to the next method which was the same method that you saw before for finding just a single stopCodon. And now we're going to find any occurrence of a stopCodon. Any occurrence of three different stopCodons. We're going to find TAA or TAG or TGA.

Downloads

Lecture Video mp4
Subtitles (English) WebVTT
Transcript (English) txt

Would you like to help us translate the transcript and subtitles into additional languages?