# What the Hack: Indexing & Slicing | Episode 7 Challenges Solutions

*Finish testing your knowledge? Take a look at some solutions to the challenges!*

## Challenge 1: Another Secret Message! Decode It With Slicing!

```
#first, save the secret message into a variable
secretMessage = 'joalGak owrooi;sdlkd weiJr hojasbd;o! Diok;jsdkl;ajdoi skjdS
faCslikj ioweojw*e k'

###The start index
##"This secret message begins at the index that is the same as the number of
times a semicolon (;)"
#this means that we need to count how many times a semicolon appears in our
secret message!
#for this, we can use the .count() method
#pass in the semicolon enclosed by quotation marks as a string to the .count()
method, and print it out to see what the number is!
print(secretMessage.count(';'))
#we can save the result of this .count() method into a variable, let's call it
startIndex
startIndex = secretMessage.count(';')

###The stop index
##"(This secret message) ends at where the exclamation mark (!) is"
#this means that we need to find the index for the character ! in our gibberish
string!
#we can do that by using the .index() method
#pass in the exclamation mark (!) enclosed by quotation marks as a string to the
.index() method, and print it out to see what the number is!
print(secretMessage.index('!'))
#but also, remember when doing slicing, the stop index is the index right after
where the slicing actually stops!
#so, we need to add 1 to the index of the exclamation mark
#we can save the result of this .index() method plus 1 into a variable, let's
call it stopIndex
stopIndex = secretMessage.index('!') + 1
```

```
###The skip value
##"The characters that make up the actual message are embedded in this gibberish string in a specific pattern: they have a number of characters in between them."
#This means that we need to skip a number of characters when we do slicing!
##"How many characters in between them? I don't know, maybe 1, maybe 2, maybe 3, maybe 4, maybe 5."
#this means that we have 5 numbers to try for the skip value: 1, 2, 3, 4 and 5!

#while we can definitely try each number one by one by changing the number everytime we run the code, utilizing a for loop or a while loop can automate this task for us, meaning that we can ask the computer to do the job and we can just relax and see the result!

##for loop solution
#let's see a solution with for loop
#remember in a for loop, we need to specify something to loop through. In our case, we need to specify a range of number (1,2,3,4,5). We can do that using the range() function in Python! remember that the second number that we give to the range function needs to be 1 value greater than the last number that we want to have in this range
#remember a for loop also needs a variable that will loop through, or iterate through the range of numbers that we have. We can call it skipValue
for skipValue in range(1,6):
 #each time skipValue loops through a number, we want it to try it as the skip value for slicing, and we want it to print the message out and see if this is the correct skip value or not
 #we can slice the secret message using the start index, stop index and the skip value that we have
 #we can save the sliced string into a variable called decodedMessage
 decodedMessage = secretMessage[startIndex:stopIndex:skipValue]
 #now let's print out the skip value that the for loop tries for us
 print("Skip Value: " + str(skipValue))
 #and then print out the decoded message when our for loop tried that skpi value for us
 print(decodedMessage)
 #we can print a blank line of message, to make the terminal a little cleaner to look at
 print("")

##while loop solution
#what we did with a for loop, we can achieve it with a while loop as well! let's
take a look!
#remember when using a while loop, we need to declare a variable to contain a
start value for our looping.
#the only thing that we need to change when we want to try to decode the message,
is the skip value
#because we want to try the skip value of 1 first, we can assign our variable
skipValue an integer 1
skipValue = 1
#remember for a while loop, we only need to specify a condition before the
compile
#here, because the spirit of computer science mentioned that the skip value can
be 1, 2, 3, 4, 5, so we can set 5 to be the largest number that we want our while
loop to try for us! here we say, while the skipValue is less than or equal to 5,
keep trying. If the skipValue becomes greater than 5, stop trying.
while skipValue <= 5:
 #here, let's slice the secret message with the startIndex, stopIndex and the
skipValue
 #and save it into the decodedMessage variable
 decodedMessage = secretMessage[startIndex:stopIndex:skipValue]
 #now let's print out the skip value that the for loop tries for us
 print("Skip Value: " + str(skipValue))
 #and then print out the decoded message when our for loop tried that skpi value
for us
 print(decodedMessage)
 #we can print a blank line of message, to make the terminal a little cleaner to
look at
 print("")
 #the last step in a while loop is to change the value of the variable in the
condition. In our case, we need to increase the skipValue by 1, so that in the
next iteration, the while loop will try the next integer as the skip value when
slicing for us
 skipValue += 1

#Have a look at the result, and, what was the secret message?
```