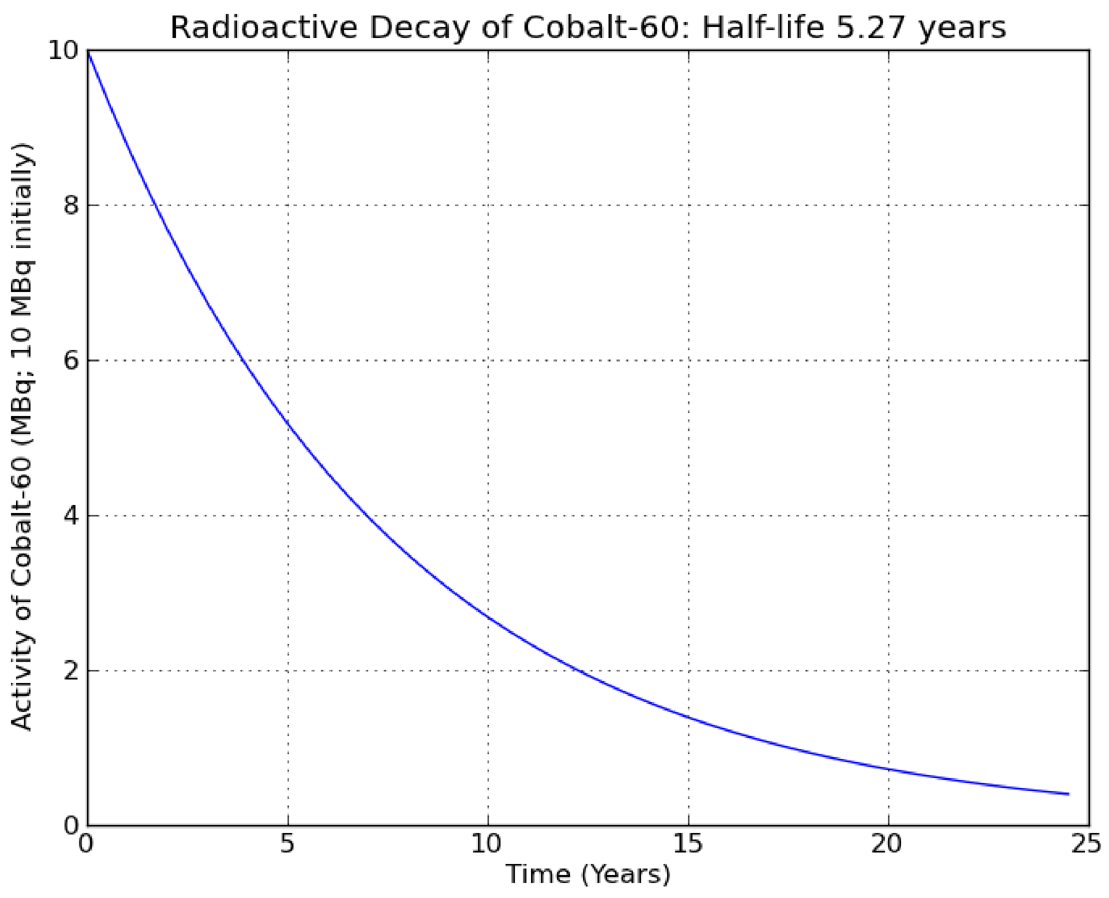
## RADIATION EXPOSURE (25/25 points)

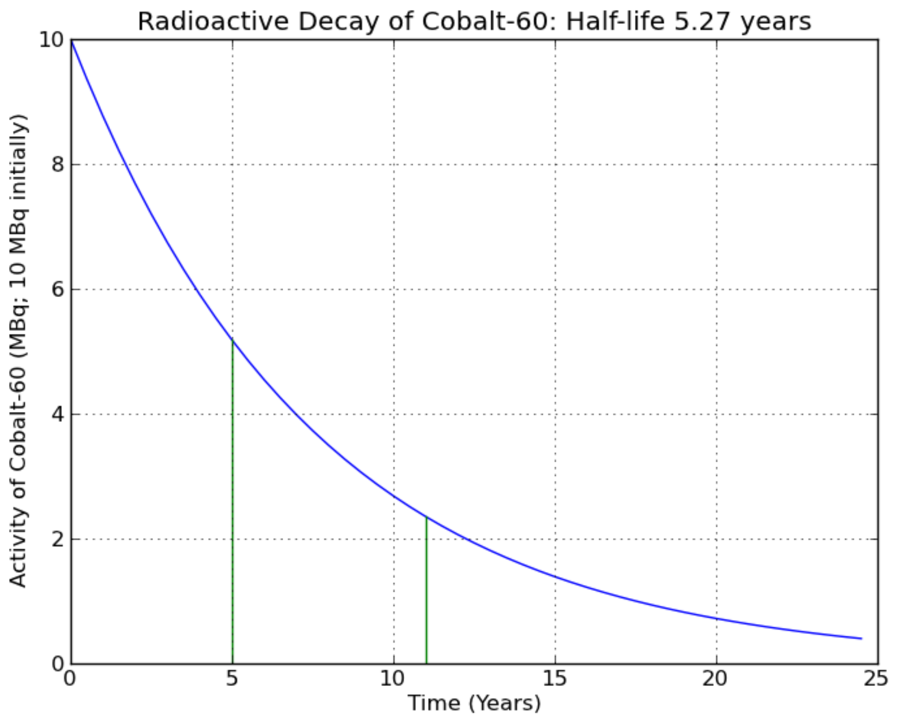
"Radioactive decay" is the process by which an unstable atom loses energy and emits ionizing particles - what is commonly referred to as radiation. Exposure to radiation can be dangerous and is very important to measure to ensure that one is not exposed to too terribly much of it.

The radioactivity of a material decreases over time, as the material decays. A radioactive decay curve describes this decay. The x-axis measures time, and the y-axis measures the amount of *activity* produced by the radioactive sample. 'Activity' is defined as the rate at which the nuclei within the sample undergo transitions - put simply, this measures how much radiation is emitted at any one point in time. The measurement of activity is called the Becquerel (Bq). Here is a sample radioactive decay curve:

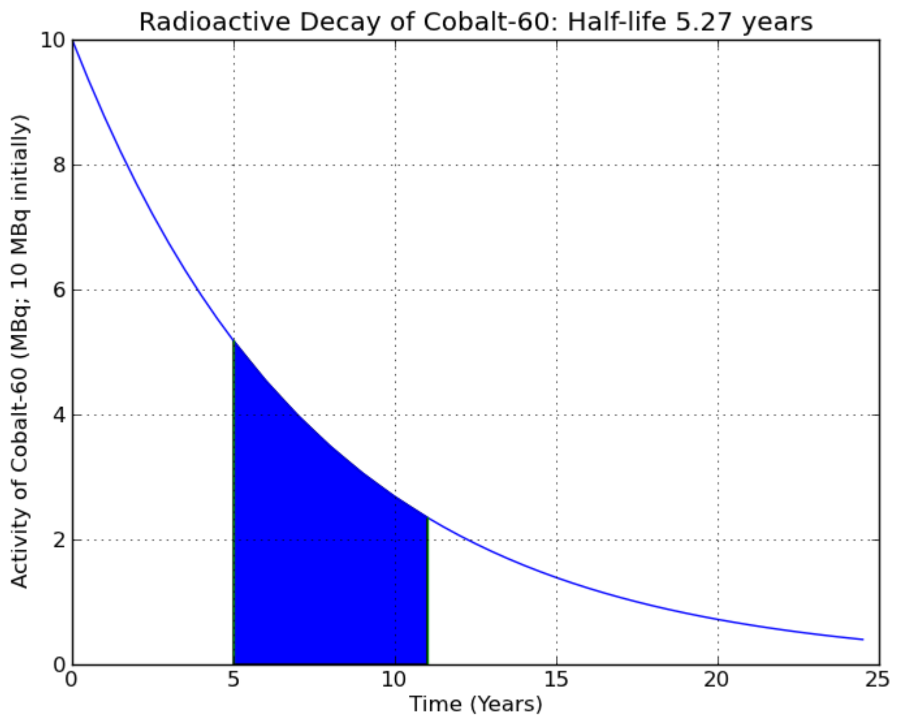


Now here's the problem we'd like to solve. Let's say Sarina has moved into a new apartment. Unbeknownst to her, there is a sample of Cobalt-60 inside one of the walls of the apartment. Initially that sample had 10 MBq of activity, but she moves in after the sample has been there for 5 years. She lives in the apartment for 6 years, then leaves. How much radiation was she exposed to?

We can actually figure this out using the radioactive decay curve from above. What we want to know is her *total radiation exposure* from year 5 to year 11.

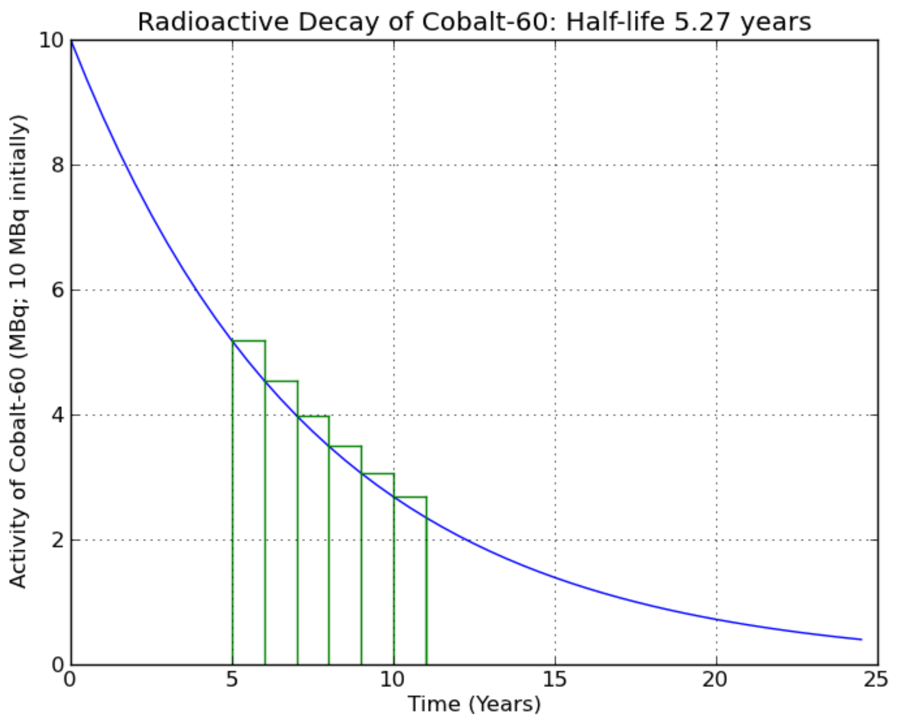


Total radiation exposure corresponds to the area between the two green lines at time = 5 and time = 11, and under the blue radioactive decay curve. This should make intuitive sense - if the x axis measures time, and the y axis measures activity, then the area under the curve measures (time * activity) = MBq*years, or, approximately the total number of MBq Sarina was exposed to in her time in the radioactive apartment (technically, this result is the combination of gamma rays and beta particles she was exposed to, but this gets a bit complicated, so we'll ignore it. Sorry, physicists!).

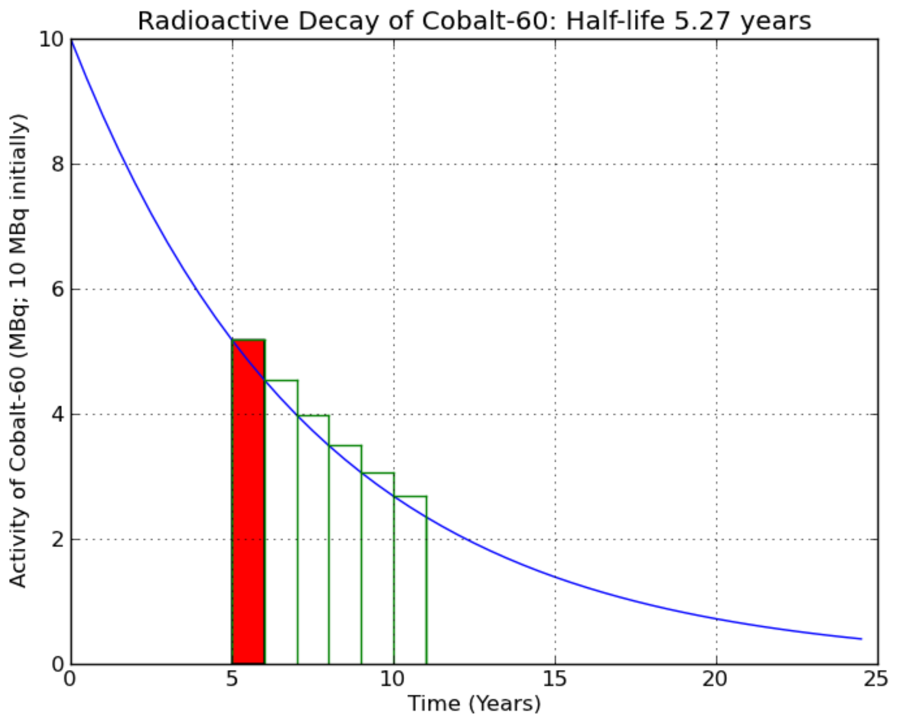


So far, so good. But, how do we calculate this? Unlike a simple shape - say a square, or a circle - we have no easy way to tell what the area under this curve is.

However, we have learned a technique that can help us here - *approximation*. Let's use an approximation algorithm to estimate the area under this curve! We'll do so by first splitting up the area into equally-sized rectangles (in this case, six of them, one rectangle per year):



Once we've done that, we can figure out the area of each rectangle pretty easily. Recall that the area of a rectangle is found by multiplying the height of the rectangle by its width. The height of this rectangle:



is the value of the curve at 5.0. If the curve is described by a function, `f`, we can obtain the value of the curve by asking for `f(5.0)`.

```
f(5.0) = 5.181
```

The width of the rectangle is 1.0. So the area of this single rectangle is `1.0*5.181 = 5.181`. To approximate how much radiation Sarina was exposed to, we next calculate the area of each successive rectangle and then sum up the areas of each rectangle to get the total. When we do this, we find that Sarina was exposed to nearly 23 MBq of radition (technically, her apartment was bombarded by 23e6 * 3.154e6 = 7.25e13 neutrons, for those interested...).

Whether or not this will kill Sarina depends exactly on the type of radiation she was exposed to (see this link which discusses more about the ways of measuring radiation). Either way, she should probably ask her landlord for a substantial refund.

In this problem, you are asked to find the amount of radiation a person is exposed to during some period of time by completing the following function:

```
def radiationExposure(start, stop, step):
    '''
    Computes and returns the amount of radiation exposed
    to between the start and stop times. Calls the
    function f (defined for you in the grading script)
    to obtain the value of the function at any point.

    start: integer, the time at which exposure begins
    stop: integer, the time at which exposure ends
    step: float, the width of each rectangle. You can assume that
      the step size will always partition the space evenly.

    returns: float, the amount of radiation exposed to
      between start and stop times.
    '''
```

To complete this function you'll need to know what the value of the radioactive decay curve is at various points. There is a function `f` that will be defined for you that you can call from within your function that describes the radioactive decay curve for the problem.

You should implement this function on your own machine. Open a new Canopy Python file and title it "radiationExposure.py". Complete your work inside this file. Test your code well in Canopy, and when you are convinced it is correct, cut and paste your definition into this tutor window.

**Test Cases to Test Your Code With. Be sure to test these on your own machine - and that you get the same output! - before running your code on this webpage!**

Test Cases

Assume that the curve function `f` is defined as follows:

```
def f(x):
    import math
    return 10*math.e**(math.log(0.5)/5.27 * x)
```

Test case 1:

```
>>> radiationExposure(0, 5, 1)
39.10318784326239
```

Test case 2:

```
>>> radiationExposure(5, 11, 1)
22.94241041057671
```

Test case 3:

```
>>> radiationExposure(0, 11, 1)
62.0455982538
```

Test case 4:

```
>>> radiationExposure(40, 100, 1.5)
0.434612356115
```

A note on these test cases: Your answers should be within 0.01 of the correct answer.

A Mathematical Note of Interest

The technique of finding the area under a curve is called *integration*. This comes to us from calculus. What we're doing in this problem is an approximation of finding the integral under the curve using the summation of rectangular areas known as a Riemann integral.

This approximation becomes more and more correct the smaller the width of the rectangles becomes.

So there you have it. If you've not learned calculus before, you've now got one of the basics - integration - covered!

```
def radiationExposure(start, stop, step):
    '''
    Computes and returns the amount of radiation exposed to between the start and stop times. Calls the
    function f (defined for you in the grading script) to obtain the value of the function at any point.

    start: integer, the time at which exposure begins
    stop: integer, the time at which exposure ends
    step: float, the width of each rectangle. You can assume that the step size will always partition the space evenly.

    returns: float, the amount of radiation exposed to between start and stop times.
    '''
    # FILL IN YOUR CODE HERE...
```

## A WORDGAME: HANGMAN

Note: Do not be intimidated by this problem! It's actually easier than it looks. We will 'scaffold' this problem, guiding you through the creation of helper functions before you implement the actual game.

For this problem, you will implement a variation of the classic wordgame Hangman. For those of you who are unfamiliar with the rules, you may read all about it here. In this problem, the second player will always be the computer, who will be picking a word at random.

In this problem, you will implement a **function**, called `hangman`, that will start up and carry out an interactive Hangman game between a player and the computer. Before we get to this function, we'll first implement a few helper functions to get you going.

For this problem, you will need the code files ps3_hangman.py and words.txt. Right-click on each and hit "Save Link As". **Be sure to save them in same directory.** Open and run the file `ps3_hangman.py` without making any modifications to it, in order to ensure that everything is set up correctly. By "open and run" we mean do the following:

- Go to Canopy. From the File menu, choose "Open".
- Find the file `ps3_hangman.py` and choose it.
- The template `ps3_hangman.py` file should now be open in Canopy. Click on it. From the Run menu, choose "Run File" (or simply hit Ctrl + R).

The code we have given you loads in a list of words from a file. If everything is working okay, after a small delay, you should see the following printed out:

```
Loading word list from file...

55909 words loaded.
```

If you see an `IOError` instead (e.g., "No such file or directory"), you should change the value of the `WORDLIST_FILENAME` constant (defined near the top of the file) to the **complete** pathname for the file`words.txt` (This will vary based on where you saved the file).

For example, if you saved `ps3_hangman.py` and `words.txt` in the directory "C:/Users/Ana/" change the line:

`WORDLIST_FILENAME = "words.txt"` to something like

```
WORDLIST_FILENAME = "C:/Users/Ana/words.txt"
```

**This folder will vary depending on where you saved the files.**

The file `ps3_hangman.py` has a number of already implemented functions you can use while writing up your solution. You can ignore the code between the following comments, though you should read and understand how to use each helper function by reading the docstrings:

```
# -----------------------------------

# Helper code

# You don't need to understand this helper code,

# but you will have to know how to use the functions

# (so be sure to read the docstrings!)

    .

    .

    .

# (end of helper code)

# -----------------------------------
```

You will want to do all of your coding for this problem within this file as well because you will be writing a program that depends on each function you write.

*Requirements*

Here are the requirements for your game:

1. The computer must select a word at random from the list of available words that was provided in`words.txt`. The functions for loading the word list and selecting a random word have already been provided for you in `ps3_hangman.py`.

2. The game must be interactive; the flow of the game should go as follows:

- At the start of the game, let the user know how many letters the computer's word contains.

- Ask the user to supply one guess (i.e. letter) per round.

- The user should receive feedback immediately after each guess about whether their guess appears in the computer's word.

- After each round, you should also display to the user the partially guessed word so far, as well as letters that the user has not yet guessed.

3. Some additional rules of the game:

- A user is allowed 8 guesses. Make sure to remind the user of how many guesses s/he has left after each round. Assume that players will only ever submit one character at a time (A-Z).

- A user loses a guess **only** when s/he guesses incorrectly.

- If the user guesses the same letter twice, do not take away a guess - instead, print a message letting them know they've already guessed that letter and ask them to try again.

- The game should end when the user constructs the full word or runs out of guesses. If the player runs out of guesses (s/he "loses"), reveal the word to the user when the game ends.

## Sample Output

The output of a winning game should look like this...

```
Loading word list from file...

55900 words loaded.
```

```
Welcome to the game, Hangman!

I am thinking of a word that is 4 letters long.

-------------

You have 8 guesses left.

Available letters: abcdefghijklmnopqrstuvwxyz

Please guess a letter: a

Good guess: _ a_ _

------------

You have 8 guesses left.

Available letters: bcdefghijklmnopqrstuvwxyz

Please guess a letter: a

Oops! You've already guessed that letter: _ a_ _

------------

You have 8 guesses left.

Available letters: bcdefghijklmnopqrstuvwxyz

Please guess a letter: s

Oops! That letter is not in my word: _ a_ _

------------

You have 7 guesses left.

Available letters: bcdefghijklmnopqrtuvwxyz

Please guess a letter: t

Good guess: ta_ t

------------
```

```
You have 7 guesses left.

Available letters: bcdefghijklmnopqruvwxyz

Please guess a letter: r

Oops! That letter is not in my word: ta_ t

------------

You have 6 guesses left.

Available letters: bcdefghijklmnopquvwxyz

Please guess a letter: m

Oops! That letter is not in my word: ta_ t

------------

You have 5 guesses left.

Available letters: bdefghijklmnopquvwxyz

Please guess a letter: c

Good guess: tact

------------

Congratulations, you won!
```

And the output of a losing game should look like this...

```
Loading word list from file...

55900 words loaded.

Welcome to the game Hangman!

I am thinking of a word that is 4 letters long
```

```
-----------

You have 8 guesses left

Available Letters: abcdefghijklmnopqrstuvwxyz

Please guess a letter: a

Oops! That letter is not in my word _ _ _ _

-----------

You have 7 guesses left

Available Letters: bcdefghijklmnopqrstuvwxyz

Please guess a letter: b

Oops! That letter is not in my word _ _ _ _

-----------

You have 6 guesses left

Available Letters: cdefghijklmnopqrstuvwxyz

Please guess a letter: c

Oops! That letter is not in my word _ _ _ _

-----------

You have 5 guesses left

Available Letters: defghijklmnopqrstuvwxyz

Please guess a letter: d

Oops! That letter is not in my word _ _ _ _

-----------

You have 4 guesses left

Available Letters: efghijklmnopqrstuvwxyz
```

```
Please guess a letter: e

Good guess: e_ _ e

-----------

You have 4 guesses left

Available Letters: fghijklmnopqrstuvwxyz

Please guess a letter: f

Oops! That letter is not in my word e_ _ e

-----------

You have 3 guesses left

Available Letters: ghijklmnopqrstuvwxyz

Please guess a letter: g

Oops! That letter is not in my word e_ _ e

-----------

You have 2 guesses left

Available Letters: hijklmnopqrstuvwxyz

Please guess a letter: h

Oops! That letter is not in my word e_ _ e

-----------

You have 1 guesses left

Available Letters: ijklmnopqrstuvwxyz

Please guess a letter: i

Oops! That letter is not in my word e_ _ e

-----------
```

```
Sorry, you ran out of guesses. The word was else.
```

On the next page, we'll break down the problem into logical subtasks, creating helper functions you will need to have in order for this game to work.

## HANGMAN PART 1: IS THE WORD GUESSED? (5/5 points)

Please read the Hangman Introduction before starting this problem. The helper functions you will be creating in the next three exercises are simply suggestions, but you DO have to implement them if you want to get points for this Hangman Problem Set. If you'd prefer to structure your Hangman program in a different way, feel free to redo this Problem Set in a different way. However, if you're new to programming, or at a loss of how to construct this program, we strongly suggest that you implement the next three helper functions before continuing on to Hangman Part 2.

We'll start by writing 3 simple functions that will help us easily code the Hangman problem. First, implement the function `isWordGuessed` that takes in two parameters - a string, `secretWord`, and a list of letters, `lettersGuessed`. This function returns a boolean - True if `secretWord` has been guessed (ie, all the letters of `secretWord` are in `lettersGuessed`) and False otherwise.

Example Usage:

```
>>> secretWord = 'apple'
>>> lettersGuessed = ['e', 'i', 'k', 'p', 'r', 's']
>>> print isWordGuessed(secretWord, lettersGuessed)
False
```

For this function, you may assume that all the letters in `secretWord` and `lettersGuessed` are lowercase.

```
def isWordGuessed(secretWord, lettersGuessed):
    '''
    secretWord: string, the word the user is guessing
    lettersGuessed: list, what letters have been guessed so far
    returns: boolean, True if all the letters of secretWord are in lettersGuessed;
      False otherwise
    '''
```

## PRINTING OUT THE USER'S GUESS (5/5 points)

Next, implement the function `getGuessedWord` that takes in two parameters - a string, `secretWord`, and a list of letters, `lettersGuessed`. This function returns a string that is comprised of letters and underscores, based on what letters in `lettersGuessed` are in `secretWord`. This shouldn't be too different from `isWordGuessed`!

Example Usage:

```
>>> secretWord = 'apple'
>>> lettersGuessed = ['e', 'i', 'k', 'p', 'r', 's']
>>> print getGuessedWord(secretWord, lettersGuessed)
'_ pp_ e'
```

When inserting underscores into your string, it's a good idea to add at least a space after each one, so it's clear to the user how many unguessed letters are left in the string (compare the readability of `____` with `_ _ _ _ `). This is called *usability* - it's very important, when programming, to consider the usability of your program. If users find your program difficult to understand or operate, they won't use it!

For this problem, you are free to use spacing in any way you wish - our grader will only check that the letters and underscores are in the proper order; it will not look at spacing. We do encourage you to think about usability when desigining

For this function, you may assume that all the letters in `secretWord` and `lettersGuessed` are lowercase.

```
def getGuessedWord(secretWord, lettersGuessed):
    '''
    secretWord: string, the word the user is guessing
    lettersGuessed: list, what letters have been guessed so far
    returns: string, comprised of letters and underscores that represents
      what letters in secretWord have been guessed so far.
    '''
```

## PRINTING OUT ALL AVAILABLE LETTERS (5/5 points)

Next, implement the function `getAvailableLetters` that takes in one parameter - a list of letters, `lettersGuessed`. This function returns a string that is comprised of lowercase English letters - all lowercase English letters that are **not** in `lettersGuessed`.

Example Usage:

```
>>> lettersGuessed = ['e', 'i', 'k', 'p', 'r', 's']
>>> print getAvailableLetters(lettersGuessed)
abcdfghjlmnoqtuvwxyz
```

Note that this function should return the letters in alphabetical order, as in the example above.

For this function, you may assume that all the letters in `lettersGuessed` are lowercase.
**Hint:** You might consider using `string.ascii_lowercase`, which is a string comprised of all lowercase letters:

```
>>> import string
>>> print string.ascii_lowercase
abcdefghijklmnopqrstuvwxyz
```

```
def getAvailableLetters(lettersGuessed):
    '''
    lettersGuessed: list, what letters have been guessed so far
    returns: string, comprised of letters that represents what letters have not
      yet been guessed.
    '''
```

## HANGMAN PART 2: THE GAME **(15/15 points)**

Now you will implement the function `hangman`, which takes one parameter - the `secretWord` the user is to guess. This starts up an interactive game of Hangman between the user and the computer. Be sure you take advantage of the three helper functions, `isWordGuessed`, `getGuessedWord`, and `getAvailableLetters`, that you've defined in the previous part.

### *Hints:*

- You should start by noticing where we're using the provided functions (at the top of `ps3_hangman.py`) to load the words and pick a random one. Note that the functions `loadWords` and `chooseWord` should only be used on your local machine, not in the tutor. When you enter in your solution in the tutor, you only need to give your `hangman` function.
- Consider using `lower()` to convert user input to lower case. For example:
- `guess = 'A'`

```
guessInLowerCase = guess.lower()
```

- Consider writing additional helper functions if you need them!

- There are four important pieces of information you may wish to store:

  1. `secretWord`: The word to guess.
  2. `lettersGuessed`: The letters that have been guessed so far.
  3. `mistakesMade`: The number of incorrect guesses made so far.
  4. `availableLetters`: The letters that may still be guessed. Every time a player guesses a letter, the guessed letter must be removed from `availableLetters` (and if they guess a letter that is not in`availableLetters`, you should print a message telling them they've already guessed that - so try again!).

### Sample Output

The output of a winning game should look like this...

```
        Loading word list from file...
        55900 words loaded.
        Welcome to the game, Hangman!
        I am thinking of a word that is 4 letters long.
        -------------
        You have 8 guesses left.
        Available letters: abcdefghijklmnopqrstuvwxyz
        Please guess a letter: a
        Good guess: _ a_ _
        ------------
        You have 8 guesses left.
        Available letters: bcdefghijklmnopqrstuvwxyz
        Please guess a letter: a
        Oops! You've already guessed that letter: _ a_ _
        ------------
        You have 8 guesses left.
        Available letters: bcdefghijklmnopqrstuvwxyz
        Please guess a letter: s
        Oops! That letter is not in my word: _ a_ _
        ------------
        You have 7 guesses left.
        Available letters: bcdefghijklmnopqrtuvwxyz
        Please guess a letter: t
        Good guess: ta_ t
        ------------
        You have 7 guesses left.
        Available letters: bcdefghijklmnopqruvwxyz
        Please guess a letter: r
        Oops! That letter is not in my word: ta_ t
        ------------
        You have 6 guesses left.
        Available letters: bcdefghijklmnopquvwxyz
        Please guess a letter: m
        Oops! That letter is not in my word: ta_ t
        ------------
        You have 5 guesses left.
        Available letters: bcdefghijklnopquvwxyz
        Please guess a letter: c
        Good guess: tact
        ------------
        Congratulations, you won!
```

And the output of a losing game should look like this...

```
        Loading word list from file...
        55900 words loaded.
        Welcome to the game Hangman!
        I am thinking of a word that is 4 letters long
        -----------
        You have 8 guesses left
        Available Letters: abcdefghijklmnopqrstuvwxyz
        Please guess a letter: a
        Oops! That letter is not in my word: _ _ _ _
        -----------
        You have 7 guesses left
        Available Letters: bcdefghijklmnopqrstuvwxyz
        Please guess a letter: b
        Oops! That letter is not in my word: _ _ _ _
        -----------
        You have 6 guesses left
        Available Letters: cdefghijklmnopqrstuvwxyz
        Please guess a letter: c
        Oops! That letter is not in my word: _ _ _ _
        -----------
        You have 5 guesses left
        Available Letters: defghijklmnopqrstuvwxyz
        Please guess a letter: d
        Oops! That letter is not in my word: _ _ _ _
        -----------
        You have 4 guesses left
        Available Letters: efghijklmnopqrstuvwxyz
        Please guess a letter: e
        Good guess: e_ _ e
        -----------
        You have 4 guesses left
        Available Letters: fghijklmnopqrstuvwxyz
        Please guess a letter: f
        Oops! That letter is not in my word: e_ _ e
        -----------
        You have 3 guesses left
        Available Letters: ghijklmnopqrstuvwxyz
        Please guess a letter: g
        Oops! That letter is not in my word: e_ _ e
        -----------
        You have 2 guesses left
        Available Letters: hijklmnopqrstuvwxyz
        Please guess a letter: h
        Oops! That letter is not in my word: e_ _ e
        -----------
        You have 1 guesses left
        Available Letters: ijklmnopqrstuvwxyz
        Please guess a letter: i
        Oops! That letter is not in my word: e_ _ e
        -----------
        Sorry, you ran out of guesses. The word was else.
```

Note that if you choose to use the helper functions `isWordGuessed`, `getGuessedWord`, or `getAvailableLetters`, you do not need to paste your definitions in the box. We have supplied our implementations of these functions for your use in this part of the problem. If you use additional helper functions, you will need to paste those definitions here.
Your function should include calls to `raw_input` to get the user's guess.

```python
def hangman(secretWord):
    '''
    secretWord: string, the secret word to guess.
    Starts up an interactive game of Hangman.
    * At the start of the game, let the user know how many
      letters the secretWord contains.
    * Ask the user to supply one guess (i.e. letter) per round.
    * The user should receive feedback immediately after each guess
      about whether their guess appears in the computers word.
    * After each round, you should also display to the user the
      partially guessed word so far, as well as letters that the
      user has not yet guessed.
    Follows the other limitations detailed in the problem write-up.
    '''
```