Whenever attempting to solve a problem with code, the very first thing you need to do is understand the problem without code. If you do not yet have a solid strategy for approaching the problem using your human problem-solving skills, adding code to the mix will only make it more difficult! Ideally, you do not want to bring code into the equation until the final step.

It is often necessary to break a problem apart into smaller problems before trying to solve them, and it may even be helpful to solve the problem by using a pencil and paper! Pay attention to the different actions and decisions you make as you solve it. Write down every step and every thought you have! Then you might be in a good place to try to figure out what basic code tools you've seen in action would be best suited to mimic the things you did with your brain.

- Do you need to remember or track any values as you solve it? (create a new variable)
- Do you choose to do one action or another based on different situations? (if/else statement with comparisons)
- Do you repeat an action or assessment on each item/letter/number in a pre-defined collection? (for loop)
- Do you want something to continue happening until a certain point is reached? (while loop)

Before coding out any solutions to the below problems, make sure to write a step-by-step breakdown of the problem-solving process. Once complete, then begin implementing a solution in code.

1. Write a function that takes in a list of programming languages and prompts the user for their favorite programming language. If the user's favorite programming language exists in the list, return it and print the returned result to the console.
2. Write a function that takes in a minimum number and maximum number, and return a random number between the minimum and maximum range.
3. Write a function that takes in a word and return the reversal of that word.
    a. Example: "packers" will be returned as "srekcap"
4. Write a function that prints every number from 100 to 1 (descending).
    a. If the number is divisible by 4, print "Banana" instead of the number
    b. If the number is divisible by 7, print "Flamingo" instead of the number
    c. If the number is divisible by 4 and 7, print "Flamingo -Banana!"
5. Write a function that takes in a list of numbers. Return a new list that contains only the elements that are less than 5. Print to the console the contents of the returned list.
    a. [1, 2, 3, 7, 8, 9, 45, 134, 43, 2, 3, 1, 6, 7, 5, 4]
        i. Bonus for fun: No duplicates in the new list.
6. Write a function that prompts the user for their name and age, and then prints out the user's name and the year they will turn 100 years old.
7. Write a function that returns a list that contains only the elements that are common between the lists (without duplicates). Make sure your program works on two lists of different sizes. Examples of two lists:
        i. a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
        ii. b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
        iii. Bonus for fun: Randomly generate two lists to test this.
8. Write a function that takes in two words and determines if the two words are an anagram of each other.

    a. An anagram is a word or phrase that is formed by rearranging the letters of another word or phrase.
    b. Assume any word that is passed in is a word that exists in the English language
    c. If the two words are an anagram, return true. Otherwise, return false.

9. Write a function that takes in a phrase and reverse each word inside the string, but keeps the same order of the phrase.
    a. “Hello world I am a programmer”
    b. “olleh dlrow I ma a remmargorp”
10. Print downward Pyramid Pattern with Star (asterisk).
    a.
    ```
    *****
     ****
      ***
       **
        *
    ```
    b. Bonus for fun: Allow user input to decide how many stars on the first row.