- Why does the program skip a line of code asking for text right after an input of numbers?

```
System.out.print("enter a number: ");
int number = scanner.nextInt();
System.out.print("enter text: ");
String line = scanner.nextLine();
System.out.println("you entered: "+number+" / "+line);
```

```
___42_
_hi_there___
```

  - user's input is a continuous stream
    - each use of `Scanner` to read reads the next bit, blocking if there's nothing available
  - `Scanner` behavior
    - `nextInt()`, `nextDouble()`, etc skip initial whitespace, then read up to (but don't consume) the next whitespace, returning what was read as an `int`/`double`/etc
    - `nextLine()` reads up to and consumes the next newline, returning everything read up to (but not including) the newline itself

- What does the text book mean by pseudocode?
  - pseudocode is English, written in a Java-like structure

```
Ask the user for the initial investment
Read the user's response
Ask the user for the interest rate
Read the user's response
years = 0
while years < 5:
    years = years + 1
    Compute interest = value * interest rate
    Add the interest to the value
    Display the value
```

  - it contains all the steps needed (though not necessarily all the details of those steps) and captures control structures (conditionals and loops)
    - allows expression of the key ideas without getting bogged down in the details of the code itself

Combinations of Loops

- single loop
  - use when the task involves repeating something (which may involve multiple steps)
- one loop after another
  - use where there are separate tasks (one can be completed before starting the next), both of which involve repetition
- nested loops – a loop inside another loop
  - use when the task being repeated itself involves repetition
    - i.e. "what repeats?" involves repetition
    - i.e. "what changes?" involves more than one thing, but sometimes a loop variable stays the same and sometimes it changes


```
int a, b;
for ( a = 0 ; a <= 5 ; a++ ) {
   for ( b = 0 ; b < a ; b++ ) {
     System.out.print("*");
   }
   System.out.println("!");
}
```

Program Development With Loops

Loop forms –

- repeat until condition
- repeat as long as condition
- repeat n times
- repeat from a to b by interval c

Loop patterns –

- *counting loop* – aka repeat n times
- *error-checking user input* – when user is asked to enter a value but only some values of that type are acceptable
- *fencepost pattern* – when the determination of whether to continue or not is in the middle of the repeated unit
- *accumulator pattern* – when answer is built up over the repetitions e.g. summing a bunch of numbers
  - *counting-up-things pattern* – when the answer is a number of repetitions
- *best-so-far pattern* – when the answer is the value from one of the repetitions e.g. find minimum or maximum of a bunch of numbers