1. *Number Data Types*
   a. What data type is the number 2? How about 20? 200? Keep adding zeros and watch the data type change until it reaches `BigInteger`. Then do the same for 2.0.
   b. Declare a variable `x` of type `def` and assign it the sum of 1 and 1.5. What is the resulting data type?
   c. What do you get when you divide 5 by 2? What is the resulting data type? If you wanted to do integer division (no remainder), what method would you call?
2. *Wrapper Classes*
   From the associated wrapper classes, find the min and max values for the Java primitives: `byte`, `short`, `int`, `long`, `float`, `double`.
3. *2s Complement*
   Create a `byte` variable with its maximum value. What do you get when you add 1 to it?
4. *Strings and GroovyStrings*
   a. How many characters are in the string "Hello, Groovy!"?
   b. Define a string variable containing a name. Print a hello statement with your name using string concatenation, then using a Groovy string.
   c. Demonstrate that "racecar" is a palindrome by comparing it to its reverse. Do the same with "Bob", removing case sensitivity first.
   d. Define a string variable containing the sentence, "Hello, World. How are you?". Split the sentence into an array using the `split` method. Count the number of words. Do the same using the `tokenize` method.
   e. Using the same sentence, use array notation (square brackets) to print the substring "World".
   f. Use array notation to print the last word, but reversed.
5. *Prime Numbers*
   Write a method called `isPrime` that takes an integer argument and returns a boolean. Determine whether the number is prime by dividing it by all numbers from 2 up to one less than the number. That limit is too high, of course. How high do you have to check to be sure whether you've gone far enough?

1. *Sorting Strings*
   Create a list of strings. Sort them alphabetically. Sort them by length. Sort them by length in descending order.
   Advanced: Sort by length, then sort equal length strings alphabetically
2. *Processing a list of numbers*
   Create a list of numbers. Add them together. First double each number, then add them up. Compute their average.
3. *Reading a web page*
   Using the Groovy JDK, access your home page and display the source code. Print the length of each line of the home page.
4. *Closures as a filter*
   Create a list of numbers. Print all elements greater than zero.
5. *Multi-line strings*
   Make a multi-line string. Compute the number of vowels on each line.
6. *Padded binary output*
   Print the numbers from 0 to 15 in binary (use Java's `Integer.toBinaryString()` method). Use a method in String from the Groovy JDK to make all the output values have four digits.

1. *Encode and decode*
   i. Create two strings, one for a username and one for a password. Concatenate them together, separated by a colon. Use a method from the Groovy JDK to convert the resulting String to a byte array. Then use the `encodeBase64` method on byte array to create an encoded string.
   ii. Decode the string by using the `decodeBase64` method, and using the result as an argument to the String constructor. Use the split method to return the original username and password.
2. *Sorting a list*
   Create a class called `Course`, with a `String` attribute called `name` and an `int` attribute called `days`. Create a list of four course instances, where at least two have the same number of days. Sort the list by number of days. Then, sort the list by days, but when the days are equal, sort by name.
3. *Operator overloading*

i. Create a class called `Money` with a double `amount` and a `String` `currency` (like USD or EUR). Implement a `plus` method that checks that the currencies are the same and, if so, returns a new `Money` instance with the sum of the amounts and the correct currency. Write a similar `minus` method.
ii. Write a `MoneyTest` class in Groovy that uses + and - and verifies that they work properly.