Hi there me,

This is level one so we will be using jshell to type java …

* Type the word “jshell” in order to start jshell in the cmd

**C:\Users\USER>jshell**

* This is how we use the print statement to type our console

**System.out.print("Hello World");**

* To exit the jshell you may do as follows,

**jshell> /exit**

* Creating a variable and messing with it…

**int my\_first\_variable = 10;**

these types of statements are called declaration statements.

These are used to define a variable with the data type that stores and stores a value in it.

**code**

System.out.print(my\_first\_variable);

**Console**

10

We may also do crazy stuff like this …

**Code**

Int cat = 10 + 5;

System.out.print(cat);

**Console**

15

We could have a list of variables we created within the jshell by the following command,

**jshell> /var**

* Finding the min and max value that can be assigned to an int variable can be found this way

**Code**

System.out.print("The smallest integer that can be assigned to a int variable would be: " + Integer.MIN\_VALUE);

System.out.print("The max integer that can be assigned to an int variable would be: " + Integer.MAX\_VALUE);

System.out.print("The amount of integers that can be assigned to an int variable would be: " + (Integer.MAX\_VALUE - Integer.MIN\_VALUE))

System.out.print(“The integer value range is from (” + Integer.MIN\_VALUE + “ to ” + Integer.MAX\_VALUE + “)”);

**Console**

The smallest integer that can be assigned to a int variable would be: -2147483648

The max integer that can be assigned to a int variable would be: -2147483647

The amount of integers that can be assigned to an int variable would be: -1

The integer value range is from (-2147483648 to 2147483647)

This unusual error in their output in the console happens due to the behavior of integer overflow in Java.

The maximum value, when it overflows, wraps around to the minimum value, and just continues processing without an error.

The minimum value, when it underflows, wraps around to the maximum value, and continues processing.

This is not usually behavior you really want, and as a developer, you need to be aware that this can happen, and choose the appropriate data type.

int cat = Integer.MAX\_VALUE + 1

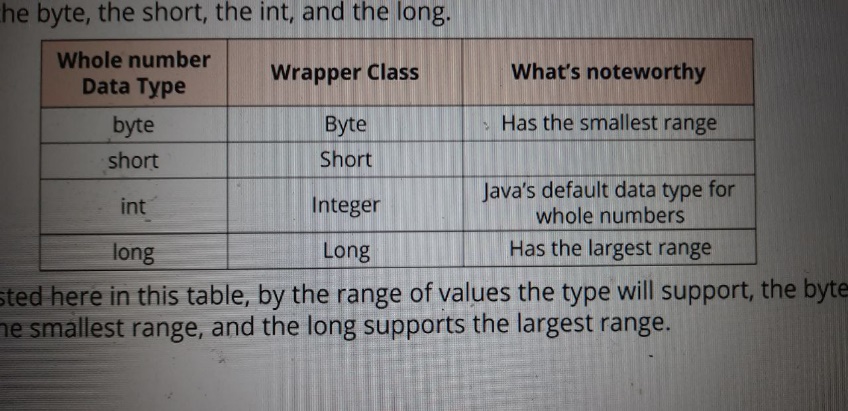
int cat = 2147483647 + 1

The both statements above will get compiled without creating an error while creating an overflow yet the following statement will create an error

int cat = 2147483648

* A computer screen with white text

  Description automatically generatedHow to make large unclear numbers clear within the code…,



* **Mind when using other data types when storing whole numbers…**

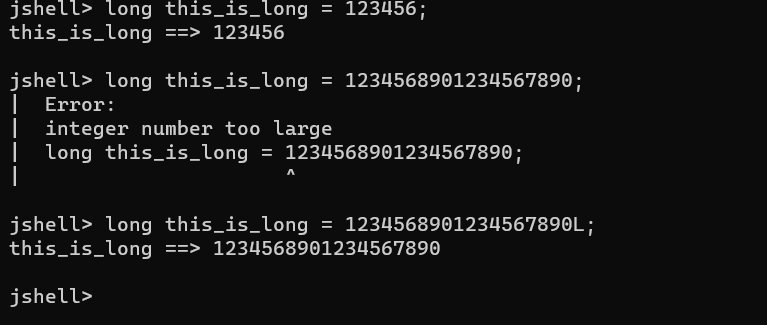
When using long data type to store data type keep in mind to use the suffix “L” in front of the number you are inserting …(this is a one place what java language wont be case sensitive… which means that you can use both “L” and “l”…but the best is to “L” because “l” can often be misunderstood as one )

**Code**



The code will work as planned if the assigning value is within the range of int…but will cause and error is exceeds that…

**Code**



But this is not a necessity when dealing with other data types other than int

Ex- short, bye (suffices are not defined to those data types )

* **There are ways of having multiple code statements in a single line of code in java…**

(mind that unlike in Python there is a “;” indicating the end of the statement

So having multiple statements each ending with a “;” is perfectly valid syntax in java)

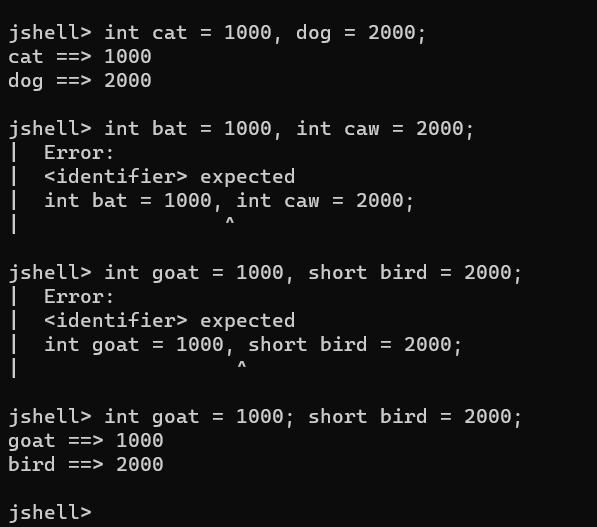
**Code**

**A black screen with white text

Description automatically generated**

* **We may also create multiple variables in a single code statement of code.**

(keep in mind that in this case both of the variables should be in same data type)

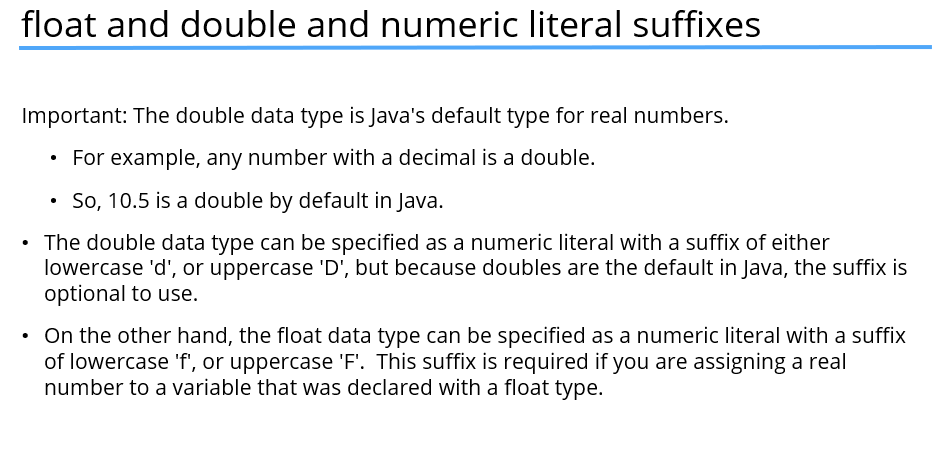


* **Dealing with decimal point numbers when creating variables**

java has two data types when it comes to dealing with decimal point numbers.

EX- float

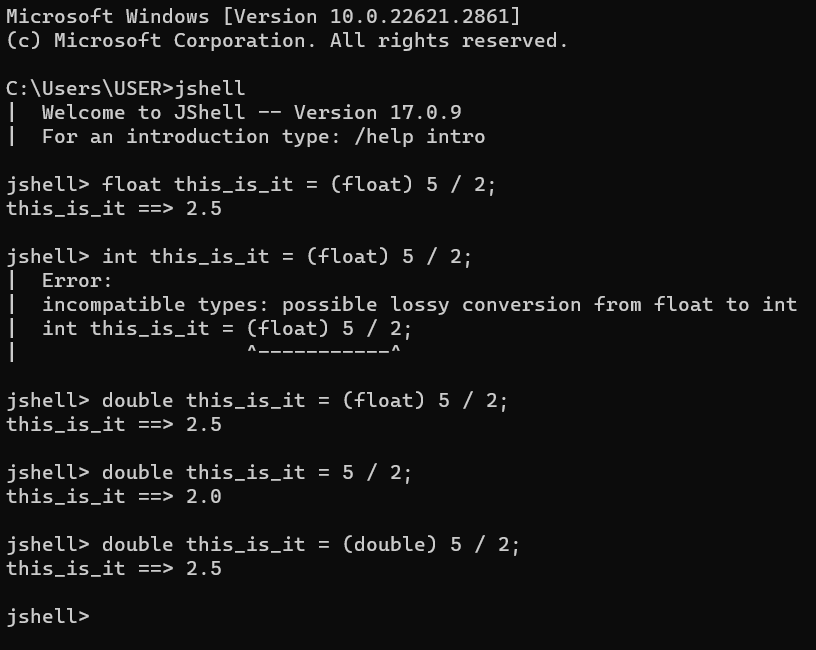
decimal

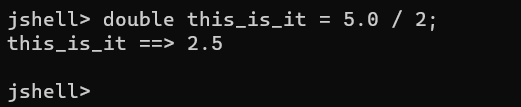


A computer screen with white text

Description automatically generated

More messy stuff when it comes to dealing with decimal point numbers.





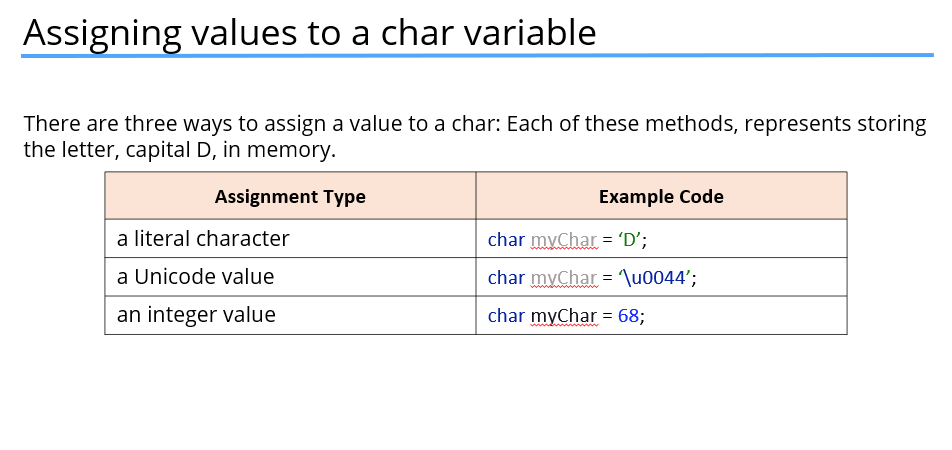
* **Here we start dealing with char variables…**

mind that these types of variables can only store one type of character at a time.

Though there are multiple ways of accessing these values that we plan on storing in the char variables.

A screenshot of a computer program

Description automatically generated



* **Strings**

Always remember to start a string with the first letter capital….

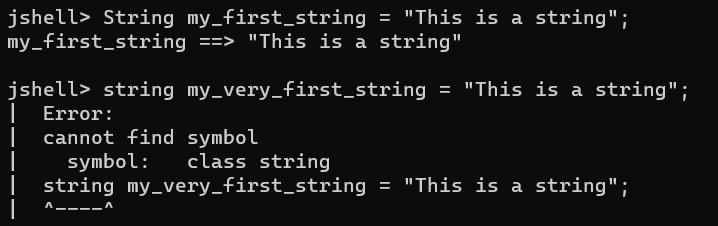
Ex- string will throw an error …

String will get executed as expected.

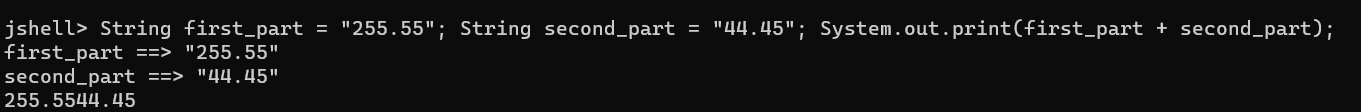
A computer screen shot of a black screen

Description automatically generated

Some more playing around with string (a little like char but not really)

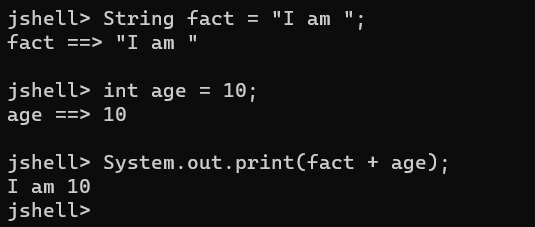


Concatenating strings…



* **Mind that “+” in java work a little different than it does in python…**

Adding a string to an integer will not throw an error as like in python but combines the two variables.



In this case the integer value in the variable “age” is converted to a string and finally the two strings get concatenated.

Mind that same as in python string is considered as an immutable data type.

There is a class called StringBuilder while act a little like a mutable string yet it doesn’t give all the features (using the “+” with integer values and more) that a string class does.

* **Operators, Operands and expression…**

To explain the concept let’s consider the following equation,

**Code**

int my\_int = 20;

int the\_total = (20 + 30) \* my\_int;

so when considering the second line of code ,

operators : =, +, \*

operands : 20, 30, my\_int

expression: (20 + 30) \* my\_int

point: the “=” is also called the assignment operator…

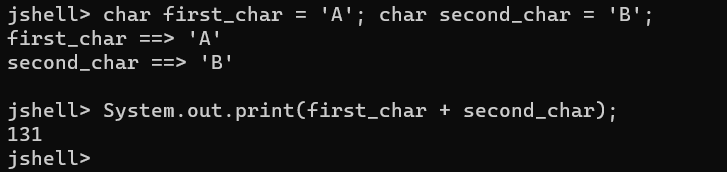
* **Using the “+” operator on char variables.**

You might remember that we said chars are stored as 2 byte numbers in memory.

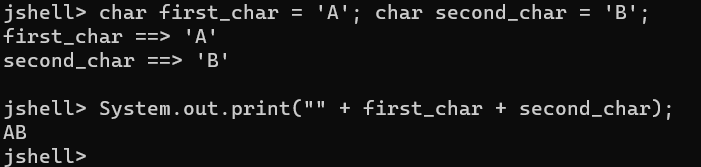
When you use the plus operator with chars, it is this numbers in memory that get added together.

Char values don’t get concatenated.

As a result, the final out come may deviate from the expected outcome as follows…



We can over come this problem as follows.

****

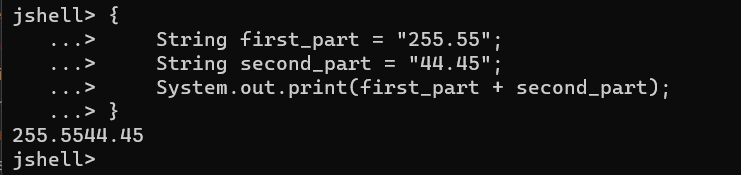
Notice that there is an empty string the very biggening on what needs to be printed.

As we know any datatype can be concatenated to a string in java and when we try concatenating a different data type with a string the other data type will get converted to a string before getting concatenated.

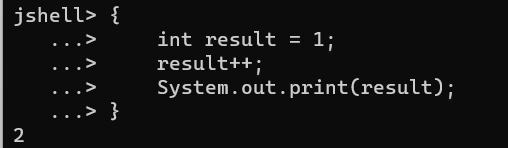
Show above is a good evidence for that …

Notice how the two chars get converted to strings and give the expected result.

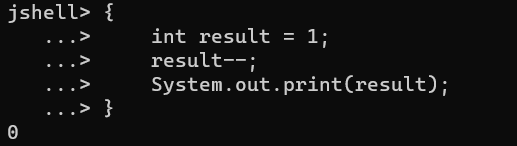
* **Typing code with multiple lines using jshell (Typing within {})**



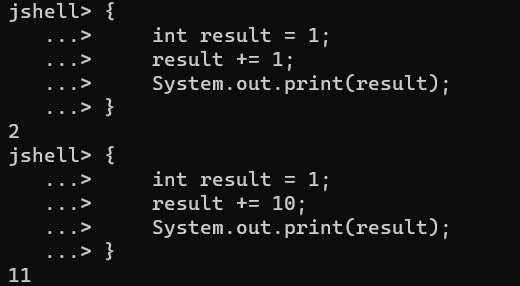
* **Incrementing a value by 1 (using “++”)**

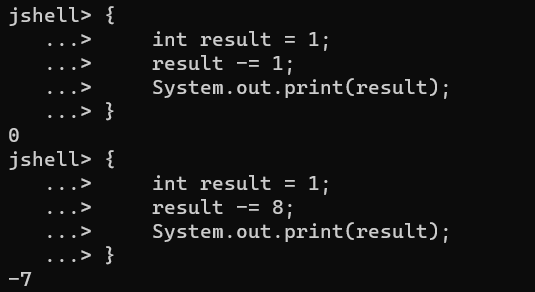


* **Decrementing by 1 (using “--”)**

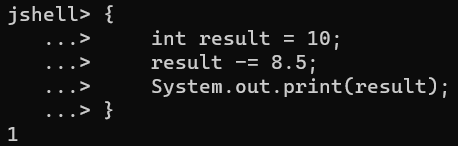


* **Both “++” and “--” are called shorthand operators**
* **Adding/Subtracting values from the existing values of a variable (using compound operators)**





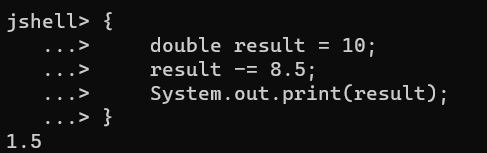
* **Trying using compound assignment operators with non-identical data types in java.**



When considering the compound operators…,

X -= Y is,

X = (data type of X) (X - Y)



* All ++, --, +=, -= are considered as abbreviating operators…  
  this is because they only need one operand in order to execute…