# CSCI 111, Bonus Lab 1
## Add Variables to the RPN Calculator

**Individual work:** All work must be your own. Do not share code with anyone other than the instructor and teaching assistants. This includes looking over shoulders at screens with the code open. You may discuss ideas, algorithms, approaches, *etc.* with other students but NEVER actual code.

**RPN with variables:** Add the ability to define variables to the RPN calculator. You will need to have a working RPN calculator for this!

Variables are defined with the `def` operator. For example, in this session we find the sine of 3.14159, and then we define the variable `pi` to have the value 3.14159 and take the sine of `pi` to get the same value:

```
RPN>>> 3.14159 sin
Stack: [2.65358979335273e-06] Tokens: [3.14159, 'sin']
Stack: [2.65358979335273e-06, 3.14159] Tokens: ['sin']
[2.65358979335273e-06, 2.65358979335273e-06]
RPN>>> clr
Stack: [2.65358979335273e-06, 2.65358979335273e-06] Tokens: ['clr']
[]
RPN>>> 3.14159 pi def
Stack: [] Tokens: [3.14159, 'pi', 'def']
Stack: [3.14159] Tokens: ['pi', 'def']
Stack: [3.14159, 'pi'] Tokens: ['def']
[]
RPN>>> pi sin
Stack: [] Tokens: ['pi', 'sin']
Stack: ['pi'] Tokens: ['sin']
[2.65358979335273e-06]
RPN>>>
```

**The store:** Variables and their values are kept in a global variable called `store`.[1] The store is a dictionary mapping variables to values.

**Parsing:** You will have to account for variables in your tokenizer. Variables are not keywords, but consist of all alphabetic letters, and are represented after tokenizing just by the string, like keywords.

**Processing:** When interpreting the tokenized list variables are passed onto the stack just like numbers.

When the new keyword `def` is encountered, two items are popped from the stack, the variable should be on top, and its value underneath. The `store` dictionary stores the value under the variable as key.

When an item is removed from the stack, its value is found. The value of a number is the number, the value of a string (variable) is the value found in the `store`.

**Errors:** You do not have to handle any errors, such as a variable not defined, etc.

[1] Think "storehouse" and not "marketplace."