Review
# Changing Variable Types

<table>
<thead>
<tr>
<th>Int to double?</th>
<th>Int to String?</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int x = 5;</code></td>
<td><code>int x = 5;</code></td>
</tr>
<tr>
<td><code>double xDbil = x;</code></td>
<td><code>String xStr = &quot;&quot; + x</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String to int?</th>
<th>String to double?</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>String xStr = &quot;5&quot;;</code></td>
<td><code>String xStr = &quot;5.6&quot;;</code></td>
</tr>
<tr>
<td><code>int x = Integer.parseInt(x);</code></td>
<td><code>double x = Double.parseDouble(xStr);</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Casting double to int</th>
<th>GObject to GRect</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>double x = 5.2;</code></td>
<td><code>GObject obj = getElementAt(5, 2);</code></td>
</tr>
<tr>
<td><code>int y = (int)x;</code></td>
<td><code>GRect objRect = (GRect)obj;</code></td>
</tr>
</tbody>
</table>

**int to char**

```java
int diff = 'C'-'A';
char next = (char)('a' + diff);
```
Where are we?

- Karel the Robot
- Java
- Console Programs
- Graphics Programs
- Text Processing
- **Data Structures**
- Defining our own Variable Types
- GUIs

Piech, CS106A, Stanford University
/Review
What does this say?

Puzzle in Gold Bug by Edgar Allan Poe
A **cryptogram** is a puzzle in which a message is encoded by replacing each letter in the original text with some other letter. The substitution pattern remains the same throughout the message. Your job in solving a cryptogram is to figure out this correspondence.

One of the most famous cryptograms was written by Edgar Allan Poe in his short story “The Gold Bug.”

In this story, Poe describes the technique of assuming that the most common letters in the coded message correspond to the most common letters in English, which are E, T, A, O, I, N, S, H, R, D, L, and U.

Edgar Allan Poe (1809-1849)
### Letter Frequency

<table>
<thead>
<tr>
<th>Letter</th>
<th>Count</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>445.2</td>
<td>12.49%</td>
</tr>
<tr>
<td>T</td>
<td>330.5</td>
<td>9.28%</td>
</tr>
<tr>
<td>A</td>
<td>286.5</td>
<td>8.04%</td>
</tr>
<tr>
<td>O</td>
<td>272.3</td>
<td>7.64%</td>
</tr>
<tr>
<td>I</td>
<td>269.7</td>
<td>7.57%</td>
</tr>
<tr>
<td>N</td>
<td>257.8</td>
<td>7.23%</td>
</tr>
<tr>
<td>S</td>
<td>232.1</td>
<td>6.51%</td>
</tr>
<tr>
<td>R</td>
<td>223.8</td>
<td>6.28%</td>
</tr>
<tr>
<td>H</td>
<td>180.1</td>
<td>5.05%</td>
</tr>
<tr>
<td>L</td>
<td>145.0</td>
<td>4.07%</td>
</tr>
<tr>
<td>D</td>
<td>136.0</td>
<td>3.82%</td>
</tr>
<tr>
<td>C</td>
<td>119.2</td>
<td>3.34%</td>
</tr>
<tr>
<td>U</td>
<td>97.3</td>
<td>2.73%</td>
</tr>
<tr>
<td>M</td>
<td>89.5</td>
<td>2.51%</td>
</tr>
<tr>
<td>F</td>
<td>85.6</td>
<td>2.40%</td>
</tr>
<tr>
<td>P</td>
<td>76.1</td>
<td>2.14%</td>
</tr>
<tr>
<td>G</td>
<td>66.6</td>
<td>1.87%</td>
</tr>
<tr>
<td>W</td>
<td>59.7</td>
<td>1.68%</td>
</tr>
<tr>
<td>Y</td>
<td>59.3</td>
<td>1.66%</td>
</tr>
<tr>
<td>B</td>
<td>52.9</td>
<td>1.48%</td>
</tr>
<tr>
<td>V</td>
<td>37.5</td>
<td>1.05%</td>
</tr>
<tr>
<td>K</td>
<td>19.3</td>
<td>0.54%</td>
</tr>
<tr>
<td>X</td>
<td>8.4</td>
<td>0.23%</td>
</tr>
<tr>
<td>J</td>
<td>5.7</td>
<td>0.16%</td>
</tr>
<tr>
<td>Q</td>
<td>4.3</td>
<td>0.12%</td>
</tr>
<tr>
<td>Z</td>
<td>3.2</td>
<td>0.09%</td>
</tr>
</tbody>
</table>

By **Peter Norvig**

Head of Google Research

Former CS221 Instructor

Based on 3.8 trillion letters
Arrays: A new variable type that is an object that represents an ordered, homogeneous list of data. Arrays have many elements that you can access using indices.

#majorkey of the day
```java
int[] numbers = new int[5];
```

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Creating Arrays

The type is an array of ints

```
int[] numbers = new int[5];
```

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Creating Arrays

The array has the name “numbers”

```java
int[] numbers = new int[5];
```

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Creating Arrays

It's an object so it needs you to make a new one.

```java
int[] numbers = new int[5];
```

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Creating Arrays

This array has five ints

```java
int[] numbers = new int[5];
```

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Creating Arrays

type[] name = new type[length];

int[] numbers = new int[5];

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Java automatically initializes elements to 0.
int[] numbers = new int[5];
int[] numbers = new int[5];
int[] numbers = new int[5];
```java
int[] numbers = new int[5];
```
int[] numbers = new int[5];

stack

run

numbers

heap

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
You can create arrays of any variable type. For example:

```java
double[] results = new double[5];

String[] names = new String[3];

boolean[] switches = new boolean[4];

GRect[] rects = new GRect[5];
```

- Java initializes each element of a new array to its default value, which is 0 for int and double, \'\0\' for char, false for boolean, and null for objects.
You can create arrays of any variable type. For example:

```java
char[] oldSchoolString = new char[5];
```

- Java initializes each element of a new array to its default value, which is 0 for int and double, '\0' for char, false for boolean, and null for objects.
# Data Structures

<table>
<thead>
<tr>
<th>Operation</th>
<th>Strings</th>
<th>Arrays</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make a new one</td>
<td><code>String str = “abc”;</code></td>
<td></td>
</tr>
<tr>
<td>Get length?</td>
<td><code>str.length()</code></td>
<td></td>
</tr>
<tr>
<td>Get element?</td>
<td><code>str.charAt(i)</code></td>
<td></td>
</tr>
<tr>
<td>Set element?</td>
<td><em>Not allowed</em></td>
<td></td>
</tr>
<tr>
<td>Loop?</td>
<td><code>for(int i = 0; i &lt; str.length(); i++)</code></td>
<td></td>
</tr>
</tbody>
</table>
## Data Structures

<table>
<thead>
<tr>
<th>Operation</th>
<th>Strings</th>
<th>Arrays</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make a new one</td>
<td><code>String str = “abc”</code>;</td>
<td><code>int arr = new int[5];</code></td>
</tr>
<tr>
<td>Get length?</td>
<td><code>str.length()</code></td>
<td><code>arr.length</code></td>
</tr>
<tr>
<td>Get element?</td>
<td><code>str.charAt(i)</code></td>
<td><code>arr[i]</code></td>
</tr>
<tr>
<td>Set element?</td>
<td><em>Not allowed</em></td>
<td><code>arr[i] = 5;</code></td>
</tr>
<tr>
<td>Loop?</td>
<td><code>for(int i = 0; i &lt; str.length(); i++)</code></td>
<td><code>for(int i = 0; i &lt; arr.length; i++)</code></td>
</tr>
</tbody>
</table>
name[index] // get element at index

• Like Strings, indices go from 0 to the array's length - 1.

    for (int i = 0; i < numbers.length; i++) {
        println(numbers[i]);
    }
    println(numbers[9]); // exception
    println(numbers[-1]); // exception

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>
name[index] = value;  // set element at index
**Setting values**

```java
name[index] = value;     // set element at index

• Like Strings, indices go from 0 to the array's length - 1.

```java
int[] numbers = new int[7];
for (int i = 0; i < 7; i++) {
    numbers[i] = i;
}

numbers[8] = 2;        // exception
numbers[-1] = 5;       // exception
```

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>
Q: What are the contents of numbers after executing this code?

```java
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 7;
numbers[6] = 5;
int x = numbers[1];
numbers[x] = 2;
numbers[numbers[numbers[4]]] = 9;
```

// 0 1 2 3 4 5 6 7
A. {0, 3, 0, 2, 7, 0, 5, 9}
B. {0, 3, 0, 0, 7, 0, 5, 0}
C. {3, 3, 5, 2, 7, 4, 5, 0}
D. {0, 3, 0, 2, 7, 6, 4, 4}
Q: What are the contents of numbers after executing this code?

```java
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 7;
numbers[6] = 5;
int x = numbers[1];
numbers[x] = 2;
numbers[numbers[numbers[4]]] = 9;
```

// 0 1 2 3 4 5 6 7
A. {0, 3, 0, 2, 7, 0, 5, 9}
B. {0, 3, 0, 0, 7, 0, 5, 0}
C. {3, 3, 5, 2, 7, 4, 5, 0}
D. {0, 3, 0, 2, 7, 6, 4, 4}
Q: What are the contents of `numbers` after executing this code?

```java
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 7;
numbers[6] = 5;
int x = numbers[1];
numbers[x] = 2;
numbers[numbers[numbers[4]]] = 9;
```

A. `{0, 3, 0, 2, 7, 0, 5, 9}`
B. `{0, 3, 0, 0, 7, 0, 5, 0}`
C. `{3, 3, 5, 2, 7, 4, 5, 0}`
D. `{0, 3, 0, 2, 7, 6, 4, 4}`
Q: What are the contents of numbers after executing this code?

```java
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 7;
numbers[6] = 5;
int x = numbers[1];
numbers[x] = 2;
numbers[numbers[numbers[4]]] = 9;
```

// 0 1 2 3 4 5 6 7

A. {0, 3, 0, 2, 7, 0, 5, 9}
B. {0, 3, 0, 0, 7, 0, 5, 0}
C. {3, 3, 5, 2, 7, 4, 5, 0}
D. {0, 3, 0, 2, 7, 6, 4, 4}
Q: What are the contents of numbers after executing this code?

```java
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 7;
numbers[6] = 5;
int x = numbers[1];
numbers[x] = 2;
numbers[numbers[4]] = 9;
```

// 0 1 2 3 4 5 6 7
A. {0, 3, 0, 2, 7, 0, 5, 9}
B. {0, 3, 0, 0, 7, 0, 5, 0}
C. {3, 3, 5, 2, 7, 4, 5, 0}
D. {0, 3, 0, 2, 7, 6, 4, 4}
Q: What are the contents of numbers after executing this code?

```java
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 7;
numbers[6] = 5;

int x = numbers[1];
numbers[x] = 2;
numbers[numbers[numbers[4]]] = 9;
```

// 0 1 2 3 4 5 6 7
0 3 0 0 7 0 5 0
A. {0, 3, 0, 2, 7, 0, 5, 9}
B. {0, 3, 0, 0, 7, 0, 5, 0}
C. {3, 3, 5, 2, 7, 4, 5, 0}
D. {0, 3, 0, 2, 7, 6, 4, 4}
Q: What are the contents of numbers after executing this code?

```java
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 7;
numbers[6] = 5;
int x = numbers[1];
numbers[x] = 2;
numbers[numbers[numbers[4]]] = 9;
```

// 0 1 2 3 4 5 6 7
A. {0, 3, 0, 2, 7, 0, 5, 9}
B. {0, 3, 0, 0, 7, 0, 5, 0}
C. {3, 3, 5, 2, 7, 4, 5, 0}
D. {0, 3, 0, 2, 7, 6, 4, 4}
Q: What are the contents of numbers after executing this code?

```java
int[] numbers = new int[8];
numbers[1] = 3;
numbers[4] = 7;
numbers[6] = 5;
int x = numbers[1];
numbers[x] = 2;
numbers[numbers[4]] = 9;
```

// 0 1 2 3 4 5 6 7
A. {0, 3, 0, 2, 7, 0, 5, 9}
B. {0, 3, 0, 0, 7, 0, 5, 0}
C. {3, 3, 5, 2, 7, 4, 5, 0}
D. {0, 3, 0, 2, 7, 6, 4, 4}
Getting “length”

Similar to a String, you can get the length of an array by saying

```
myArray.length
```

Note that there are *no parentheses* at the end!

**Practice:**

- What is the index of the *last element* of an array in terms of its length?
- What is the index of the *middle element* of an array in terms of its length?
Just like with Strings, we can use an array’s length, along with its indices, to perform cool operations.
Just like with Strings, we can use an array’s length, along with its indices, to perform cool operations.
For instance, we can efficiently initialize arrays.

```java
int[] numbers = new int[8];
for (int i = 0; i < numbers.length; i++) {
    numbers[i] = 2 * i;
}
```

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>14</td>
</tr>
</tbody>
</table>
Just like with Strings, we can use an array’s length, along with its indices, to perform cool operations. For instance, we can read in numbers from the user:

```java
int length = readInt("# of numbers? ");
int[] numbers = new int[length];
for (int i = 0; i < numbers.length; i++) {
    numbers[i] = readInt("Elem " + i + ": ");
}
```
Just like with Strings, we can use an array’s length, along with its indices, to perform cool operations. Try it out! *sum up* all of an array’s elements.

```java
// assume that the user has created int[] numbers
int sum = 0;
for (int i = 0; i < numbers.length; i++) {
    sum += numbers[i];
}
println(sum);
```
Sometimes, we want to hardcode the elements of an array.

```java
int numbers = new int[7];
numbers[0] = 5;
numbers[1] = 32;
numbers[3] = 12;
...

// This is tedious!
```
Sometimes, we want to hardcode the elements of an array. Luckily, Java has a special syntax for initializing arrays to hardcoded numbers.

```
type[] name = { elements };  // Java infers the array length
int[] numbers = {5, 32, 12, 2, 1, -1, 9};
```
Limitations of Arrays

• An array’s length is **fixed**. You cannot resize an existing array:

```java
int[] a = new int[4];
a.length = 10; // error
```

• You cannot compare arrays with `==` or `equals`:

```java
int[] a1 = {42, -7, 1, 15};
int[] a2 = {42, -7, 1, 15};
if (a1 == a2) { ... } // false!
if (a1.equals(a2)) { ... } // false!
```

• An array does not know how to print itself:

```java
println(a1); // [I@98f8c4]
```
The class Arrays in package java.util has useful methods for manipulating arrays:

<table>
<thead>
<tr>
<th>Method name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrays.equals(<em>array1</em>, <em>array2</em>)</td>
<td>returns true if the two arrays contain same elements in the same order</td>
</tr>
<tr>
<td>Arrays.fill(<em>array</em>, <em>value</em>);</td>
<td>sets every element to the given value</td>
</tr>
<tr>
<td>Arrays.sort(<em>array</em>);</td>
<td>arranges the elements into sorted order</td>
</tr>
<tr>
<td>Arrays.toString(<em>array</em>)</td>
<td>returns a string representing the array, such as &quot;[10, 30, -25, 17]&quot;</td>
</tr>
</tbody>
</table>
Arrays.toString accepts an array as a parameter and returns a string representation of its elements.

```java
int[] e = {0, 2, 4, 6, 8};
println("e is " + Arrays.toString(e));
```

Output:
```
e is [0, 14, 4, 6, 8]
```
Arrays as Parameters

- Arrays are just another variable type, so methods can take arrays as parameters and return an array.

```java
private int passArray(int[] numbers) {
    ...
}

private int[] returnArray(…) {
    ...
    return newArray;
}
```
Arrays as Parameters

```java
public void run() {
    int[] numbers = new int[7];
    fillArray(numbers);
    println(Arrays.toString(numbers));
}

private void fillArray(int[] arr) {
    for (int i = 0; i < arr.length; i++) {
        arr[i] = 2 * i;
    }
}
```
Arrays are objects. An array variable stores an address of a heap location with the array data.
Let’s write a method called `swapElements` that swaps two elements of an array. How can we do this?

What parameters should it take (if any)? What should it return (if anything)?

```java
private ??? swapElements(???) {
    ...
}
```
public void run() {
    int[] array = new int[5];
    ...
    swapElements(array[0], array[1]);
    ...
}

private void swapElements(int x, int y) {
    int temp = x;
    x = y;
    y = temp;
}
public void run() {
    int[] array = new int[5];
    ...
    swapElements(array[0], array[1]);
    ...
}

private void swapElements(int x, int y) {
    int temp = x;
    x = y;
    y = temp;
}
public void run() {
    int[] array = new int[5];
    ...
    swapElements(array, 0, 1);
    ...
}

private void swapElements(int[] arr, int pos1, int pos2) {
    int temp = arr[pos1];
    arr[pos1] = arr[pos2];
    arr[pos2] = temp;
}
public void run() {
    int[] array = new int[5];

    Arrays are **objects**, so they are passed by **reference**! Their variable boxes store their **address**. So changes to the parameter do **affect the original**.
}

private void swapElements(int[] arr, int pos1, int pos2) {
    int temp = arr[pos1];
    arr[pos1] = arr[pos2];
    arr[pos2] = temp;
}
Example: Reverse Array Program

```java
public void run() {
    int n = readInt("Enter number of elements: ");
    int[] intArray = createIndexArray(n);
    println("Forward: " + arrayToString(intArray));
    reverseArray(intArray);
    println("Reverse: " + arrayToString(intArray));
}
```

```
Enter number of elements: 10
Forward: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Reverse: [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```
What does this say?

53AAB305) 6*; 4826; 4AK) 4A); 806*; 48B8X60) 85; 1A(; :A*8B83 (88) 5*B; 46 (; 88*96*?; 8) *A(; 485); 5*B2:*A(; 4956*2 (5*–4) 8X8*; 4069285); ) 6B8) 4AA; 1 (A9; 48081; 8: 8A1; 48B85; 4) 485B528806*81 (A9; 48; (88; 4 (A?34; 48) 4A; 161; : 188; A?;

Puzzle in Gold Bug by Edgar Allan Poe
Letter Frequency

By Peter Norvig
Head of Google Research
Former CS221 Instructor

Based on 3.8 trillion letters
The basic idea behind the program to count letter frequencies is to use an array to keep track of how many times each letter appears. As the program reads the text, it increments the array element that corresponds to each letter.

```
TWAS BRILLIG
```
| Code | Char | Code | Char | Code | Char | Code | Char | Code | Char | Code | Char | Code | Char | Code | Char | Code | Char |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 32   | [space] | 48   |   | 64   | @   | 80   | P   | 96   |  | 112  | p   |
| 33   | !    | 49   |   | 65   | A   | 81   | Q   | 97   | a   | 113  | q   |
| 34   | "    | 50   |   | 66   | B   | 82   | R   | 98   | b   | 114  | r   |
| 35   | #    | 51   |   | 67   | C   | 83   | S   | 99   | c   | 115  | s   |
| 36   | $    | 52   |   | 68   | D   | 84   | T   | 100  | d   | 116  | t   |
| 37   | %    | 53   |   | 69   | E   | 85   | U   | 101  | e   | 117  | u   |
| 38   | &    | 54   |   | 70   | F   | 86   | V   | 102  | f   | 118  | v   |
| 39   | (    | 55   |   | 71   | G   | 87   | W   | 103  | g   | 119  | w   |
| 40   | )    | 56   |   | 72   | H   | 88   | X   | 104  | h   | 120  | x   |
| 41   | ;    | 57   |   | 73   | I   | 89   | Y   | 105  | i   | 121  | y   |
| 42   | :    | 58   |   | 74   | J   | 90   | Z   | 106  | j   | 122  | z   |
| 43   | +    | 59   |   | 75   | K   | 91   | [   | 107  | k   | 123  | {   |
| 44   | ,    | 60   |   | 76   | L   | 92   | \   | 108  | l   | 124  | |   |
| 45   | .    | 61   |   | 77   | M   | 93   | ]   | 109  | m   | 125  | }   |
| 46   | ?    | 62   |   | 78   | N   | 94   | ^   | 110  | n   | 126  | [   |
| 47   | /    | 63   |   | 79   | O   | 95   | ~   | 111  | o   | 127  | ]   |
To the code!
Poe’s Cryptographic Puzzle

53AAB305) 6*; 4826) 4AK) 4A) 806*; 48B8X60) 85; 1A (; 8*8B83 (88) 5*B; 46 (88*96*?
?; 8) *A (; 485) 5*B2; *A (4956*2 (5*4) 8X8*; 4069285);) 6B8) 4AA; 1 (A9; 48081; 8: 8A
1; 48B85; 4) 485B528806*81 (A9; 48; (88; 4 (A?34; 48) 4A; 161; : 188; A?;)

AaBcD0aSs6BntB26SNAKsHAsB06BnBdBBX
605$SsraTFAryANBEGKGBsANBTH06RTH0N6NN
UTSNaKHrBAStANBBBoYNArTMA6NBRANsHsEx
BnTH06nBraStS6BrSsAATFRAuTH0RoBIFyra
FTbHrBAHTsHABABBBB06NB1fKoUTHRKBrTHK
AUCbTHREsHAf6FfYErETaut