Type char

Notice that the first letter of the word char is lower case – this is how Processing spells it.

Processing comes “complete” with eight built in or so-called primitive types of data. One of these is the type, **`char`**. The type name, **`char`** is short for character. A nontechnical definition of **`char`** that is sufficient for 257 is any character that we can type with the keyboard or what some authors refer to as printing characters. There are other **`char`**s but we will ignore them and leave their explanation to the next course you take.

Look at your keyboard. Every key that makes a change to the display when you are typing is a char. These keys includes the white space keys which are:

- the space
- the tab,
- and the return / enter / new-line

The numbers, punctuation, and special characters (e.g. $ % @ #) are chars.

A Digression

There is a small problem with characters. The computer really only understands numbers. And there is a problem there. It only understands two numbers: 0 and 1. This is because modern computers use electricity to store data. Electricity can either be flowing through a circuit or not flowing through the circuit. Generally a flow of electricity is translated as the value 1 and the lack of a flow is translated 0.

If we use only the numbers 0 and 1, we are using the binary number system. Translating binary numbers to decimal (int) numbers and decimal (int) to binary can be done in processing using API functions:

- **`binary( )`**
- **`unbinary( )`**

You can pursue these if you want to but we will no use the directly in class assignments. Binary numbers grow in physical size very quickly:

decimal (int) value	binary value
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000

The ASCII Code

In order to use binary numbers to represent chars a “code” is needed. This code is a simple substitution code where one binary number represents one char. This code is called the ASCII code (American Standard Code for Information Interchange). You do not need to memorized the code or remember it. You just need to be aware of its existence.

Jim has survived thirty years of teaching programming knowing only three ASCII values:

- ‘A’ ➔ ASCII 65
- ‘a’ ➔ ASCII 97
- ‘ ‘ [space] ➔ ASCII 32

Notice that the difference between the ASCII values ‘a’ and ‘A’ is a ‘ ‘.

ASII values go in sequence:

char	**ASCII Code**
‘a’	65
‘b’	66
‘c’	67
‘d’	68
‘e’	69

There is one thing to note. The char ‘z’ does not directly follow the char ‘Z’. There are 26 characters in the alphabet but the space between ‘A’ and ‘z’ is 32. This means that there are some chars between ‘Z’ and ‘a’ in the ASCII code.

If you look up ASCII on the web you will find tables that show the chars and their numeric codes. These charts usually show three different values for each char:

- decimal
- octal (base eight)
- hex (base sixteen)

We will ignore octal and look at hex a bit later when we look at the RBG color mode the Processing uses.

Why Do We Care About ASCII?

We have taken this very brief look at ASCII because of the term, **text** that we often see when we are reading documentation. Text refers to a document that contains only ASCII values. The only formatting in a text file or text document is the space, the tab, and the return/enter/new-line characters. The coloring and font that you see when you view a text file or text document on a computer are provided by the software that you are using to display the file or document.

Text files often have the extension .txt. However, there are other text files. The Processing files (.pde files) that you have been saving are text files. If you go in and use C++ or C or Python, those files will end with .c, cpp, and .py but they are all text files.

Text Files.
Another reason to have this background is if/when you need to bring data into your Processing program from an outside source. This data is usually text data or what we more commonly call text files.

Text files are just sets of chars separated by a specific char such as the space or the comma.

The use of the space is very common. Think about your email address. Here is Jim's
jr2u@andrew.cmu.edu

Notice that there are no spaces in the address. There cannot be any spaces. Mail programs use a space to mark the end of the address. This is why there is a period or dot where we would expect to find spaces:
jr2u @ andrew cmu edu

Another point – the address is andrew.cmu.edu and not Andrew.CMU.Edu which we might expect. It could be Andrew but, by convention it is not. And since it was originally andrew, we cannot use Andrew because the char 'A' has an different ASCII code (65) than the char 'a '(97).

`Char vs int`
The numbers on your keyboard are not really numbers – they are digits. When you press the key, 8, you are entering the char '8'. It is not the number 8 so it cannot be added or subtracted in the conventional arithmetic manner. If we want to use the character 8 as a number, we have to convert it to an int value. Some programming software does this for us. In the language C++ this code can read and convert the in put of the char '8' to the int value 8:

```
        int num;
        cout << "Enter a number: ";
        cin >> num;
```

If you press the key 8, the '8' is be converted to the int value 8.

Processing does not have this type of straightforward keyboard input. There are several ways to read keyboard input. One way to do this type of input is to use a **`keyPressed( )`** function and code it ourselves. We will look at this in a later set of notes.

Type char
You have use the char type in your keyPressed( ) coding:

```
void keyPressed( )
{
 if  (key == 'X' )
}
```

The Processing variable, key is a char variable.

Since the char values have a numerical equivalent in the ASCII code, Processing "knows" the order. This is how it "knows" that char 'A' (ASCII 65) is less than the char 'B (ASCII 66). This is also why the chars .jpg are not the same as .JPG when you are trying to load an image file. The ASCII values of jpg are different than the ASCII values for JPG.

Because the chars have an equivalent numerical value, we can do us use the for loop with chars. This code:

```
for( char ch = 'A'; ch <= 'Z'; ch++)
{
   print( ch + " " );
}
println( );
```

produces this output:

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

Converting from int to char and char to int
Because the chars have an equivalent numerical value, we can convert char values to their corresponding ASCII value and back. Processing has two function to do this:

- `char( int );`
- `int ( char );`

This code:

```
     char( 65 );
```

Converts 65 to the char A.

And this code:

```
     int ( 'A' );
```

Converts the char 'A' to 65.

Character Arithmetic

In a "traditional" first programming course, assignments can focus on what some teachers call character arithmetic. We are not really interested in this for 257 but it is worth mentioning here.

What happens if you "add" the char 'A' and the char 'B'? Do you get 'C'?? We can find out. If we run this code:

```
            text( ('A' + 'B'), 10, 50 );
```

the output is

```
            131
```

Where did this 131 come from? Remember that the ASCII value of 'A' is 65 and the values are in sequence to the ASCII value of 'B' must be 66. The sum of 65 and 66 is 131. Processing converted the result to an integer. The char that 131 represents in not printed by Processing. ASCII values beyond 127 are in the extended ASCII character set and Processing as we use in in class does not work with the extended ASCII character set.

What happens if we add the value 1 to the char 'A'? We can find out with this code:

```
            text( ('A' + 1), 10, 50 );
```

The output for this code is:

```
            66
```

The ASCII value of 'A' is 65 so 66 is the correct answer.

If we want the char 'B' instead of 66, we can use the char( ) function like this:

```
            text( char('A' + 1), 10, 50 );
```

to produce the output:

```
            B
```

You may be asking how you might us this character arithmetic stuff in the near future and the answer is that it is available to you if you need it.

Practical Uses of the Type char

The type char is used by Processing for keyboard input of single characters as we have done in the **`keyPressed( )`** function in the course.

Another use that we begin at this time is when be combine multiple char values into what we call a **`String`**. Strings are covered in more detail in class notes #17