



## Resources: Strings and Arrays

This resource covers three topics:

- Character data type

- String data type

- Array data type

Check the Processing reference pages for further detail.

### Characters

The keyboard characters have the data type `char` in modern programming languages. So, declaring them uses that keyword:

```
char period = '.';
char doubleu = 'W';
```

Notice that character constants like the *W* in the example are enclosed in single quote (apostrophe) characters. Characters are single letters. This is the kind of data we get when we use `keyPressed( )` to read keyboard characters with the keyword [key](#).

### Strings

Strings are a composite type, and for that reason it is capitalized when used as a keyword. To declare strings, we write

```
String quotation = "Uncle Sam wants you!";
String reply = "Doh";
String nada = "";
```

Notice that `String` constants like *Doh* in the example are enclosed in double quotes characters. Strings are sequence of single characters, including a sequence with no characters in it at all, called the *empty string*; see `nada`. Notice that it uses two double quotes with no space between them. (If there were a space between them, then it would be string of one space.)

Because strings are sequences of characters, one thing we can do with them is to *concatenate* more characters onto a string using the plus sign. This simply adds on or appends the letters.

```
reply = reply + period;
String school = "U" + doubleU;
nada = nada + nada;
```

The computer is not confused about what you mean when you use the plus sign because it knows the data types of the things you are combining ... if they are numbers it will perform addition, if they are strings and characters, it will perform concatenation. Notice

that in the third example, `nada` still is nothing because nothing appended to nothing is still nothing.

## Arrays

Arrays are also a composite type, and so it is always capitalized when it is used as a keyword. An array is a list of data items called *elements*, each item must be of the same type. When we declare an array we give the type of the data first followed by square brackets as in

```
int[ ] nums = {2 4 6 8};
String[ ] phrase = {"Who" "do" "we" "appreciate"}
```

In both examples, the arrays have four elements; `nums` contains four integers, and `phrase` contains four String values. The initial values of an array are enclosed in braces because there are usually several of them.

The elements are referred to by their index position, starting with zero, as

```
int six = nums[0] + nums[1]; // Which references first elements of nums
phrase[3] = "respect";       // Change last element of array
phrase[3] = phrase[3] + '?'; // Concatenate a character onto a string item
```

Indexes are always enclosed in brackets. When we use the elements or assign to them, we always need to give the index position.

The length of an array is the number of items, and by writing `<array variable>.length` we get its length. (Notice that an array's length is always one more than the highest index because we start counting at 0.) So, for example

```
nums.length == phrase.length
```

would be true because both array have four elements. A handy place to use `length` is when we want to visit every element of an array, as in this `for`-loop:

```
for (int i = 0; i < phrase.length; i++) {
    phrase[i] = phrase[i] + ' ';           //Put a space after each word
}
```

The loop will go around four times because length of `phrase` is 4, and the index variable `i` will have the values 0, 1, 2, 3, allowing us to reference each element of the array. As another example

```
int total = 0;
for (int j = 0; j < nums.length; j++) {
    total = total + nums[j];               //add up the array elements
}
```

The result will be 20 in total.

**New Array.** If we want an array, but don't want it initialized, we tell the computer that we want a new array, as in

```
int[ ] lots = new int[100]; //make a 100 item integer array, 0 through 99
String[ ] textin = new String[1000]; //handle 1000 text strings
```

### ***Care With Types***

The data types just discussed are different. So, adding a character to a string is concatenation and uses the `+`. Adding a String to a String array, that is, making it longer by one item, uses the `append( )` function, as in

```
phrase = append(phrase, "UW");
```

To add a single character, say the exclamation point, to a String array, we need to convert the character to a string first, by using the `str("!")`, as in

```
phrase = append(phrase, str("!");
```

The exclamation point is a single character string in position `phrase[phrase.length-1]`. Why?