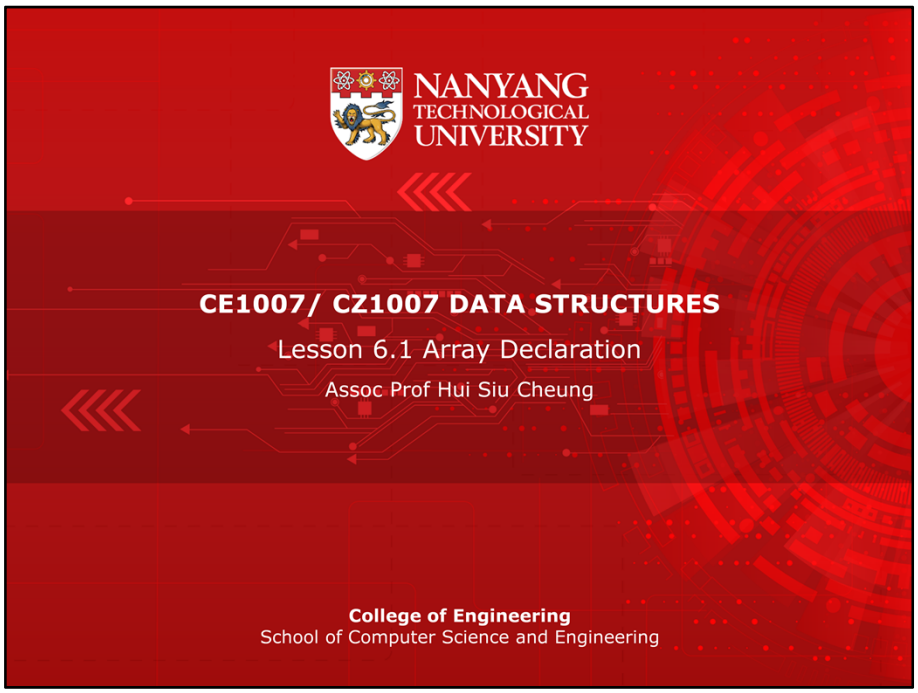
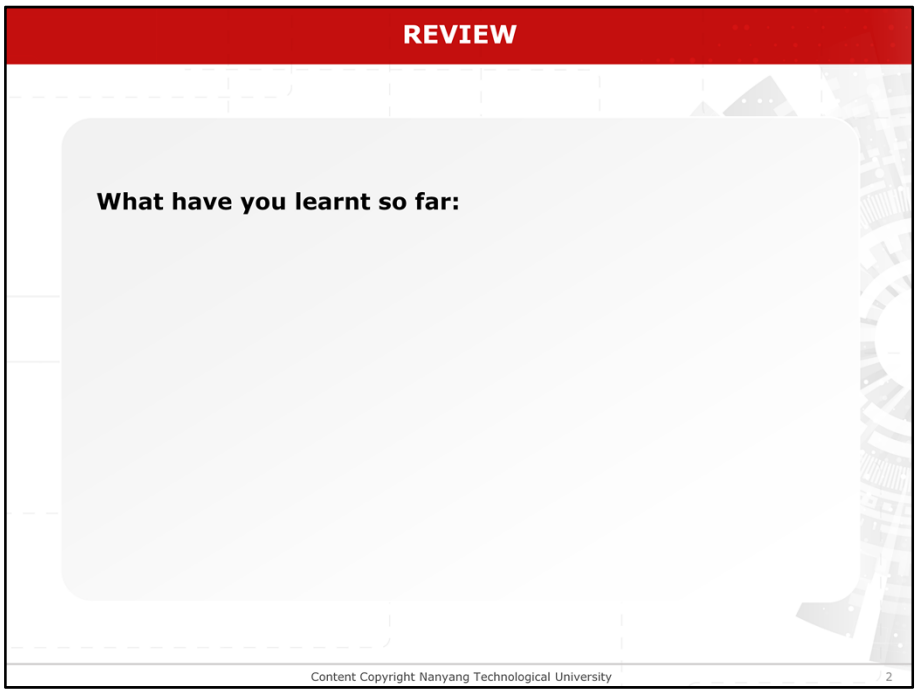
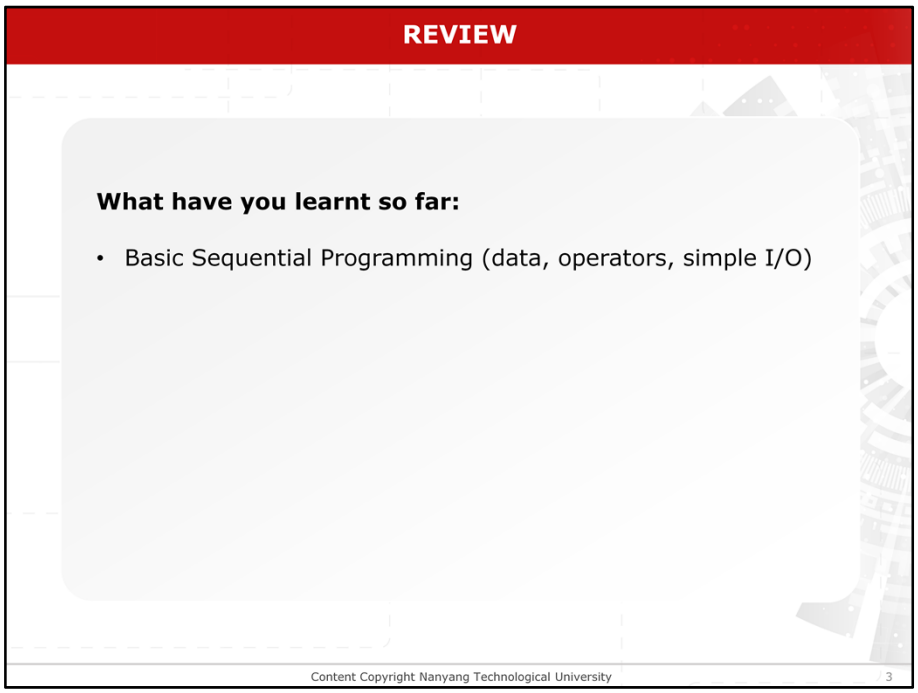


Lesson 6.1 Array Declaration.

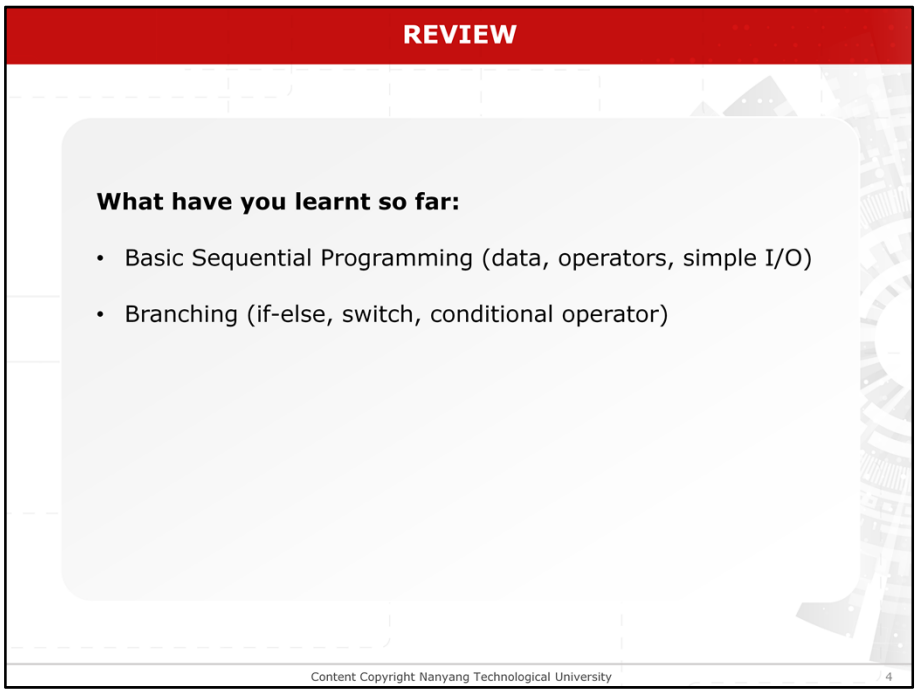
In this lesson, we will discuss on one dimensional arrays. To do that, let's do a quick review of previous lessons.



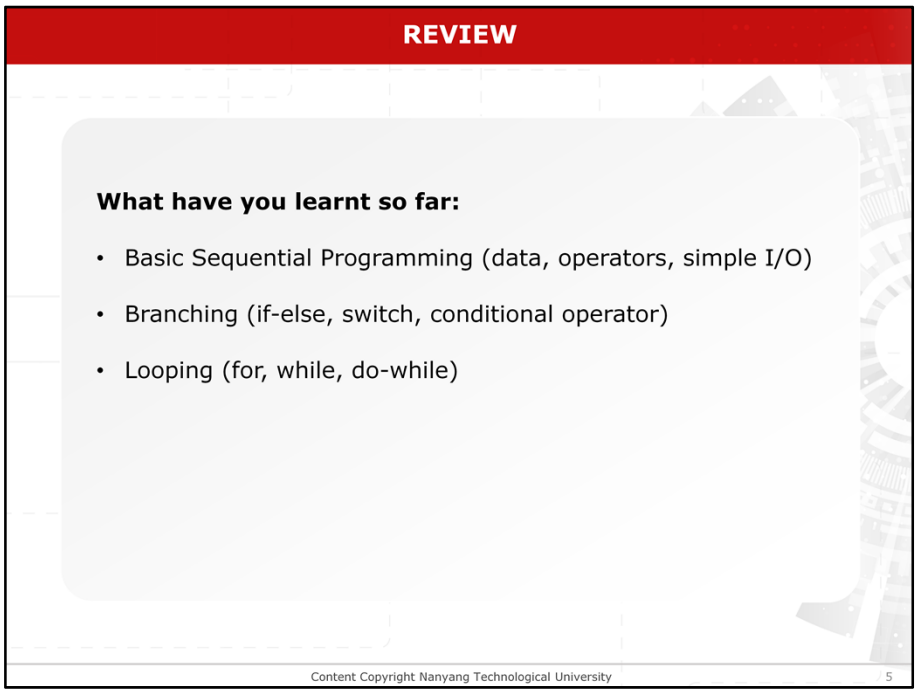
From the previous lessons, you have learnt the following:



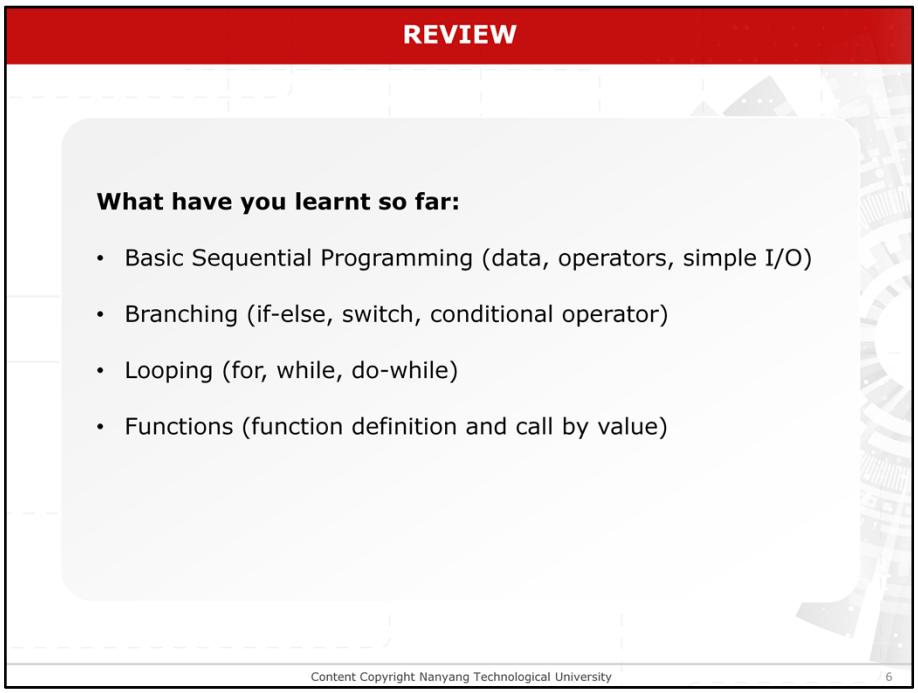
Basic Sequential Programming. We covered data, operators, simple Input, Output.



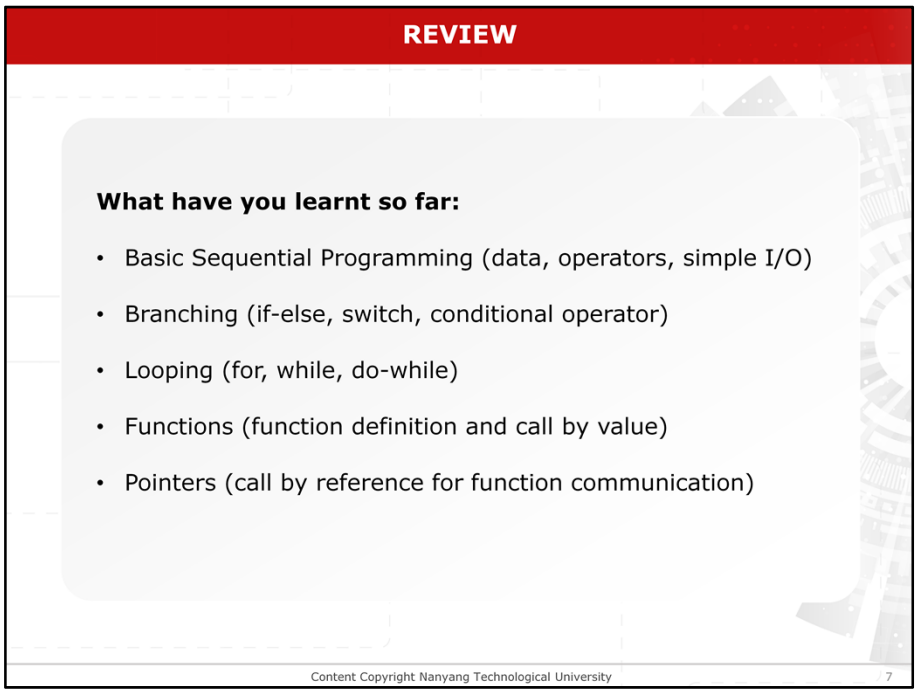
Branching using if-else, switch, conditional operator.



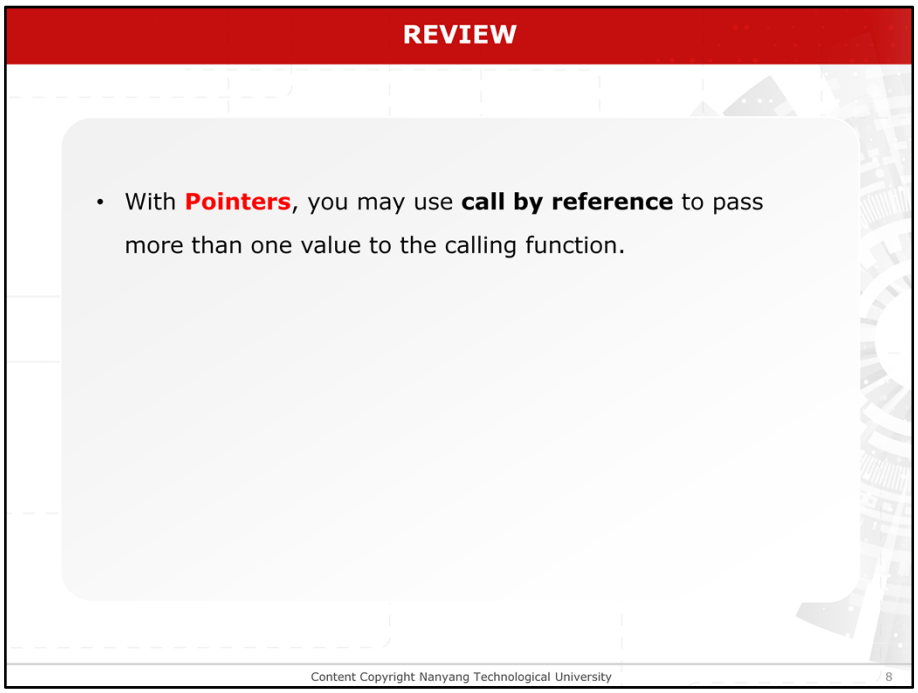
Looping using for, while, and do, while



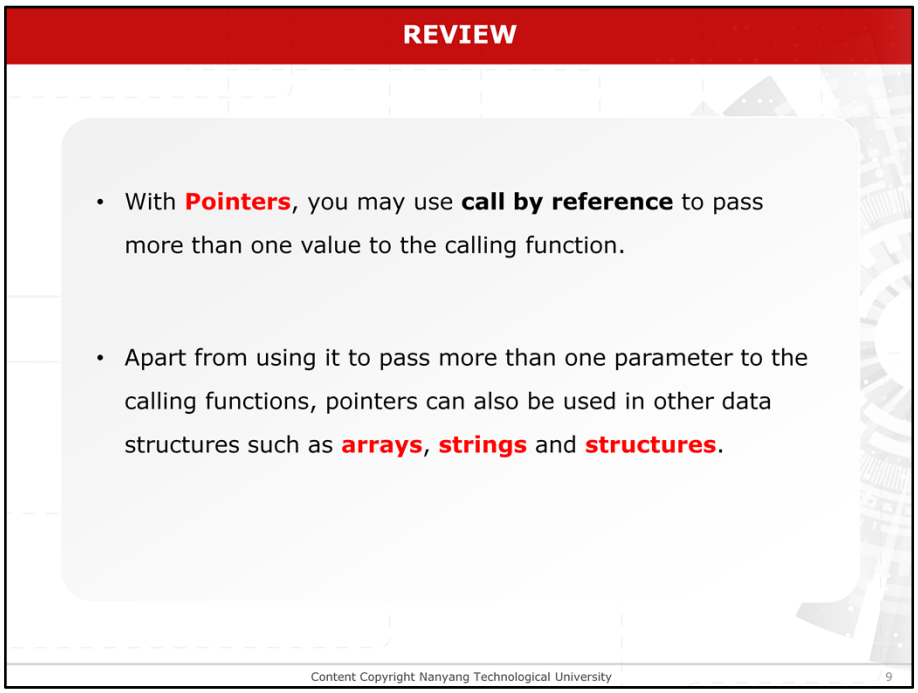
Functions where we learnt about function definition and call by value.



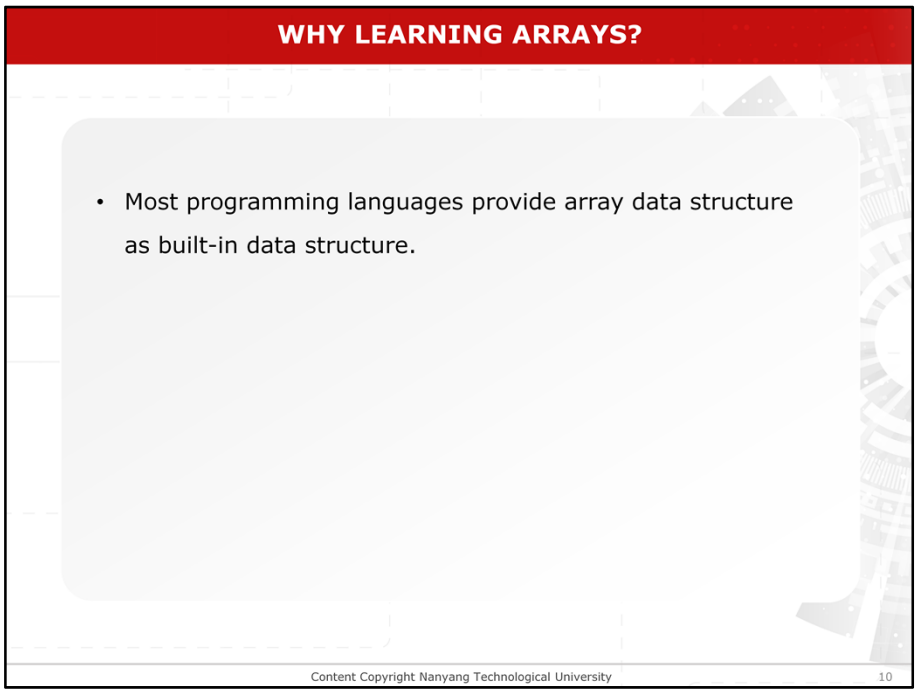
Pointers where we learnt about call by reference for function communication.



With **Pointers**, you may use **call by reference** to pass more than one value to the calling function.

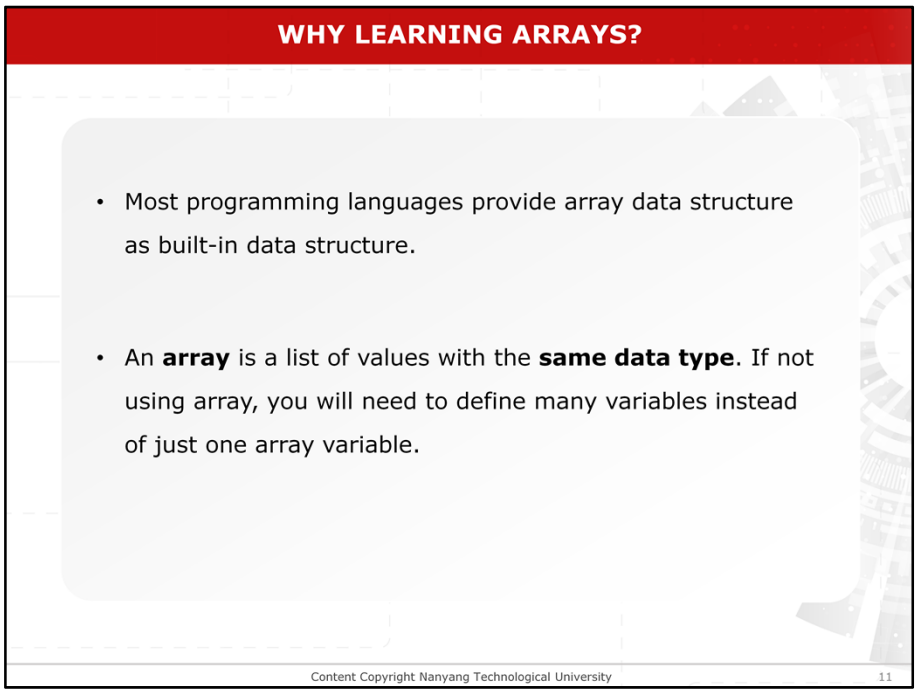


Apart from using pointers to pass more than one parameter to the calling functions, pointers can also be used in other data structures such as **arrays**, **strings** and **structures**.

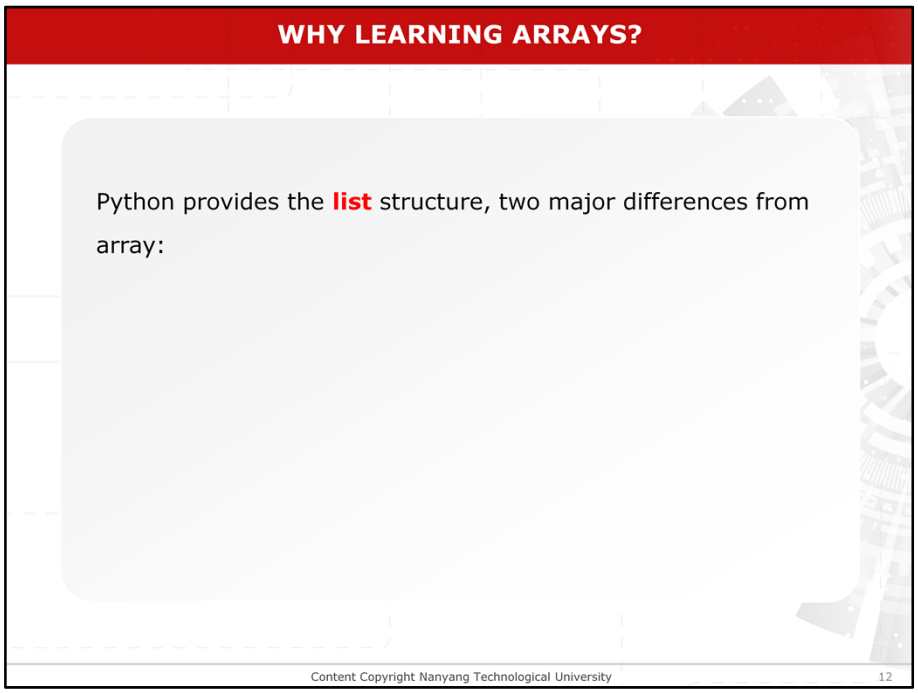


Why do we need to learn arrays?

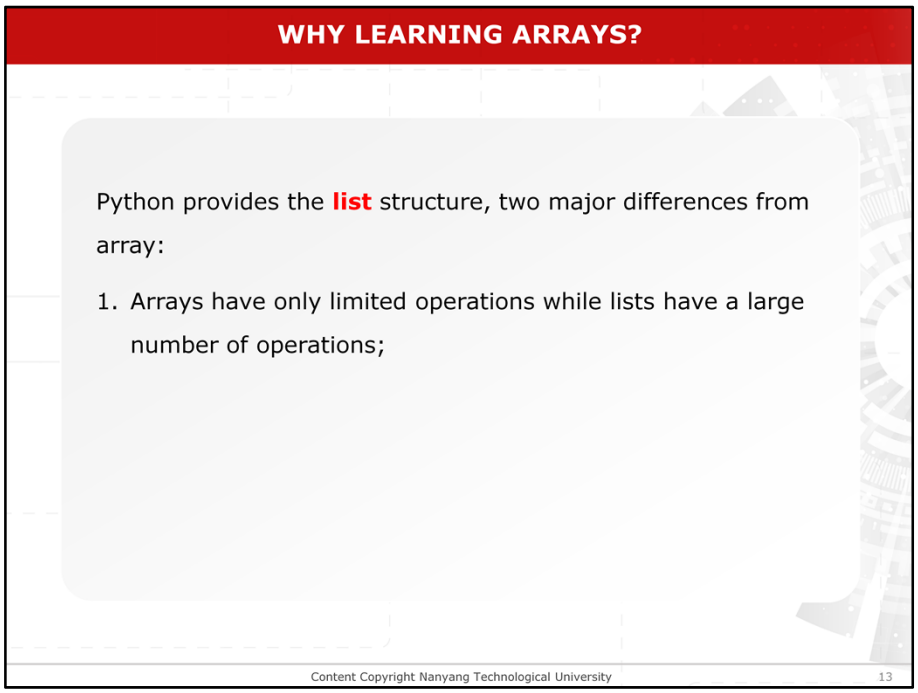
It is because most programming languages provide array data structure as built in data structure.



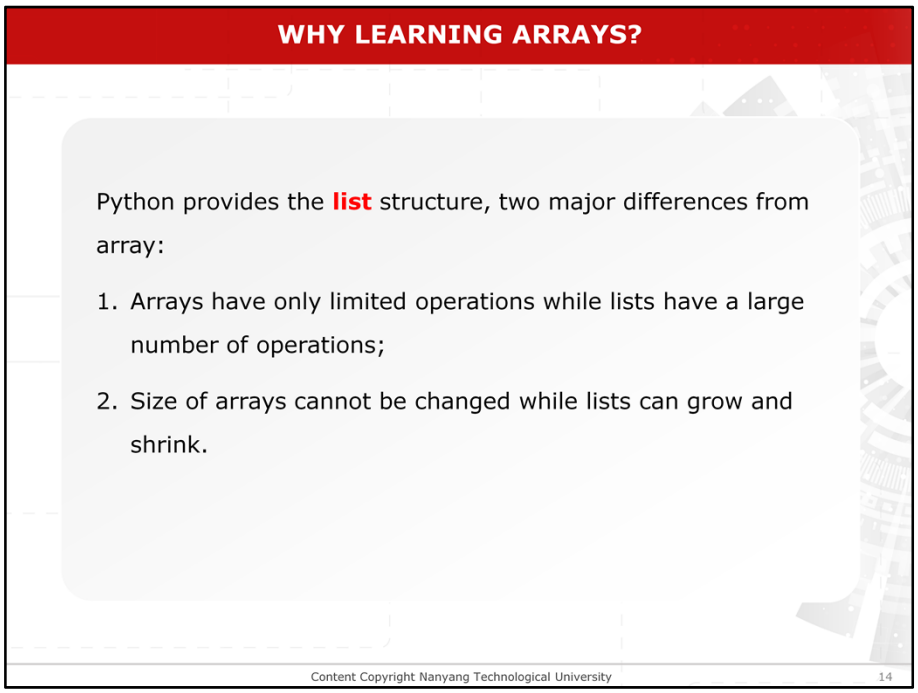
An **array** is a list of values with the **same data type**. If not using array, you will need to define many variables instead of just one array variable.



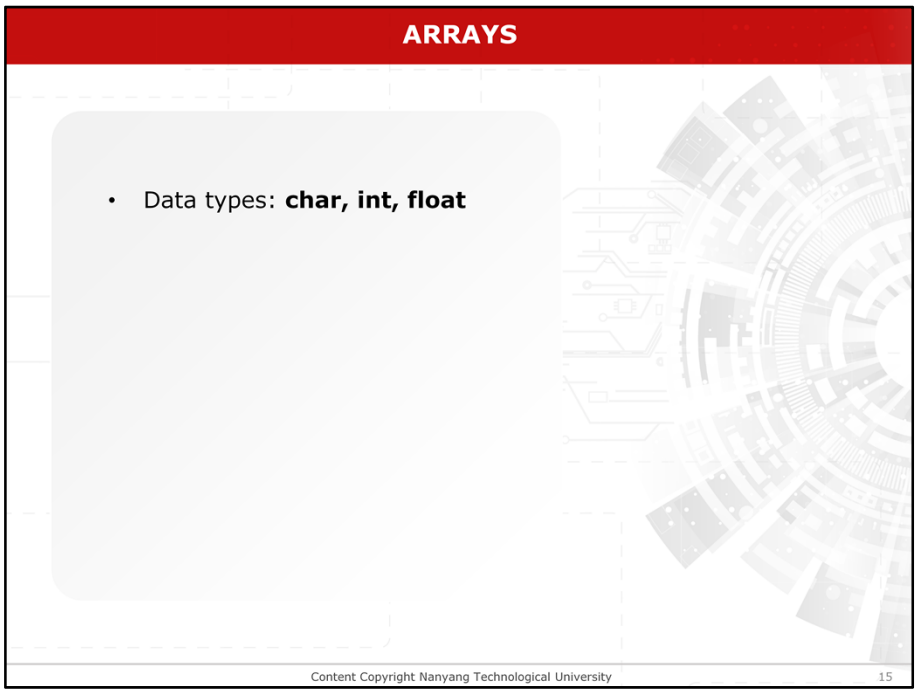
Python provides the list structure. List has two major differences from array.



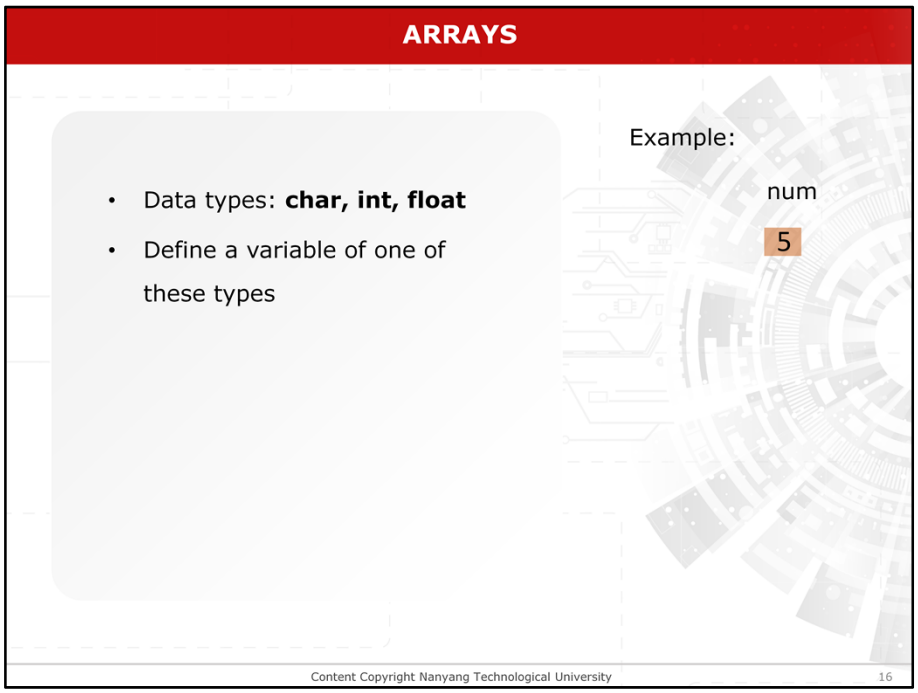
Arrays have only limited operations while lists have a large number of operations.



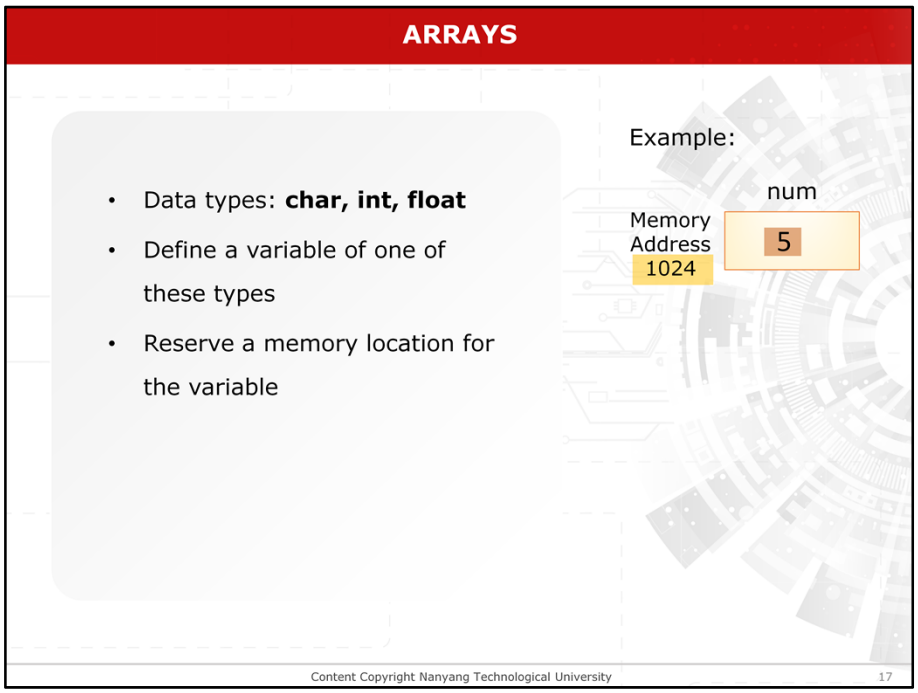
Size of arrays cannot be changed while lists can grow and shrink.



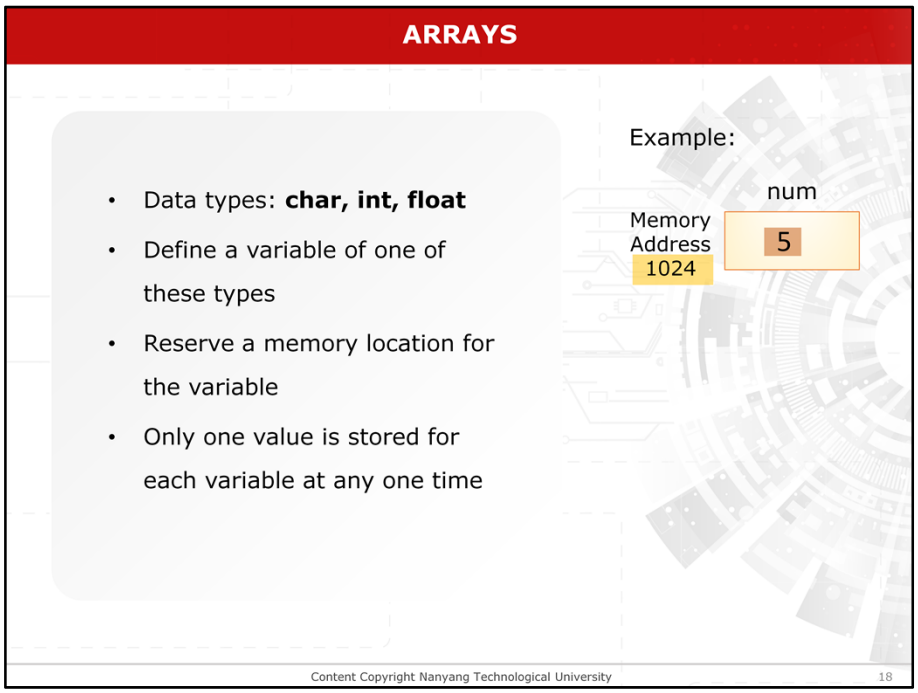
In the previous lectures, we have discussed the various data types such as **character**, **integer**, **float**, etc.



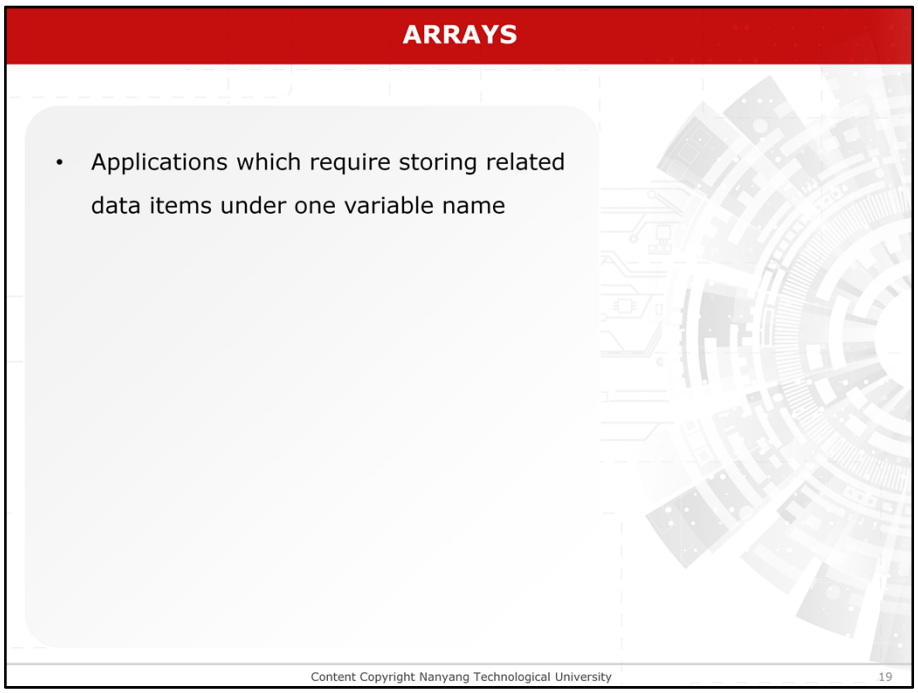
When we define a variable of one of these types,



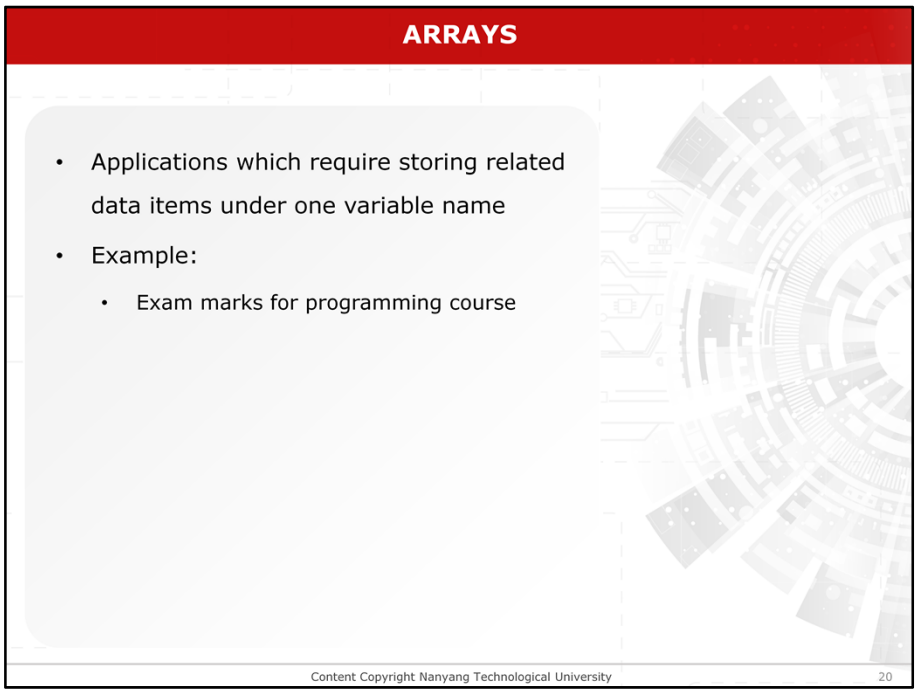
the computer will reserve a memory location for the variable.



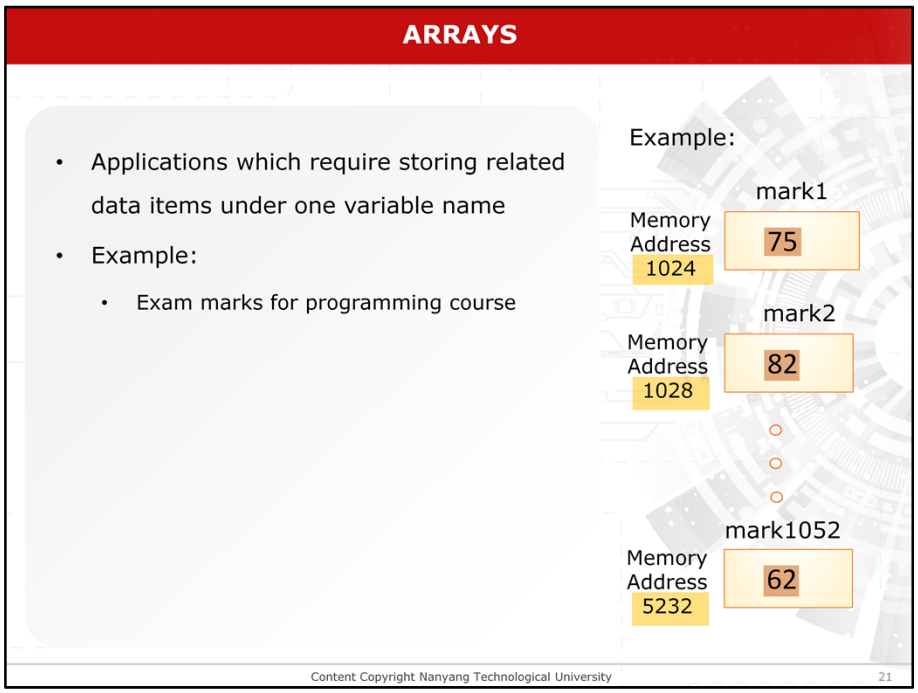
Only one value is stored for each variable at any one time.



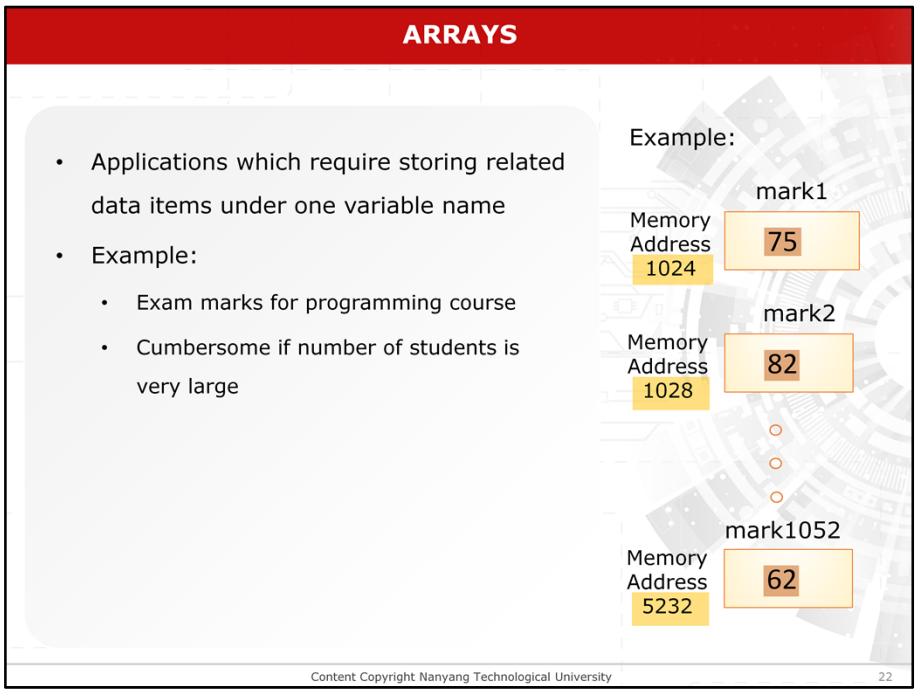
However, there are applications which require storing related data items under one variable name.



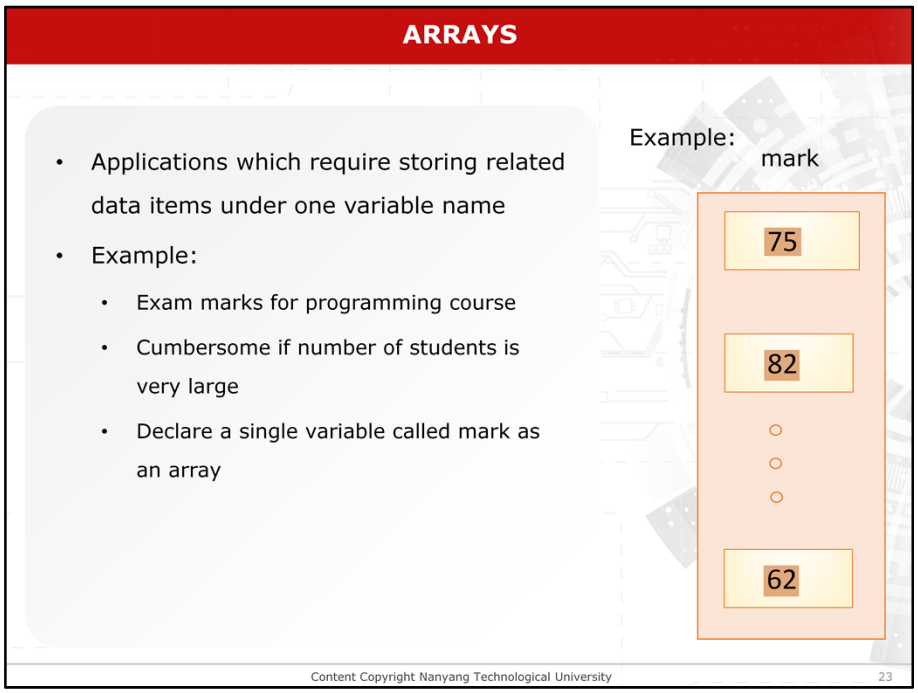
For example, if we have different items with similar nature, such as examination marks for the programming course,



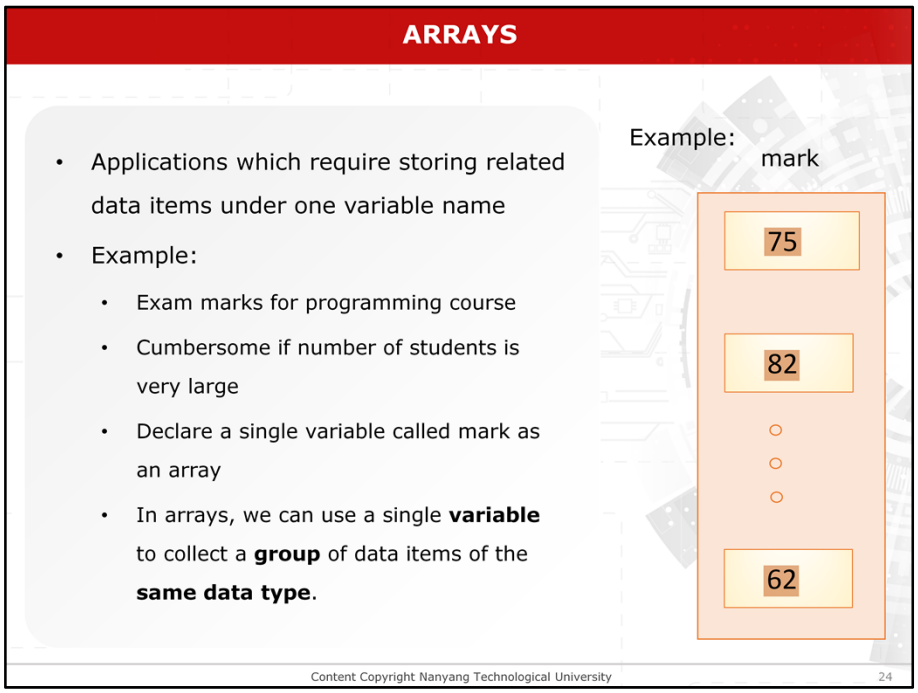
we might need to declare different variables such as mark 1, mark 2, etc. to represent the mark for each student.



This is quite cumbersome if the number of students is very large.



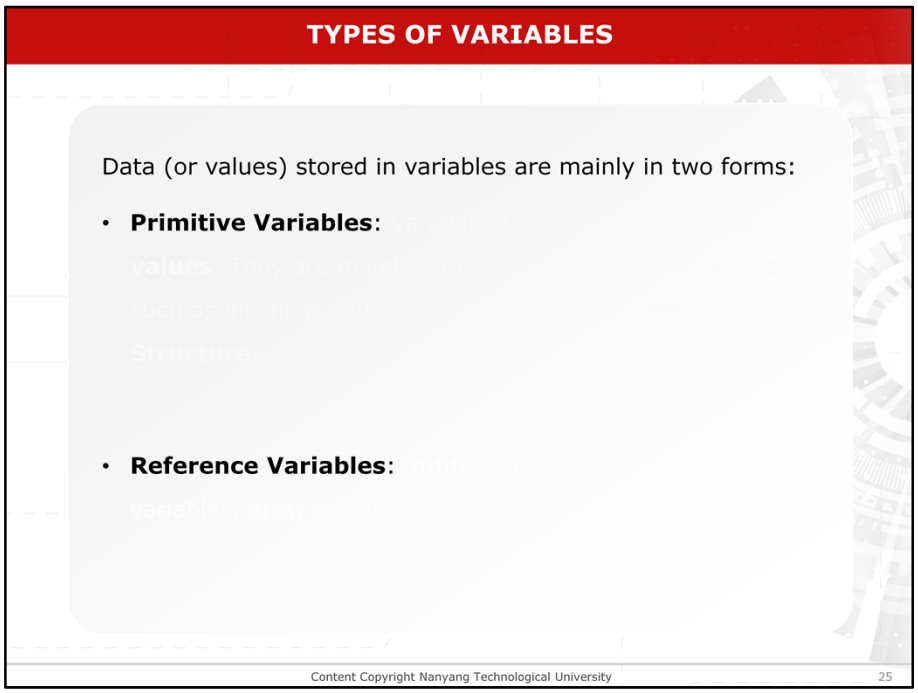
Instead, we can declare a single variable called mark as an array, and each element of the array can be used to store the mark for each student.



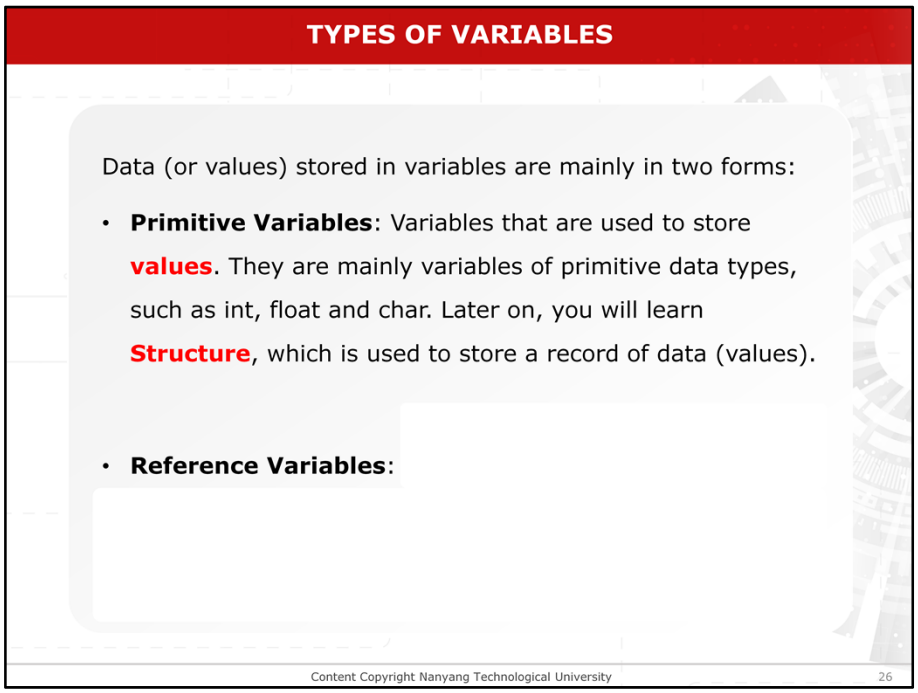
In arrays, we can use a single **variable** to collect a **group** of data items of the ***same data type***.

In this lesson, we introduce this important topic on data structure that can be used to organize and store related data items. *Arrays* are used to store related data items of the same data type. In arrays, we can categorize them as one-dimensional arrays and multi-dimensional arrays.

In this lesson, we focus on discussing one-dimensional arrays, and in the next lesson, we will discuss multi-dimensional arrays.

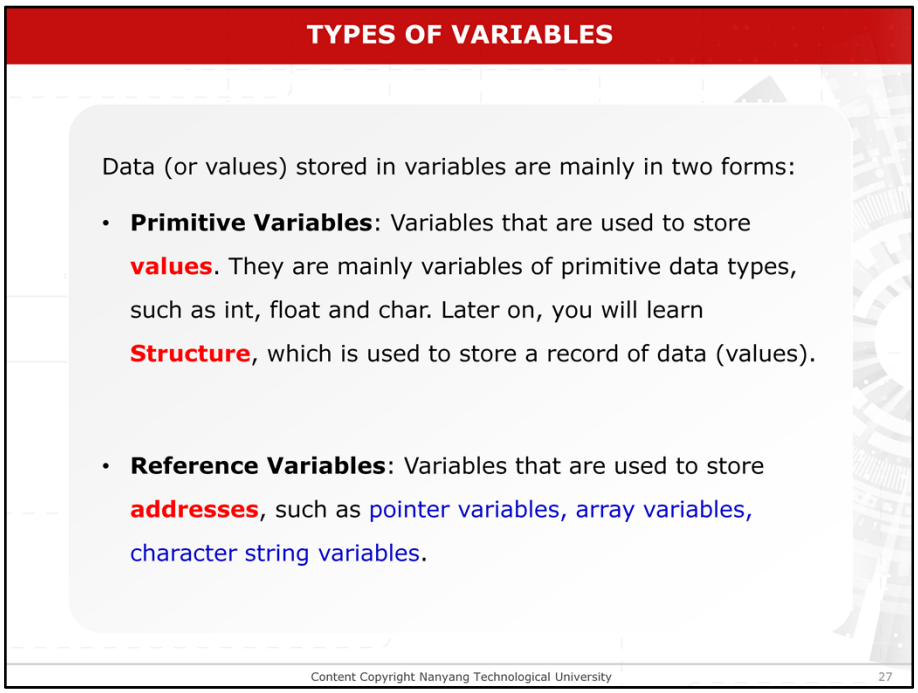


As discussed before, there are mainly two types of variables: primitive type variables and reference (pointer) variables.

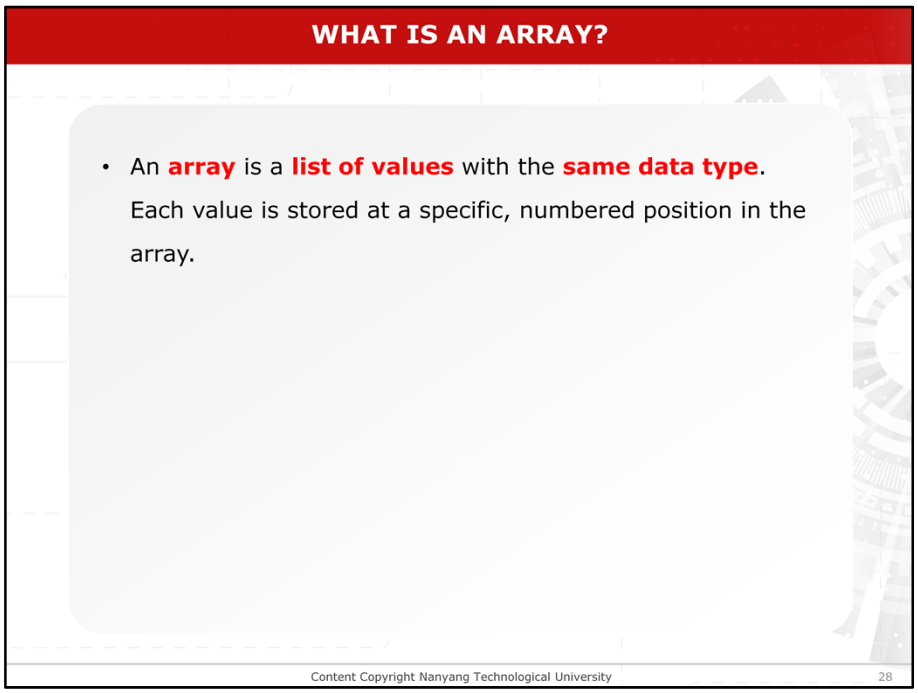


Primitive Variables are variables that are used to store values. They are mainly variables of primitive data types, such as integer, float and character.

Later on, you will learn Structure, which is used to store a record of data (values).

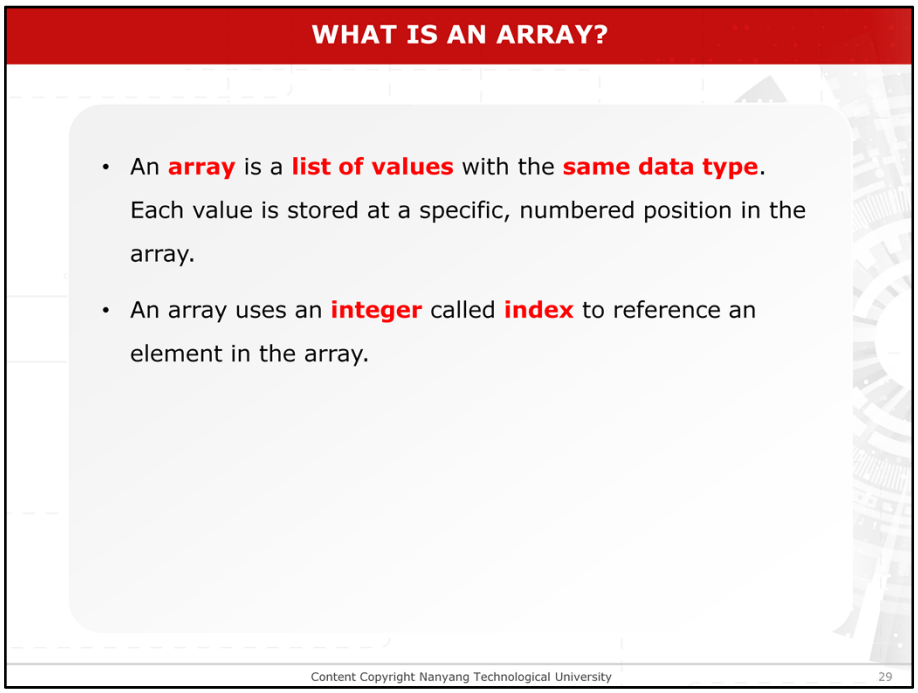


Reference variables are variables that are used to store addresses, such as pointer variables, array variables, character string variables. The content stored in an array variable is an address, not the actual data.

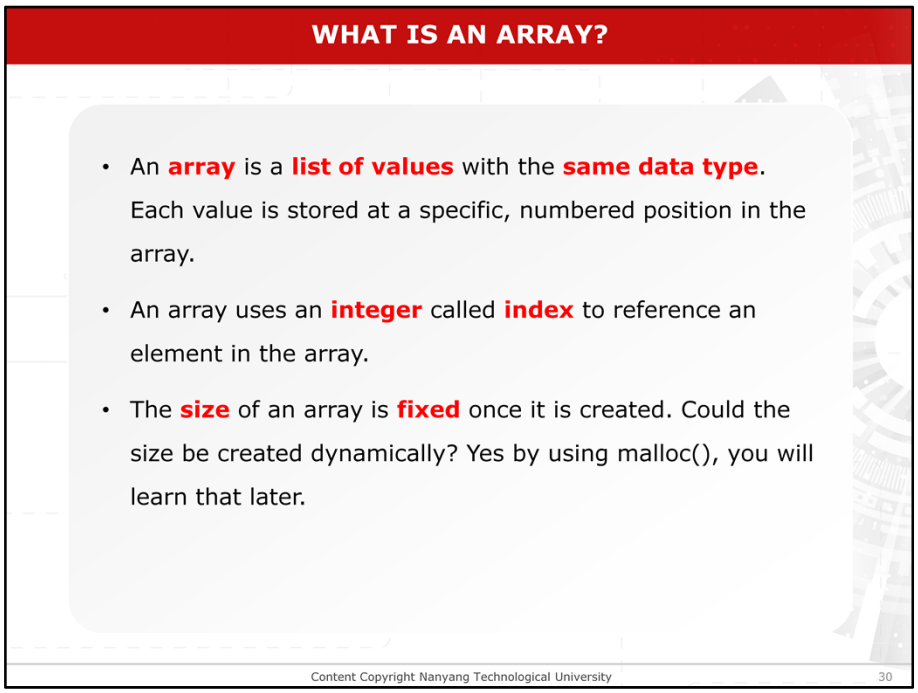


An array is a list of values with the same data type.

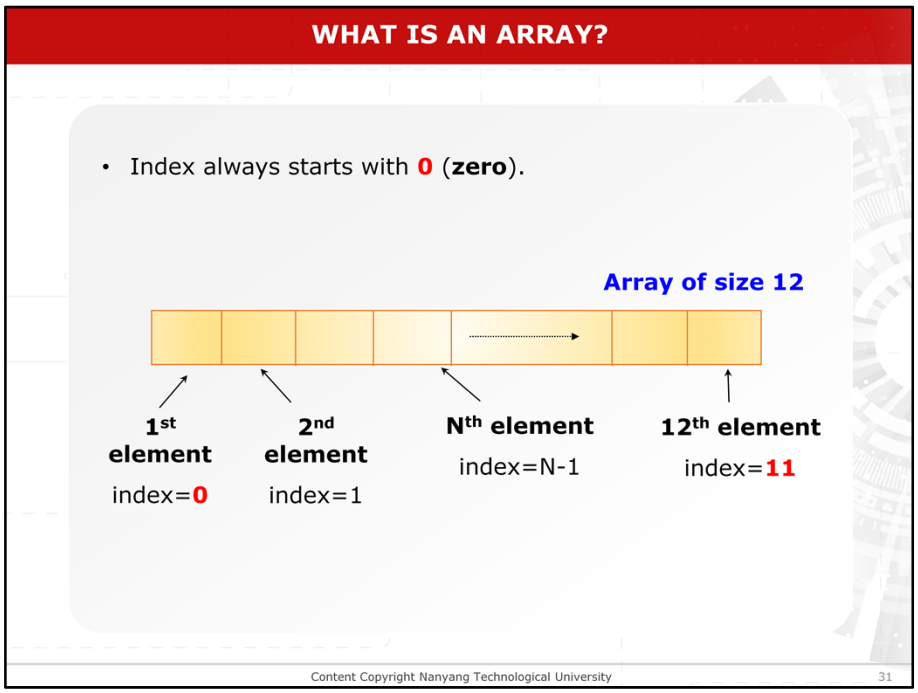
Each value is stored at a specific, numbered position in the array.



An array uses an integer called index to reference an element in the array.

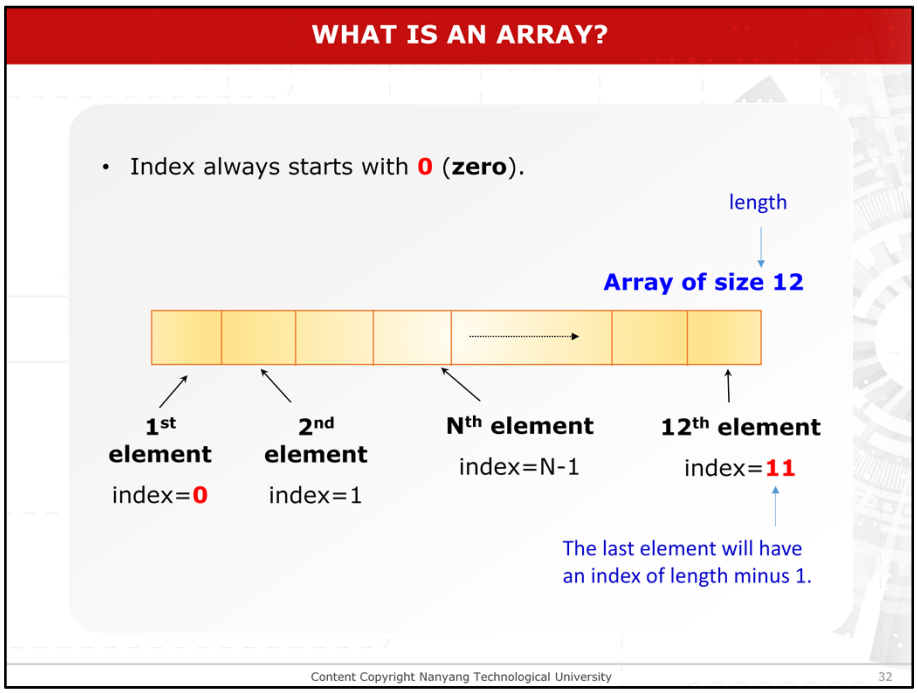


The size of an array is fixed once it is created. Could the size be created dynamically? Yes by using malloc, you will learn that later.



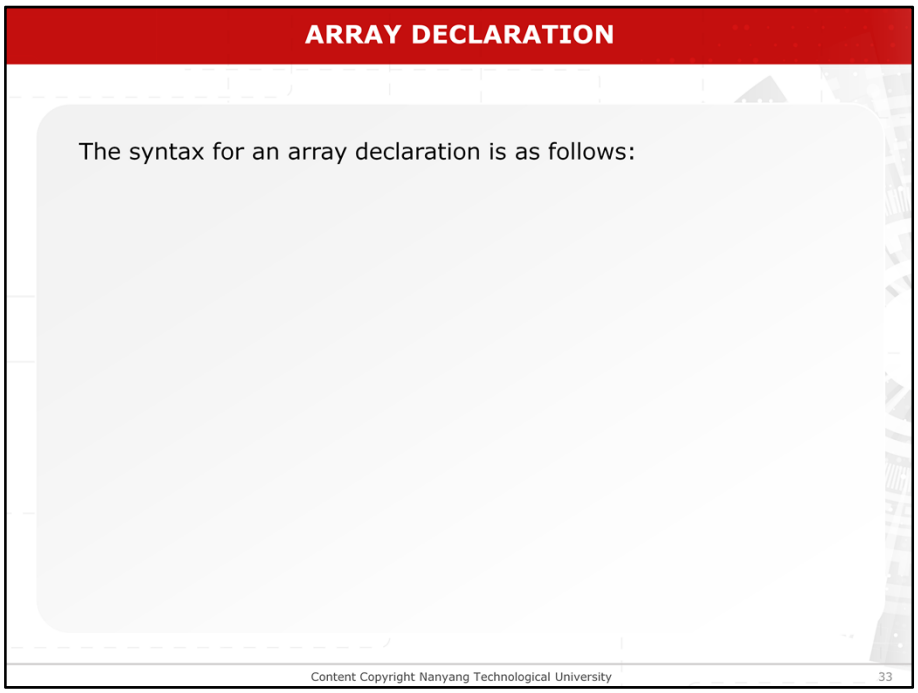
Index always starts with zero. Index is used to reference an element in the array.

In this example, there are 12 elements in the array so the array size is 12. The first element is referred by index 0.

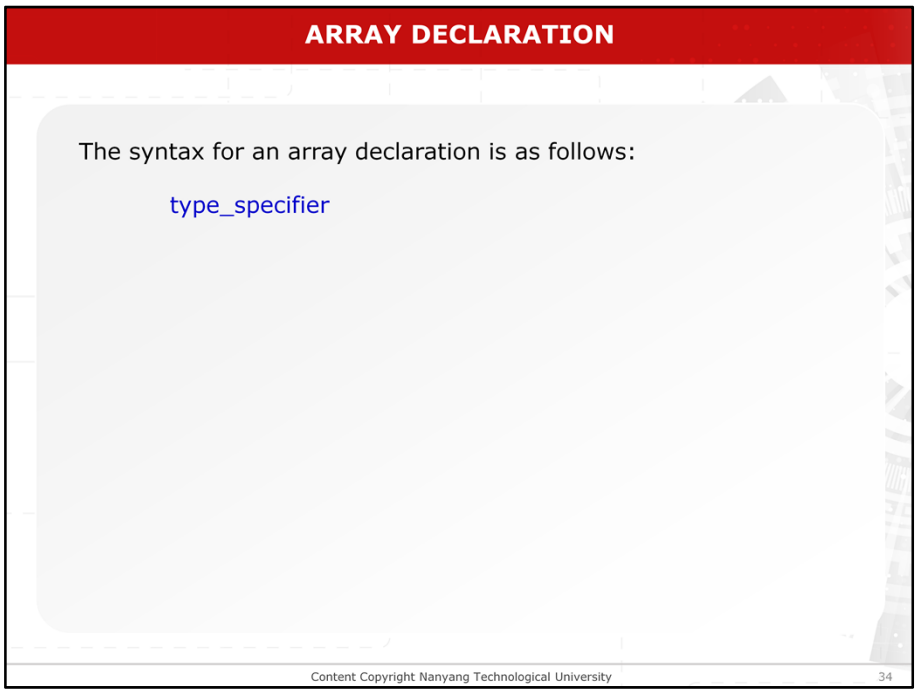


The last element will have an index of length minus 1.

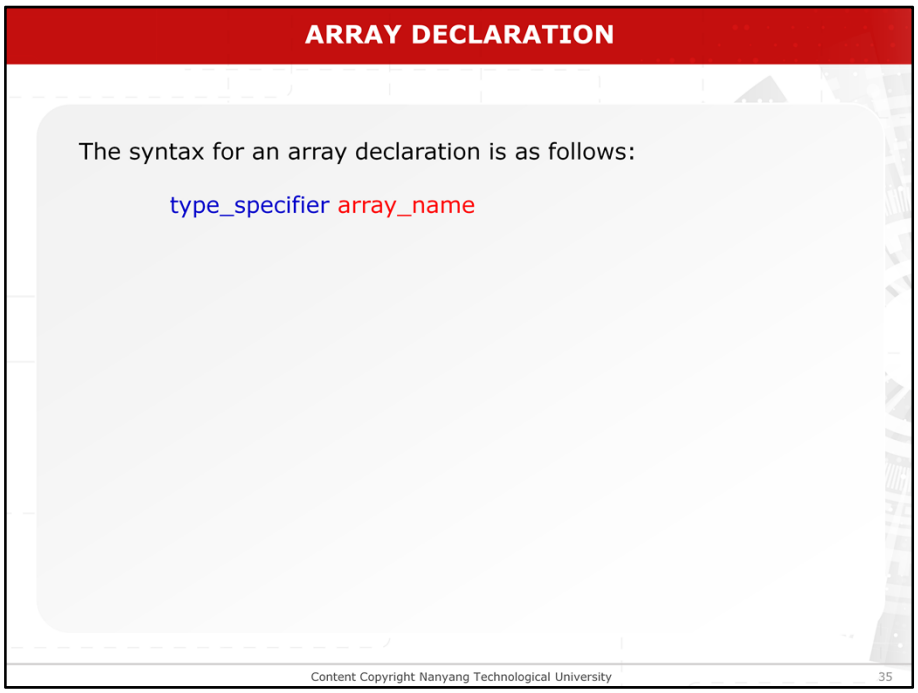
In this example, the last element, that is the 12th element has an index of 11.



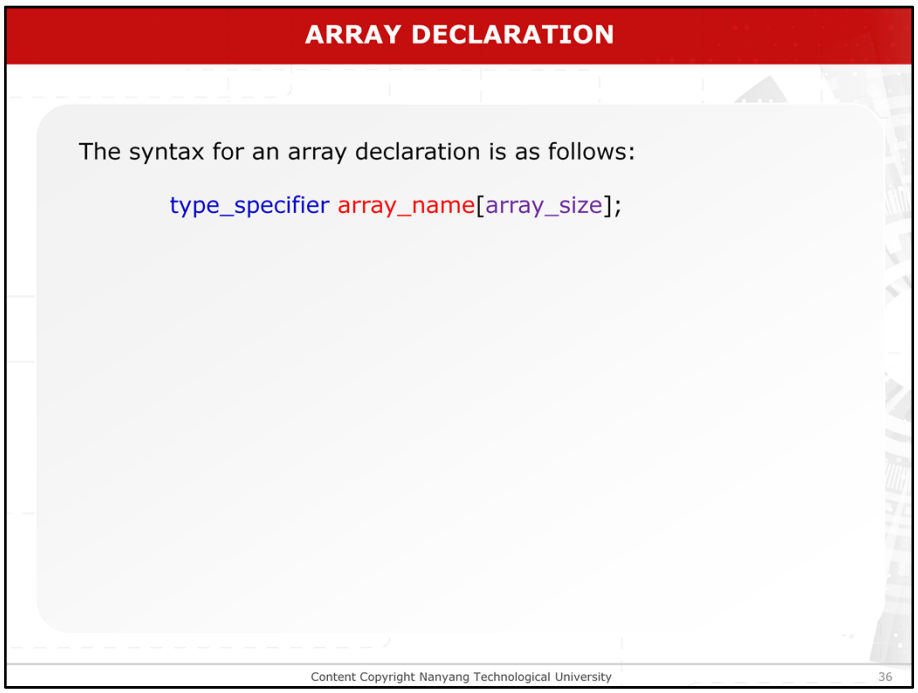
The syntax for an array declaration is as follows:



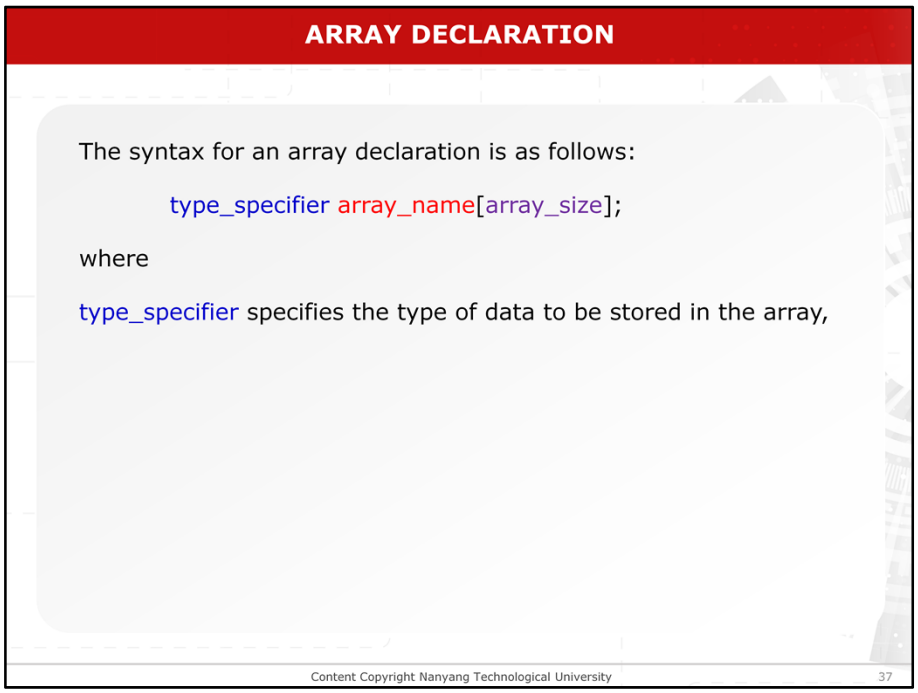
Type specifier


ARRAY DECLARATION
The syntax for an array declaration is as follows:
type_specifier array_name
Content Copyright Nanyang Technological University
35

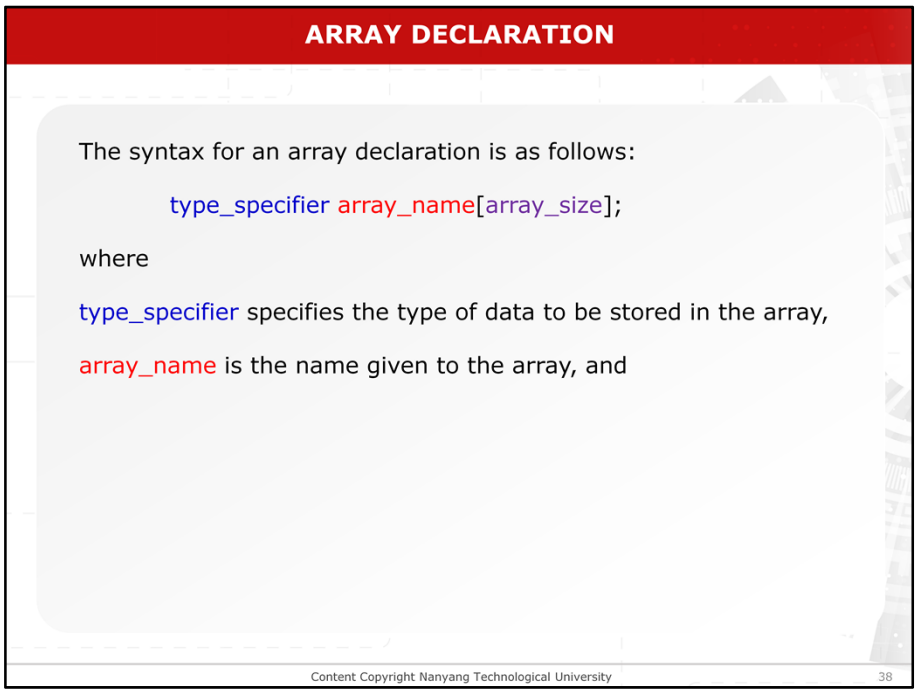
Array name



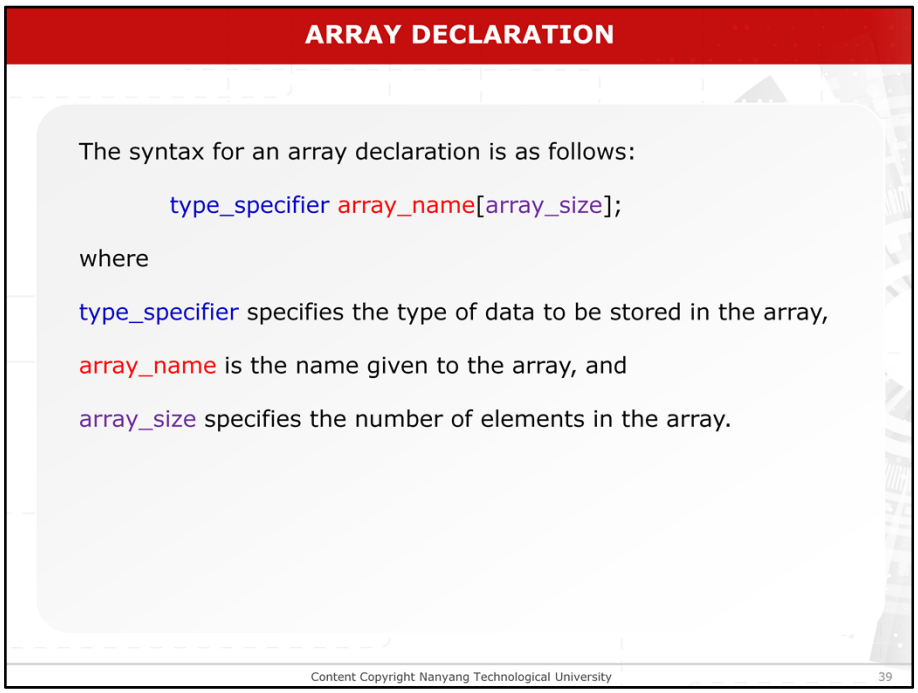
Square bracket **array size** square bracket



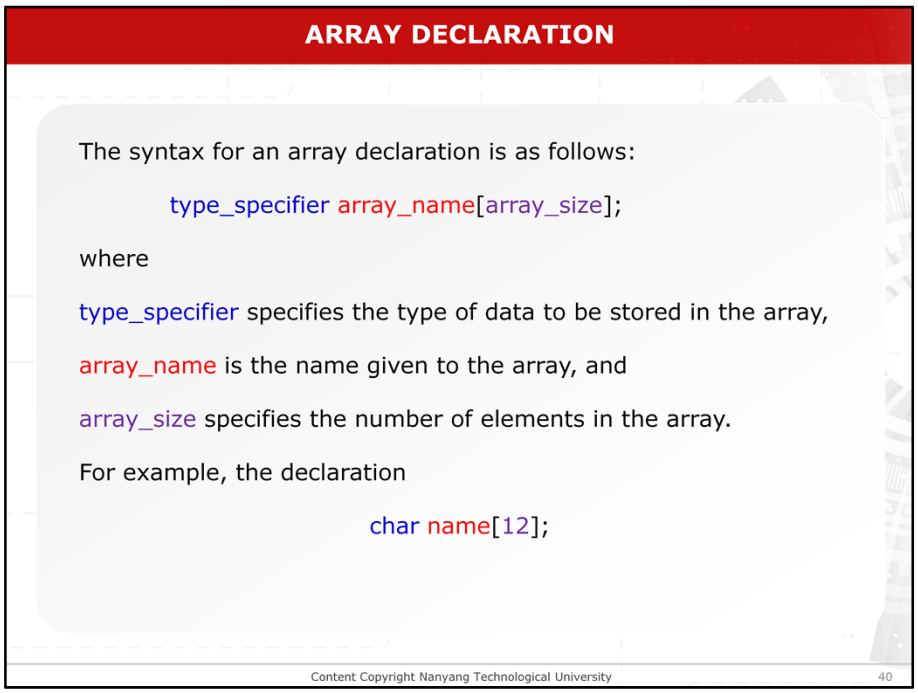
where **type specifier** specifies the type of data to be stored in the array, for example, character, integer, float etc.



Array name is the name given to the array

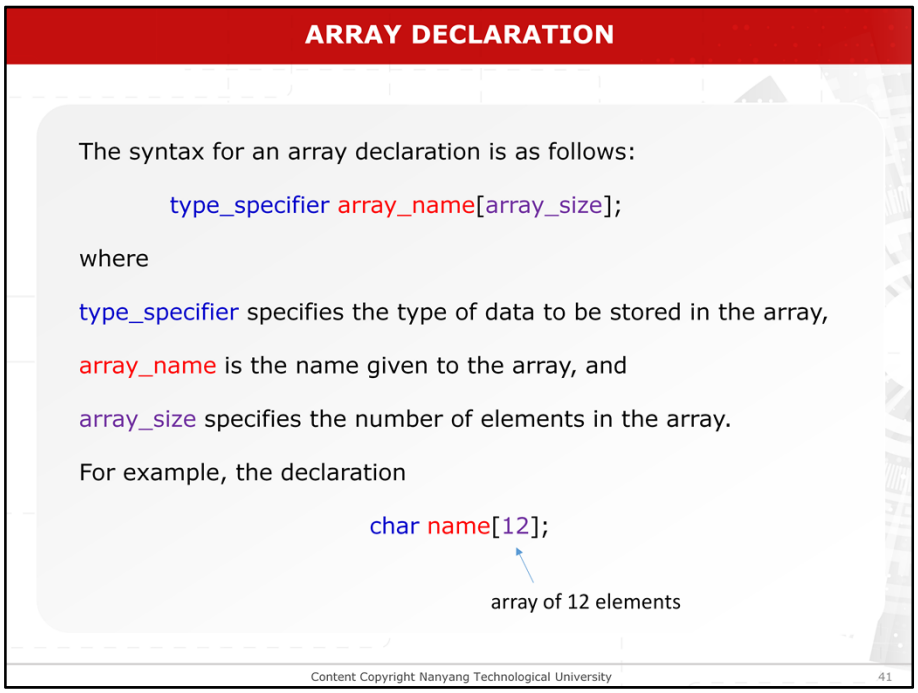


and **array size** specifies the number of elements in the array.

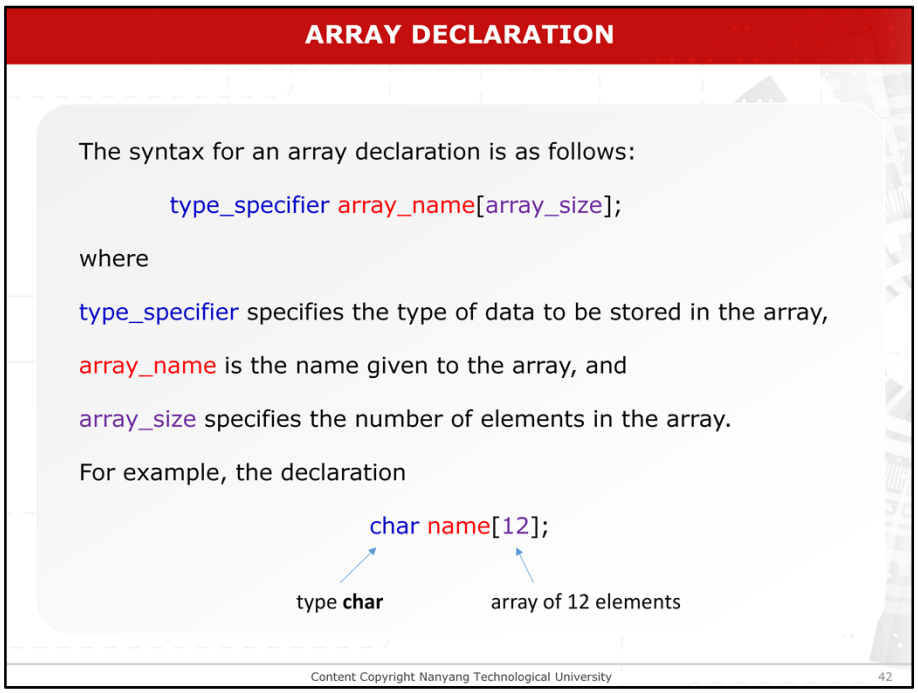


For example, the declaration

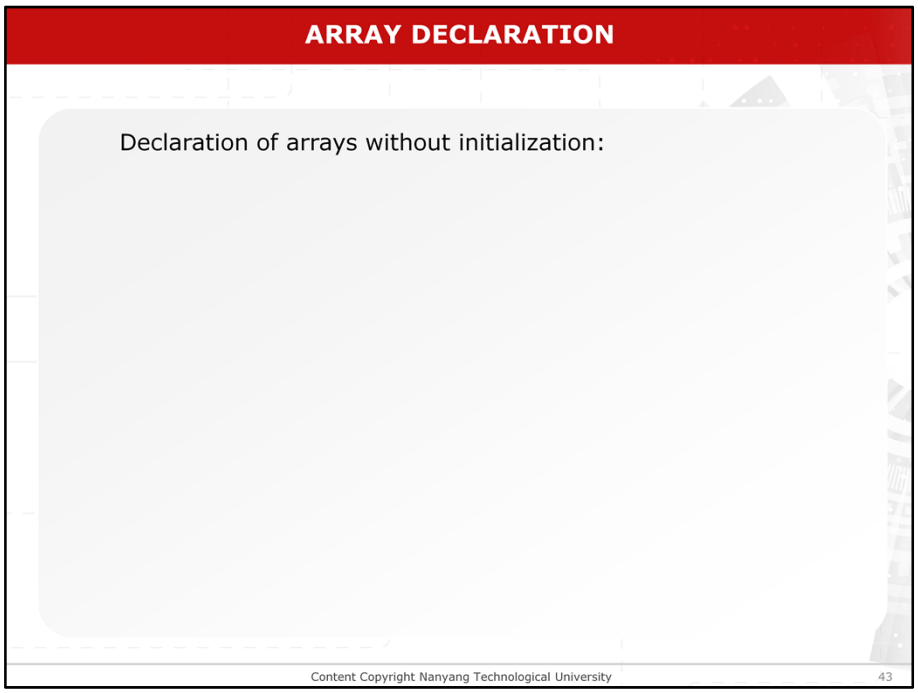
character name square bracket 12 square bracket;



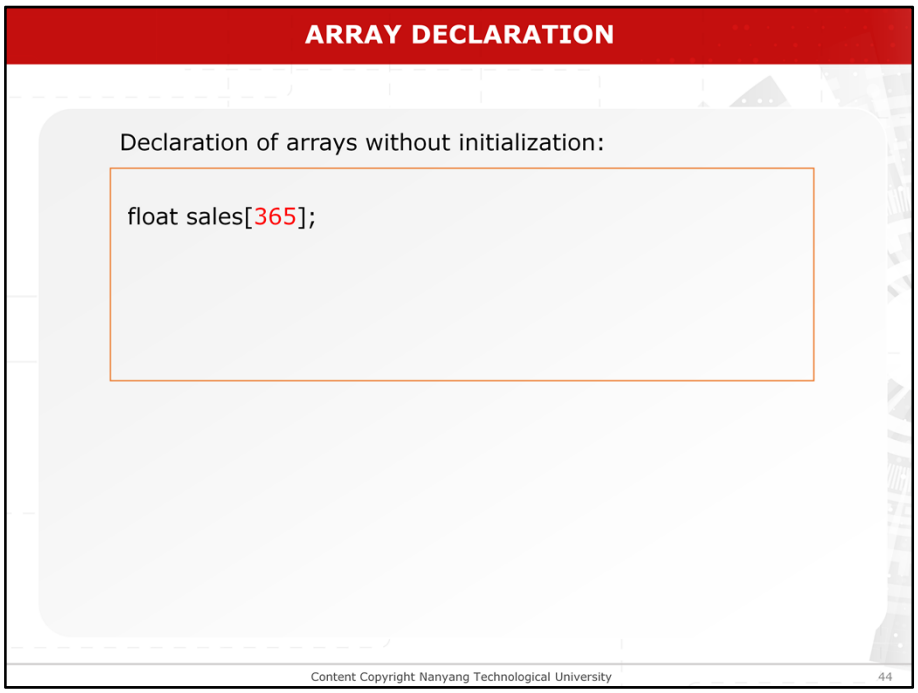
defines an array of 12 elements,



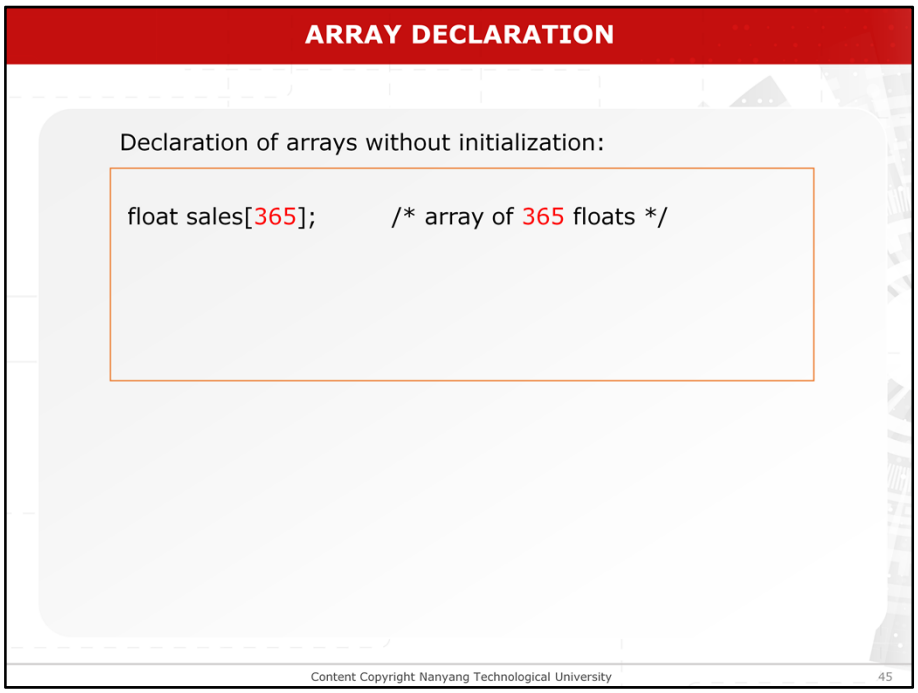
each element of the array stores data of type **character**. The elements are stored sequentially in memory. Each memory location in an array is accessed by a relative address called an *index* (or *subscript*).



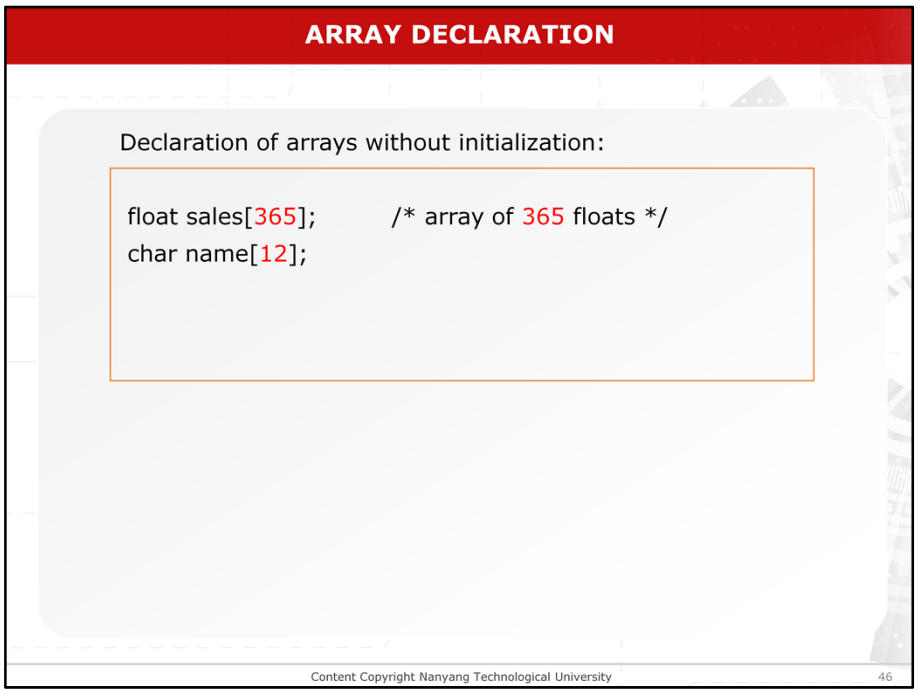
Arrays can be declared without initialization.
For example,



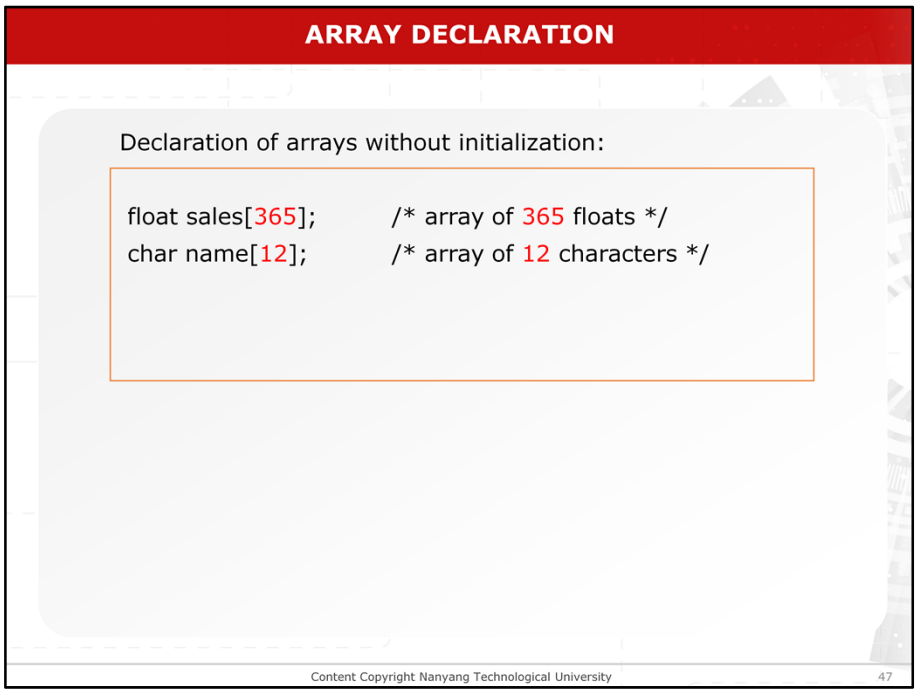
float sales square bracket 365 square bracket means



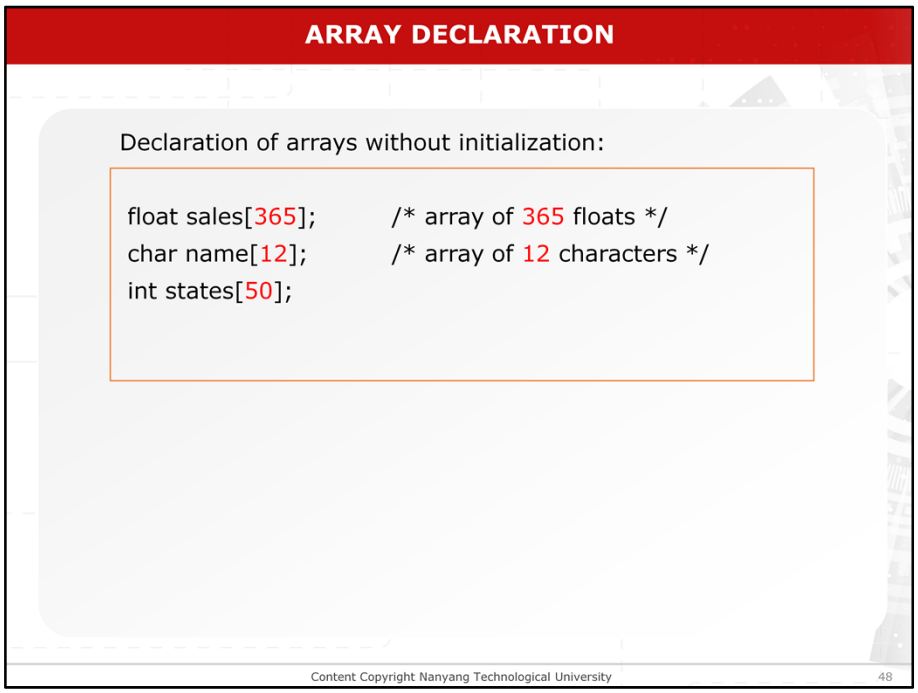
Declaring an **array of 365 floats.**



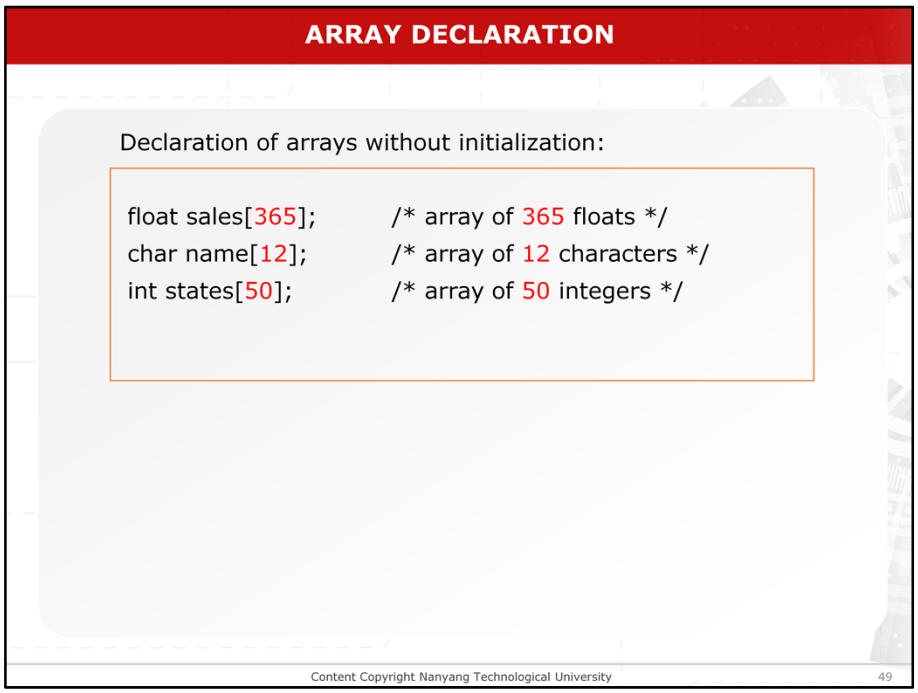
Character name 12 means



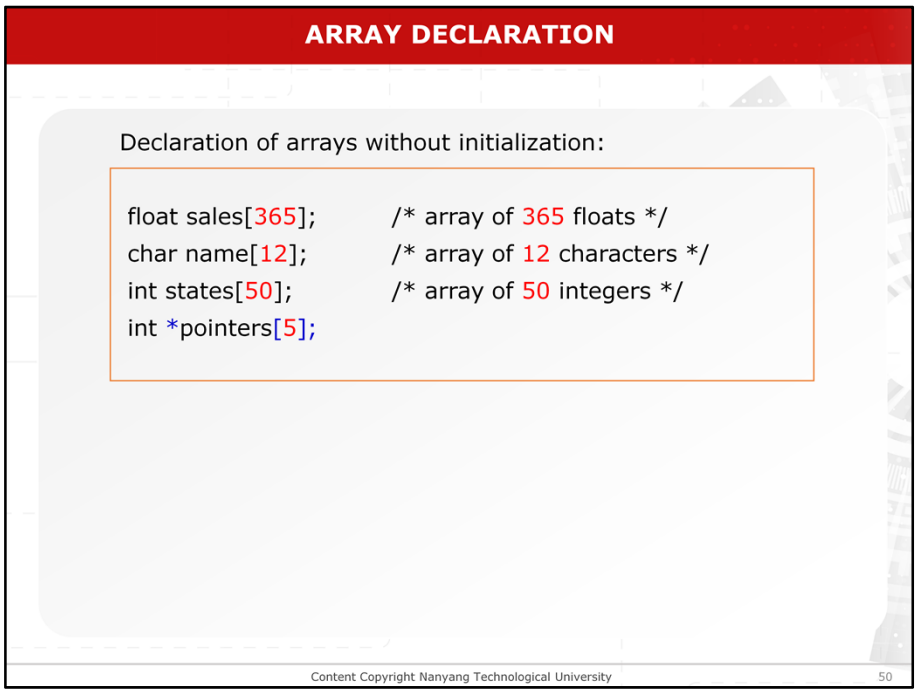
Declaring an array of 12 characters.



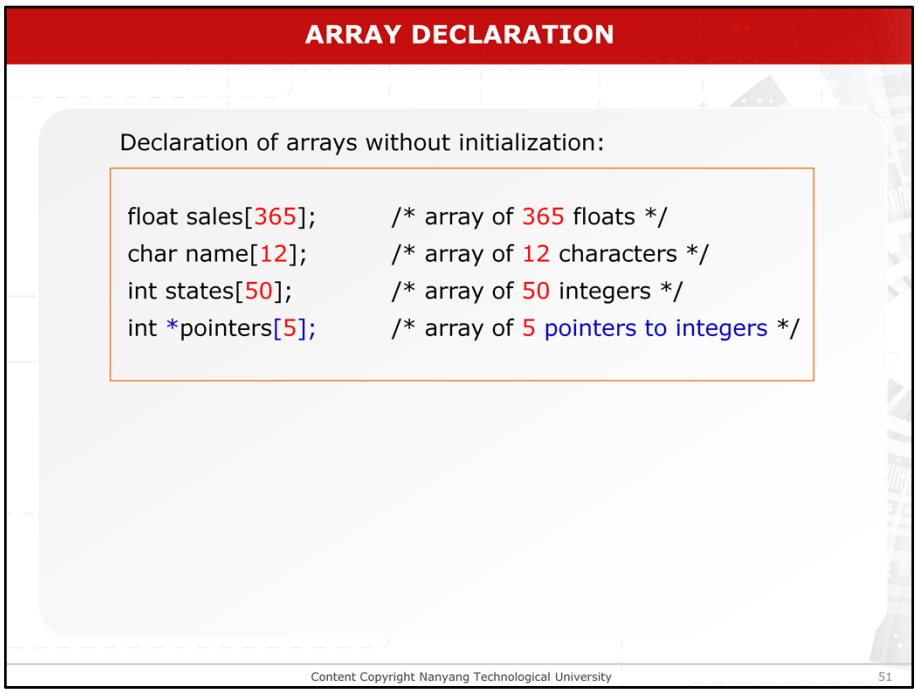
Integer states 50 means



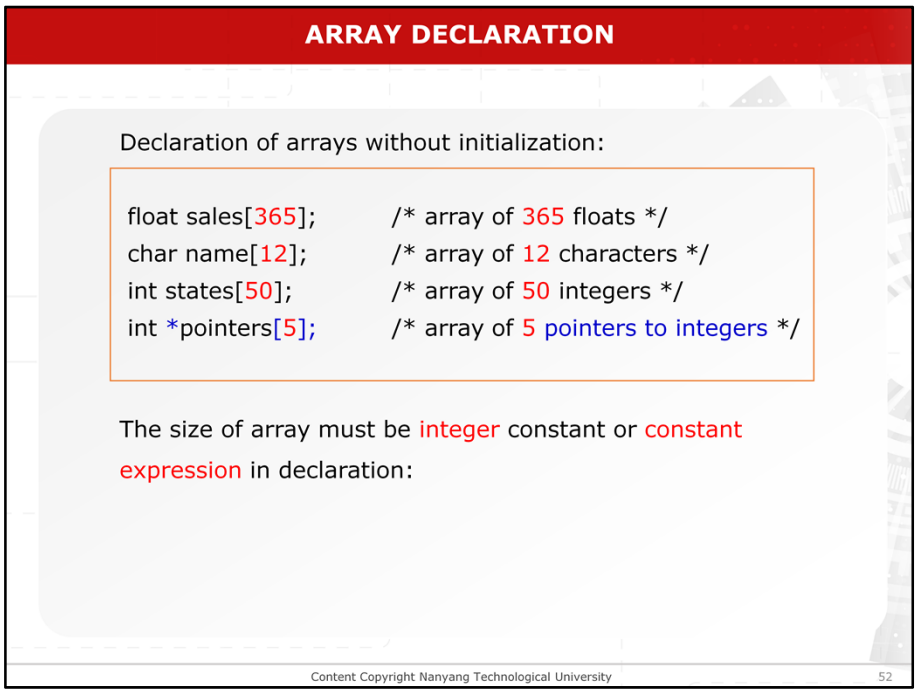
Declaring an array of 50 integers



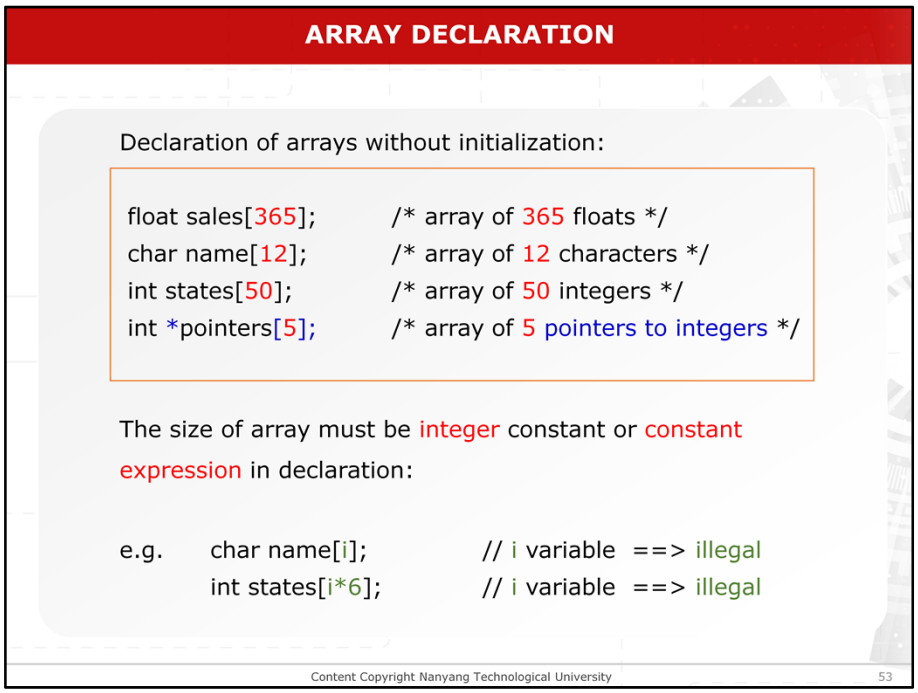
Integer asterisk pointers 5 means



Declaring an array of 5 pointers to integers



The size of array must be integer constant or constant expression in declaration.



Variables or expressions containing a variable cannot be used for the declaration of the size of the array.

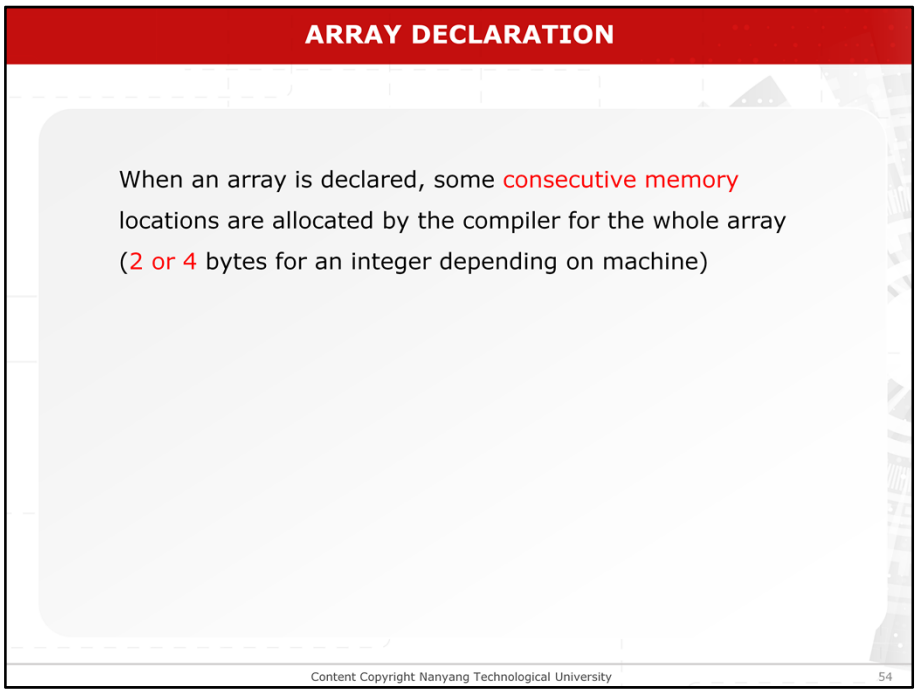
The following declarations

character name [i]; **and**

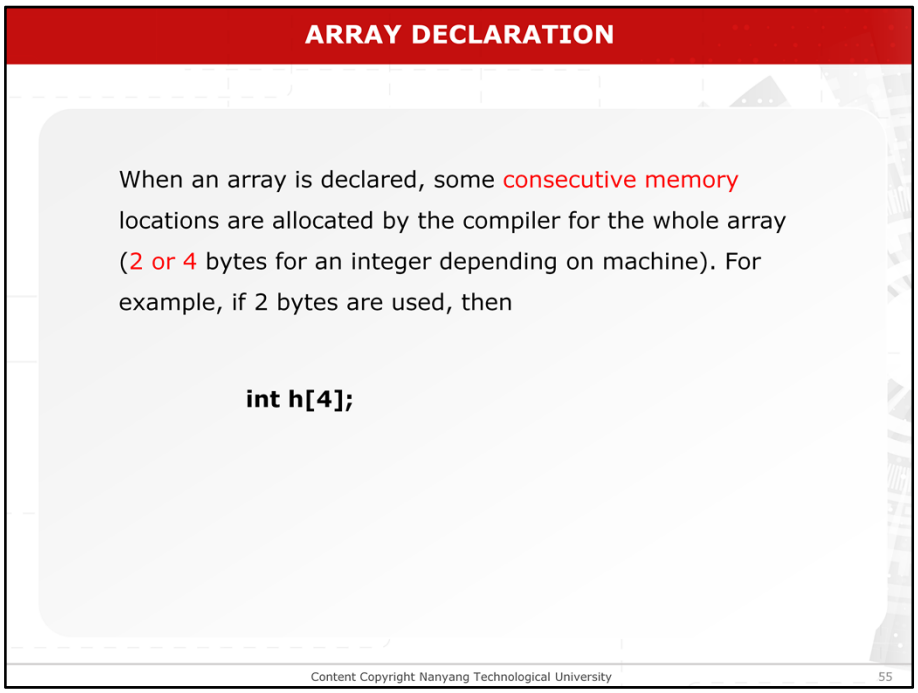
integer states [i mulitiply by 6];

are illegal.

In this case, i is a variable, not a constant thus it is ~~not~~ an illegal declaration.

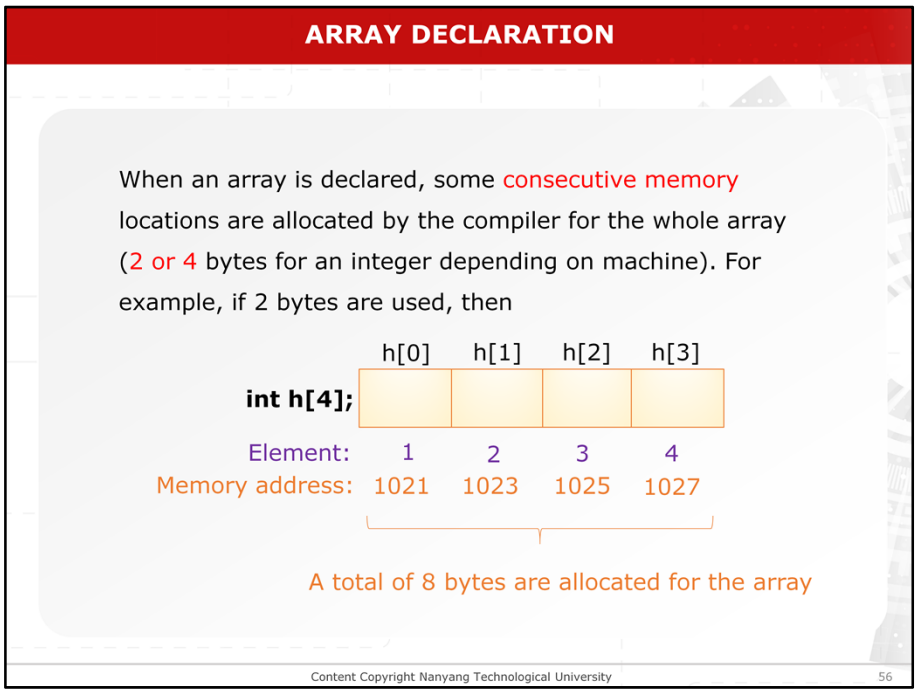


When an array is declared, **consecutive memory locations** for the number of elements specified in the array are allocated by the compiler for the whole array.



For example, if a system uses 2 bytes to store an integer, the declaration for the array

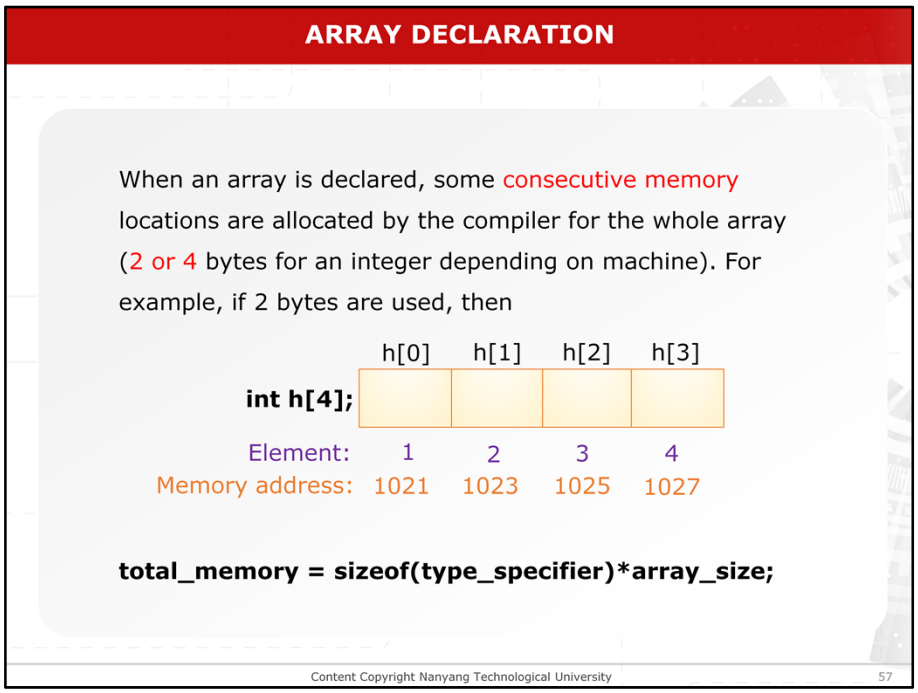
integer h[4];



will result in a total of 8 bytes allocated for the array.

The total number of bytes of storage allocated to an array will depend on the size of the array and the type of data items.

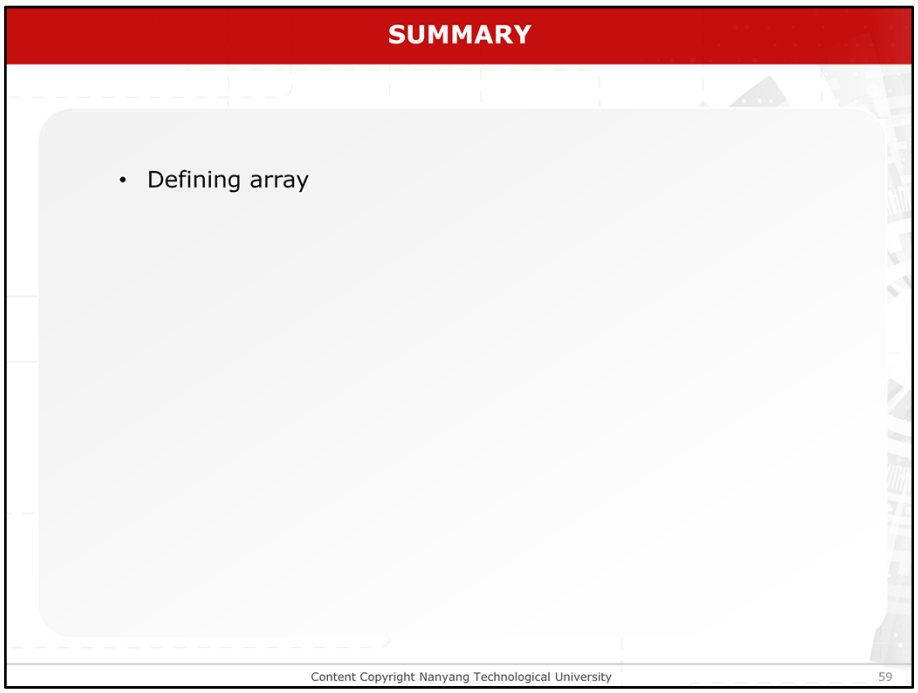
The size of memory required can be calculated using the following equation:



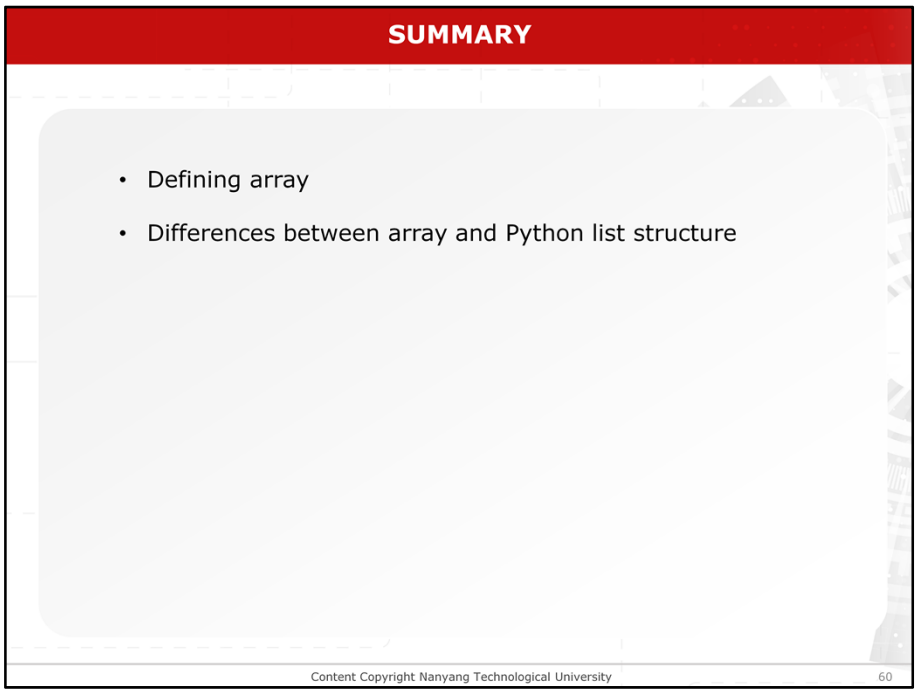
total memory equals size of (type specifier) multiply by array size;

where **size of** operator gives the size of the specified data type and **array size** is the total number of elements specified in the array.

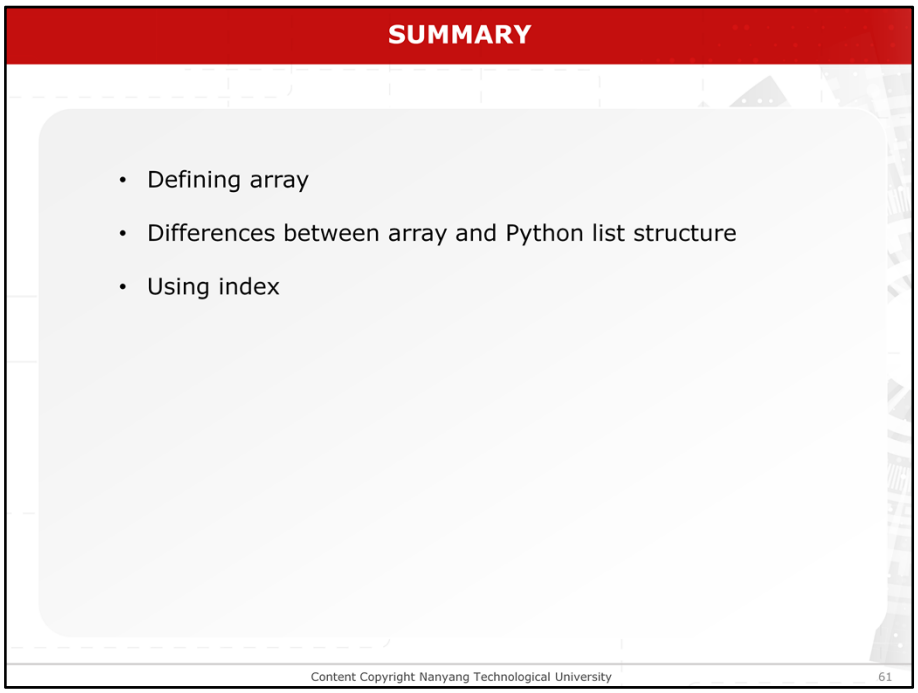
Let’s do a quick summary of the important points that have been covered.
We have learnt:



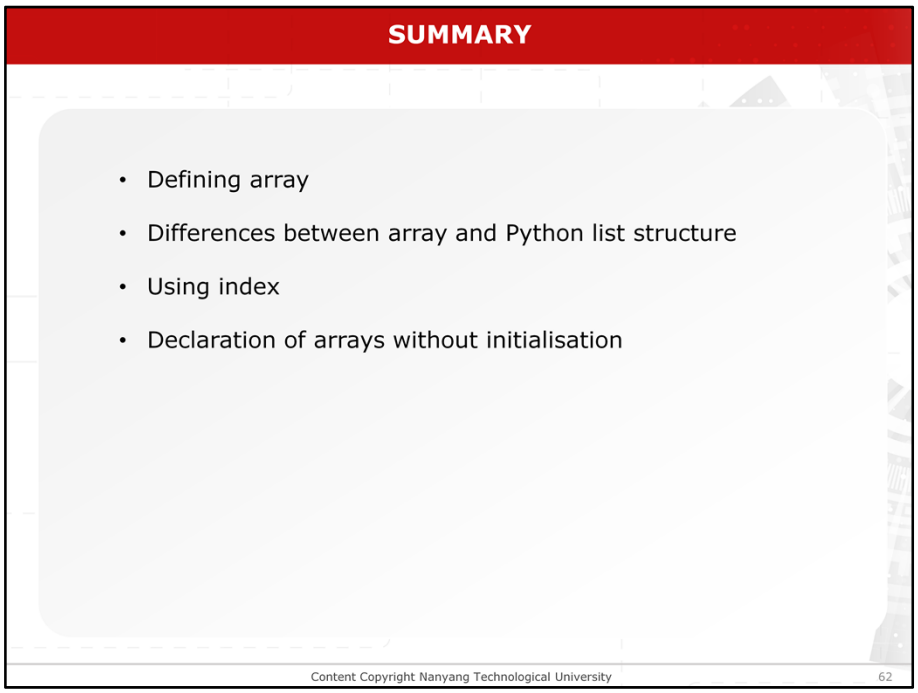
What is an array? Basically array is a list of values with the same data type.



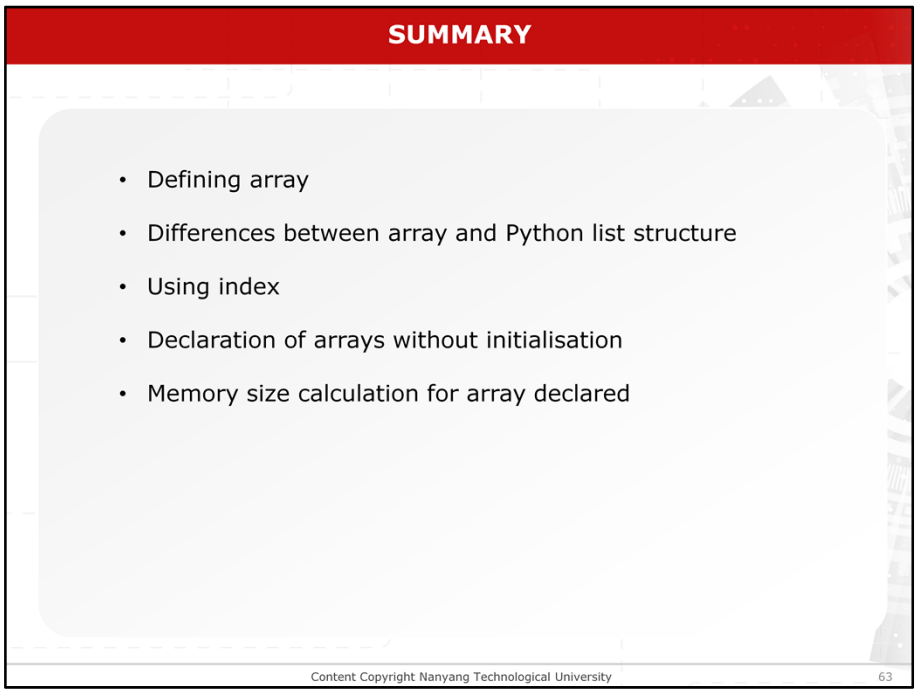
The differences between array and Python list structure.



Using index to reference an element in the array.



How to declare arrays without initialisation.



And the memory size calculation for array declared.

In the next video, we will look into how to initialize the arrays declared.