

“

Intro to Programming 2

ASSIGNMENT

Module 1– Arrays using Pseudocode

Manipulating Arrays

Define arrays

- Store, print, read and manipulate values in an array

Manipulating Arrays searching & sorting

- Distinguish between sequential search and binary search
- Demonstrate how binary search operates
- Demonstrate how selection sort operates

Working with Modules using Pseudocode

The user can perform the presentation on a projector or computer, and the powerpoint can be working with functions and procedures

- State advantages of modules
- Define functions
- Define procedures
- Distinguish between functions and procedures
- Write functions and procedures
- Write functions with parameters
- Write programs that make reference to at least one function
- Write programs that make reference to at least one procedure
- Test and trace a program design using a trace table

CONTEN

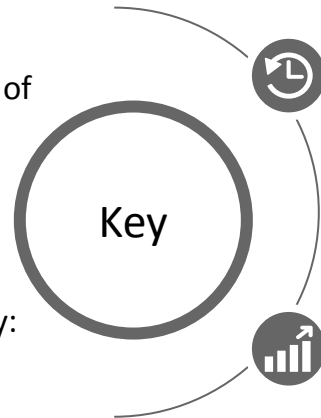


• Define Arrays

An array is a data structure that consists of a set of values of the same data type.

Format for declaring an array: `data_type array_name[number of values to hold];`

To refer to a specific element in the array: `array_name[subscript number].`



Can be initialized when they are declared

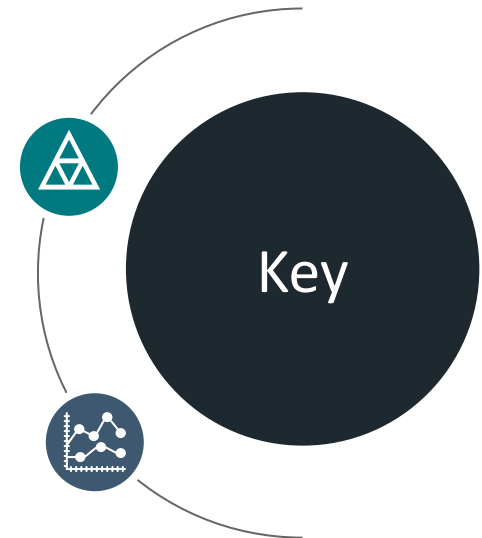
```
char vowels[5] = {'a', 'e', 'i', 'o', 'u'};
```

For numerical arrays – if all of the spaces are not initialized, the remaining ones are set to 0

```
int IntAr[10] = {2, 4, 6, 8};
```

The number of elements in the array is not always used when declaring an array

```
int IntAr[ ] = {3, 5, 7};
```



- Store, print, read, and manipulate values in an array.

Storing values in an array:
`numbers = [1, 2, 3, 4, 5]`

Storing values

Printing values in an array:
For number in numbers:
 `print(number)`

Printing values

Reading values from an array:
`first_number = numbers[0]`
`second_number = numbers[1]`

Reading values

`numbers[2] = 6` # change the
value at index 2 to 6.
`numbers.append(7)` # add 7
to the end of the array.

Manipulating values

`numbers.pop(0)` # remove
the first element from the
array.
`numbers.sort()` # sort the
array in ascending order.
`numbers.reverse()` # reverse
the order of the array.



an array.

```
if 5 in  
numbers:  
    print("5
```

Manipulating Arrays, Searching, and Sorting

```
    ),  
    else:  
        print("5  
is not in  
the  
array.")
```



Sorting an
array in
ascending
order:



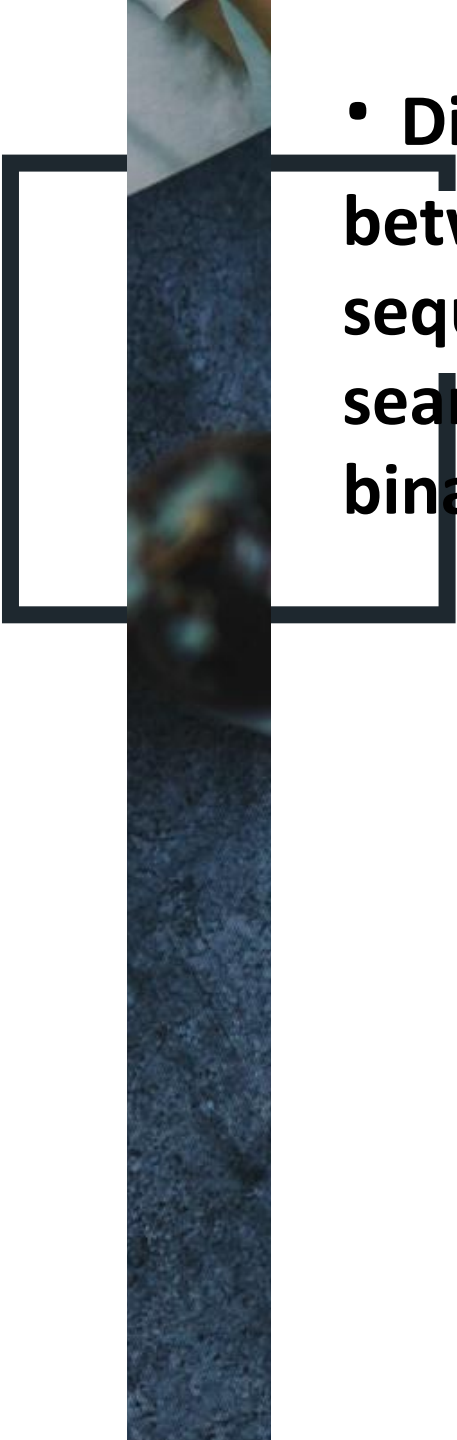
```
numbers.s  
ort()
```

Sorting an
array in
descendin
g order:



```
numbers.  
sort(rever
```





• Distinguish between sequential search and binary search.

01

Sequential search and binary search are two different algorithms used to search for an element in an array.

Sequential search is a simple method that searches through an array one element at a time, starting from the first element, until it finds the desired element or reaches the end of the array. This method is also known as "linear search." It is efficient for small arrays or arrays with a small number of elements.

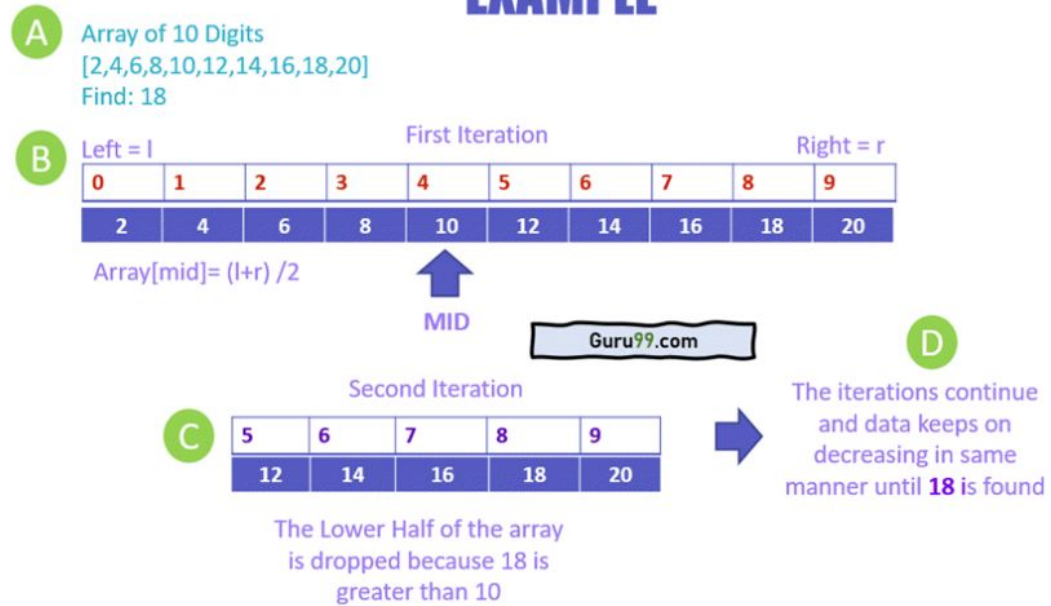
02

Binary search is a more advanced method that requires the array to be sorted. It starts by comparing the middle element of the array with the desired element. If the middle element is larger, it searches the left half of the array, and if the middle element is smaller, it searches the right half of the array. It then continues to divide the array in half and compare the middle element until it finds the desired element or determines that it is not in the array. This method is much more efficient than a sequential search for large arrays or arrays with a large number of elements.

03

In summary, sequential search is simple, efficient for small arrays, and also known as linear search. While binary search is more advanced, it is efficient for large arrays, and the array should be sorted.

EXAMPLE



A.

You have an array of sorted values ranging from 2 to 20 and need to locate 18.

B.

The average of the lower and upper limits is $(l + r) / 2 = 4$. The value being searched is greater than the mid which is 4.

C.

The array values less than the mid are dropped from search and values greater than the mid-value 4 are searched.

D.

This is a recurrent dividing process until the actual item to be searched is found.

Demonstrate how binary search operates.

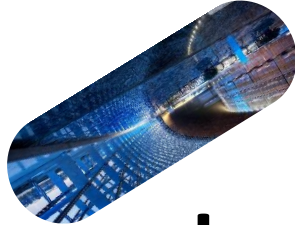
- Here are Some notable differences between Sequential Search and Binary Search.

Sequential Search	Binary Search
Sequential search is a simple technique of searching an element.	Binary Search is an efficient technique of searching an element.
Sequential Search does not require the list to be sorted	The binary search technique requires the list to be sorted.
Every element of the list is compared with the key element.	Only the midpoint element of the list is compared with a key element.
Algorithm is iterative in nature	Algorithm technique is Divide and Conquer.
The algorithmAlgorithm is easy to implement.	The algorithmAlgorithm is slightly complex.
Time complexity: $O(n)$.	Time complexity: $O(\log n)$.

- Binary search is an efficient algorithm for finding an item from a sorted list of items. It works by repeatedly dividing in half the portion of the list that could contain the item, until you've narrowed down the possible locations to just one.



- <https://www.youtube.com/watch?v=xrMppTpoqdw>



Demonstrate how selection sorting operates.



Selection sort works by taking the smallest element in an unsorted array and bringing it to the front. You'll go through each item (from left to right) until you find the smallest one. The first item in the array is now sorted, while the rest of the array is unsorted.



01

What are modules?

Modules are simply files with the “.py” extension that contain Python code that can be imported inside another Python program. In simple terms, we can consider a module to be the same as a code library or a file that contains a set of functions that you want to include in your application.



02

A modular program is easy to understand as the user has to analyze the individual modules which in itself are small programs, each performing a specific task rather than analyzing the entire program.



03

Many programmers can work at the same time on a large program as each individual module can be developed separately in isolation. Thus, parallel designing of the modules speeds up the program design job.



04

Modules may be developed in a general way so that these can be used with other programs also. In this way, a library of standard and commonly used modules can be developed for future use.



Define Functions

Key

- A function is a block of code that only runs when it is called, A function is defined in Python by using the def keyword, which is followed by the function's name, parentheses, and a colon.
- `def functionName():`

Function

- An example of a basic Python function printing “Hello World” to the terminal looks like this:
- `def greeting():`
- `print("Hello World")`

Words

- To call this function, write the name of the function followed by parentheses:
- `def greeting():`



- Define procedures

A procedure allows us to group a block of code under a name, known as a procedure name. We can call the block of code from anywhere in the program to execute the instructions it contains. We can also pass values to the procedure to change how it works.



```
def showMenu():  
    print('Main Menu')  
    print('1. Play game')  
    print('2. View high scores')  
    print('3. Quit')  
showMenu()
```

showMenu() is an example of a procedure call. We can call the procedure as many times as we wish in the program. A procedure needs to be defined earlier in the program than when it is called.



Show output:

Main Menu

1. Play game
2. View high scores
3. Quit





- Distinguish between functions and procedures.

A function would return the returning value to the code or calling function. A procedure, on the other hand, would return the control but would not return any value to the calling function or the code. We cannot use the DML statements in a function (such as Update, Delete, or Insert).

- Write functions and procedures

```
def func1 ():  
    Print ("I am writing a code")  
Func1()
```

Output

I am writing a code

Procedures

```
def storyStart(name):  
    print('Once upon a time, ' + name + ' was imprisoned in a  
castle.')  
    print('They were desperate to escape, but couldn\'t.')  
userName = input('What is your name? ')  
storyStart(userName)
```

Show Output

What is your name? Joe
Once upon a time, Joe was imprisoned in a castle.
They were desperate to escape, but couldn't.

- **Write functions with parameters**

```
def simple_addition (num1, num2):  
    answer = num1 + num2  
num1 = 3  
num2 = 5  
print('num1 is', num1, 'num2 is', num2)  
print (num1 + num2)  
simple_addition (num2=5, num1=3)
```

Output

8

Test and trace a program design using a trace table

Algorithm		Trace Table			
1	number = 3	Line	number	i	OUTPUT
2	PRINT number	1	3		
3	FOR i from 1 to 3:	2			3
4	number = number + 5	3		1	
5	PRINT number	4	8		
6	PRINT " ? "	5			8
		3		2	
		4	13		
		5			13
		3		3	
		4	18		
		5			18
		6			?

“

THANK
YOU

