|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name: |  | Date |  | Class |  |

# Data structures

In python, the data structures that you have seen are 1D arrays (list) and 2D arrays. For the A-level you are required to know more. The 1D array in Python is described as being a dynamic data structure.

An example of a 1D array in Python:

countries\_not\_in\_the\_eu = [“Switzerland”,”Norway”,”Serbia”,”Albania”,”Kosovo”,”UK”]

## Static and dynamic data structures

|  |
| --- |
| **Word bank** |
| size | memory | grow |
| heap | fixed | back |
| additional | monthly temperatures | efficient |

A static data structure is ............ in size and cannot increase in ............ or free up memory while the program is running. A programmer might choose a static array to store something like “the months of the year”, sales per month, ............ ............. The programmer will know how much ............ to allocate/how big the data structure will be.

A dynamic data structure refers to a collection of data in memory that has the ability to ............ or shrink in size. It does this with the aid of the **heap**. Unused blocks of memory are placed on a ............ (which can be used on the program). A dynamic data structure is able to take ............ memory off the heap if needed or place unused memory ............ on the heap. This is a more ............ use of resources and is often used in stacks, queues and binary trees. (the 1D array in Python is actually dynamic)

## On the dotted line, write static or dynamic depending on the description:

Inefficient as memory is allocated that may not be needed.....................

Fast access to each element of data as the memory location is fixed when the program is written..... ............

Efficient as the amount of memory varies as needed...........

Slower access to each element as the memory location is allocated at run time............

Memory addresses allocated will be contiguous so quicker to access.............

Memory addresses allocated may be fragmented so slower to access...........

Structures are a fixed size, making them more predictable to work with. For example, they can contain a header............

The relationship between different elements of data does not change...........

Structures vary in size so there needs to be a mechanism for knowing the size of the current structure............

The relationship between different elements of data will change as the program is run..........

## Stack

|  |
| --- |
| **Word bank** |
| static | top | variables |
| Last In, First Out | stack | size |

A ............ is a data structure that can be either ............ or dynamic. A stack is a ............ (LIFO) data structure. This means that, like a stack of plates in a cafe, items are added to the top and removed from the ............. When you use the Back button in your web browser, you will be taken back through the previous pages you looked at, in reverse order as their URLs are removed from the stack and reloaded. The same idea when you use Undo in Microsoft office. Stacks are also used in calculations.

A static data structure such as an array can be used with two additional ............, one being a pointer to the top of the stack and the other holding the ............ of the array (the maximum size of the stack)



The maximum size of the stack above is ....................

The LIFO in the stack above is.....................

## How stacks work

Here is a simplified example of a stack in use. Note that this stack can only store 6 data items.



The pointer is used to show where the top of the stack is



“Linda” has been pushed to the top of the stack so the pointer moves up

The stack is **popped** (see below) so the data at the pointer (“Linda”) is read and the pointer moves down



## Overflow and underflow

|  |
| --- |
| **Word bank** |
| indefinitely | stack pointer | overflow |
| maximum | underflow | onto |

A stack will always have a ............ size, because memory cannot grow ............. If the stack is implemented as an array, a full stack can be tested for by examining the values of the ............. An attempt to push another item ............ the stack would cause an ............ so an error message can be given to the user to avoid this, similarly if the stack pointer is -1, the stack is empty and ............ will occur if an attempt is made to pop an item.

## Operations on a stack

The following operations are required to implement a stack

* Push(item) adds a new item to the top of the stack
* Pop() removes and returns the top item from the stack
* Peek() returns the top item from the stack but does not remove it
* isEmpty() tests to see whether the stack is empty, and returns a Boolean value
* isFull() tests to see whether the stack is full, and returns a Boolean value

|  |  |  |
| --- | --- | --- |
| Stack operation | Stack contents | Return value |
| s.isEmpty | [] | True |
| s.push(‘Blue’) | [‘Blue’] |  |
| s.push(‘Red’) | [‘Blue’, ‘Red’] |  |
| s.push(‘Green’) | [‘Blue’, ‘Red’, ‘Green’] |  |
| s.isEmpty | [‘Blue’, ‘Red’, ‘Green’] | False |
| s.peek() | [‘Blue’, ‘Red’, ‘Green’] | ‘Green’ |
| s.pop() | [‘Blue’, ‘Red’] | ‘Green’ |

Using the above as an example, please complete the (grey bits) table:

|  |  |  |
| --- | --- | --- |
| Stack operation | Stack contents | Return value |
| s.isEmpty | [] |  |
| s.push(‘Arsenal’) | [‘Arsenal’] |  |
|  | [‘Arsenal’] | False |
| s.push(‘Barcelona’) |  |  |
|  | [‘Arsenal’, ‘Barcelona’, ‘Juventus’] |  |
| s.peek() | [‘Arsenal’, ‘Barcelona’, ‘Juventus’] |  |
| s.pop() | [‘Arsenal’, ‘Barcelona’] |  |
| s.pop() | [‘Arsenal’] |  |
|  |  | False |

## Questions:

1. A stack is LIFO, what is LIFO?
2. Is the memory in a computer ultimately limited or unlimited and how does this relate to a stack overflow?
3. Apart from stacks, list any other data structures that exist