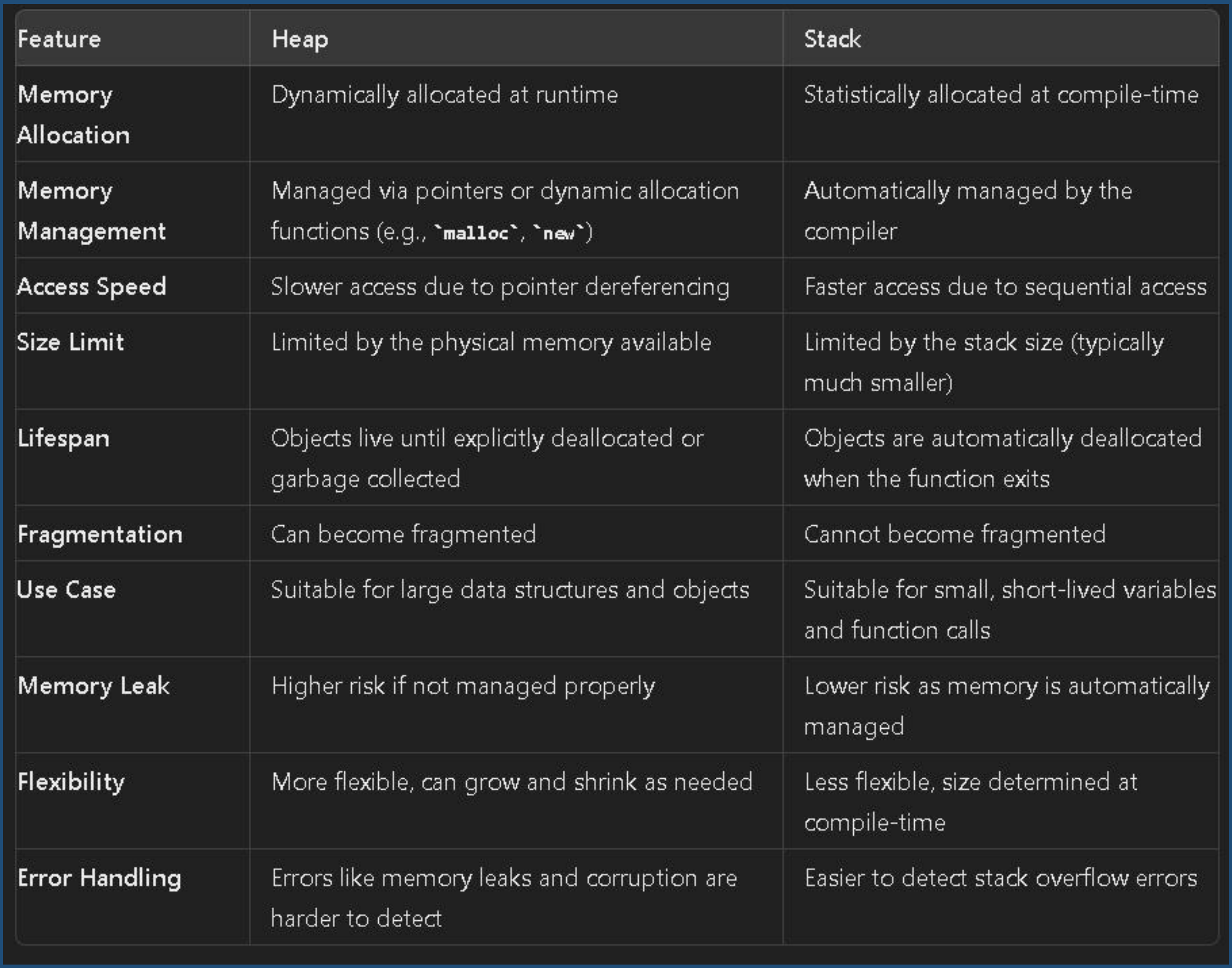
What's is difference between heap and stack?



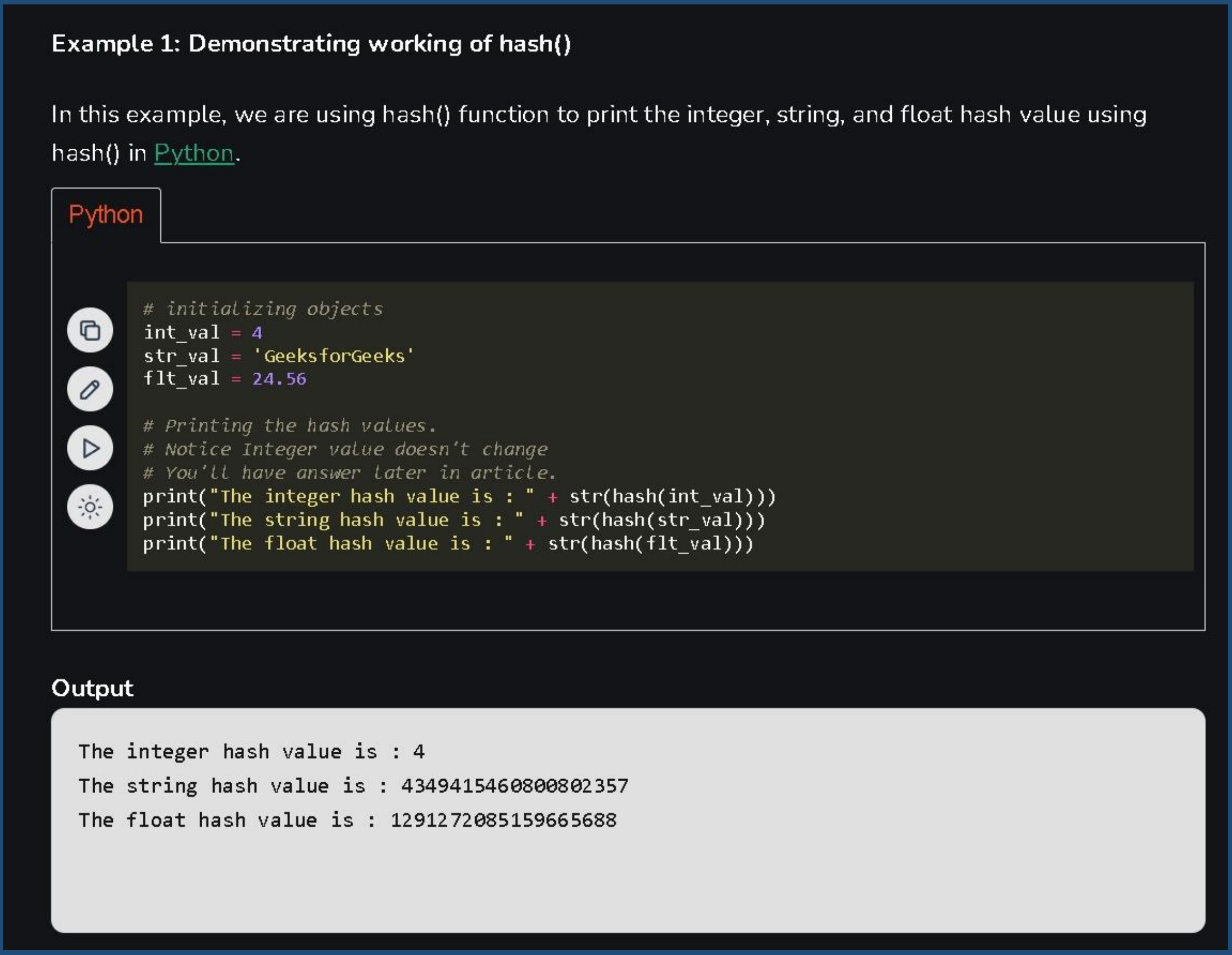
Feature	Heap	Stack
Memory Allocation	Dynamically allocated at runtime	Statistically allocated at compile-time
Memory Management	Managed via pointers or dynamic allocation functions (e.g., `malloc`, `new`)	Automatically managed by the compiler
Access Speed	Slower access due to pointer dereferencing	Faster access due to sequential access
Size Limit	Limited by the physical memory available	Limited by the stack size (typically much smaller)
Lifespan	Objects live until explicitly deallocated or garbage collected	Objects are automatically deallocated when the function exits
Fragmentation	Can become fragmented	Cannot become fragmented
Use Case	Suitable for large data structures and objects	Suitable for small, short-lived variables and function calls
Memory Leak	Higher risk if not managed properly	Lower risk as memory is automatically managed
Flexibility	More flexible, can grow and shrink as needed	Less flexible, size determined at compile-time
Error Handling	Errors like memory leaks and corruption are harder to detect	Easier to detect stack overflow errors

What is a hash function?

Python hash() function is a built-in function and returns the hash value of an object if it has one. The hash value is an integer that is used to quickly compare dictionary keys while looking at a dictionary.

Properties of hash() function

- **Objects hashed using hash() are irreversible, leading to loss of information.**
- **hash() returns hashed value only for immutable objects, hence can be used as an indicator to check for mutable/immutable objects.**
- **We can encode data for security in Python by using hash() function.**



Example 1: Demonstrating working of hash()

In this example, we are using hash() function to print the integer, string, and float hash value using hash() in Python.

Python

```python
# initializing objects
int_val = 4
str_val = 'GeeksforGeeks'
flt_val = 24.56

# Printing the hash values.
# Notice Integer value doesn't change
# You'll have answer later in article.
print("The integer hash value is : " + str(hash(int_val)))
print("The string hash value is : " + str(hash(str_val)))
print("The float hash value is : " + str(hash(flt_val)))
```

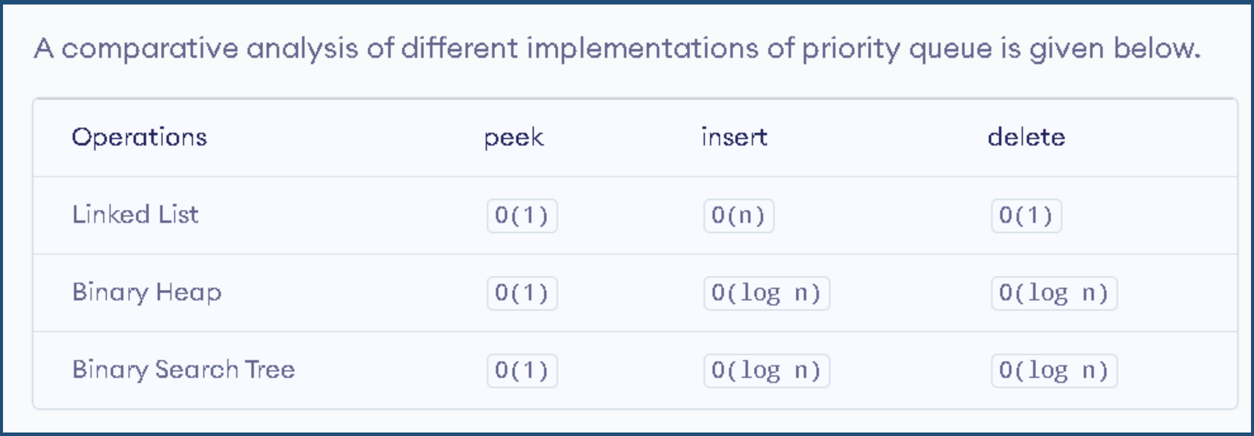
Output

```
The integer hash value is : 4
The string hash value is : 4349415460800802357
The float hash value is : 1291272085159665688
```

how memory store in pariority queue in python?

Priority queue can be implemented using an array, a linked list, a heap data structure, or a binary search tree. Among these data structures, heap data structure provides an efficient implementation of priority queues. Hence, we will be using the heap data structure to implement the priority queue in this tutorial.

Hence, we will be using the heap data structure to implement the priority queue in this tutorial. A max-heap is implemented in the following operations. If you want to learn more about it, please visit max-heap and min-heap.


A comparative analysis of different implementations of priority queue is given below.

Operations	peek	insert	delete
Linked List	`O(1)`	`O(n)`	`O(1)`
Binary Heap	`O(1)`	`O(log n)`	`O(log n)`
Binary Search Tree	`O(1)`	`O(log n)`	`O(log n)`