Glossary for This Course

A

- allocation, memory [**Why do we need to manage the memory?**; **Allocating memory**]
    - Reserving a chunk of memory for a certain bit of information
- arena (Python) [**Memory management in Python**]
    - The largest bits of memory, containing sequential blocks of memory
- automatic memory management [**Why do we need to manage the memory?**]
    - A system of managing memory that is built into the programming language and runtime; Python, Java, and JavaScript take care of memory management for you.
- automatic variables (C language) [**The C way: Allocating memory**]
    - Another name for local variables; they are automatically freed when no longer needed.

B

- block (Python) [**Memory management in Python**]
    - The actual memory addresses that are going to be assigned when doing memory management
- block starting symbol (C language) [**The C way: Allocating memory**]
    - Block starting symbol—a special area of memory in the C runtime for static and global variables that are un-initialized at compile time
- BSS [**The C way: Allocating memory**]
    - Block starting symbol—a special area of memory in the C runtime for static and global variables that are un-initialized at compile time

C

- call stack [**Stack memory**]
    - A stack that keeps track of the order of function calls
- compacting [**Actual removing or sweeping**]
    - Closing gaps in heap memory when removing objects
- concurrency [**Heap vs. stack memory**]
    - Multiple things happening at the same time
- copy, deep [**Best practices with memory**]
    - A copy of the object and its attributes, and their attributes, and so on; the data itself is copied.
- copy, shallow [**Best practices with memory**]

    - A copy of an object reference; the data referred to is not copied.
- CPU [**What is memory?**]
    - Central processing unit of a computer
- cyclic reference [**Garbage collection**]
    - Two objects that reference each other

D

- dangling pointer [**The C way: Deallocating and reallocating memory**]
    - A reference to a memory area that has been deallocated
- data segment (C language) [**The C way: Allocating memory**]
    - Data segment—an area of memory in the Cruntime for initialized global and static variables
- deallocation, memory [**Why do we need to manage the memory?**; **Deallocating memory**]
    - Releasing memory when information is no longer needed
- deep copy [**Best practices with memory**]
    - A copy of the object and its attributes, and their attributes, and so on; the data itself is copied.
- DS [**The C way: Allocating memory**]
    - Data segment—an area of memory in the Cruntime for initialized global and static variables
- dynamic memory allocation [**Why do we need to manage the memory?**; **Allocating memory**]
    - Memory allocation that occurs during runtime

E

- error, out of memory [**Out of Memory error**]
    - An error that occurs when the system cannot allocate a requested memory block anymore
- error, stack overflow [**Stack memory**]
    - An error that occurs when a program exceeds the size limit of stack memory
- escaping references [**Best practices with memory**]
    - The situation when an object is accessed unintentionally through another function

F

- fragmentation, heap [**Deallocating memory**]
    - The situation where gaps occur in the heap when memory is allocated and deallocated

- free (C language) [**The C way: Deallocating and reallocating memory**]
    - A function that deallocates memory

G

- garbage collection [**Deallocating memory**]
    - A process by which the runtime system keeps track of which items in memory are no longer needed and deletes them

H

- heap fragmentation [**Deallocating memory**]
    - The situation where gaps occur in the heap when memory is allocated and deallocated
- heap memory [**Heap memory**]
    - A part of RAM used for storing values that need to be accessed throughout the entire application; it also holds larger values that might not fit in the limited stack memory.

M

- main memory [**What is memory?**]
    - Analogous to human short-term memory; when you shut down a computer, the main memory gets erased. This is called volatile.
- malloc (C language) [**The C way: Allocating memory**]
    - A function that allocates heap memory
- manual memory management [**Why do we need to manage the memory?**]
    - A system of managing memory in which the developer has to allocate and deallocate memory; this is the way C and C++ work.
- memory allocation [**Why do we need to manage the memory?**; **Allocating memory**]
    - Reserving a chunk of memory for a certain bit of information
- memory allocation, dynamic [**Why do we need to manage the memory?**; **Allocating memory**]
    - Memory allocation that occurs during runtime
- memory allocation, static [**Why do we need to manage the memory?**; **Allocating memory**]
    - Memory allocation that occurs before a program is executed
- memory deallocation [**Why do we need to manage the memory?**; **Deallocating memory**]
    - Releasing memory when information is no longer needed
- memory, heap [**Heap memory**]
    - A part of RAM used for storing values that need to be accessed throughout the entire application; it also holds larger values that might not fit in the limited stack memory.

- memory leak [**Garbage collection**]
    - A situation where memory should be deallocated but has not and therefore cannot be reached anymore
- memory management [**Why do we need to manage the memory?**]
    - The allocation and releasing of memory
- memory management, automatic [**Why do we need to manage the memory?**]
    - A system of managing memory that is built into the programming language and runtime; Python, Java, and JavaScript take care of memory management for you.
- memory management, manual [**Why do we need to manage the memory?**]
    - A system of managing memory in which the developer has to allocate and deallocate memory; this is the way C and C++ work.
- memory profiler [**Best practices with memory**]
    - A program that lets you see how memory is used when a program runs
- memory, stack [**Stack memory**]
    - A stack that stores the variables created by functions; when entering a function, memory is allocated on the stack. When the function is done, the stack memory is deallocated.
- multi-threading [**Memory management in Python**]
    - An approach to writing code that executes multiple threads (tasks) concurrently

O

- out of memory error [**Out of Memory error**]
    - An error that occurs when the system cannot allocate a requested memory block anymore

P

- pointer (C language) [**The C way: Allocating memory**]
    - A reference to a memory address where a value is stored
- pool (Python) [**Memory management in Python**]
    - A subdivision of an arena containing blocks of memory of a specific same size
- profiler, memory [**Best practices with memory**]
    - A program that lets you see how memory is used when a program runs

R

- RAM [**What is memory?**]
    - Random-access memory; see main memory.

- realloc (C language) [**The C way: Deallocating and reallocating memory**]
    - A function that re-allocates memory
- reference counter [**Garbage collection**; **Memory management in Python**]
    - A counter that tells how many references exist to an object on the heap
- references, escaping [**Best practices with memory**]
    - The situation when an object is accessed unintentionally through another function
- ROM [**What is memory?**]
    - Read-only memory—slower than RAM but is non-volatile and doesn't disappear when the computer is shut down; usually used for the logic needed to start a computer and operating system

S

- secondary storage [**What is memory?**]
    - Analogous to human long-term memory. This storage persists even when the computer is shut down. An example is files and programs on your hard drive. This is called non-volatile.
- shallow copy [**Best practices with memory**]
    - A copy of an object reference; the data referred to is not copied.
- stack, call [**Stack memory**]
    - A stack that keeps track of the order of function calls
- stack memory [**Stack memory**]
    - A stack that stores the variables created by functions; when entering a function, memory is allocated on the stack. When the function is done, the stack memory is deallocated.
- stack overflow error [**Stack memory**]
    - An error that occurs when a program exceeds the size limit of stack memory
- static memory allocation [**Why do we need to manage the memory?**; **Allocating memory**]
    - Memory allocation that occurs before a program is executed
- sweeping [**Actual removing or sweeping**]
    - The actual removal of objects on the heap during garbage collection

T

- text segment (C language) [**The C way: Allocating memory**]
    - An area of memory for compiled code that is being stored
- thread [**Heap vs. stack memory**]

    - A part of execution within a concurrent application; every thread has its own stack.

V

- virtual memory [**What is memory?**]
    - Secondary storage used to manage data that cannot fit into main memory