Storage Classes:
1 ) Default Value: When memory allocated by compiler, for a variable,what is the value of that memory location.
2 ) Place of Storage:Whether the variable is stored in stack / data segment.
3 ) Scope:The area of program, in which an identifier can be used/accessed.
4 ) Lifetime: The time period between memory allocation and deallocation for a variable, is called its lifetime.

Variable Declaration: <scs> <dt> var;

<scs> : storage class specifier - tells you about the variable's properties like default value, place of storage, scope and lifetime.

automatic storage class:
applied by default to any local variable declared without a storage class specifier. Here auto keyword is automatically applied.
keyword : auto
place of storage: stack
default value : garbage value

an auto variable will be recreated everytime it is declared. once it is out of its scope, its memory is deallocated.

Auto variable cannot be used out of its scope. Within its scope, it can be used in any sub scope.
Two variables of same name cannot exist in same scope.
you can declare a variable of same name, in a sub scope.
And the nearest variable declaration always takes priority, and after coming out of that scope, again the outer scope variable is considered back.

external storage class:
applied by default to any globally declared variable, without a storage class specifier.extern keyword is not automatically applied. It has a different meaning.
keyword : extern
place of storage: data
default value : 0

Any variable declared in the global area, without any storage class, is by default an external variable.
It is created and initialised only once, on data segment.
its scope starts from the line of declaration, and ends with file. If you want to use the external variable out of its scope, then you must redeclare the variable with extern keyword.
extern declaration is a duplicate declaration. You should not initialise a variable in extern declaration.
extern declaration can be local to a function or global to several functions.
extern variable can be superseded by a auto variable of same name, as auto variable will be the nearest declaration.
A function declaration automatically contains extern keyword as function definitions are global, and they also need scope extension. This is in gcc.

static storage class:
keyword : static
default value : 0
Place of storage: Data Segment
Lifetime : Till end of the program
Scope:
local static variable : Scope is with in block of declaration.
global static variable : Scope starts from line of declaration, till end of file.
Scope of a global static variable cannot be extended to other functions or files.

Another usage of static keyword is static functions:
a static function can be called only after its definition, from the same file. It cannot be called in other files.

static variables will be created / initialised only once. and the value persists between function calls.

register storage class:
keyword : register
default value:garbage value
place of storage: CPU registers, if they are available. Otherwise on stack.
scope: is with in the block of declaration.
lifetime : with in the function.
Scope and lifetime of a register storage class are similar to auto storage class.
if CPU register is not available, a register storage class variable will work like a auto storage class variable.
register storage class variable's does not have any address. So we cannot apply addressof(&) operator on register variables.

When to use what?

auto : When you need to keep your data for a short duration, and you want to use it only with in a part of the program.
external : When you need to have the data stay till end of the program, and you want to use it across multiple functions and files.
static: When you need to have the data stay till end of the program, but you want to limit its usage to only a function or file.
register : When a variable is going to be very frequently accessed, it is better to store it on register, instead of memory to save some processing time.