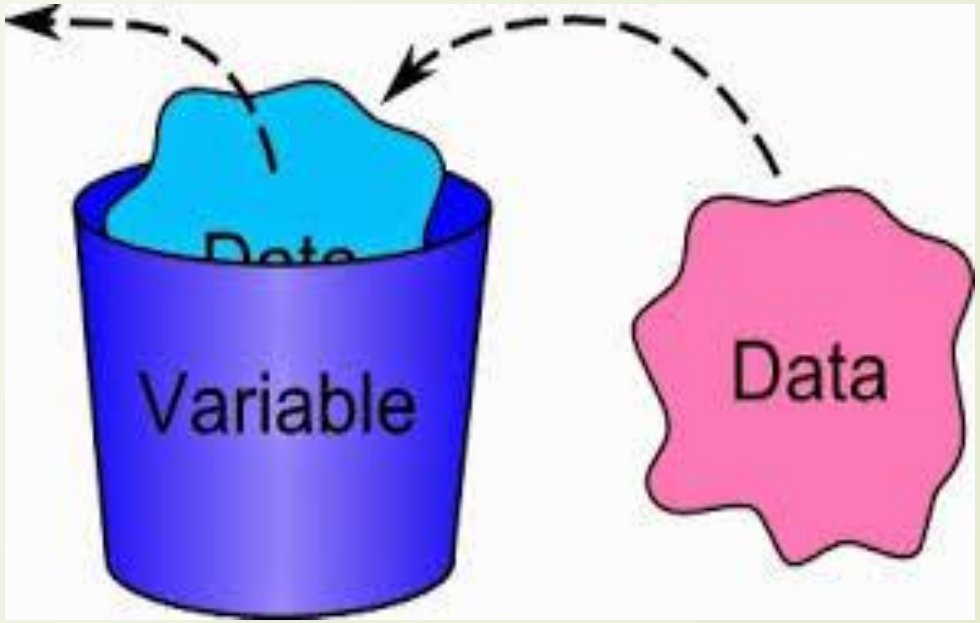
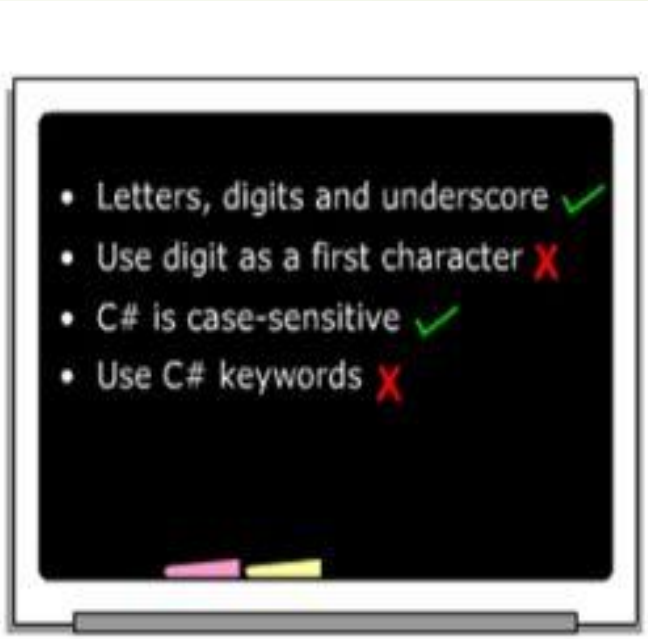
What is variable



Rules to create variable



Variable Name	Valid/Invalid
Employee	Valid
student	Valid
_Name	Valid
Emp_Name	Valid
@goto	Valid
static	Invalid as it is a keyword
4myclass	Invalid as a variable cannot start with a digit
Student&Faculty	Invalid as a variable cannot have the special character &

Declaring Variables

- To declare variables in C# you need to use the pattern:

```
{data type / var} {variable name} = {value};
```

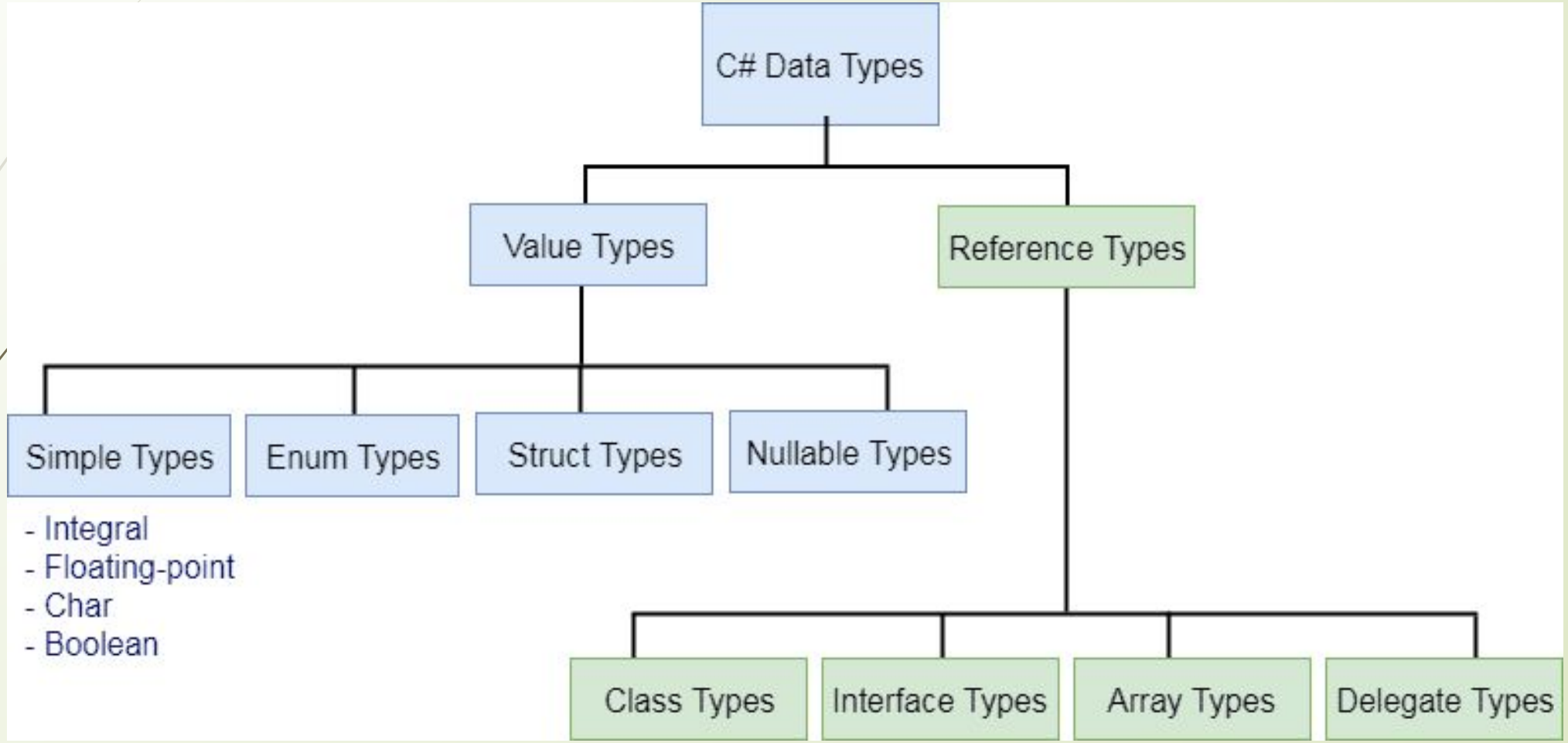
- Examples:

```
var firstNumber = 5;
var name = "Pesho";
var isPassed = false;
var gender = 'F';
var mathGrade = 5.49;
```

```
int firstNumber = 5;
string name = "Pesho";
bool isPassed = false;
char gender = 'F';
double mathGrade = 5.49;
```

- Type is inferred from the right side of the expression (use **var**)

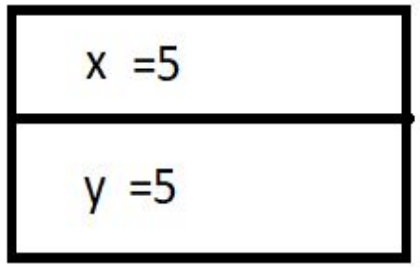
Data types


C# Data Types
Value Types
Reference Types
Simple Types
Enum Types
Struct Types
Nullable Types
- Integral
- Floating-point
- Char
- Boolean
Class Types
Interface Types
Array Types
Delegate Types

Value and Reference

VALUE

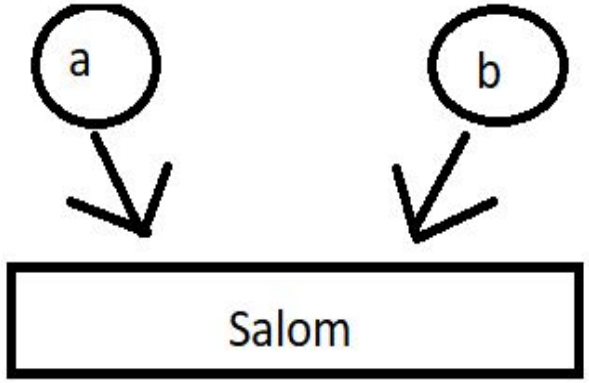
int x = 5; int y =x;



Stack

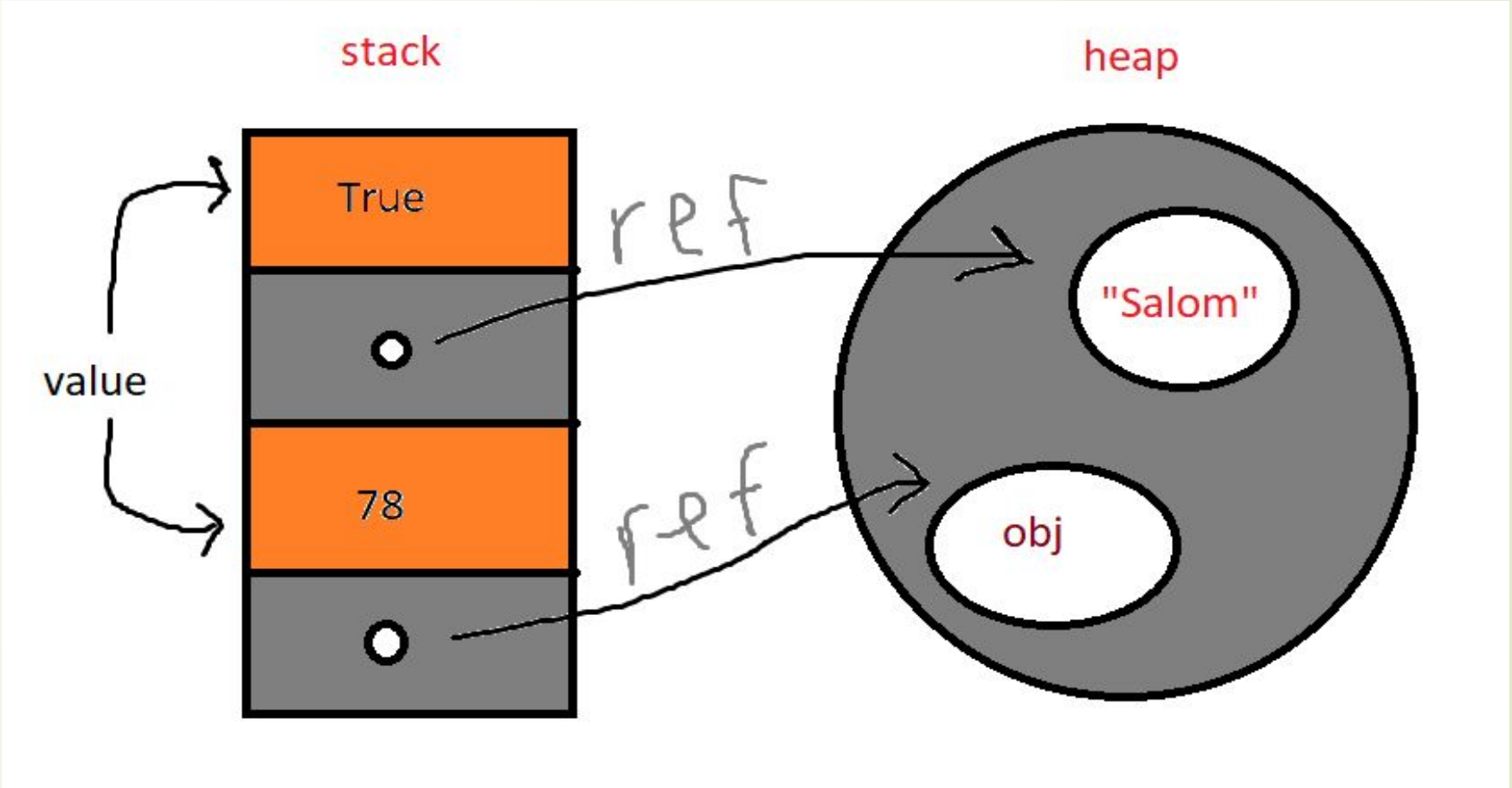
REFERENCE

string a ="Salom"; string b=a;



Heap

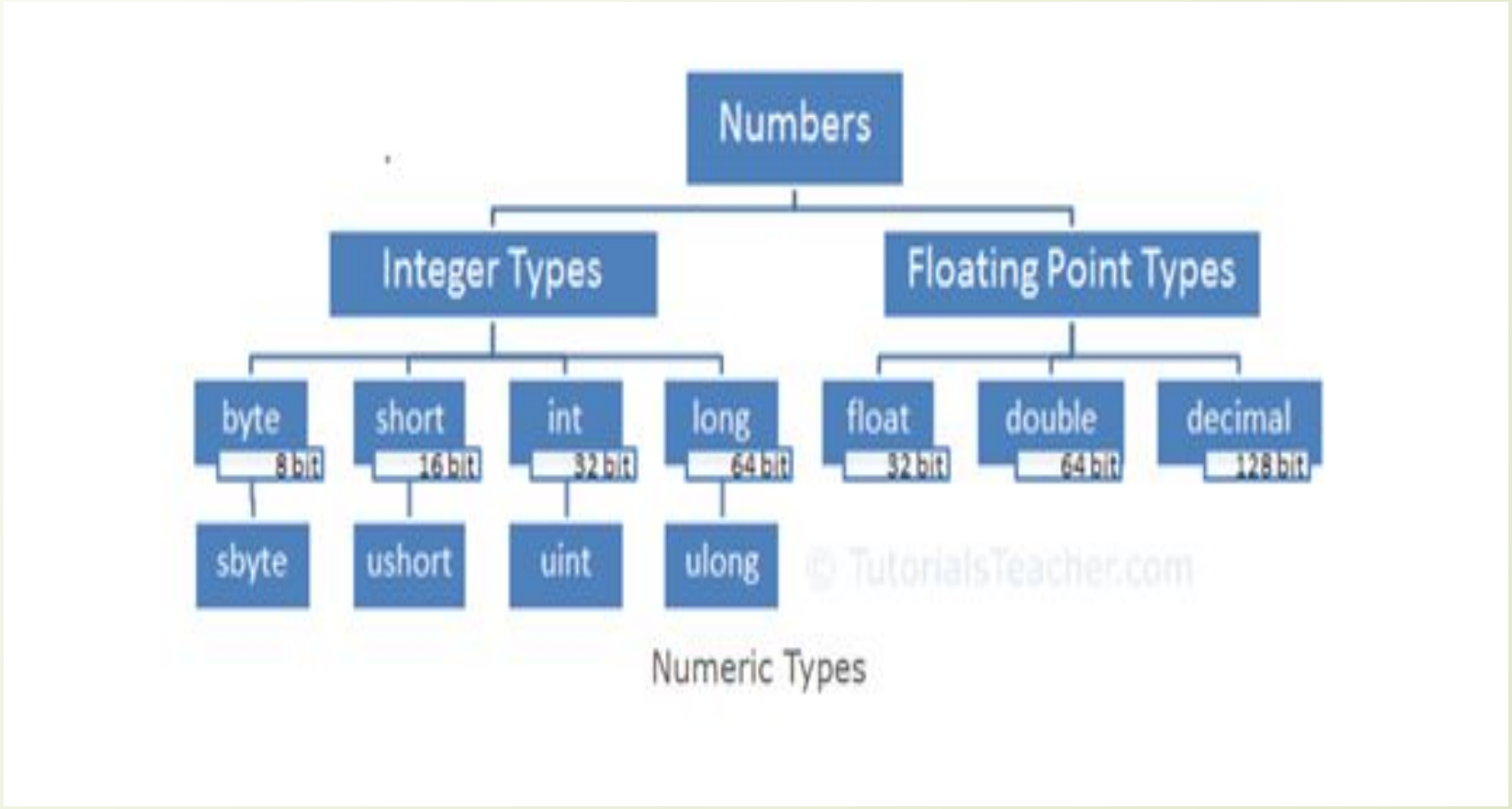
Stack and Heap



Key difference

- Stack is a linear data structure whereas Heap is a hierarchical data structure.
- Stack memory will never become fragmented whereas Heap memory can become fragmented as blocks of memory are first allocated and then freed.
- Stack accesses local variables only while Heap allows you to access variables globally.
- Stack variables can't be resized whereas Heap variables can be resized.
- Stack memory is allocated in a contiguous block whereas Heap memory is allocated in any random order.
- Stack doesn't require to de-allocate variables whereas in Heap de-allocation is needed.
- Stack allocation and deallocation are done by compiler instructions whereas Heap allocation and deallocation is done by the programmer.

Numeric types



Numeric Types

Description of some advanced types

Type	Description	Range	Suffix
byte	8-bit unsigned integer	0 to 255	
sbyte	8-bit signed integer	-128 to 127	
short	16-bit signed integer	-32,768 to 32,767	
ushort	16-bit unsigned integer	0 to 65,535	
int	32-bit signed integer	-2,147,483,648 to 2,147,483,647	
uint	32-bit unsigned integer	0 to 4,294,967,295	u
long	64-bit signed integer	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	l
ulong	64-bit unsigned integer	0 to 18,446,744,073,709,551,615	ul
float	32-bit Single-precision floating point type	-3.402823e38 to 3.402823e38	f
double	64-bit double-precision floating point type	-1.79769313486232e308 to 1.79769313486232e308	d
decimal	128-bit decimal type for financial and monetary calculations	(+ or -)1.0 x 10e-28 to 7.9 x 10e28	m

Definition

- Integer Types:

Integer type numbers are positive or negative whole numbers without decimal points. C# includes four data types for integer numbers: ***byte, short, int, and long.***

- Floating Point Types:

Floating-point numbers are positive or negative numbers with one or more decimal points. C# includes three data types for floating-point numbers: ***float, double, and decimal***.