Diploma in

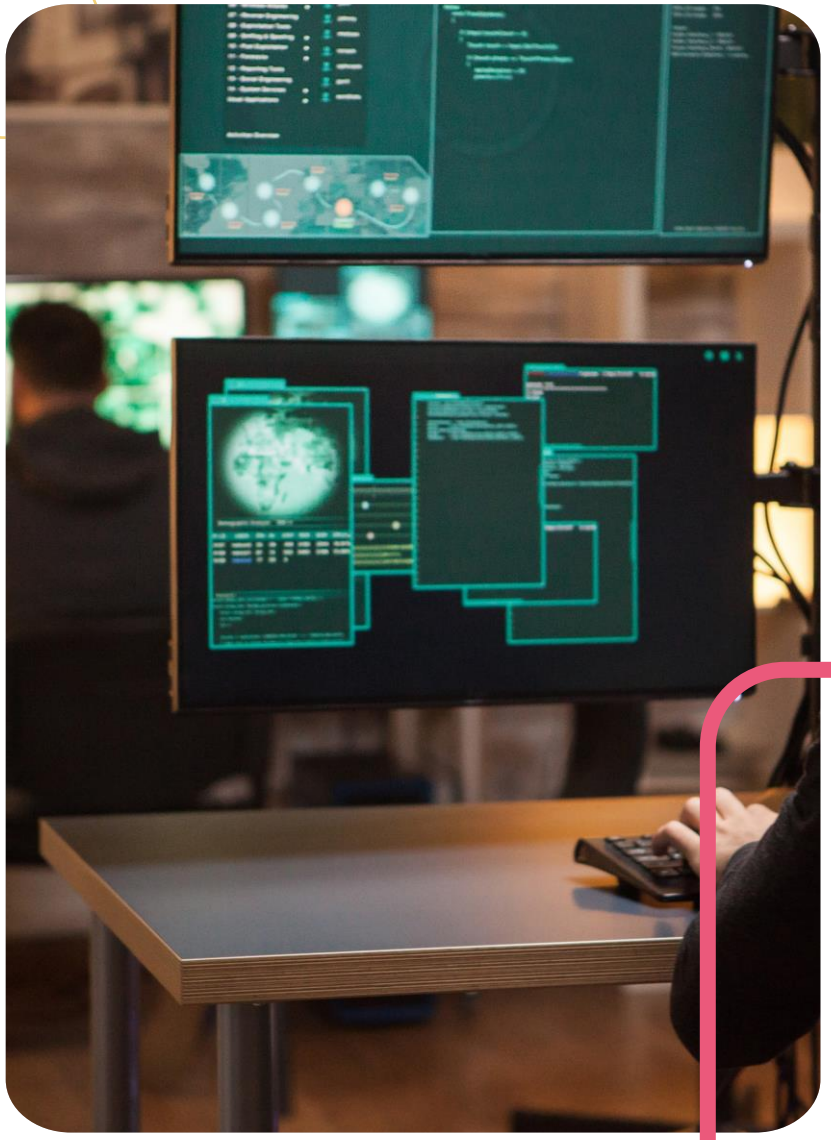
Computer Science

Variables and Constants

Lesson 5 Challenge Feedback

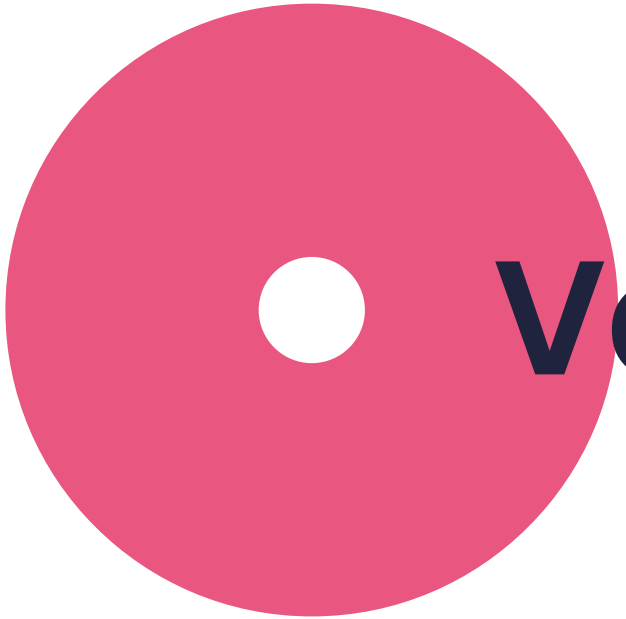
Challenge:
Given a task to create a program that generates lottery numbers for the state lottery, what would be the appropriate data type for the numbers?

We can't store anything other than whole numbers when using the int data type, which makes it the perfect candidate!



Objectives

- Develop an understanding of variables
- Appreciate the role of variables in behaviour of your code at runtime
- Discuss the difference between constants and variables
- Identify good programming practices when using variables and constants



Variables

Variable types

- Syntax of a variable:
  data_type variable_name;
- Area in memory is ‘formatted’
  into specific type

Types of variables

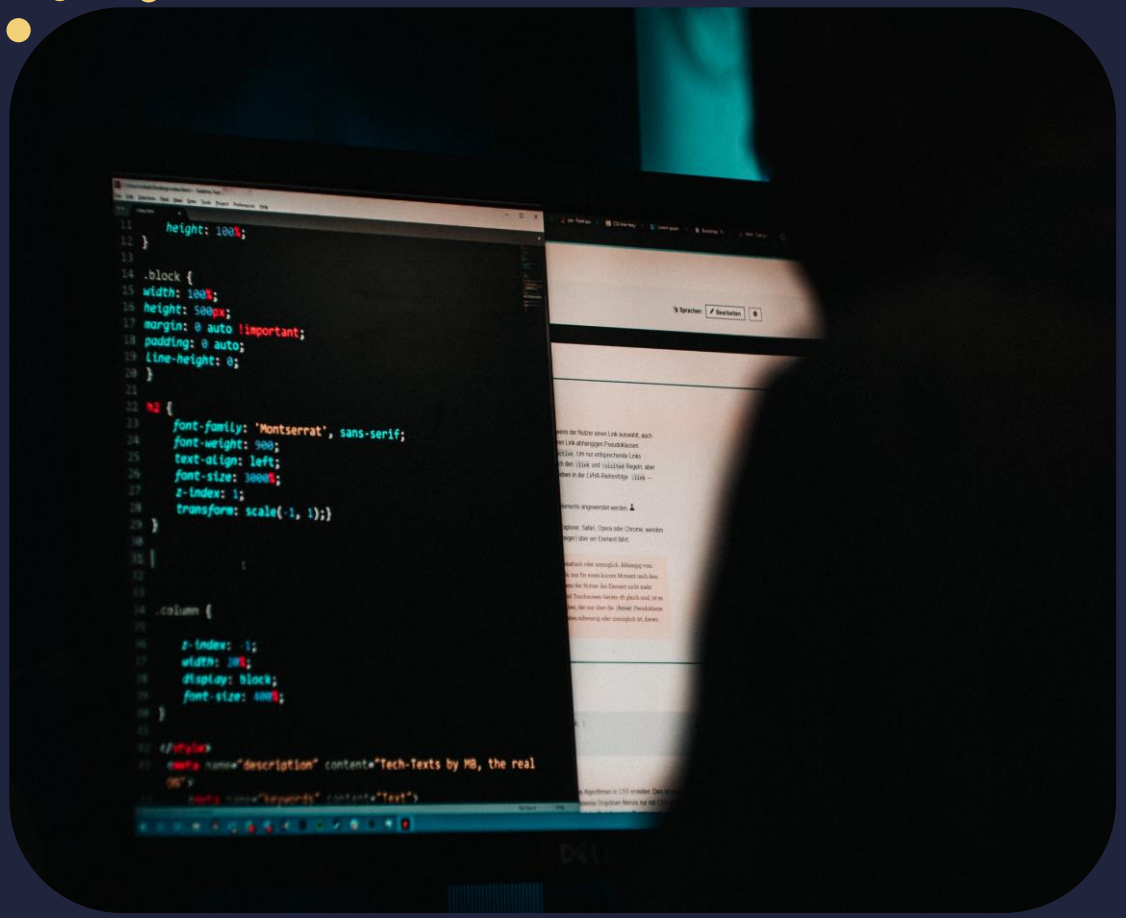
- Local
- Global
- Static
- Automatic
- External

Did you know?

The idea of using letters to represent variables was first suggested by French mathematician François Viète.

Variable declaration

Syntax for a variable:
data_type variable_name;



Variable declaration

Can declare a variable anywhere

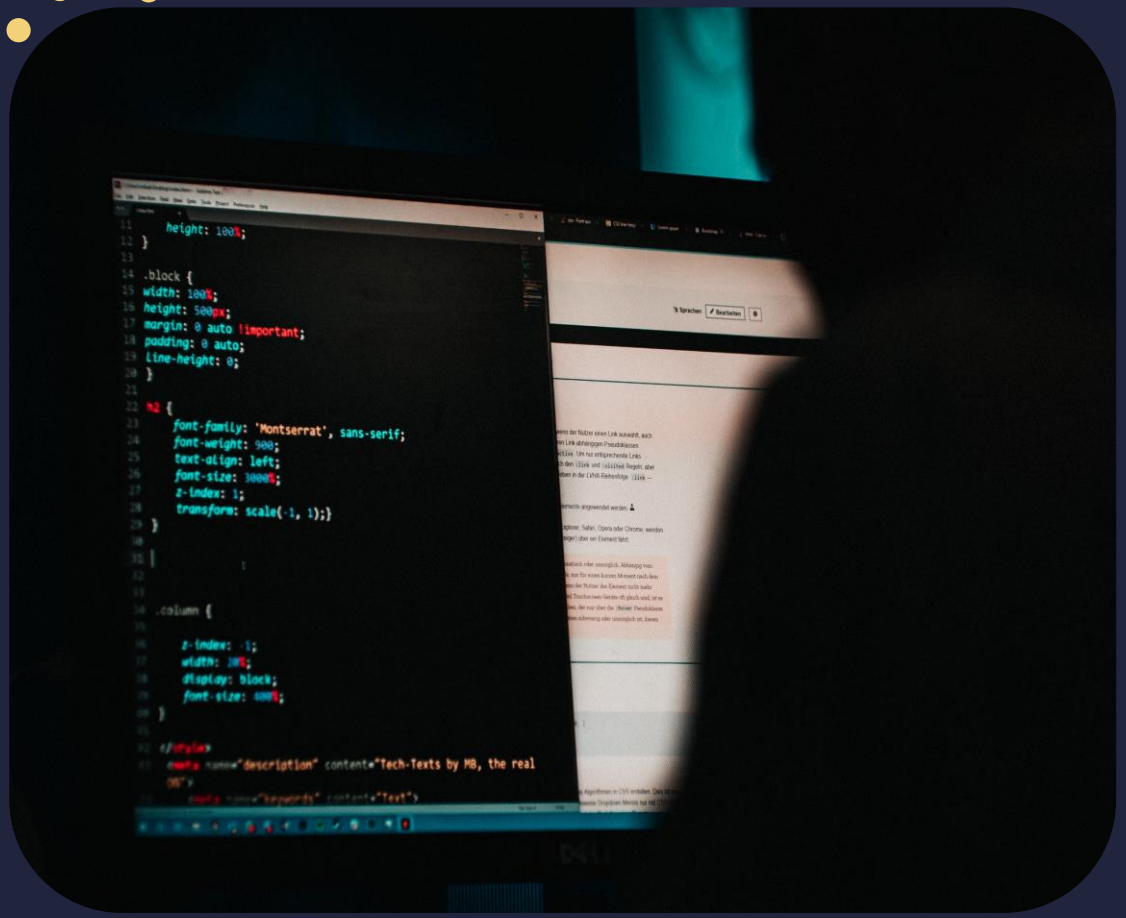
Position of variable affects its visibility

Naming conventions

- Variable name must make sense
- Key is clarity
- Name variables with names related to program’s function or procedure
- Program needs to have flow and logic

Naming conventions

- Example: program to calculate age must include ‘age’
- Separate words in variable name without white space



Four options to space out:

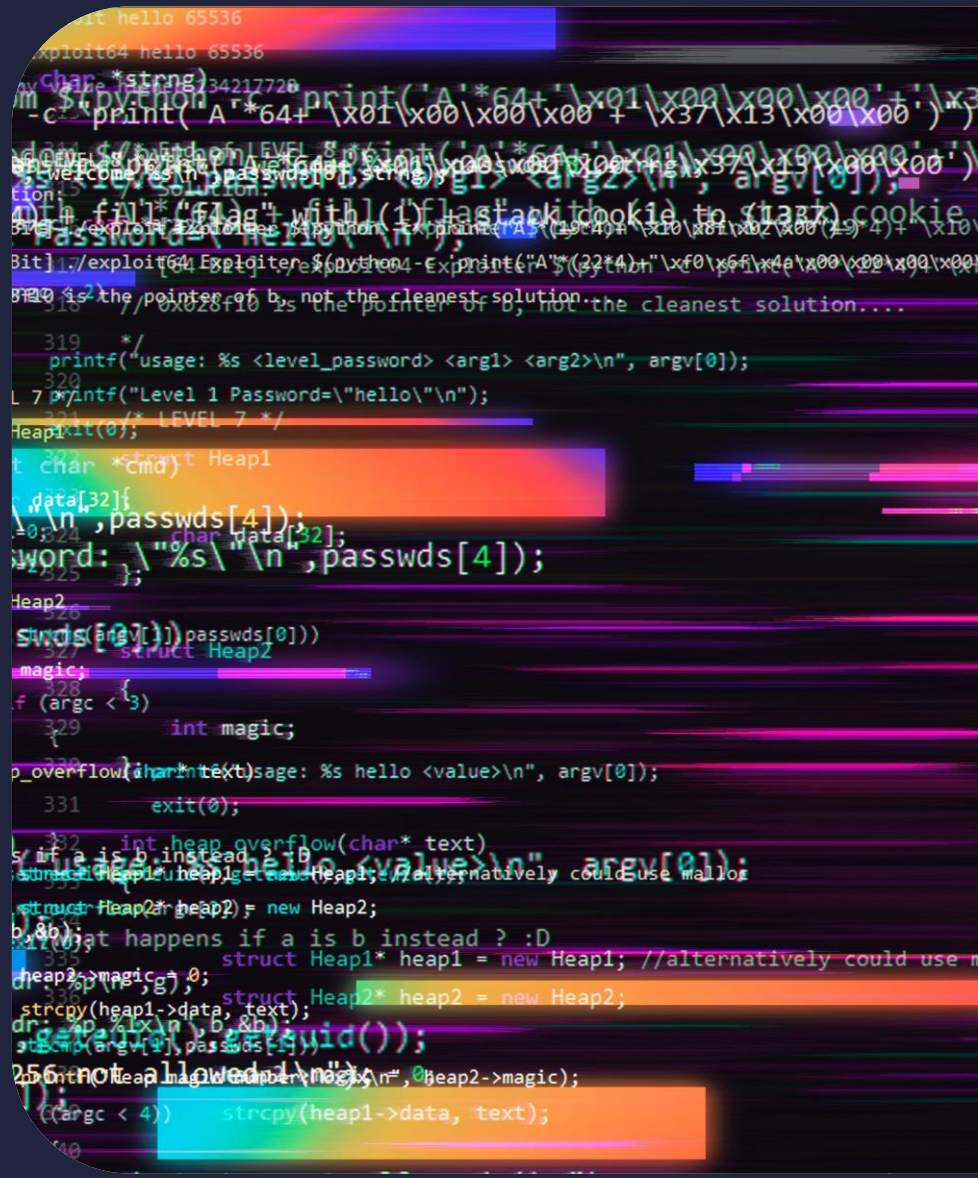
Snakecase	Example: variable_one
Pascalcase	Example: VariableOne
Camelcase	Example: variableOne
Hungarian Notation	Example: sUserName

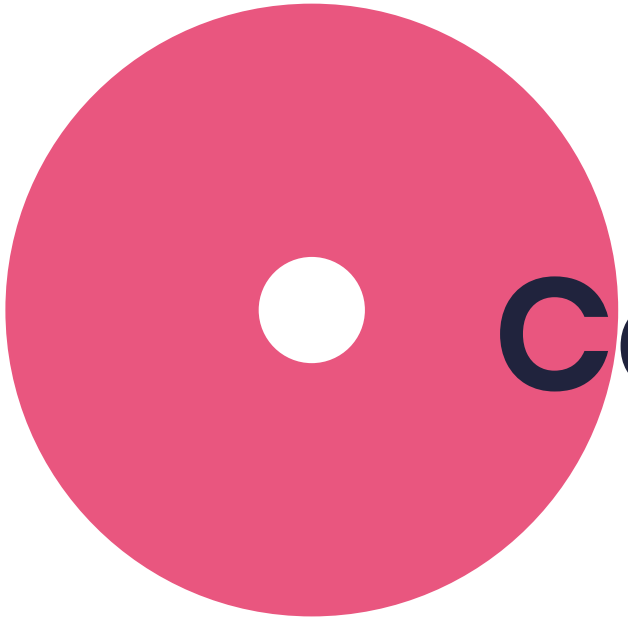
A good variable name tells half the story

lvalues and rvalues

The left side of an expression refers to an object that can hold a value.

The right side of the expression is where the calculation is performed.





Constants

How do you use fixed values?

Can declare a variable but not recommended

Declare a constant because a constant doesn’t change

Example

pi = 22/7 or 3.14

Why use constants?

To assign a value that doesn't change

To eliminate as many errors as possible

To make code more readable

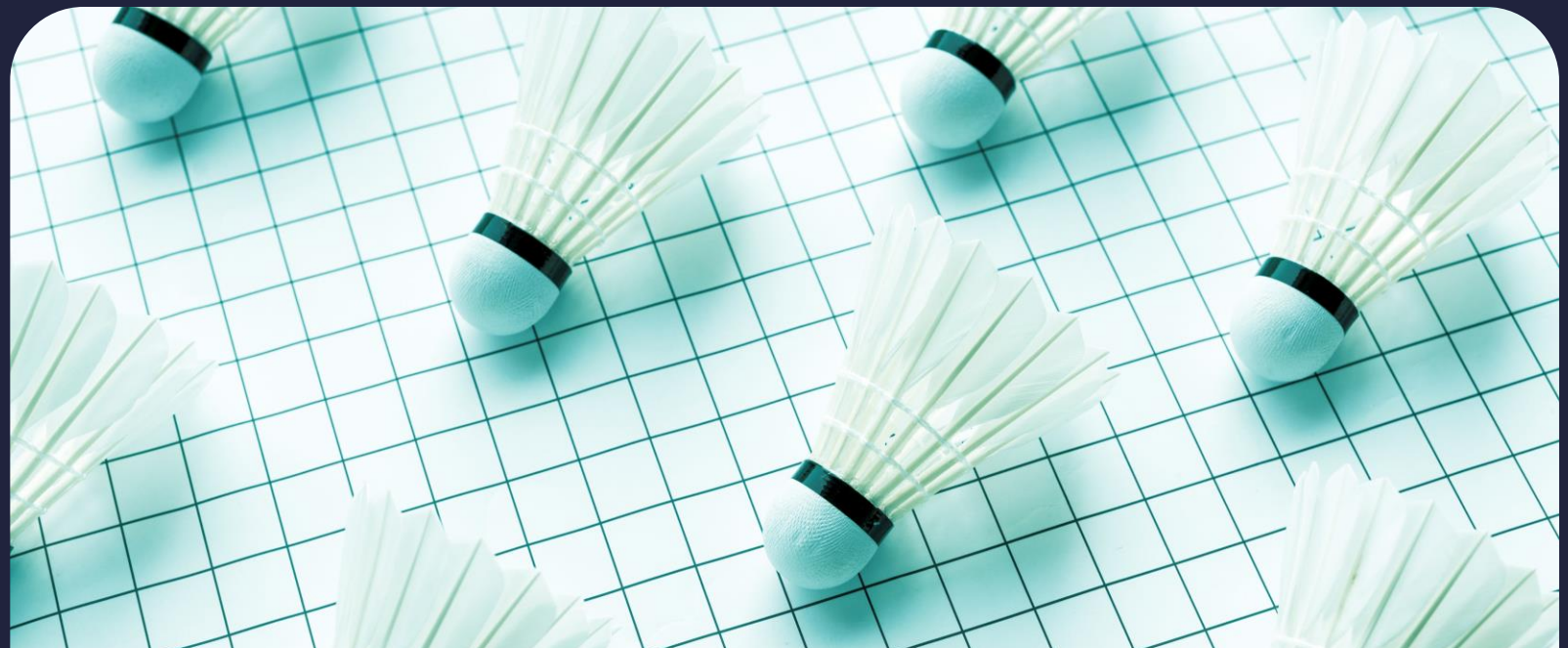
Properties of a constant

Value will never change during runtime of program

Value is probably thread safe

Value is likely referenced in many areas throughout the code

Value name tells something important about its meaning



Types of constants

#define

const

Constant definition

General syntax option 1:
#define identifier value

General syntax option 2:
const DataType
constantName=value;

Same naming conventions for variables apply

Use capital letters when defining constants

Assign a value straight away

A constant is like cement.
A variable is like a bottle of water.

Scope

Local variables

Only accessible from within the program space where declared

Global variables

Accessible from anywhere in the program

Parameters

Passing or assigning variables to functions to be used in the execution of that function

Refer to variables in the declaration of function

Two main forms:

- Formal parameters
- Actual parameters

Did you know?

A *function prototype* is just another way to refer to a *function declaration*.

Parameters

Function prototype = function declaration

Parameter variables are used to import arguments into functions

Function parameters = names listed in function’s definition

Function arguments = real values passed to the function

Formal parameters

Variable and type as they appear

Int shapes(int length, int width);

Example

Referring to parameters without actually knowing their value

A sequence of pre-processing tokens in comma-separated list bounded by brackets

Actual parameters

What you actually pass to the function when you call it

Comparison

ARGUMENTS	PARAMETERS
The values are passed during the function call.	The values are defined during function definition.
They are used in the function call statement to send values from the calling function to the called function.	They are used in the function header of the called function to store the value from the arguments.
Each argument is always assigned to the parameter in the function definition.	Parameters are local variables which are assigned value of the arguments when the function is called.
They are also called ‘actual parameters’.	They are also called ‘formal parameters’.

Arguments passed by value

Copy of parameter's value is made in memory

Arguments passed by value

Caller function passes an argument by value

Called function does not have access to actual variable in the calling code

Copy of the data in the original element is sent to the called function

Changes made to the passed variable do not affect the actual value of the variable

Arguments passed by reference/address

Uses reference of an argument in calling function to corresponding formal parameter of called function

Arguments passed by reference/address

Caller function passes an argument by reference

Called function gives a direct reference to the programming element in the calling code

The memory address of the stored data is passed

Changes to the value have an effect on the original data

When to use what

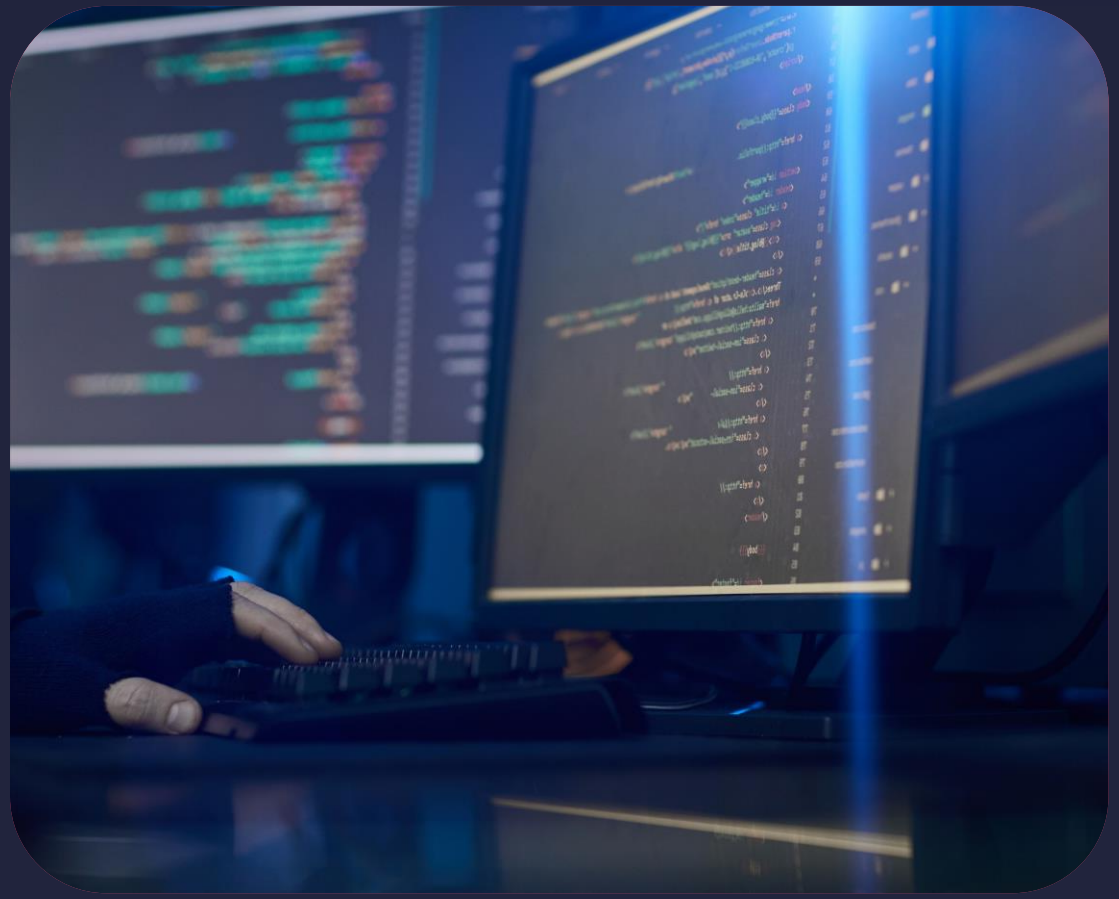
Pass by value

- With multi-threaded applications, don’t worry if objects modified by other threads
- In distributed applications, pass by value saves over network overhead to keep the objects in sync

Pass by reference

- No new copy of variable so the program uses less time and resources
- Especially efficient when passing objects of large structs or classes

To summarise...



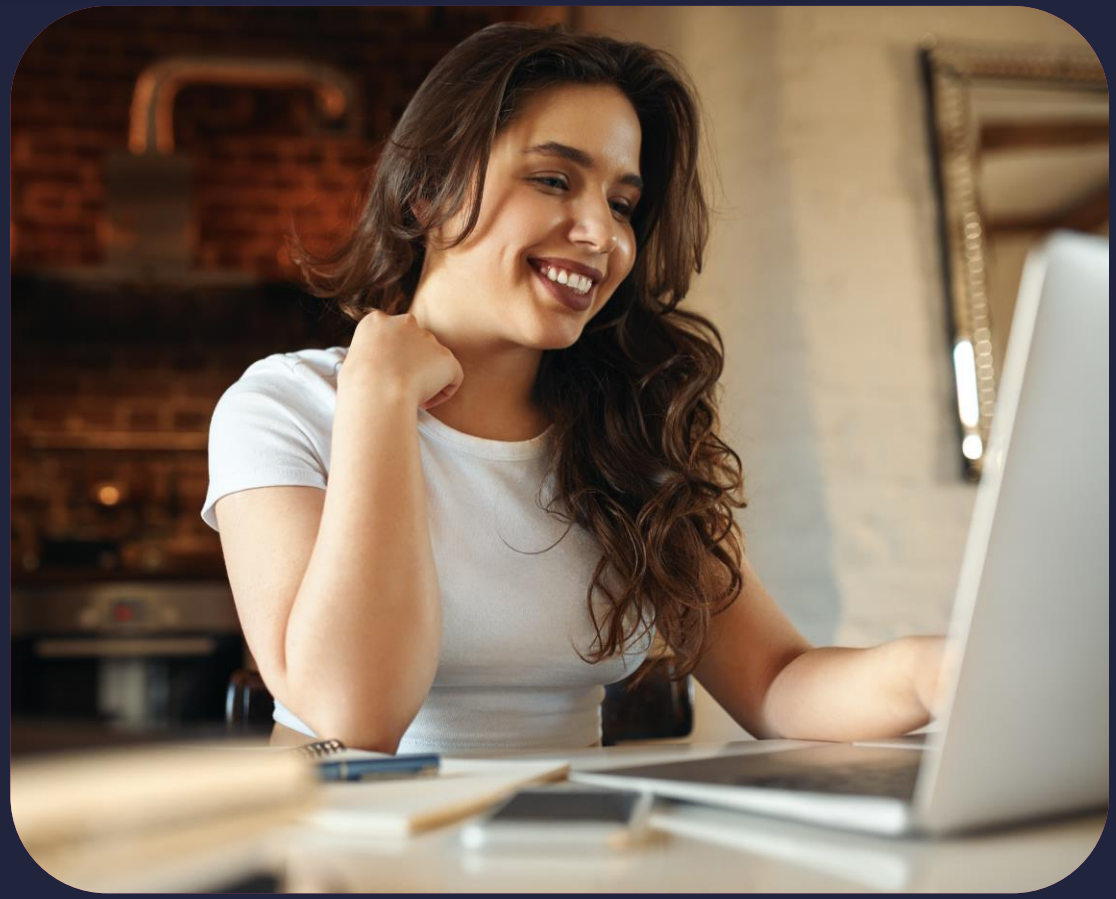
- Parameter = formal parameter or argument
- Argument = actual argument or actual parameter
- Parameter is optional
- Parameter has name, data type, and calling method
- Argument is an expression

To summarise...

Scope of parameter is the function in which the parameter has been called

Number of arguments in a function call should match number of parameters in the function declaration

Data type of arguments in a function call should match data type of parameters in the function definition



Did you know?

Computer technology has led to the mind-boggling expansion of music resources.

Initialisation – why?

- Ensures garbage value in memory location gets overwritten
- When memory is reassigned it is not always cleared
- Can cause major problems when writing large programs

Initialisation – how?

- Assign a value to your variable
- Set value to zero to ensure memory location is overwritten
- Makes your program stable