Introduction to Programming

Duration : 3 days

Course Overview

The Introduction to Programming course comprises sessions dealing with variables, expressions, conditional statements, collections, iterative statements, functions, objects, compilation and execution, and best practices.

This hands-on course does not concentrate on any one language in particular, rather its aim is to familiarise delegates with standard programming terminology, structures, and principles. Examples are given in three languages - Python, Java, and JavaScript - and delegates may choose any one of these languages with which to carry out the practical exercises.

Exercises and examples are used throughout the course to give practical hands-on experience with the techniques covered.

The delegate will learn and acquire skills as follows:

- Writing to reading from the console
- Declaring and initialising variables
- Constructing expressions
- Constructing conditional statements
- Working with arrays/lists
- Constructing iterative statements
- Declaring and invoking/calling functions
- Writing procedural programs
- Working with classes and objects
- Writing object oriented programs
- Compiling and executing code

Target Audience

This Introduction to Programming course is designed for those new to programming, who want to learn about the terminology, structures, and principles of programming generally.

Attending this course will provide delegates with the prerequisite knowledge and required skills to go on to learn any programming language in detail, e.g. Java, JavaScript, Python, C, C++, C#, PHP, Perl, Ruby, etc.

Prerequisites

Delegates should be able to navigate the file system, edit a file, and browse the web. No programming experience is necessary.

Objectives

This course aims to provide the delegate with the knowledge to be able to produce simple computer programs that demonstrate an understanding of the three core principles of programming - sequence, selection, and iteration. Delegates will also be exposed to functions, objects, and both procedural and object-oriented programming paradigms. The course further aims to prepare delegates to go on to learn any one of many programming languages in detail.

Course Content

DAY 1

Session 1: INTRODUCTION
Thinking Like a Computer
Input/Output
Storage
Arithmetic
Comparison
Decisions
Repetition
Reuse
What is a Program?
Statements
Comments
What is Code?
From Source Code to Runtime
Why So Many Languages?
What Does a Programmer Do?
Hello World
stdin and stdout
The Console

Session 2: VARIABLES
Variables - What and Why
Name and Value
Literals
Data Types
Declaration
Initialisation
Assignment
Constants

Session 3: EXPRESSIONS
Expressions - What and Why
Operators and Operands
Unary and Binary Operators
Arithmetic Operators
Assignment Operators
Precedence
Associativity
Complex Expressions

DAY 2

Session 4: CONDITIONAL STATEMENTS
Conditional Statement - What and Why
Comparison/Relational Operators
Logical Operators
if else
switch
The Ternary Operator
Code Blocks
Variable Scope

Session 5: COLLECTIONS
Collections - What and Why
Strings
Arrays/Lists
Declaration
Initialisation
Getting and Setting Elements

Session 6: ITERATIVE STATEMENTS
Iterative Statements - What and Why
while
do
for
break
continue
Array/List Traversal

Session 7: FUNCTIONS
Functions - What and Why
Declaration
Parameters
Return Type
Invocation/Call
Arguments
Return Value
Variable Scope (Review)
Modules
Libraries
Procedural Programming

DAY 3

Session 8: OBJECTS
Object - What and Why
Object Literals and Object Properties
The Trouble with Object Literals
Classes
Fields
Methods
Instances
Reference Variables and Primitive Variables
Passing by Val/Ref
Object Oriented Programming
The Three Principles

Session 9: COMPILATION & EXECUTION
From Source Code to Runtime (Review)
Compilation
Debugging
Linking
Execution
Interpretation
Platform Dependence
Compilation and Interpretation (Bytecode)

Session 10: BEST PRACTICES
Program Design
Stating the Problem
Devising the Solution
Pseudocode
Coding Conventions
White Space
Indenting
Naming
Coding Style
Readability, Flexibility, Scalability
Unit Testing
Test Driven Development (TDD)