Programming Foundations

Learn via: **Classroom / Virtual Classroom / Online**

Duration: **3 Gün**

Overview

Basic programming skills are a fundamental requirement for many IT professionals. An entry-level programmer can have difficulty with jargon, and knowing where to start. Learning language syntax can be an uphill struggle when it cannot be put into context.

As scripting languages become more powerful and available traditional tasks of a programmer are invading other IT functions. System administrators may have to write complex scripts which impact on mission critical systems, often with no programming experience or training.

Support staff often have to communicate with development staff, and misunderstandings easily arise from cultural differences.

This course gives a basic understanding of how computer systems work from a programmer's perspective, and how to use this knowledge to produce good code. It also enables technical staff who are not programmers to gain a perspective of software development.

The course is language neutral and teaches general concepts. Python is used as the language in exercises, but examples will be shown in other languages.

Who Should Attend
System Administrators and support staff who require a technical knowledge of programming, to help them produce better code, to understand programming concepts, or as a precursor to further training. The course is also suitable for trainee programmers who have little or no in-depth knowledge of programming. It can act as a primer for delegates new to programming who are looking to train on languages such as C, C , Java, Perl, Python, PHP, C# and Visual Basic in a later course.

Prerequisites

Delegates must be computer literate and have recent experience as a computer user.

Please note: Before attending this class delegates must have a Microsoft account (signing up one is free). The instructions on how to set up a Microsoft account can be found here.

What You Will Learn

On Completion, Delegates will be able to:

- Describe the components of a computer system
- Understand the purpose of Operating Systems and third-party libraries
- Understand the underlying structure of data types
- Differentiate between difference container types and their use
- Choose a suitable data type for a specific task
- Use basic operators, and understand precedence
- Understand how the stack is used to pass data
- Recognise different abstract file types, and their uses
- Be familiar with different program execution regimes
- Understand and apply good coding techniques

Outline

Chapter 1: System Components

- Computer system components
- Central hardware components
- Moore's Law
- Software components
- Operating systems
- Processes
- Virtual memory

- Virtualisation
- Cloud Computing

Chapter 2: First Steps

- What is a program?
- What does a programmer do?
- Components of a program
- Programming languages
- Standards
- Applications and libraries
- Code used by more than one program
- What does a program do?
- Paper to program
- Creating a program
- A first program
- Running a program
- Error messages are your friends
- Giving names to data items
- Assignment statements
- Console vs. GUI
- Simple graphical messages
- Special characters

Chapter 3: Data

- Representing data
- Common numbering systems
- Numbers in programming languages
- Bits, bytes and words
- Conventions
- Fundamental types
- Getting it wrong
- Representing characters
- The problem with the Euro
- Representing integers
- Representing floating point
- E numbers
- Representing time
- Representing nothing

Chapter 4: Variables and Operators

- Variables and constants
- Objects
- Life of a variable - scope
- An alternative to scope
- Namespaces
- Choosing variable names
- Names you should not use
- Operations on data
- Choosing variable types
- Assignment and types
- Simple operations?
- Operator precedence
- Comments

Chapter 5: Containers

- Arrays and lists
- Sorting
- Other linear types - stacks, queues, deques
- Downside of linear structures
- Non-linear types and keys
- Containers for records - tree structures
- Associative arrays
- Compound types
- Class
- Object Orientation

Chapter 6: Flow Control

- Flow control
- Altering program flow

- Simple decision statements
- What is truth?
- Boolean operators
- Logical operators
- Using logical operators
- Loops
- Simple loop statements
- Array processing
- Language supplied iterators
- Interrupt handling
- Exception handling

Chapter 7: Program Structure

- Scope revisited
- Named blocks
- Calling a function
- Arguments and Parameters
- Passing arguments by copy
- Passing arguments by reference
- Returning results
- Function call syntax
- Recursion
- Entry points
- Modules and Libraries
- Why hide code and data? Encapsulation
- Asynchronous subroutines - Threads

Chapter 8: Input and Output

- What is a file?
- File systems
- Exchangeable file systems
- I/O Libraries and Layers
- File data types
- File names
- Opening a file
- Opening a file - checks
- Opening a file - modes
- Sequential access
- Random access
- Buffering
- Concurrency issues
- Locking strategies

Chapter 9: Building Programs

- Compilation
- Linking
- Loading and running
- Process attribute inheritance
- Portability
- Emulators
- Interpretation
- The third way: Byte-code
- Optimisation
- Debuggers

Chapter 10: Coding Style

- Virtues of a programmer
- Readability and style
- Naming conventions
- Error handling
- Programming for change
- The need for speed
- Programming for performance
- Constants - aren't
- Portability and flexibility
- Help!