1 Fundamentals of Programming

Objectives

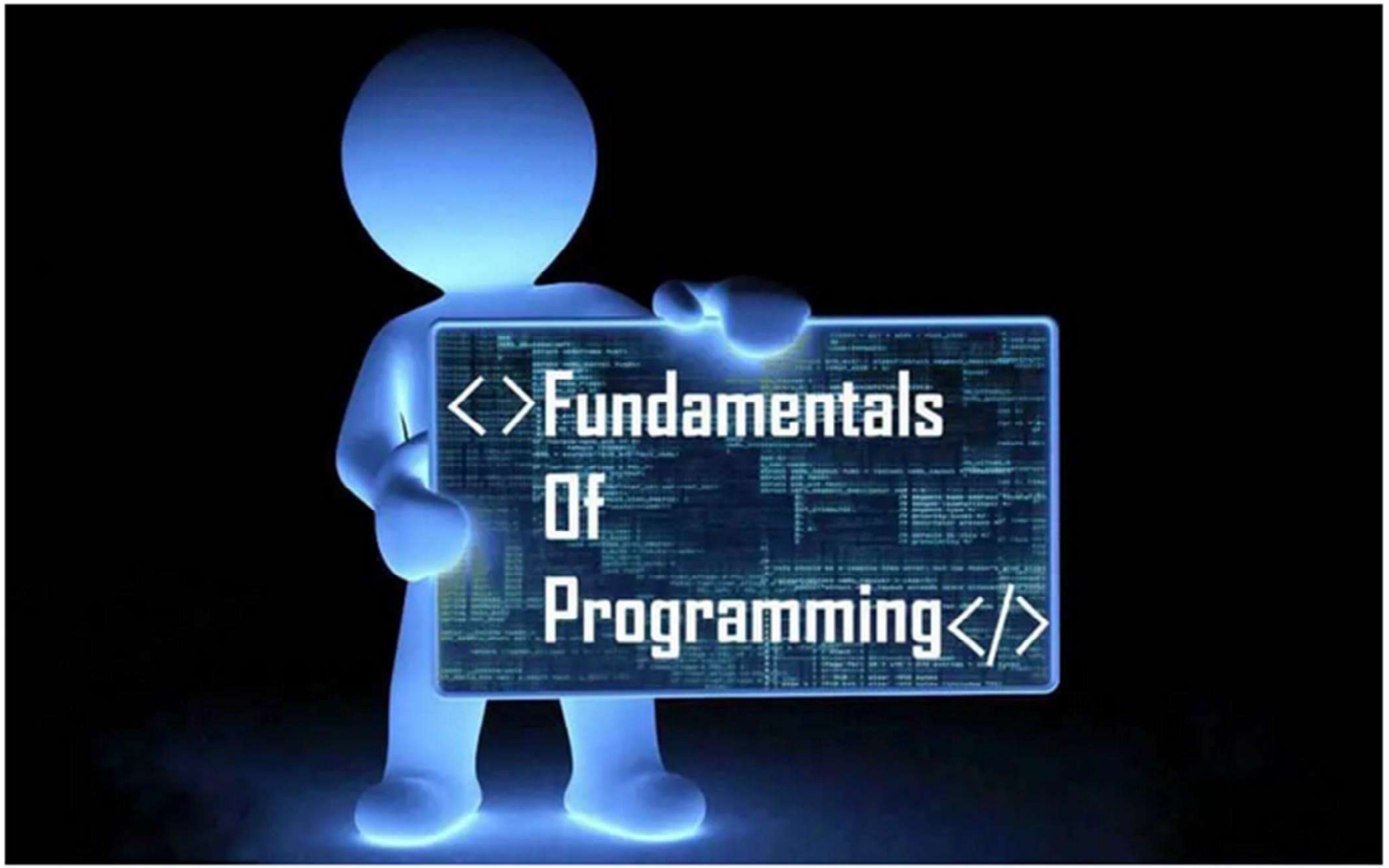
After completing this lesson, you should be able to do the following:

- Software Development Process
- Methods Used to Develop and Test Programs
- IDE's, Version Controls
- Pseudocode and Flow Chart

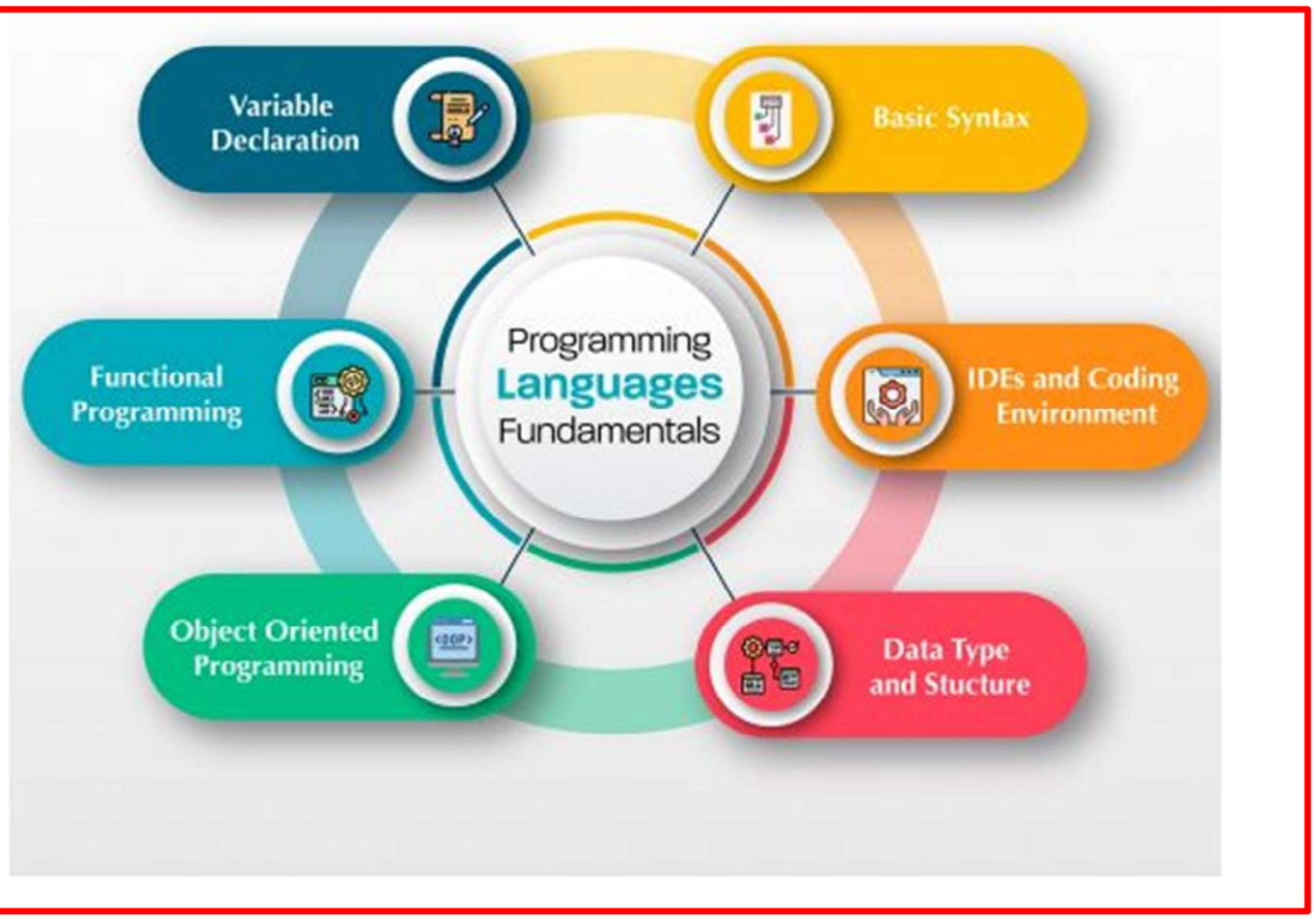
Course Roadmap

Fundamentals of Programming

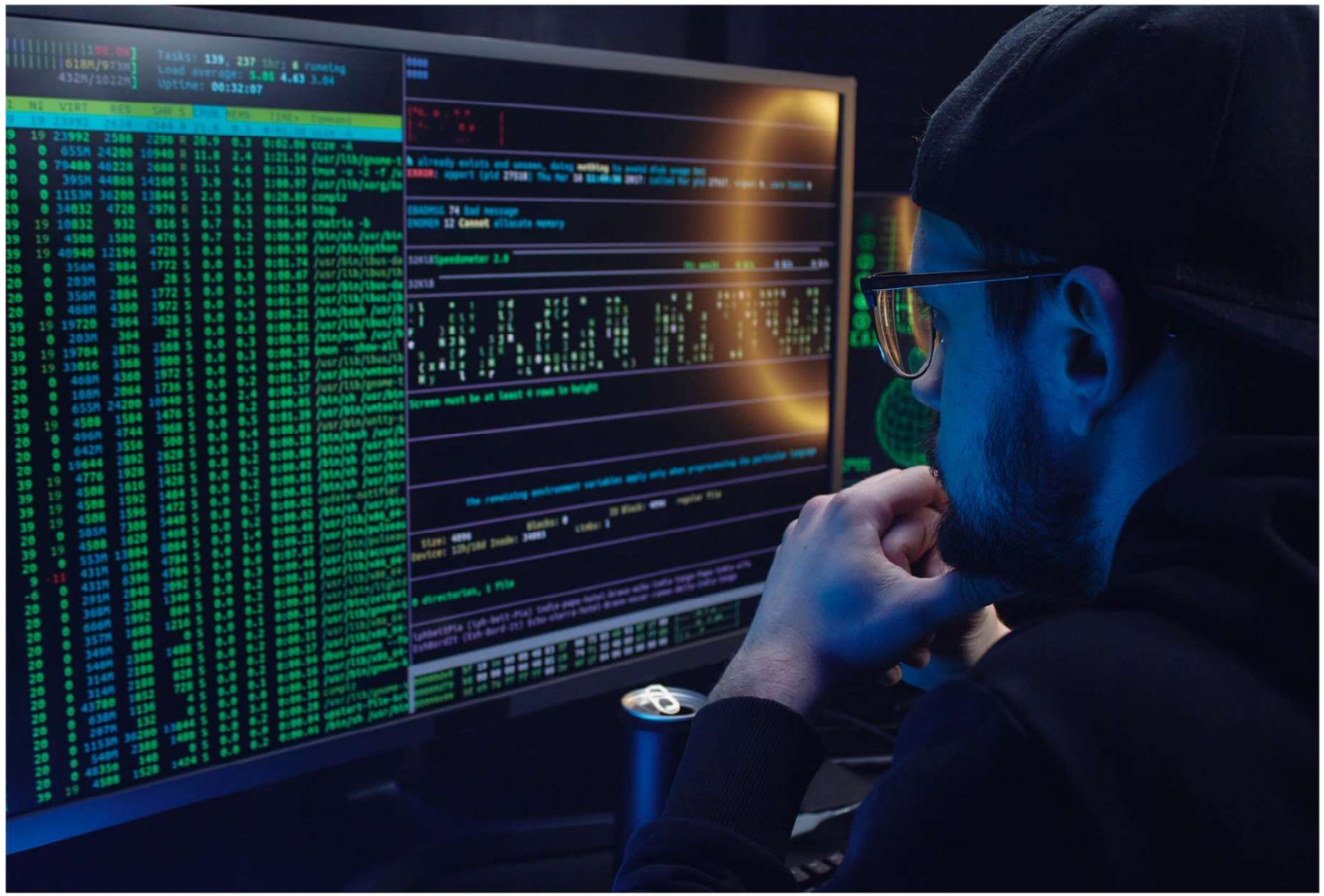
- **Lesson 1: Introduction to Fundamentals of Programming** — You are here!
- Lesson 2: How Programming Languages Work
- Lesson 3: Data and Types
- Lesson 4: Control Flow Statement Errors & Debugging
- Lesson 5: : Structured Modular & Object Oriented


<>Fundamentals
Of
Programming</>

Aspects of Fundamentals of Programming

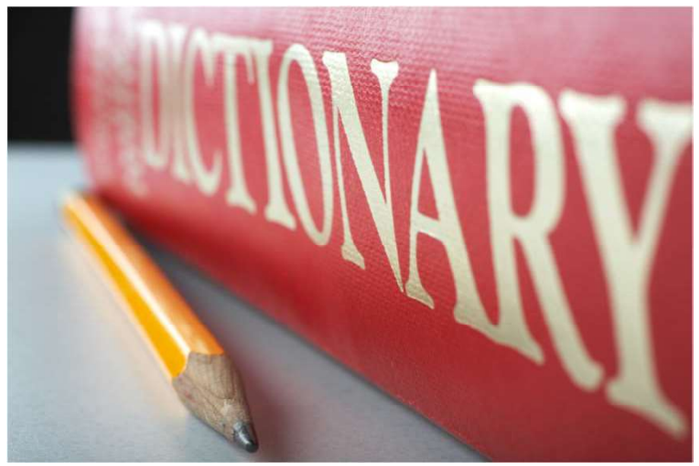


What is Programming ?



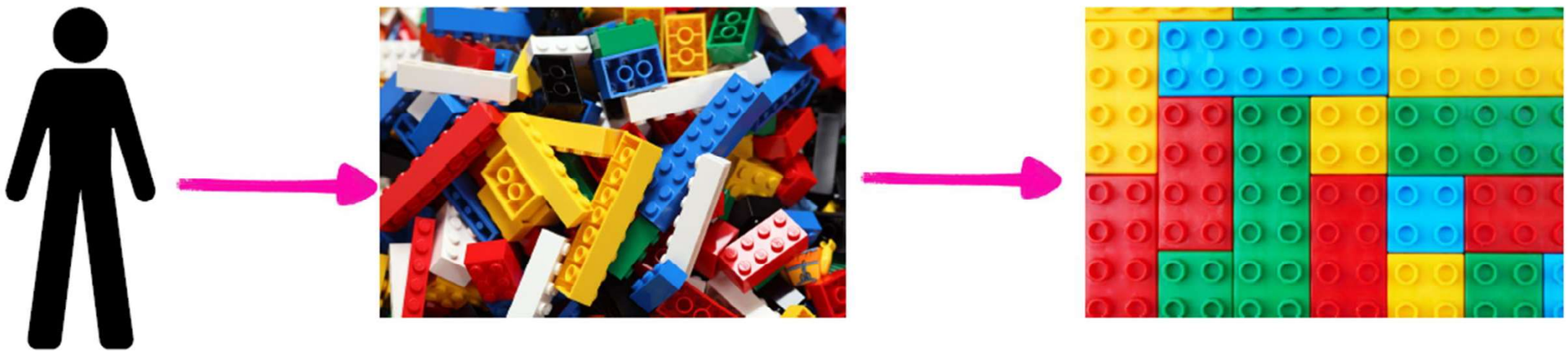
What is a Programming ?

- Set of Instructions used to perform a Particular Task.
- The process of preparing an instructional program for a device
- Attempting to get a computer to complete a ***specific task*** without making mistakes
- A few popular names are *program*, *computer program*, *application* (app), and *executable*.



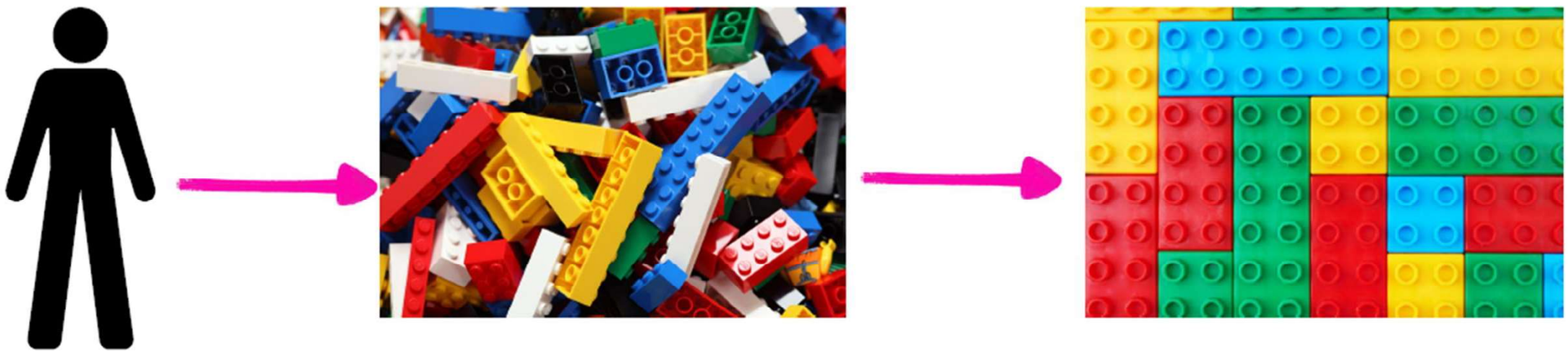
Live Illustration

- You instruct your younger siblings to build a lego set
- He/She lost the instructions so he/she can only build based on your commands



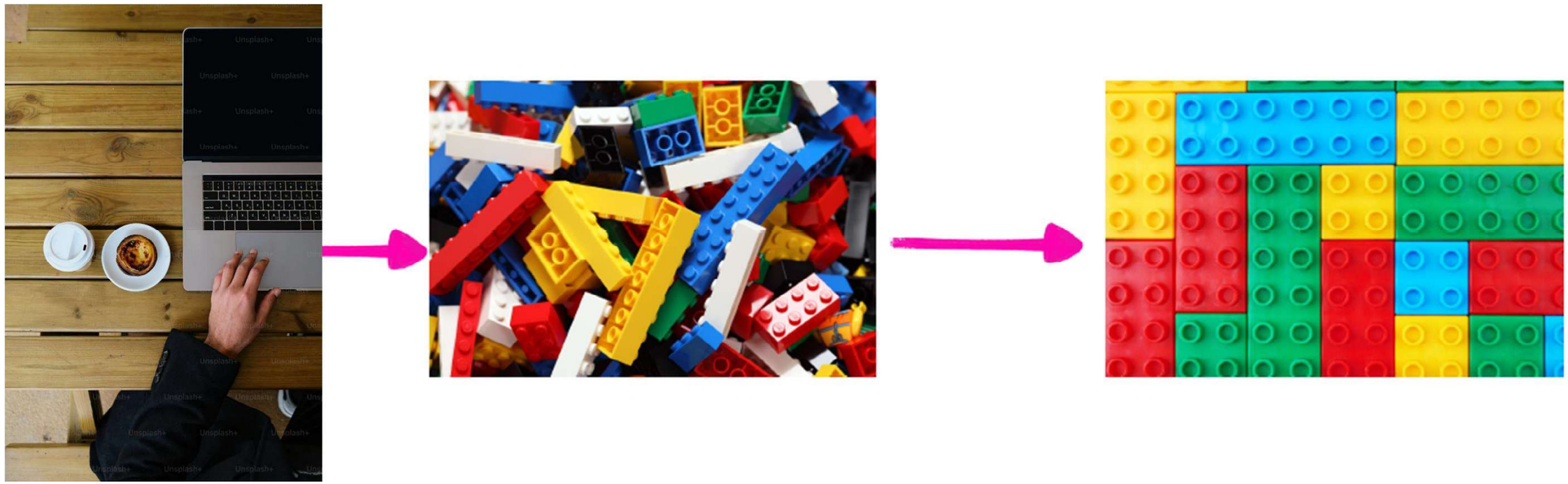
Cont …

- Remember your younger siblings is too young to build on his/her own and so you must give him EXACT instructions on how to build
  - If there is even one piece wrong the entire lego set will be ruined



What is Programming ?

- Giving instructions to your siblings is how ***Programmers Code***
- Replace your siblings with a computer and that's how we code



Computers are Incomplete Without Us..

- Computer's are only smart because ***We Program*** them to be.

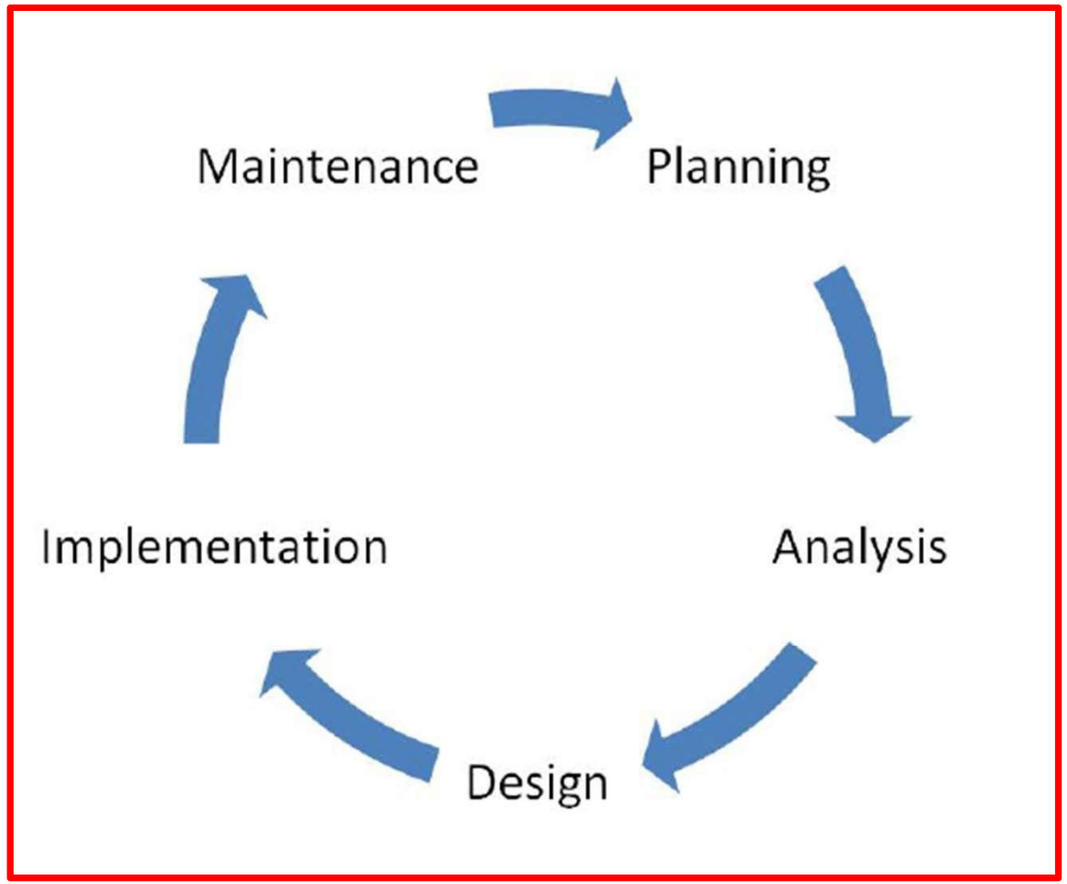
What is a Program ?

- A program can be anything that is written with code. A few examples of programs that run on different devices are:
  1. Websites
  2. Games
  3. Phone apps
- Although it's possible to create a program without writing code, the underlying logic is interpreted to the device and that logic was most likely written with code.
- A program that runs or executes code is carrying out instructions. The device that you're currently using to read this module is running a program that prints it to your screen.

Systems Development Life Cycle

- The Systems Development Life Cycle is the big picture of creating an information system that handles a major task (referred to as an application).
- The **applications** usually consist of many programs.
- An example would be the
  - Department of Defense supply system,
  - The customer system used at your local bank,
  - The repair parts inventory system used by car dealerships.
  - There are thousands of applications that use an information system created just to help solve a business problem.

- Computer professionals that are in charge of creating applications often have the job title of **System Analyst**. The major steps in creating an application include the following and start at **Planning** step.


Maintenance
Planning
Analysis
Design
Implementation

- During the **Design** phase, the System Analyst will document the inputs, processing, and outputs of each program within the application.
- During the **Implementation** phase, programmers would be assigned to write the specific programs using a programming language decided by the System Analyst. Once the system of programs is tested the new application is installed for people to use. As time goes by, things change and a specific part or program might need repair.
- During the **Maintenance** phase, it goes through a mini planning, analysis, design, and implementation. The programs that need modification are identified and programmers change or repair those programs. After several years of use, the system usually becomes obsolete. At this point, a major revision of the application is done. Thus the cycle repeats itself.

1. Program Design

- **Program design** consists of the steps a programmer should do before they start coding the program in a specific language. These steps when properly documented will make the completed program easier for other programmers to maintain in the future. There are three broad areas of activity:
  - Understanding the Program
  - Using Design Tools to Create a Model
  - Develop Test Data

Understanding the Program

- If you are working on a project as one of many programmers, the system analyst may have created a variety of documentation items that will help you understand what the program is to do. These could include screen layouts, narrative descriptions, documentation showing the processing steps, etc. If you are not on a project and you are creating a simple program you might be given only a simple description of the purpose of the program. Understanding the purpose of a program usually involves understanding its:
  1. Inputs
  2. Processing
  3. Outputs

- This **IPO** approach works very well for beginning programmers. Sometimes, it might help to visualize the program running on the computer. You can imagine what the monitor will look like, what the user must enter on the keyboard and what processing or manipulations will be done.

Using Design Tools to Create a Model

- At first, you will not need a hierarchy chart because your first programs will not be complex. But as they grow and become more complex, you will divide your program into several modules (or functions).
- The first modeling tool you will usually learn is **pseudocode**. You will document the logic or algorithm of each function in your program. At first, you will have only one function, and thus your pseudocode will follow closely the IPO approach.

Several methods or tools for planning the logic of a program

1. Flowcharting
2. Hierarchy or structure charts
3. Pseudocode,
4. HIPO
5. Nassi-Schneiderman charts
6. Warnier-Orr diagrams

Programmers are expected to be able to understand and do flowcharting and pseudocode.

Develop Test Data

- **Test data** consists of the programmer providing some input values and predicting the outputs. This can be quite easy for a simple program and the test data can be used to check the model to see if it produces the correct results

2. Program Quality

- **Program quality** describes fundamental properties of the program's source code and executable code, including
  1. Reliability
  2. Robustness
  3. Usability
  4. Portability
  5. Maintainability
  6. Efficiency
  7. Readability

Pseudocode

- **Pseudocode** is an informal high-level description of the operating principle of a computer program or other algorithm.

- Pseudocode is one method of designing or planning a program. **Pseudo** means false, thus pseudocode means false code. A better translation would be the word fake or imitation. Pseudocode is fake (not the real thing). It looks like (imitates) real code but it is NOT real code. It uses English statements to describe what a program is to accomplish. It is fake because no compiler exists that will translate the pseudocode to any machine language. Pseudocode is used for documenting the program or module design (also known as the algorithm).

```
Input
    display a message asking the user to enter the first age
    get the first age from the keyboard
    display a message asking the user to enter the second age
    get the second age from the keyboard

Processing
    calculate the answer by adding the two ages together and dividing b

Output
    display the answer on the screen
    pause so the user can see the answer
```

- we use the pseudocode to write code in a language (like C++, Java, Python, etc.) where you must follow the rules of the language (syntax) in order to code the logic or algorithm presented in the pseudocode.
- Pseudocode usually does not include other items produced during programming design such as identifier lists for variables or test data.

Flowcharts

- A **flowchart** is a type of diagram that represents an algorithm, workflow or process. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.
- This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

Simple Flowcharting Symbols

1. *Terminal*

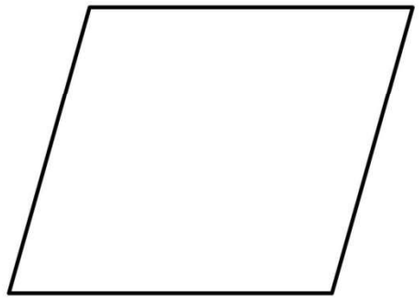
The rounded rectangles, or terminal points, indicate the flowchart's starting and ending points.

2. *Flow Lines*

The default flow is left to right and top to bottom (the same way you read English).

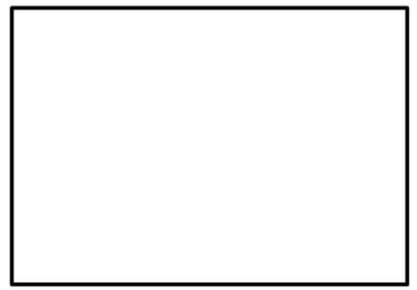
3. *Input/Output*

The parallelograms designate input or output operations.



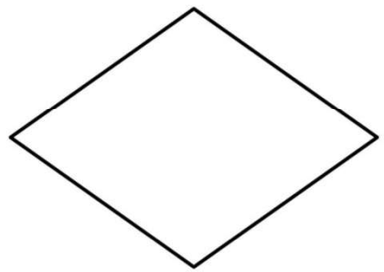
4. Process

The rectangle depicts a process such as a mathematical computation, or a variable assignment.



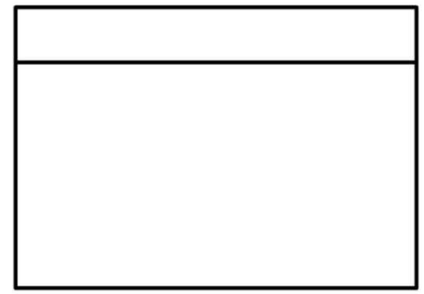
5. *Decision*

The diamond is used to represent the true/false statement being tested in a decision symbol.

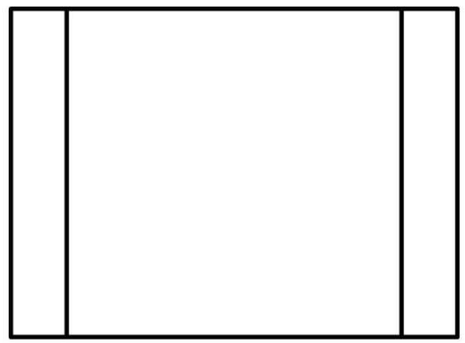


6. *Module Call*

A program module is represented in a flowchart by rectangle with some lines to distinguish it from process symbol



7. Library module: usually a specific task function.

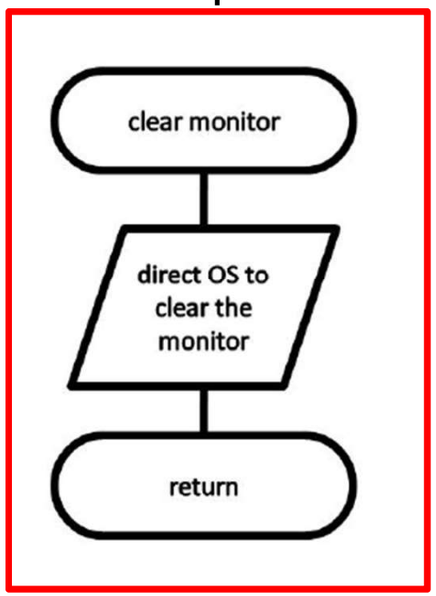


- Set of Instruction to perform used to perform a particular task

pseudocode: Function with no parameter passing

```
Function clear monitor
    Pass In: nothing
    Direct the operating system to clear the monitor
    Pass Out: nothing
End function
```

Flow Chart Representation

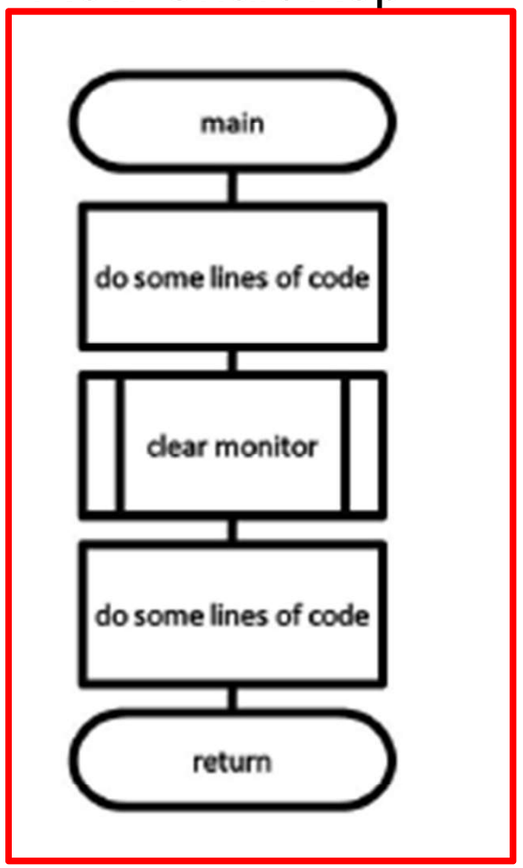


Function with library modules

pseudocode: Function main calling the clear monitor function

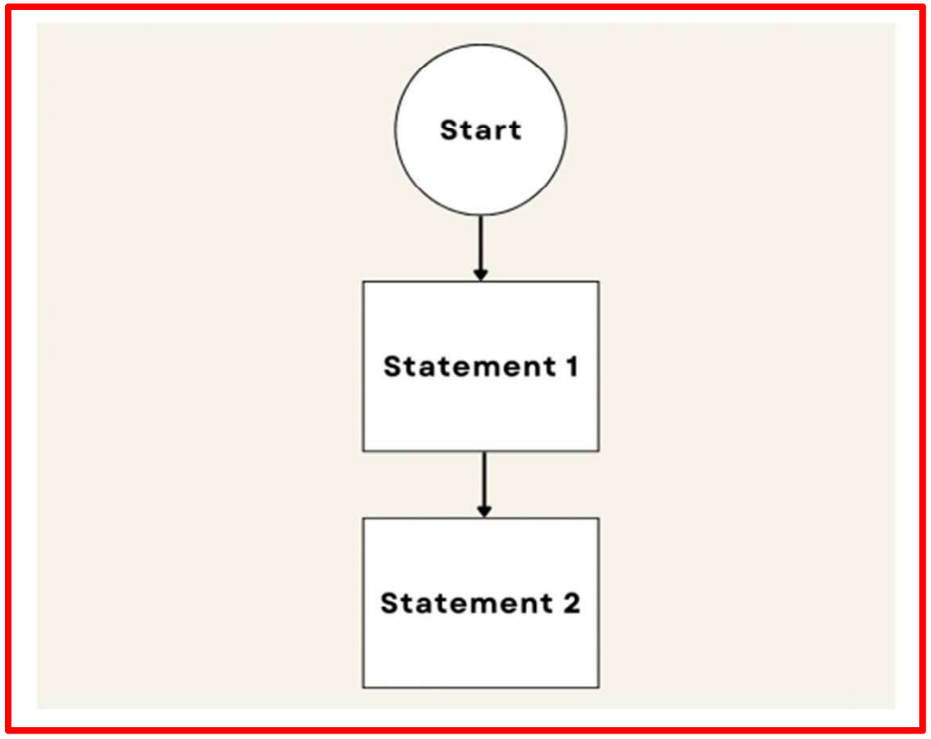
```
Function main
    Pass In: nothing
    Doing some lines of code
    Call: clear monitor
    Doing some lines of code
    Pass Out: value zero to the operating system
End function
```

Flow Chart Rep



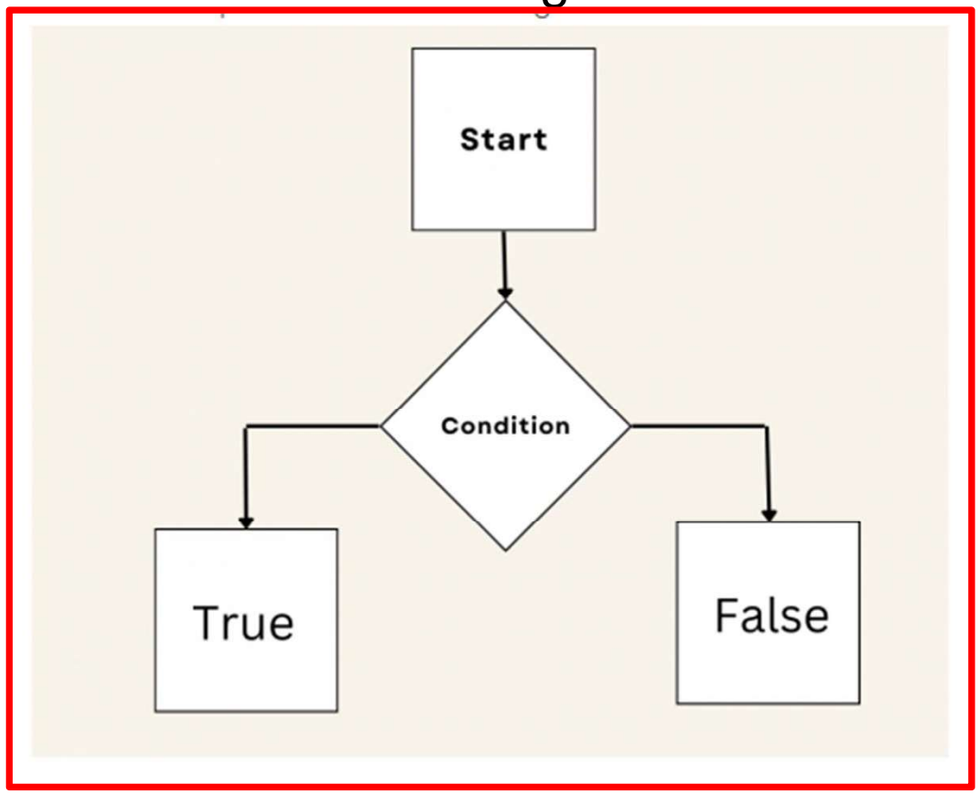
Sequential

- Sequential control flow is the basic control flow. Here the computer decides to justify the statement as true or false.



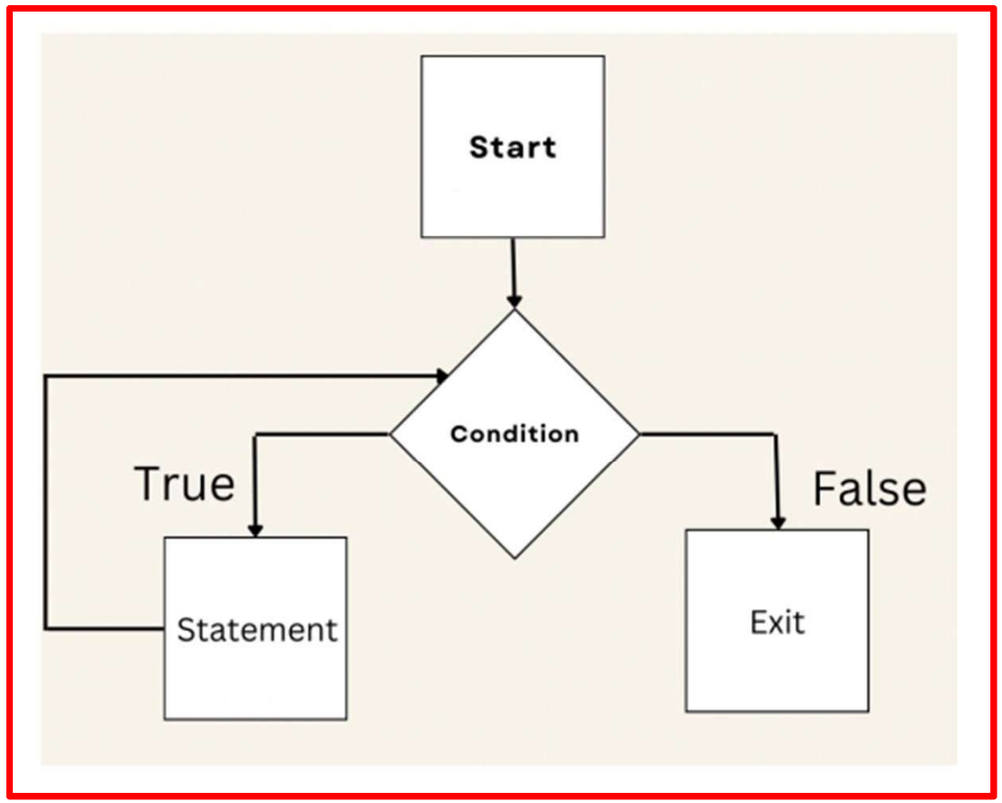
Selection (Conditionals):

- Selection flow control is the basic premise where the computer decides the taking of action. This action is the testing of the condition as true or false



Iteration (Loops):

- A loop is defined as a programming structure that enables the repeating execution of a particular statement or the block of code. This is done until a specified condition is no longer true. Loops are the powerful concept of programming languages.

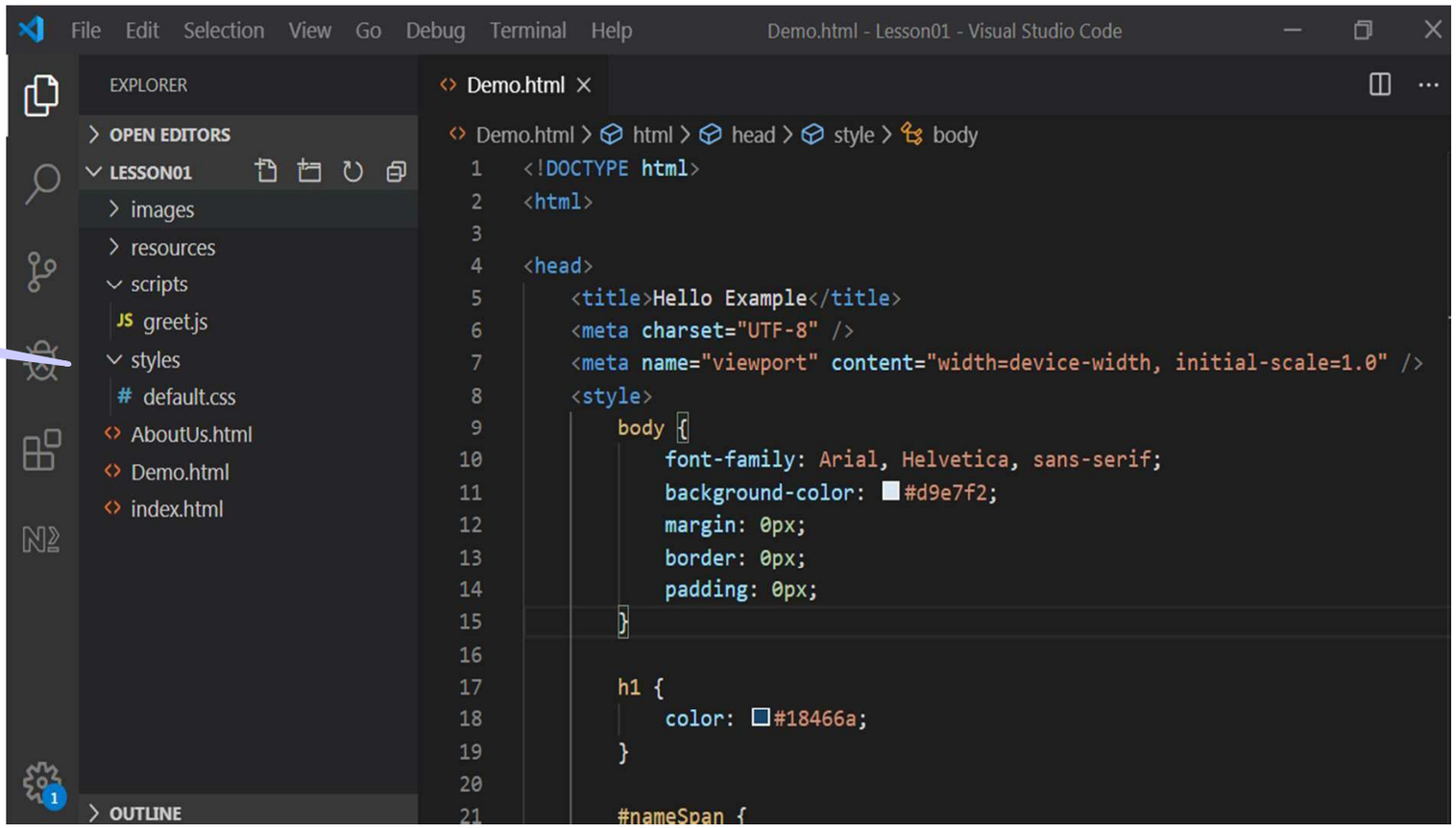


Integrated Development Environment

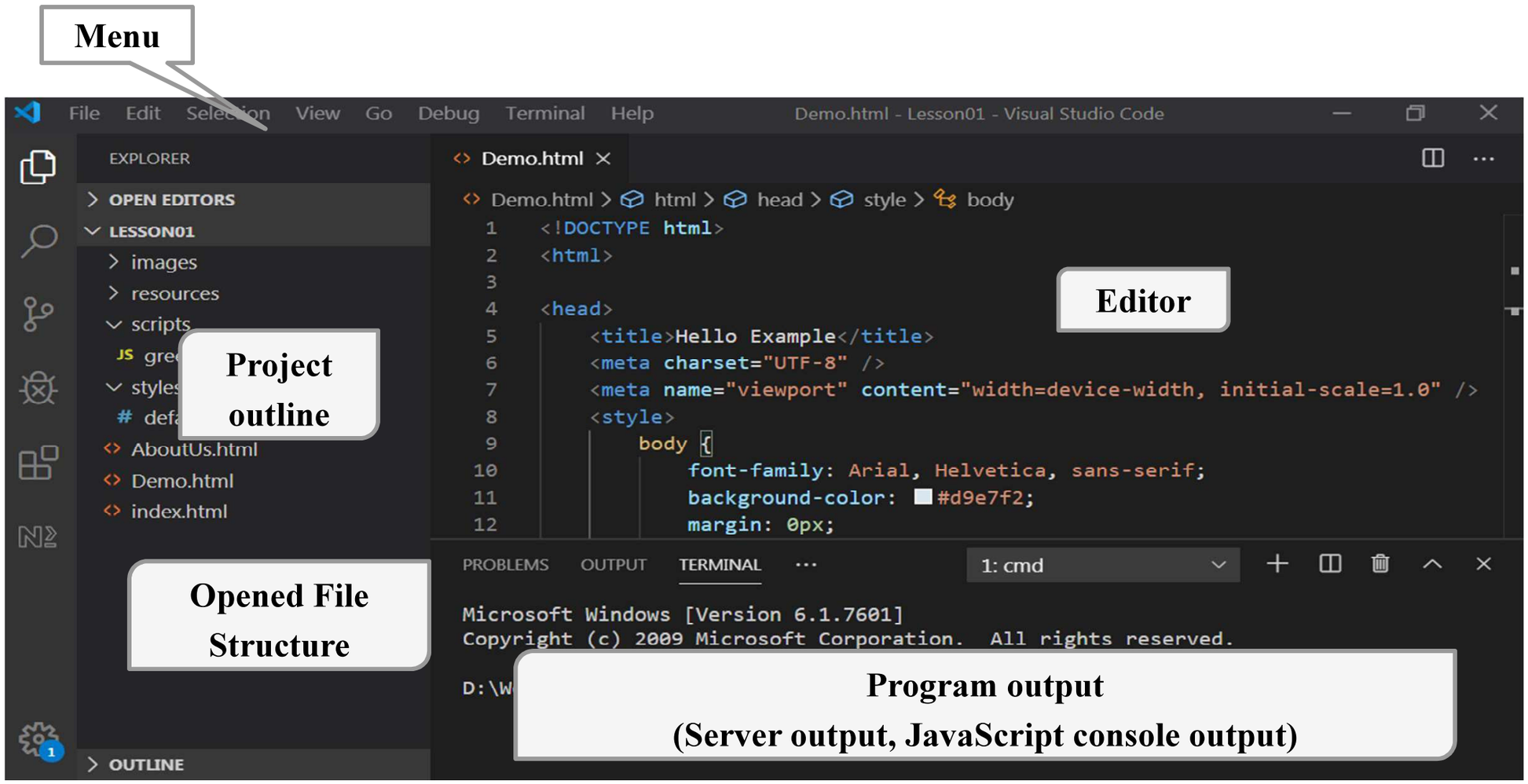
Visual Studio Code / Eclipse

- Integrated Development Environment:
  - Organizes all your application files and settings into projects
  - Automates processes such as running, debugging, testing, and profiling applications
- An IDE also provides many useful features, such as:
  - Multiple language support
  - JavaScript syntax checking and suggestions
  - HTML5 structure check, tag completion, and attribute suggestions
  - CSS syntax check, rules match-up, and inheritance analysis

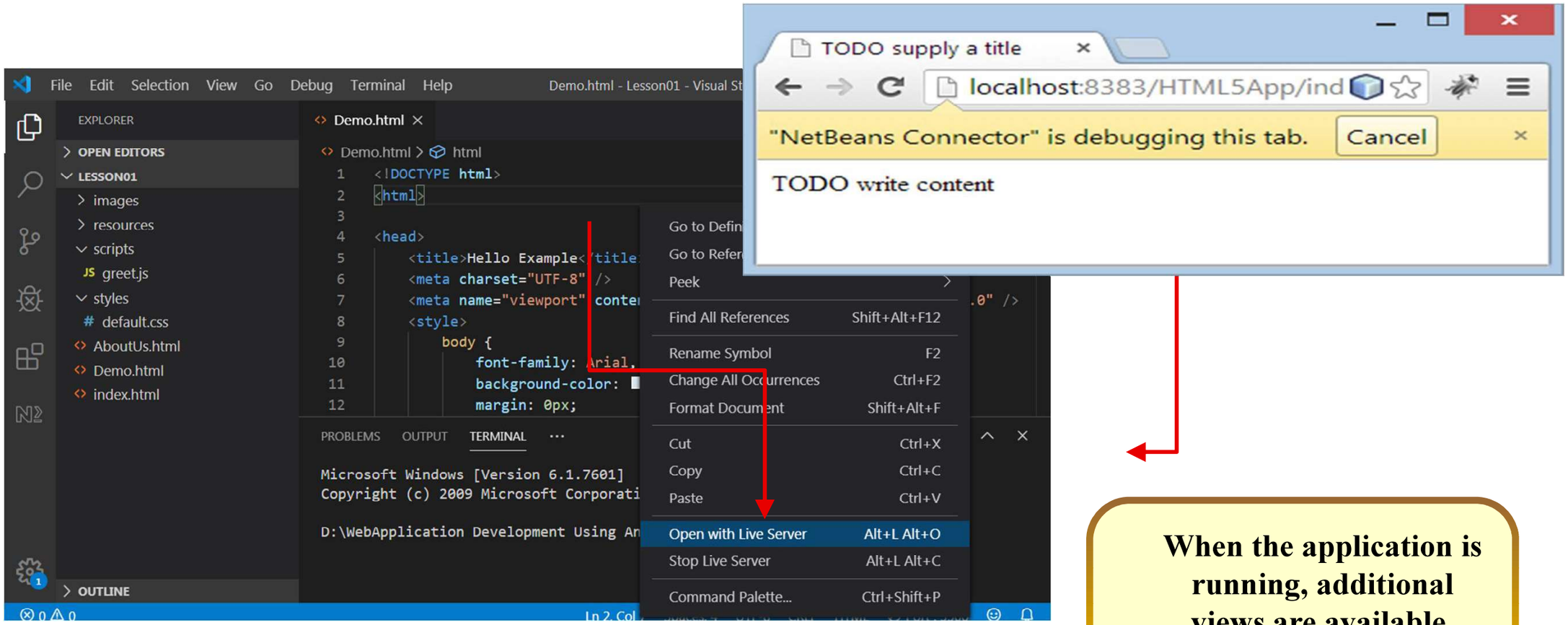
Application Files



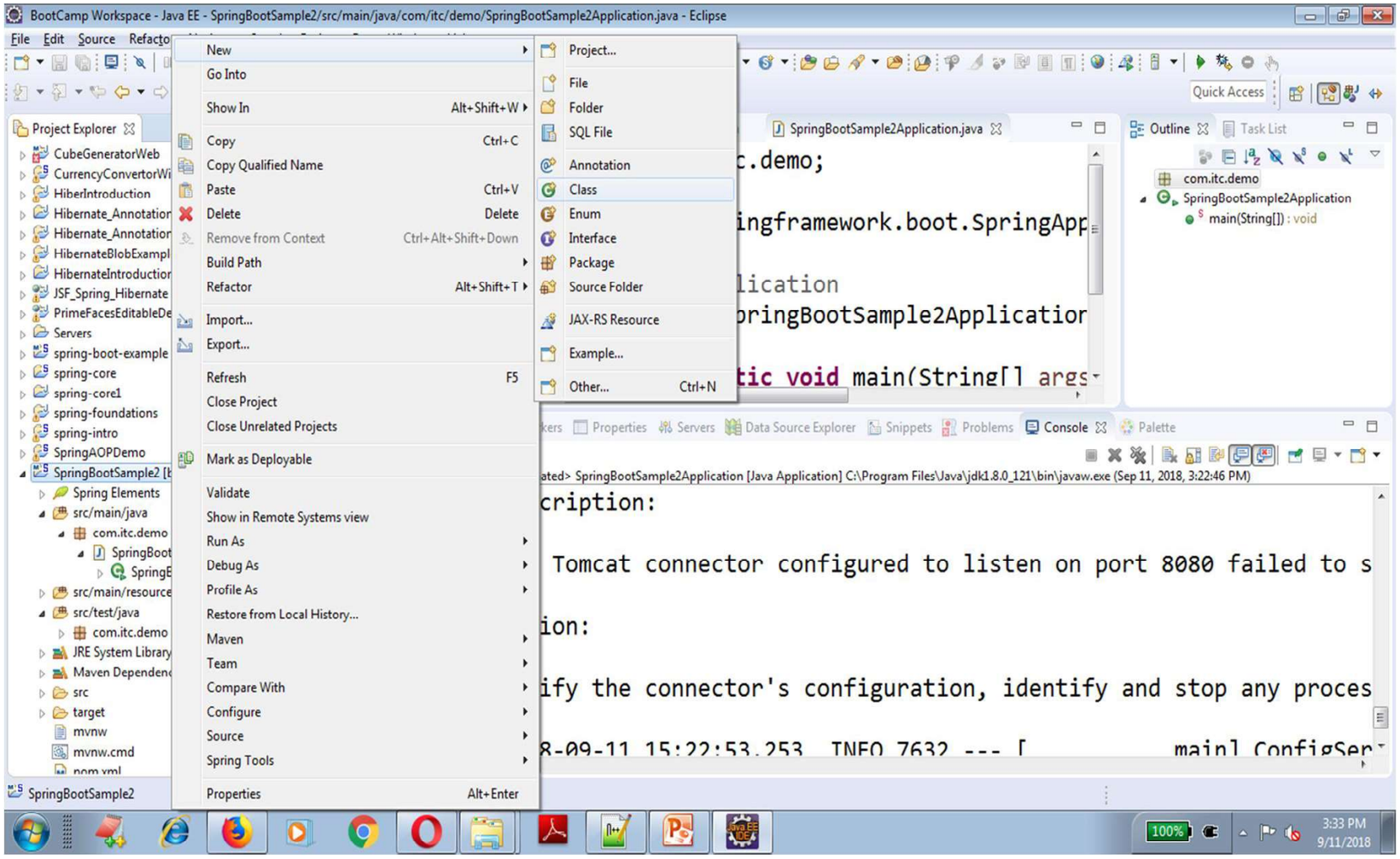
Visual Studio Elements



Running Applications



When the application is running, additional views are available.


BootCamp Workspace - Java EE - SpringBootSample2/src/main/java/com/itc/demo/SpringBootSample2Application.java - Eclipse
File Edit Source Refacto
New
Go Into
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Close Unrelated Projects
Mark as Deployable
Validate
Show in Remote Systems view
Run As
Debug As
Profile As
Restore from Local History...
Maven
Team
Compare With
Configure
Source
Spring Tools
Properties Alt+Enter
Project...
File
Folder
SQL File
Annotation
Class
Enum
Interface
Package
Source Folder
JAX-RS Resource
Example...
Other... Ctrl+N
Project Explorer
CubeGeneratorWeb
CurrencyConvertorWi
HiberIntroduction
Hibernate_Annotation
Hibernate_Annotation
HibernateBlobExampl
HibernateIntroductior
JSF_Spring_Hibernate
PrimeFacesEditableDe
Servers
spring-boot-example
spring-core
spring-core1
spring-foundations
spring-intro
SpringAOPDemo
SpringBootSample2 [b
Spring Elements
src/main/java
com.itc.demo
SpringBoot
SpringB
src/main/resource
src/test/java
com.itc.demo
JRE System Library
Maven Dependenc
src
target
mvnw
mvnw.cmd
pom.xml
SpringBootSample2
SpringBootSample2Application.java
c.demo;
ingframework.boot.SpringApp
lication
pringBootSample2Applicatior
tic void main(String[] args
Outline
Task List
com.itc.demo
SpringBootSample2Application
main(String[]) : void
Quick Access
Properties Servers Data Source Explorer Snippets Problems Console Palette
ated> SpringBootSample2Application [Java Application] C:\Program Files\Java\jdk1.8.0_121\bin\javaw.exe (Sep 11, 2018, 3:22:46 PM)
cription:
Tomcat connector configured to listen on port 8080 failed to s
ion:
ify the connector's configuration, identify and stop any proces
8-09-11 15:22:53.253 INFO 7632 --- [main] ConfigSer
100%
3:33 PM
9/11/2018

Version Controls

- **Version control**, also known as revision control or source control, is the management of changes to documents, computer programs, large websites, and other collections of information.
- Each revision is associated with a timestamp and the person making the change. Revisions can be compared, restored, and with some types of files, merged.
- Version control is particularly useful for facilitating collaboration.

1. Track developments and changes in your files
2. Record the changes you made to your file in a way that you will be able to understand later
3. Experiment with different versions of a file while maintaining the original version
4. ‘Merge’ two versions of a file and manage conflicts between versions
5. Revert changes, moving ‘backward’ through your history to previous versions of your file

Examples

1. Git
2. Helix VCS
3. Microsoft Team Foundation Server
4. Subversion

Git

- Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people.
- It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files.
- Git was created by Linus Torvalds in 2005 for development of the Linux kernel and is free and open source software.

Cloning an existing repository requires only a URL to the repository and the following git command:

- `git clone <url>`

Once cloned, repositories are synchronized by pushing and pulling changes. If the original source repository has been modified, the following git command is used to pull those changes to the local repository:

- `git pull`

Local changes must be added and committed, and then pushed to the remote repository. *Note the period (dot) at the end of the first command.*

- `git add .`
- `git commit -m "reason for commit"`
- `git push`

If there are conflicts between the local and remote repositories, the changes should be merged and then pushed. If necessary, local changes may be forced upon the remote server using:

- `git push --force`

Terms in GIT

- **Branch**

  A separate working copy of files under version control which may be developed independently from origin.

- **Clone**

  Create a new repository containing the revisions from another repository.

- **Commit**

  To write or merge the changes made in the working copy back to the repository.

- **Merge**

  An operation in which two sets of changes are applied to a file or set of files.

- **Push**

  Copy revisions from the current repository to a remote repository.

- **Pull**

  Copy revisions from a remote repository to the current repository.

Summary

In this lesson, you should have learned how to:

- Software Development Process
- Methods Used to Develop and Test Programs
- Pseudocode and Flow Chart
- IDE's, Version Controls

Practice 1:

This practice covers the following topics:

- Creating Pseudocode for few of the Use cases
- Creating Flowcharts for few of the Use Cases