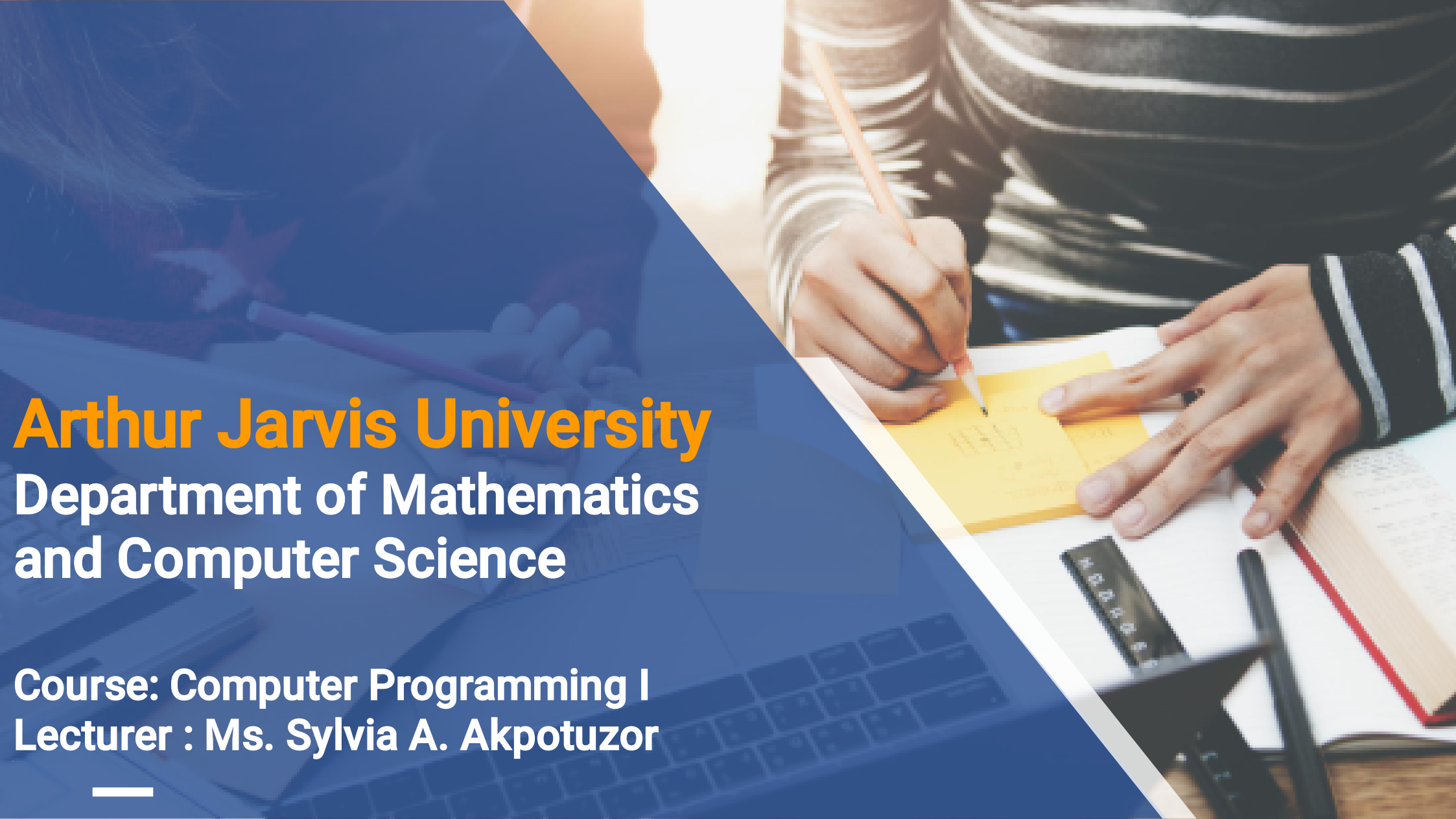


Arthur Jarvis University

Department of Mathematics and Computer Science

Course: Computer Programming I

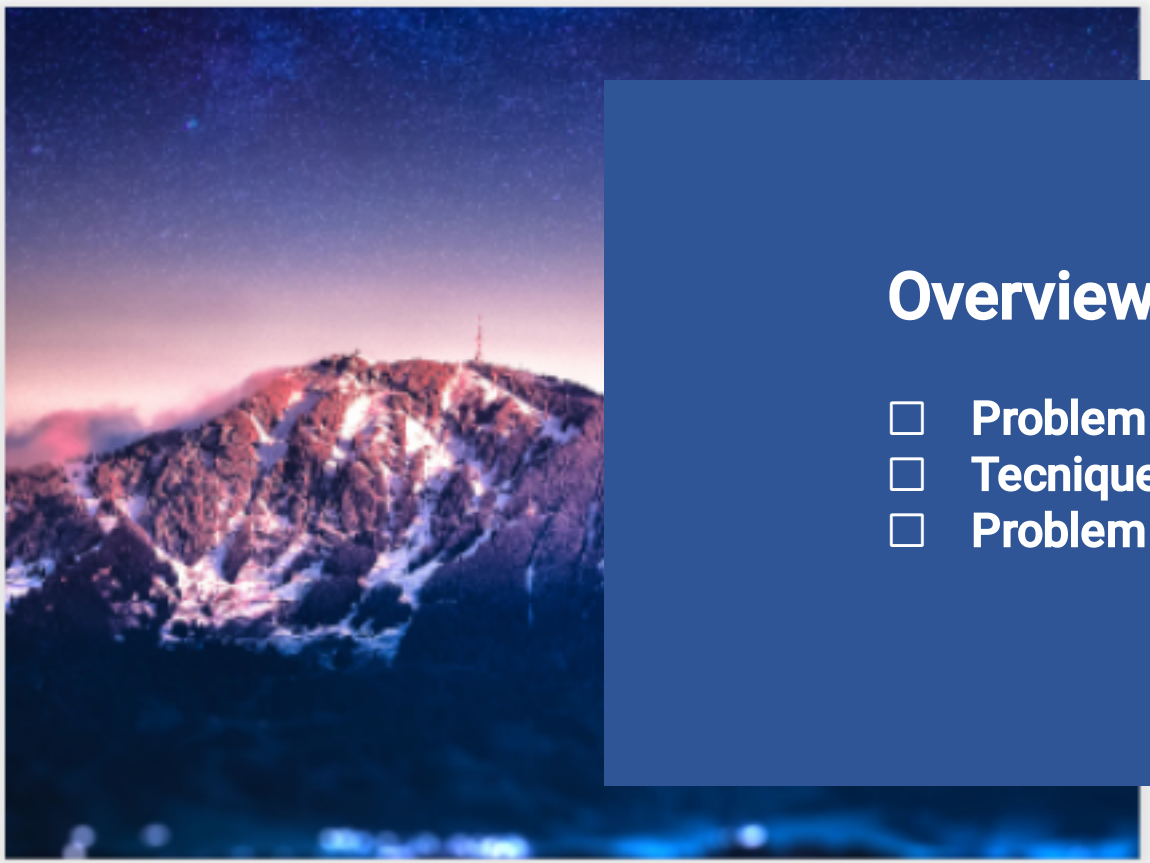
Lecturer : Ms. Sylvia A. Akpotuzor

COURSE CONTENTS

1. Overview of Problem Solving
2. Algorithm Development
3. Programming Language Design Techniques
4. Lab work

Book source:
Programming language design concepts / David A. Watt ; with contributions by William Findlay
Levitin, Anany.Introduction to the design & analysis of algorithms / Anany Levitin. — 3rd ed.

PART 1 ; WEEK 1



Overview of Problem Solving

- ☐ Problem Solving (PS) Definition
- ☐ Tecniques Applied In PS
- ☐ Problem Solving Steps

MEANING OF PROBLEM SOLVING (PS)

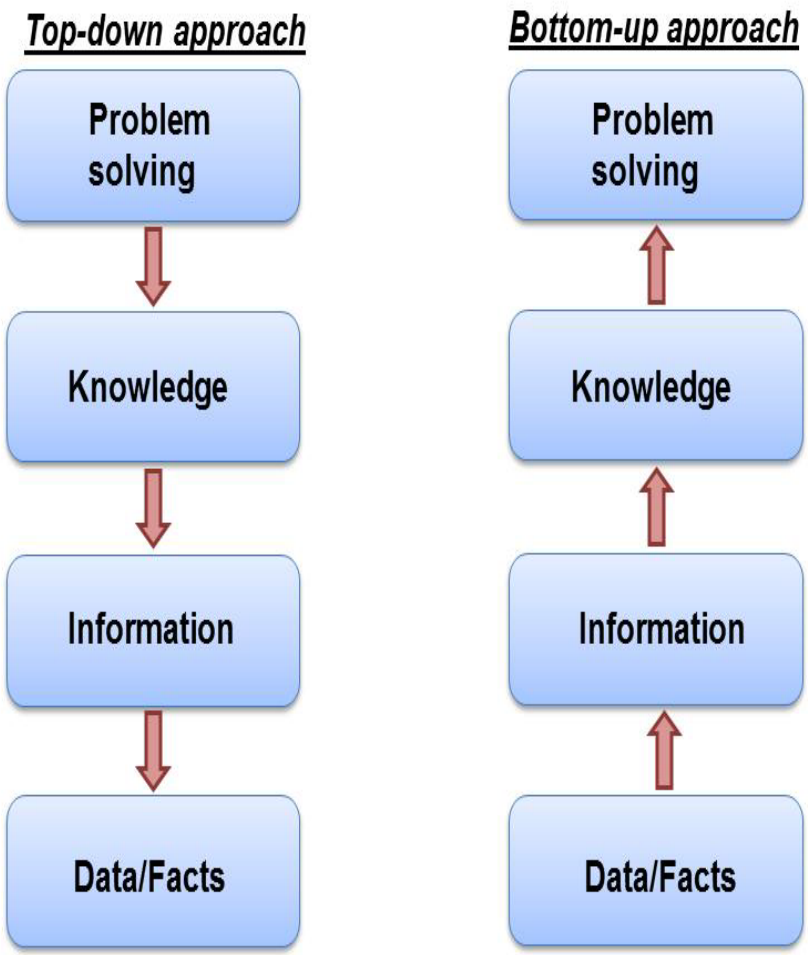
- Computers are used for solving various day-to-day problems and thus problem solving is an essential skill that a computer science student should know. It is pertinent to mention that computers themselves cannot solve a problem.
- The success of a computer in solving a problem depends on how correctly and precisely we define the problem, design a solution (algorithm) and implement the solution (program) using a programming language.
- **We can say that problem solving is the process of identifying a problem, developing an algorithm for the identified problem and finally implementing the algorithm to develop a computer program.**

TECHNIQUES APPLIED IN PROBLEM SOLVING

In Computer Science:

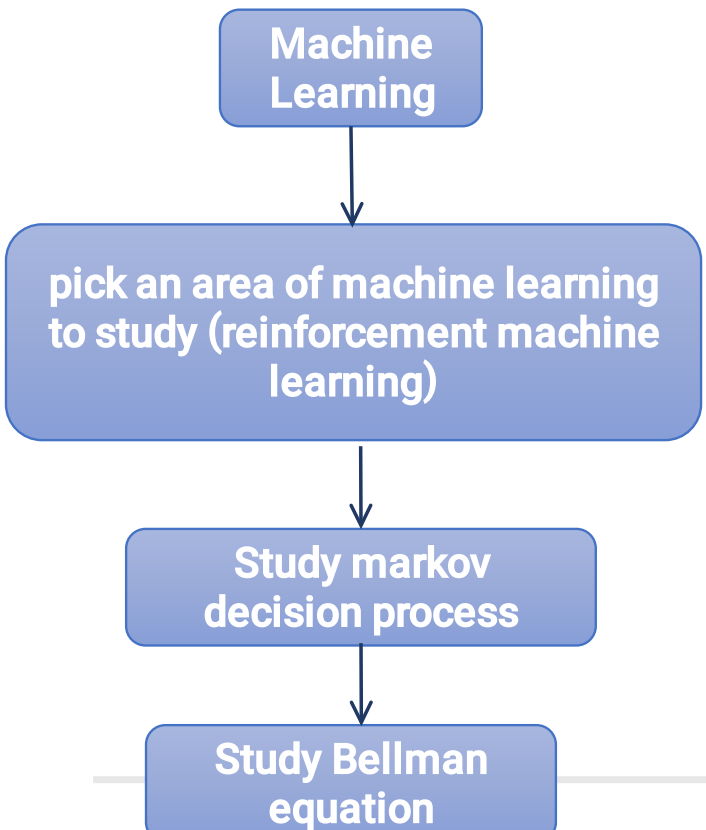
- Top–down approach deals with breaking down problems or algorithms into fragments. This process is called modularization.
- Contrary to the top–down approach, the bottom–up approach focuses on designing an algorithm by beginning at the very basic level and building up as it goes. In this approach, the modules are designed individually and are then integrated together to form a complete algorithmic design.

Generally



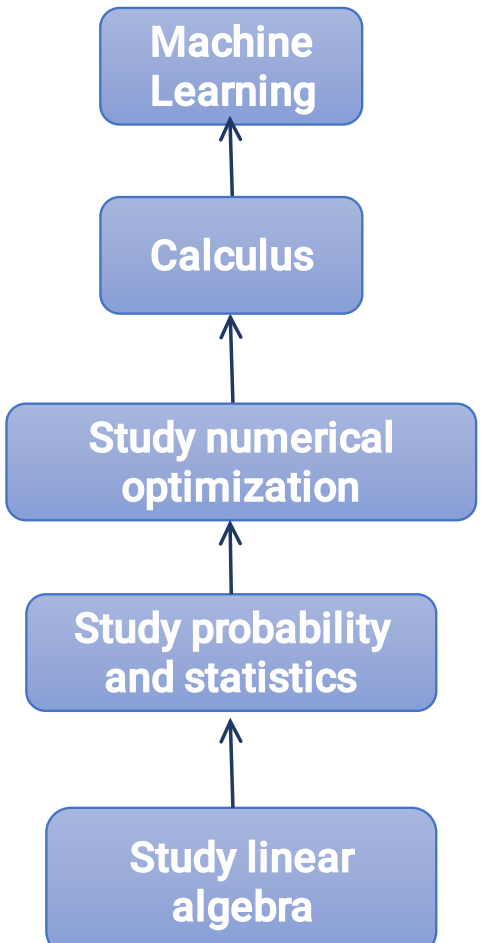
TECHNIQUES APPLIED IN PROBLEM SOLVING

Example of top–down approach

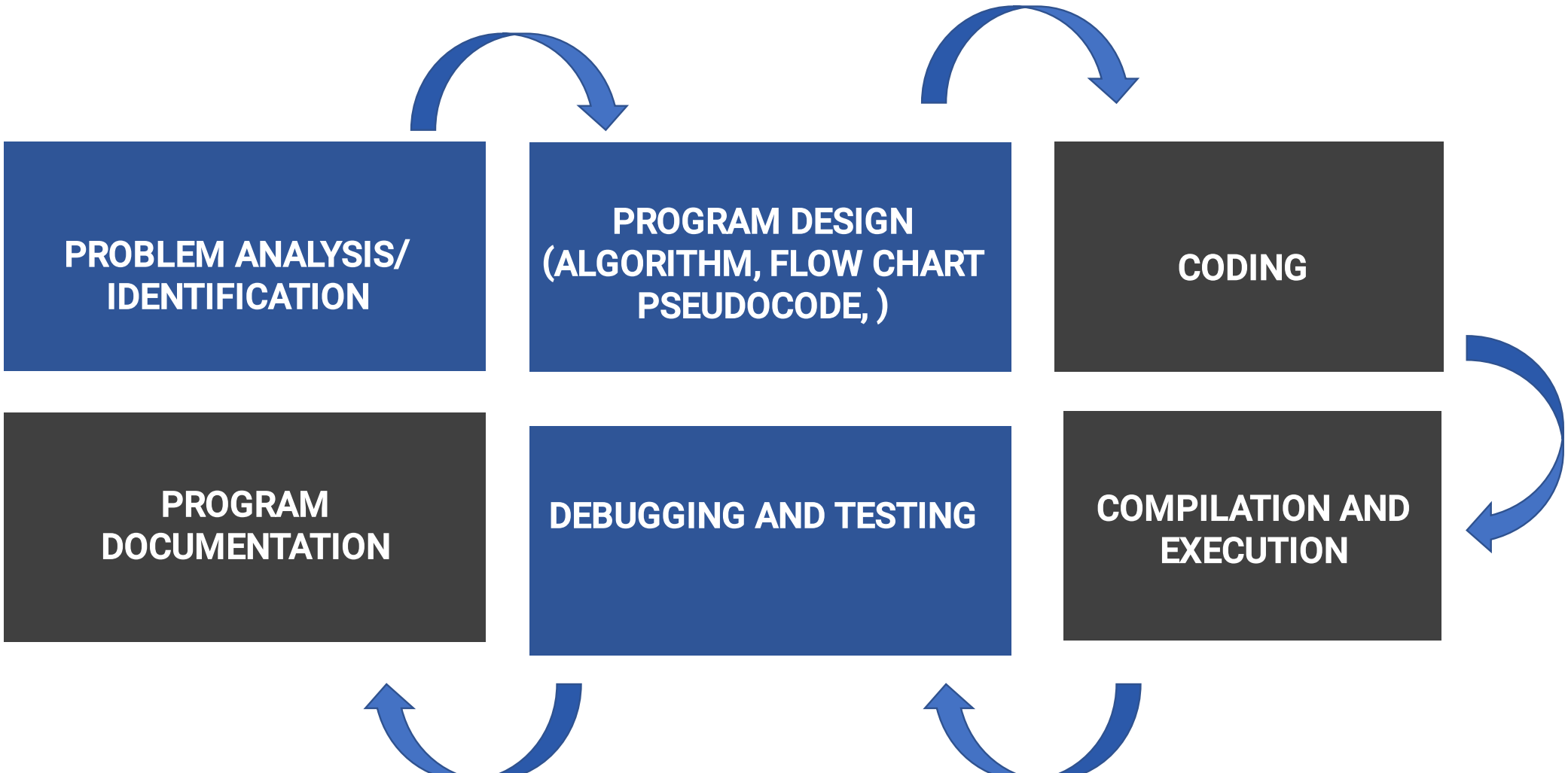


A student is faced with the challenge of determining the best way to study Machine Learning. Should the student use Top–down approach or bottom–up approach?

Example of bottom–up approach



STEPS IN PROBLEM SOLVING



STEPS IN PROBLEM SOLVING

Problem Analysis

It is important to clearly understand a problem before we begin to find the solution for it. If we are not clear as to what is to be solved, we may end up developing a program which may not solve our problem.

we need to read and analyse the problem statement carefully in order to list the principal components of the problem and decide the core functionalities that our solution should have.

By analysing a problem, we would be able to figure out the inputs that our program should accept and the outputs that it should produce.

STEPS IN PROBLEM SOLVING

Problem Analysis

The following are steps involved in problem analysis

a. Understand your problem.
b. Break the problem.
c. Define problem goals.
d. Decide how to measure progress toward goals.

STEPS IN PROBLEM SOLVING

Program Design

It is essential to device a solution before writing a program code for a given problem.

The solution can be represented in form of an algorithm. Pseudocodes and flowcharts are ways to represent algorithms.

For a given problem, more than one algorithm is possible and we have to select the most suitable solution.

STEPS IN PROBLEM SOLVING

Coding

Coding or computer programming is the process of designing and building an executable computer program to accomplish a specific computing result or to perform a specific task.

After finalising the algorithm, we need to convert the algorithm into the format which can be understood by the computer to generate the desired solution.

Different high level programming languages can be used for writing a program.

STEPS IN PROBLEM SOLVING

Compilation and Execution

Compilation is the process the computer takes to convert a high – level programming language into a machine language that the computer can understand. The software which performs this conversion is called a translator.

The translation process influences the design of computer languages, which leads to a preference of compilation or interpretation.

Execution in computing is the process by which a computer or virtual machine reads and acts on the instructions of a computer program.Each instruction of a program is a description of a particular action which must be carried out, in order for a specific problem to be solved.

STEPS IN PROBLEM SOLVING

Debugging and Testing

In computer programming and software development, debugging is the process of finding and resolving bugs (defects or problems that prevent correct operation) within computer programs, software, or systems.

Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test.

Test techniques include the process of executing a program or application with the intent of finding failures, and verifying that the software product is fit for use.

STEPS IN PROBLEM SOLVING

Program Documentation

Software documentation is written text or illustration that accompanies computer software or is embedded in the source code.

The documentation either explains how the software operates or how to use it, or may mean different things to people in different roles.

Types of documentation include:–

a. Requirements.

b. Architecture / Design.

c. Technical documentation of code.

d. End user Manual.

SOLVE A PROBLEM

INSTRUCTION

- Differentiate between top–down and bottom–up approach of solving problems.
- A peasant finds himself on a riverbank with a wolf, a goat,and a head of cabbage. He needs to transport all three to the other side of the river in his boat. However, the boat has room for only the peasant himself and one other item (either the wolf, the goat, or the cabbage). In his absence,the wolf would eat the goat, and the goat would eat the cabbage. Solve this problem for the peasant or prove it has no solution. (Note: The peasant is a vegetarian but does not like cabbage and hence can eat neither the goat nor the cabbage to help him solve the problem. And it goes without saying that the wolf is a protected species.)

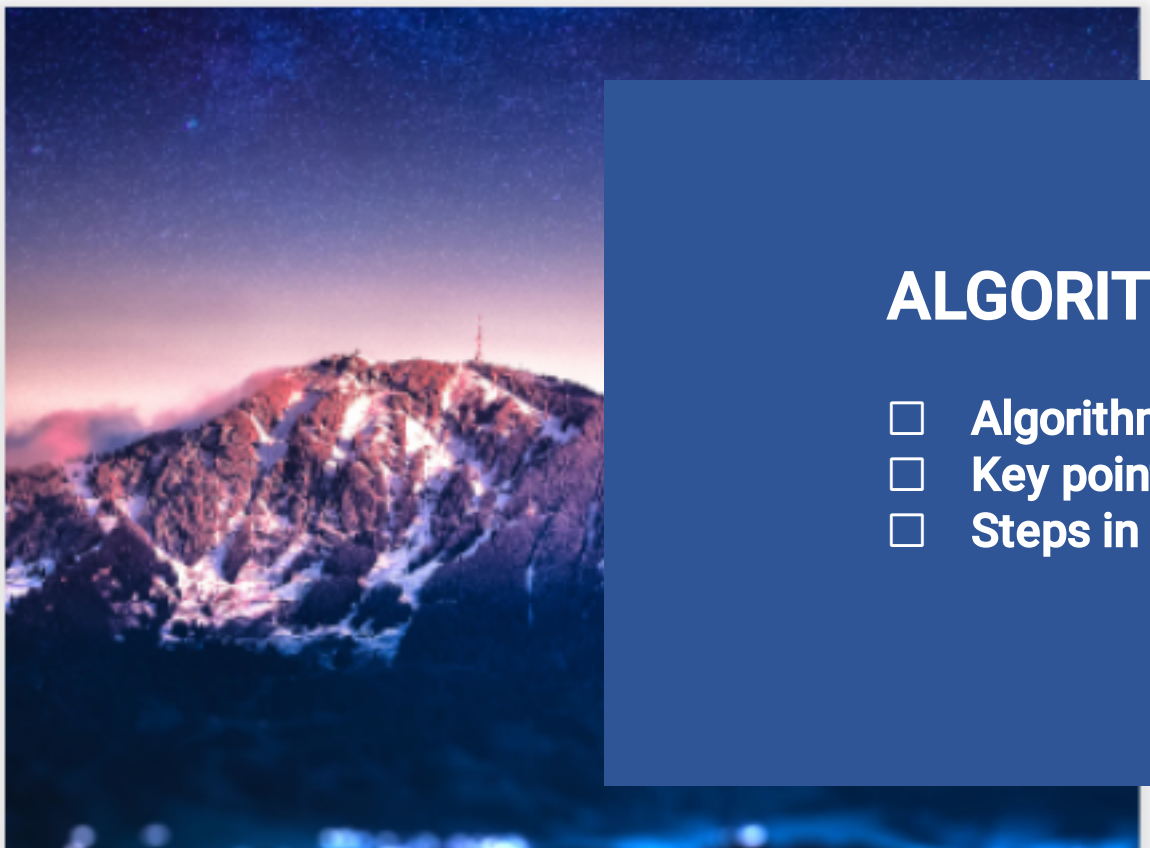
Findings will be discssed in our next class.

Thank You

Lecturer : Ms. Sylvia A. Akpotuzor

End of week 1

PART 2 ; WEEK 2



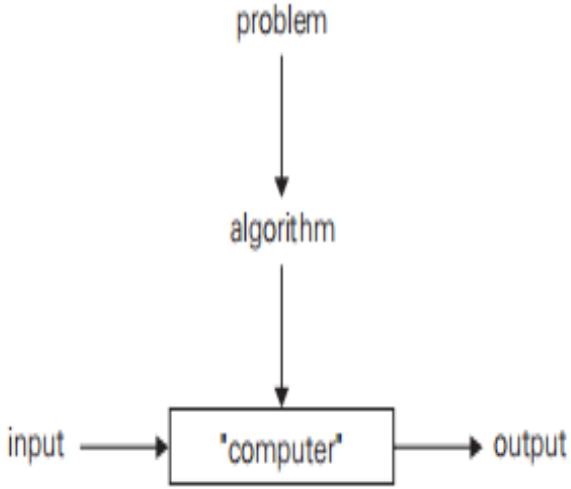
ALGORITHM DEVELOPMENT

- ☐ Algorithm meaning.
- ☐ Key points about algorithms.
- ☐ Steps in algorithm development.

ALGORITHM MEANING

Although there is no universal agreement to describing algorithms, but there is a general agreement about what the concept means:

- Algorithm can be described as a sequence of finite steps generated by the programmer to help in solving an instance of a problem. Writing an algorithm is mostly considered as a first step to programming.Once we have an algorithm to solve a problem, we can write the computer program for giving instructions to the computer in high level language.



- **We can say that An algorithm is a sequence of unambiguous instructions for solving a problem, i.e., for obtaining a required output for any legitimate input in a finite amount of time.**

KEY POINTS TO NOTE ABOUT ALGORITHMS

The important points to note about an algorithm is that:

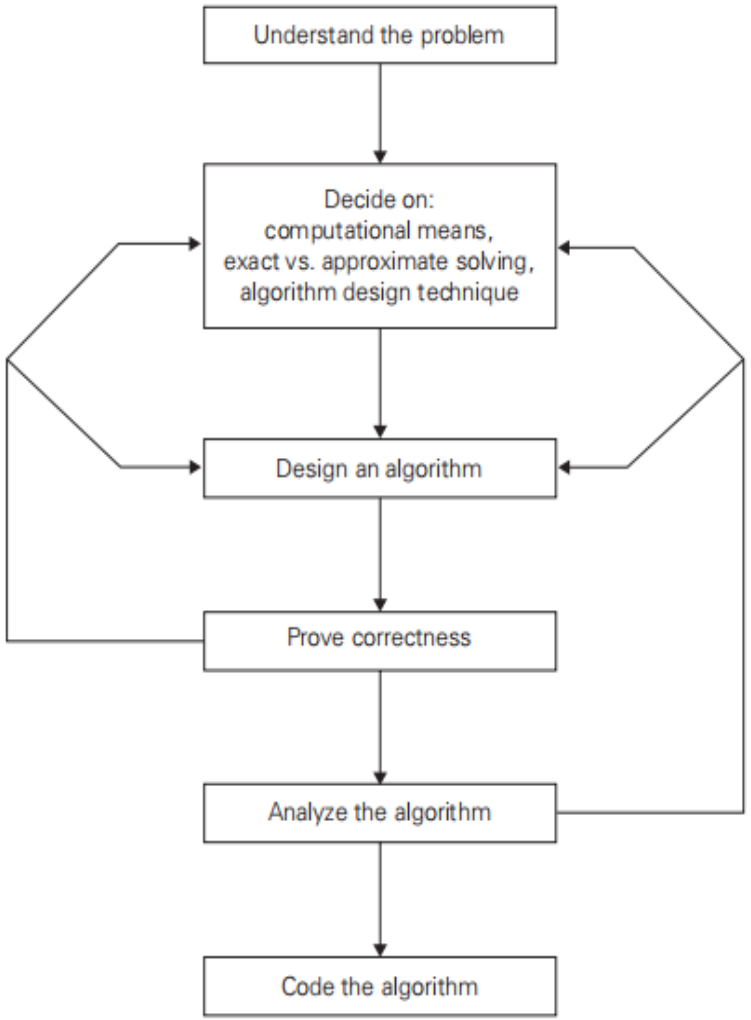
- The nonambiguity requirement for each step of an algorithm cannot be compromised.
- The range of inputs for which an algorithm works has to be specified carefully.
- The same algorithm can be represented in several different ways.
- There may exist several algorithms for solving the same problem.
- Algorithms for the same problem can be based on very different ideas and can solve the problem with dramatically different speeds.

ALGORITHM DEVELOPMENT

- Algorithm development is the act of designing the steps that solve a particular problem for a computer or any other device to follow.

The steps involved in the development of an algorithm are as follows:

- 1. Understand the problem
- 2. Decide on the capabilities of a computational device
- 3. Design an algorithm
- 4. Prove the correctness of the algorithm
- 5. Analyse the algorithm
- 6. Code the algorithm



ALGORITHM DEVELOPMENT

Understand the problem

From a practical perspective, the first thing every student needs to do before designing an algorithm is to understand completely the problem given.

understanding the problem involves either choosing an existing existing algorithm or choosing a range of algorithms based on their strengths and weaknesses or creating a new algorithm to solve the problem.

Understanding the algorithm also involves knowing the instance of the problem. An input to an algorithm specifies an instance of the problem the algorithm solves. It is very important to specify exactly the set of instances the algorithm needs to handle.

ALGORITHM DEVELOPMENT

Decide on the capabilities of a computational device

Once you completely understand a problem, you need to ascertain the capabilities of the computational device the algorithm is intended for.

Algorithms can be designed to be executed on sequential processing (Von Neumann) architectural systems or parallel architectural processing systems. Sequential processing systems executes instructions one after the other while parallel processing systems executes instructions concurrently. It is important to know the architecture of the device an algorithm is being created for.

ALGORITHM DEVELOPMENT

Decide on the capabilities of a computational device

The next principal decision is to choose between solving the problem exactly or solving it approximately. If one chooses to solve approximate problems,one needs to adopt the approximate algorithm.There are important problems that simply cannot be solved exactly for most of their instances; examples include extracting square roots, solving nonlinear equations, and evaluating definite integrals.

ALGORITHM DEVELOPMENT

Design an algorithm

Algorithms are designed based on the choice of an algorithm design technique.

An algorithm design technique (or “strategy” or “paradigm”) is a general approach to solving problems algorithmically that is applicable to a variety of problems from different areas of computing.

There is a need to pay close attention to choosing data structures appropriate for the operations performed by the algorithm design adopted to solve a problem.

ALGORITHM DEVELOPMENT

Design an algorithm

Once you have designed an algorithm, you need to specify the algorithm.

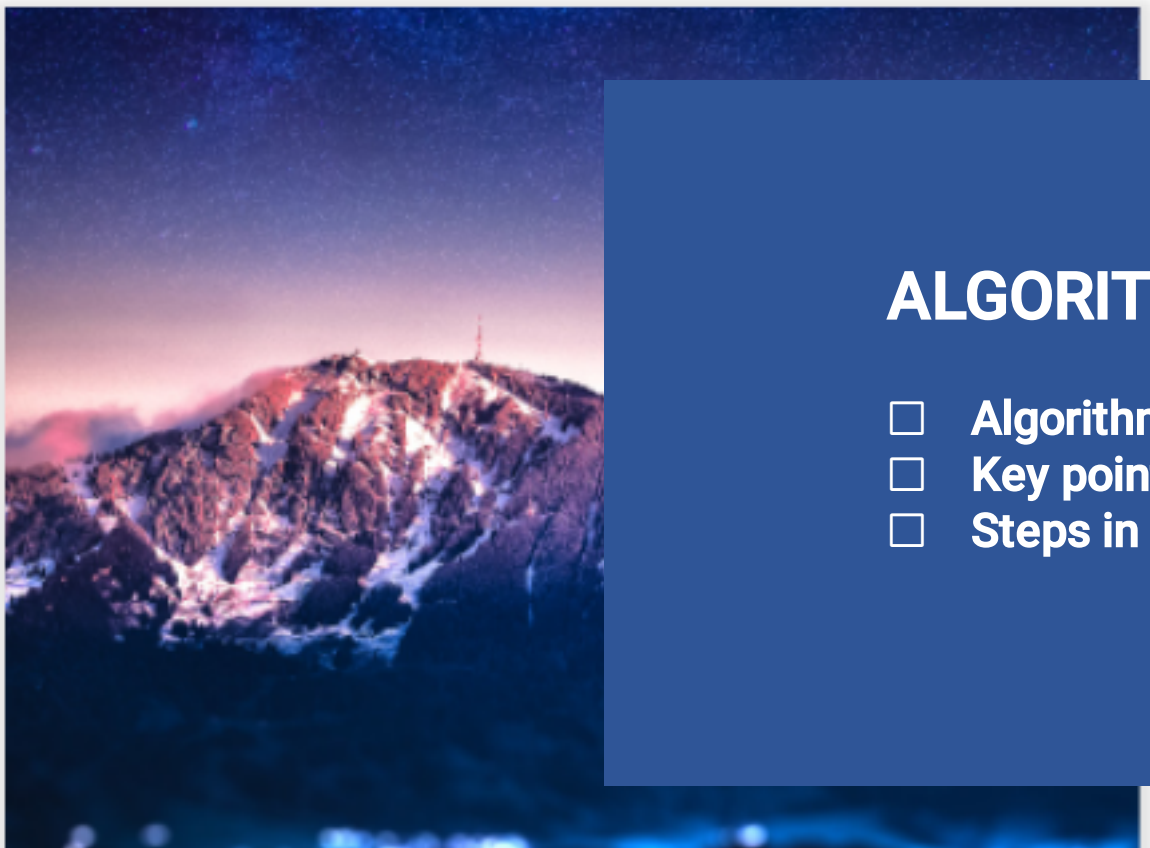
It is more direct to use natural languages in specifying algorithms.However, ambiguity of makes it difficult to describe algorithms. Pseudocode is a mixture of a natural language and programming language like constructs. Pseudocode is usually more precise than natural language, and its usage often yields more succinct algorithm descriptions. Flowchart, a method of expressing an algorithm by a collection of connected geometric shapes containing descriptions of the algorithm's steps is another way of specifying an algorithm.

Thank You

Lecturer : Ms. Sylvia A. Akpotuzor

End of week 2

PART 2 ; WEEK 3



ALGORITHM DEVELOPMENT

- ☐ Algorithm meaning .
- ☐ Key points about algorithms.
- ☐ Steps in algorithm development .

ALGORITHM DEVELOPMENT

Prove the correctness of the algorithm

Once an algorithm has been specified, you have to prove its correctness. That is, you have to prove that the algorithm yields a required result for every legitimate input in a finite amount of time.

For some algorithms, a proof of correctness is quite easy; for others, it can be quite complex. A common technique for proving correctness is to use mathematical induction because an algorithm's iteration provide a natural sequence of steps needed for such proofs.

ALGORITHM DEVELOPMENT

Analyse the algorithm

Analysing an algorithm simply means adding several qualities to the algorithm.

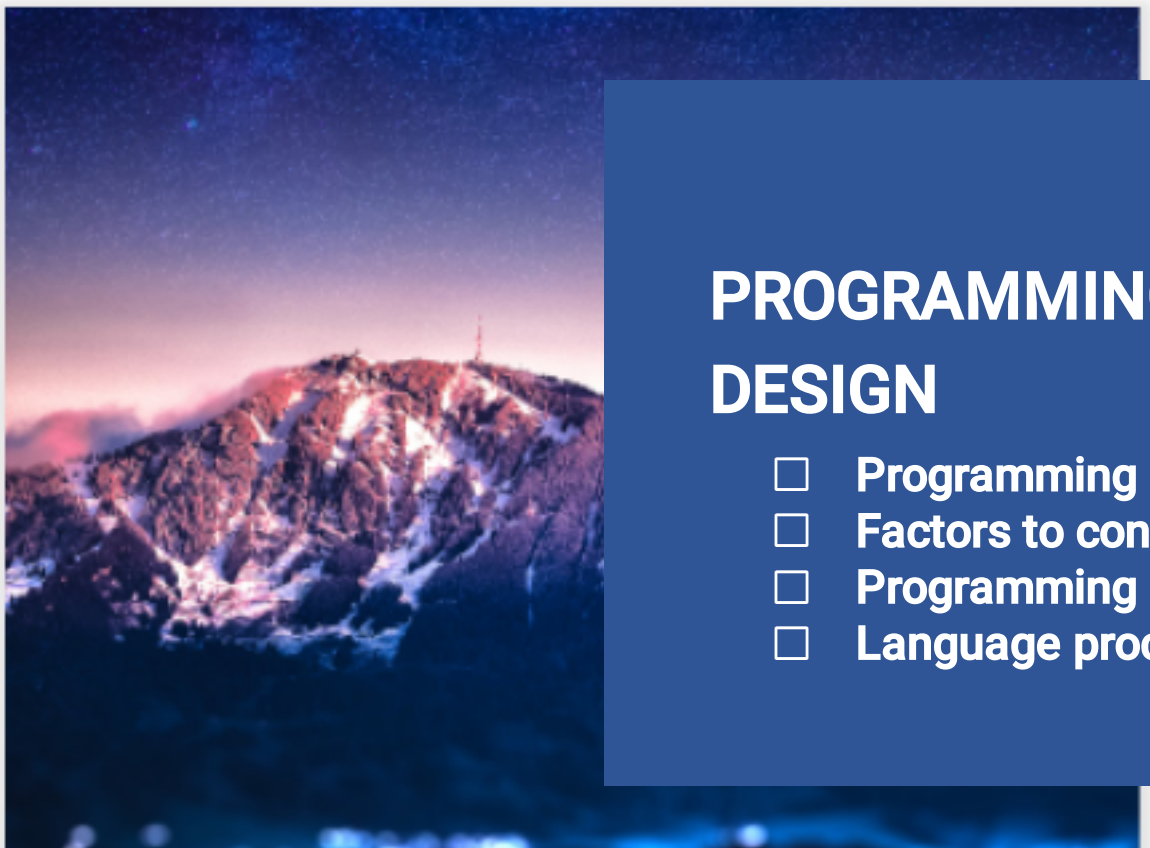
A very important quality is the quality of efficiency. There are two kinds of algorithm efficiency: **time efficiency**, which indicates how fast the algorithm runs, and **space efficiency**, which indicates how much extra memory it uses. Other qualities are simplicity and generality.

ALGORITHM DEVELOPMENT

Coding the algorithm

Coding the algorithm simply means converting every step of the algorithm into programs written in high level languages. Most algorithms are destined to be ultimately implemented as computer programs.

PART 2 ; WEEK 3



PROGRAMMING LANGUAGE DESIGN

- ☐ Programming Languages (PLs) .
- ☐ Factors to consider when deigning PLs.
- ☐ Programming Language paradigms / concepts .
- ☐ Language processors / translators.

PROGRAMMING LANGUAGE DESIGN

Programming languages

Programming is the art of performing computational tasks and converting these tasks into machine readable form.

The series of instructions for writing programs is known as a **programming language**. Every programming language is an artifact, and as such has been consciously designed. Some programming languages have been designed by a single person (such as C++), others by small groups (such as C and JAVA), and still others by large groups (such as ADA).

PROGRAMMING LANGUAGE DESIGN

Factors to consider when designing PLs

Concept selection: In designing PLs several concepts are to be taken into consideration. Concepts such as values and types, variables and storage, bindings and scope,and procedural abstraction. These concepts are found in most programming languages.

Simplicity: Simplicity should always be a goal of language design. The language is the most basic tool for programmers, which must be mastered thoroughly. The language should help programmers solve problems, it should allow programmers express solutions naturally, and indeed should help them discover these solutions in the first place. A large and complicated language creates problems by being difficult to master.

PROGRAMMING LANGUAGE DESIGN

Factors to consider when designing PLs

Efficiency: Every language should be capable of an acceptabe efficient implementation. What is acceptable depends on what is required of programs in the application area for which the language is intended. A language intended for system programming must be highly efficient. A language intended for ordinary application programming must be reasonably efficient. A language intended for programming applets or for scripting need not be particularly efficient.

Language Life Cycle: Each new language passes through a number of stages from its initial design to everyday use by programmers:The stages are as follows.

PROGRAMMING LANGUAGE DESIGN

Factors to consider when designing PLs

- The requirements for the new language are identified.
- The language's syntax and semantics are designed.
- A specification of the new language is prepared.
- Using the specification, an initial implementation of the new language is constructed.
- The specification, implementation, and programmers' experience all provide feedback to the designer, drawing attention to any un–satisfactory aspects of the design, which can then be improved.
- If the new language is successful, improved implementations are constructed, textbooks are written, and a community of programmers is builtup.

SOLVE A PROBLEM

INSTRUCTION

Which of the following formulas can be considered an algorithm for computing the area of a triangle whose side lengths are given positive numbers a, b, and c?

a. $S = \sqrt{p(p-a)(p-b)(p-c)}$, where $p = (a+b+c)/2$

b. $S = \frac{1}{2}bc \sin A$, where A is the angle between sides b and c

c. $S = \frac{1}{2}ah_a$, where h_a is the height to base a

Findings will be discssed in our next class.

Thank You

Lecturer : Ms. Sylvia A. Akpotuzor

End of week 3