**OUTLINE**

1. **Introduction to computer science**

a. Intro

b. What is computer science?

1. **Introduction to computer programming**
2. Computer programming
3. Compilers and interpreters
4. Bits & Bytes, Variables, Data types, Type Checking
5. Data Structures, control flow, Loops , Recursion
6. Big O notation, programming paradigms
7. Functional programming, Algorithms, Code Editors, Operating Systems
8. **Introduction to HTML**
9. Hello & welcome to HTML
10. What is HTML,CSS and Javascript
11. How the web works
12. History of HTML
13. HTML basics
14. **HTML**
15. Self Closing tags
16. Nesting elements, Empty elements and anatomy of an HTML Document
17. Installing pre-requisite tools
18. Let’s build our First HTML Document
19. **INTRODUCTION TO COMPUTER SCIENCE**

**What is Computer Science?**

Computer science is the study of computers including the hard ware and software . it is a broad filed and includes all aspects of computers including figuring out new ways to solve problems using computers and improving how computers work.

Practically, Computer science is all about problem solving.

**Different Fields in Computer Science**

1. Artificial Intelligence

it is a design of systems that can function without human control. Eg: programs that play chess against a human competitior or programmes that suggest music based on your listened history.

1. User interface Designer(UI)/User experience Designer(UX)

It involves designing the interface people interact with. These interactions could happen through many forms such as: websites or web apps, mobile phones, Virtual reality e.t.c.

UI/UX designers often have to collaborate with frontend engineers and backend engineers to have a finished product.

1. Data Science

Data scientists mine large data sets to gain insights or useful information.

1. Software Engineering

It is a branch of computer science that enables the application of principles of computer scince in order to design and develop software.

1. **INTRODUCTION TO COMPUTER PROGRAMMING**
2. **Computer Programming**

Computer programming also known as coding involves writing well structured instructions for a computer to execute.

Computer only understands machine Languages which are 0 and 1.

**Computer programming are divided into two**

* Low Level Language
* High Level Language

**Low level Language**

They are divided into :

* Machine Languages

They are languagues that can be understood directly by the computer.

* Assembly Languages

They are languages that needs to be translated by an assembler into machine Language. Assembly Language is less complicated than machine language and it has some human readable parts.

**High Level Language**

High level languages use compilers and interpreters.

1. **Compliers and Interpreters**

Compilers and interpreters convert computer programs writing in one language to another. Which is usually a machine language that can be processed by a computer.

However, while a compiler translates the entire program at once, an interpreter translates just one statement of the program at a time.

A compiler is generally faster than an interpreter because it analyzes each statement just once, while an interpreter must analyze it each time.

**NB;**  Data is a collection of values which can be symbol or numbers etc. Digital data are programmed in a step sequence and jumps from one value to the next. They are discrete and discontinuous.

Analog data are continuous and fluctuate infinitely within a range eg: sound

**Analog Devices include**: Loud speakers, Thermometers, amplifiers. Data can be converted from digital to analogue and vice versa using ADC and DAC converters.

Information refers to processed data . when data is interpreted, it provides context with which we can make informed decisions .

**C). BITS AND BYTES, VARIABLES, DATA TYPES, TYPE CHECKING**

**Bits and Bytes**

A bit is denoted with a small letter b while a byte is denoted with a capital letter B. Bytes are commonly used when referring to size of data on a computer while bit measurement can be seen more often in describing network speed.

**Variables**

Variables are used to store data that is likely to be reduced and or manipulated in a program. It is important to use descriptive words when naming a variable as it can get difficult to manage obscure variable names in large programs.

**Assigning a program to a variable**

In most programming languages, the equal sign/ assignment operator is used to assign a value to a variable with the variable name on the left side of the operator and the value on the right.

**Data types**

It refers to the attributes of a value which provides the compiler on what type of mathematical logical or relational operations that can be carried out on it without throwing an error.

Data types include:

* **Integers**

It represents non-decimal numbers including negative ones.

* **Floating Numbers**

It represents decimal numbers

* **Boolean Operators**

Boolean operators are used to represent conditionals. They are used to depict when something is either true or false.

* **Text/strings**

It represents words containing letters, digits, symbols and signs

* **Null**

It represents an unknown data type

* **Pointers**

It helps to store memory address of a value

**TYPE CHECKING**

It involves ensuring that operations carried out on values of each particular data type in a computer program are valid.

A type error is thrown if an invalid operation is carried out.

**Methods of type checking**

1. Static
2. Dynamic

**static**

It is done at compile time. Compile time is when a compiler translates the source code .

**Dynamic**

It does type checking at run time.

1. **Data structures, control flow, loops, Recursion**

**Data Structures**

A data structure provides you with a way of organizing data to use them effectively and efficiently.

Examples of data structure

1. Arrays
2. Stacks
3. Queues
4. Linked lists
5. Heap
6. Trees

**Arrays**

An array is a collection of data, each of which can be identified with an index or key.

**Some are zero based**: C, Javascript, Python , Java

**Some are one based** : fortran, MATLAB, pascal

**Some are n-based** :Perl, PHP

**NB:** In dynamically typed Languages such as Javascript, an array can contain different data types while in languages like C or C++, an array can only contain similar data types.

**Stacks**

Stacks are a linear data collection that only allows addition and removal of items in a first in last out order (FILO) eg: stack of plates

Stacks can be applied in different scenarios such as :

1. Call stack that stores function calls
2. Undo and Redo operations

**Queues**

A similar data structure to stacks but instead follows a first in first out order (FIFO) is the queue data structure.

Queues can be applied to the following operations

1. Job scheduling for operations which may take a long time
2. To handle congestion in network request

**Linked List**

A Linked List is a linear data structure that includes a series of connected nodes. A node contains the memory address and data to the node.

The first node refers to the Head and the last to the Null.

**Hash Tables**

A Hash table is an efficient way to store, modify and retrieve data, especially when a large amount of data is involved.

A hash function takes the key and generates an integer of fixed size which will then be associated with that key used in cryptography, security e.t.c.

A hash table can be useful when trying to keep a data base of users in an organization

**Control Flow**

The control flow of a program is the order in which a program executes.

Types of control flow

1. Sequential

Default flow of every program. The code execution is happening in a sequence at which the program was written.

1. Selection/conditionals

A statement or a set of statements is executed when a particular condition is true and ignored when the condition is false.

1. Repetition/Loops

A sequence is repeated multiple times until a particular condition is met. It is also known as Decision control. It means a program will do one multiple alternative based on if a condition resolves to true or false.

**Conditional statements**

1. If
2. If/else
3. Nested if
4. Switch

**If**

If a particular action is carried out then this other function should be carried out.

**If/else**

If something happens do this else do this other thing.

**Nested if**

If statement inside another if statement

**Switch**

Switch statements use keywords like switch , case, break e.t.c

**LOOPS**

A Loop is a programming feature that lets us repeat a sequence of instructions until a condition is met.-The break condition .

If a break condition is not specified, the loop will continue running for as long as your program is active. This called an infinite loop.

**Kinds of Loops**

1. FOR Loop
2. While Loop

**FOR Loops**

FOR Loops are more commonly used when we know the actual number of times we want the loop to run.

A FOR Loop consists of a header portion and a body portion with the header typically consisting of 3 parts and the body containing the code to be executed while our condition remains true.

**While Loops**

A while Loop runs for as long as a conditional statement passed into it remains true.

**Recursion**

Recursion is a program that calls itself until it gets to a base condition. A recursive function can call itself directly or indirectly.

It is important to have a base condition else, your function continues to call itself infinitely. Usually a recursion problem can be solved with Loops and Vice versa.

1. **Big –O- Notation, programming paradigms**

**Big-O- Notation**

It is also known as Time complexity. It defines the long time it can take a program to perform an operation.

Common Complexities

1. O(1)- Constant time complexity
2. O(n)- Linear time complexity
3. O(log n)- Logarithmic time complexity
4. O(n^2)- Quadratic time complexity

**Programming paradigms**

It refers to a way of solving problems using the features available on a language.

**Imperative Programming paradigms**

Imperative programming uses a series of commands which specify what the computer has to do and when in order to achieve a desired result eg: Java, Fortran, Assembler e.t.c

**Declarative programming paradigm**

It focuses more on what a program should accomplish and less on how to go about it. While imperative programming provides instructions for assembly . Declarative programming provides a picture of the finished piece of furniture as a template.

**Procedural programming**

It follows a linear, step- by-step approach to creating software. Each procedure is usually a function or a set of functions.

Advantages of Procedural programming

1. Very useful in general purpose programming
2. Easy to Learn
3. Code can be reused
4. Easy to transfer skills to another Language

Disadvantages of procedural programming

1. Does not work for complex application
2. The data is exposed to the whole program, hence it is not very safe.
3. It can be hard to modify and debug as the application gets managed
4. Difficult to create new data types
5. Fails to model real world applications making it difficult to design.

**Object Oriented Programming**

It structures a software program into simple, reusable pieces of code blueprints (usually called classes) which are used to create individual instances of objects.

**The Four Principles of Object Oriented Programming (OOP)**

1. Encapsulation
2. Abstraction
3. Inheritance
4. Polymorphism

**Encapsulation**

Encapsulation principle states that all important information is contained inside an object and only select information is exposed.

**Abstraction**

Applying abstraction means that each object should only expose a high level mechanism for using it.

**Inheritance**

This property of OOP forces a more thorough data analysis, reduces development time and ensures a higher level of accuracy.

**Polymorphism**

It gives a way to use a class exactly like its parent so there’s no confusion with mixing types. But each child class keeps its own methods as they are.

It allows different types of objects to pass through the same interface.

Advantages of Object oriented programming (OOP)

1. Modularity

Encapsulation enables objects to be self- contained which makes trouble shooting and collaborative development easier.

1. Flexibility

Polymorphism enables a single function to adapt to the class it is placed in. different objects can also pass through the same interface.

1. Security

Using Encapsulation and abstraction complex code is hidden, software maintenance is easier and internet protocols are protected.

1. Reusability

Code can be reused through inheritance meaning a team does not need to have to write the same code over and over again.

**Criticisms of OOP**

1. It can be complicated to write
2. It provides a steep learning curve
3. It requires a lot of work to create.
4. **Functional programming Algorithms, Code Editors, Operating Systems**

**Functional programming**

Functional programming removes interdependencies between programs by replacing procedures with pure functions which requires the use of shared and immutable state.

Functional programing languages include:

* Lisp
* Erlang
* Haskell
* Clojure

Functional programming terms

Pure Function

Pure function is a function in which given the same inputs always returns the same output and has no side effects. Doesn’t change the global variable, hence it has no side effects.

A side effect is any application state change that is observable outside the called function other than its return value.

Side effects include:

* Modifying any external variable or object property
* Writing to the screen
* Writing to a file

A shared state is a variable that can be accessed from more than one function. Another common problem associated with shared state is that changing the order in which functions are called can cause a cascade of failures because functions which act on shared state are timing dependent.

An immutable object is an object that can’t be modified after it’s created. Conversely, a mutable object is any object which can be modified after it’s created.

Advantages of functional programming

1. Functional program tends to concise, more predictable and easier to test than imperative or object oriented code
2. Use of pure functions

always produces the same output and have no external values affecting the end result.

1. Static variables They are items the user cannot modify, so variables once it has been initiated it is secure.
2. Powerful abstraction

If we write a bunch of pure functions, we get to leverage referential transparency to abstract away and hide complexity.

Disadvantages of functional programming

1. Steep learning curve
2. Input/output operations rely on side effects, so they are almost non-functional in functional programming.
3. Recursion, which is quite popular with functional programming is memory intensive which can affect speed.

**Algorithms**

An algorithm is a set of instructions for solving a problem or accomplishing a task.

**Types of algorithm**

* Binary Search
* Search algorithms : linear and Binary
* Sorting algorithms: merge sort, quick sort

**Code editors**

A Code editor is a text editor that is specialized for writing software. A source code editor may be a stand alone program or part of an integrated development environment (IDE)

An IDE is a text editor, a code editor, a debugger, compiler and more all under a simple tool belt.

**Command Line**

It is a quick powerful text based interface developers use to a more effective and efficient communication between computers to accomplish a wider set of tasks.

Examples of command line

Windows: Mintty, Git Bash

Mac OS: Zsh, powershell, iterm

Linux: Zsh, powershell

**Operating system**

An Operating system is the most important software that runs on a computer. It manages the computer’s memory and processes as well as all of its software and hardware.

**Kinds of operating system**

* Windows
* Mac
* Linux

**Linux**

It is a family of open- source operating systems, which means they can be modified and distributed by anyone around the world.

Advantages of linux are that it is free and there are many different distributions of versions you can

choose from.

**3.SSS INTRODUCTION TO HTML**

**HTML**

It is the mark up language used to define the content we display on web pages.

**CSS**

It is used to define appearances of the content that we create with HTML

**How the web works**

The web

* Client: which is basically the browser, resource and request
* Web Server: where the particular website you are trying to locate is.

URL

It is used to locate or request for resources on the internet. Each URL points to a unique resource.

HTTPS

It is the messaging language that the client and server use to communicate.

Document object Model (DOM)

It is the collection of all the contents inside a web page.

**History of HTML**

In 1989, Tim Berners-Lee invented the web with HTML as its publishing language.

**HTML Basics**

HTML elements are used to display contents in an HTML page. Attributes contain extra information about the elements you don’t want to appear in the actual content.

**Nesting Elements**

You can put elements inside other elements too.

**Installing tools**

1. Download Vs code
2. Prettier code formatter Extension
3. Live server Extension
4. Google Chrome