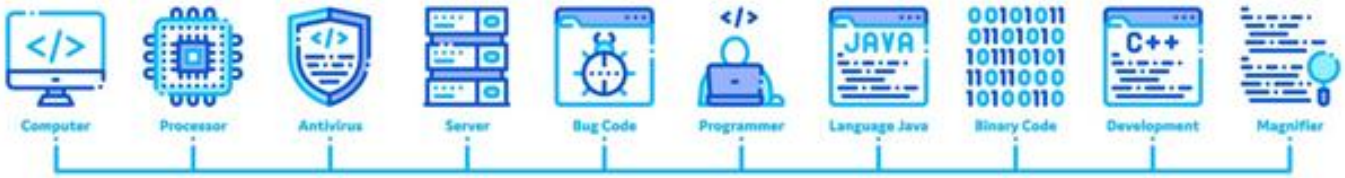
CITY GOVERNMENT OF SAN PABLO
PAMANTASAN NG LUNSOD NG SAN PABLO
CHED Recognized Local College
TESDA Recognized Programs
ALCU Commission on Accreditation – Level 1 Reaccredited
Member, Association of Local Colleges and Universities
Member, Local Colleges and Universities Athletic Association, Inc.
Patriotism • Leadership • Service • Professionalism



MODULE 1.1

Integrative Programming and Technologies
IPT222

2ND SEMESTER
A.Y. 2023 - 2024

GUIDE ON HOW TO USE THE LEARNING MODULE

For Faculty

1. The Course Instructor must review and check the list of the enrolled learners in the course together with the contact information, the learners have provided such as cell phone number and email address.

2. Before distributing the Learning Modules make sure it has been checked and reviewed by the College Program Chair, Department Chair and the College Dean and with the recommending approval from the CLAMDev (Center for Learning and Assessment Materials Development).

3. Once it has been checked and approved, the Course Instructor must provide each learner a softcopy of the Learning Modules through a provided LMS (Learning Management System); any social media platforms will also be considered.

4. It is the responsibility of the Course Instructor to ensure that all of the learners have downloaded a softcopy of the Learning Modules.

5. Encourage the class that it is way better if they will print a hardcopy of the Learning Modules; but consider this as an option if they lack of resources.

6. Orient the class on how to properly use the Learning Modules and explain the content especially the lesson proper, the activities and exercises, the grading rubrics and criteria, and the submission format.

7. The Course Instructor must be open to any concerns and inquiries of each learner regarding the lessons and activities in the Learning Modules inclusively.

8. Always provide each learner constructive feedback and a key to correction for every quiz, activities and exercises they have done. Motivate them!

9. Give each learner enough time to accomplish and submit the requirements or any activities; especially not all learners have sufficient resources. Be considerate.

GUIDE ON HOW TO USE THE LEARNING MODULE

For Learners

1. Each learner enrolled in the course will be provided by their Course Instructor a softcopy of the Learning Modules through an LMS (preferably Google Classroom) or any social media platforms available such as Facebook Messenger.

2. It is recommended for each learner to print a hardcopy of the Learning Modules for ease and clear understanding of the lessons; but again, consider it as an option.

3. Different set of Learning Modules will be sent to each learner: weekly or monthly; depending on their Course Instructors.

4. Each Learning Modules contains different topics and lessons; and at the end of each lesson the learner must accomplished different activities and exercises.

5. The activities vary to every topic covered on the Learning Modules, this activity consists of Academic and Life Activity. Academic Activities contains Quizzes and Individual or Group Activities.

6. For every group works and activities provided in the Learning Modules, make sure to work well with your teammates and collaborate with them to accomplish the work assigned smoothly.

7. When submitting an activity or exercises online, learners must follow the mode and format of submissions included in the Learning Modules.

8. Learners must ensure a good time management and prioritization regarding the tasks and activities given to them. This is a very helpful reminder since classes are all done online.

9. If learners have any questions and queries regarding the course, lessons or any activities, do not hesitate to contact and ask your Course Instructor. Always remember that we are here not only to teach you but also to help you, especially in this time of Pandemic.

FOREWORD

Integrative Programming and Technologies 1 is a three (3) unit course prefer for the second-year students of the Bachelor of Science in Information Technology and the Bachelor of Science in Information Systems from the College of Computer Studies and Technology in the Pamantasan ng Lungsod ng San Pablo.

The course introduces students to a vast range vast range of programming languages with different features and syntax. Developers can use any programming language according to their own requirements. These programming languages just didn't come out, it took years to reach this advanced level. In this module, we will learn about the history of programming languages and how our ancestors discovered and invented ways of programming that we can call paradigms.

It is a great challenge for each one of us as we battle against the Corona Virus Disease 2019 (Covid19) Pandemic, specifically in the Education sector here in the Philippines where we are still in the process of coping with what we called as the "New Normal". Year 2020 may gave us a lot of challenges but optimistically it gave us some new opportunities to continue with our life even in the midst of this pandemic, it may be different and not the way we used to be, but soon we will gradually learn to embrace these changes. Likewise, on the Education as it set a new educational platform, such as the Online Learning or the Distant Learning, where it centers the use of the Internet Technology among the Students and the Teachers.

MODULES FOR COMPUTER PROGRAMMING 2

Credits : **3 Units** (3 Hours Laboratory, 2 Hours Lecture)

Pre-Requisite : **CC123, IM211**

Lesson Objective:

At the end of the module 1, the learners will be able to:

1. To understand the context and evolution of programming languages;
2. to gain a deeper understanding of the different approaches to programming;
3. categorize a programming language based on its programming paradigm.

References:

Lestal, J. (2021, August 26). *History of Programming Languages.* DevSkiller https://devskiller.com/history-of-programming-languages/

MV, Thanoshan. (2019, November 12). *What exactly is a programming paradigm*. freeCodeCamp https://www.freecodecamp.org/news/what-exactly-is-a-programming-paradigm/

Cocca, G. (2022, May 2). *Programming Paradigms.* freeCodeCamp https://www.freecodecamp.org/news/an-introduction-to-programming-paradigms/#:~:text=Imperative%2C%20procedural%2C%20functional%2C%20declarative,topics%20of%20the%20coding%20world.

Lectures and Annotations:

History of Programming Languages

Created by computer programmers, Computer Programming is a set of instructions for a computer to facilitate specific actions like writing and testing codes that enables programs to operate successfully. It is the very basis of the technological age that we live in today.

Did you know that the first programming language was invented way back in 1843 by a woman? She came up with the first-ever machine algorithm for an early computing machine and wrote it down on a piece of paper because no computers existed at the time.

Programming languages have come a long way since then; listed below is a timeline of the history of programming languages summarized by Lestal, 2021:

- 1843: Ada Lovelace's machine algorithm

  Ada Lovelace invents the first-ever machine algorithm for Charles Babbage's Difference Machine that lays the foundation for all programming languages.

Source: https://upload.wikimedia.org/wikipedia/commons/c/c0/Ada_Lovelace_Chalon_portrait.jpg

- 1944-45: Plankalkül

  Somewhere between 1944-45, Konrad Zuse developed the first 'real' programming language called Plankalkül (Plan Calculus). Zeus's language (among other things) allowed for the creations of procedures, which stored chunks of code that could be invoked over and over to perform routine operations.

Source: https://commons.wikimedia.org/wiki/File:Konrad_Zuse_und_Heinz_Nixdorf.JPG

- 1949: Assembly Language

  Assembly language was used in the Electronic Delay Storage Automatic Calculator (EDSAC). Assembly language was a type of low-level programming language that simplified the language of machine code. In other words, the specific instructions necessary to operate a computer.

- 1949: Shortcode

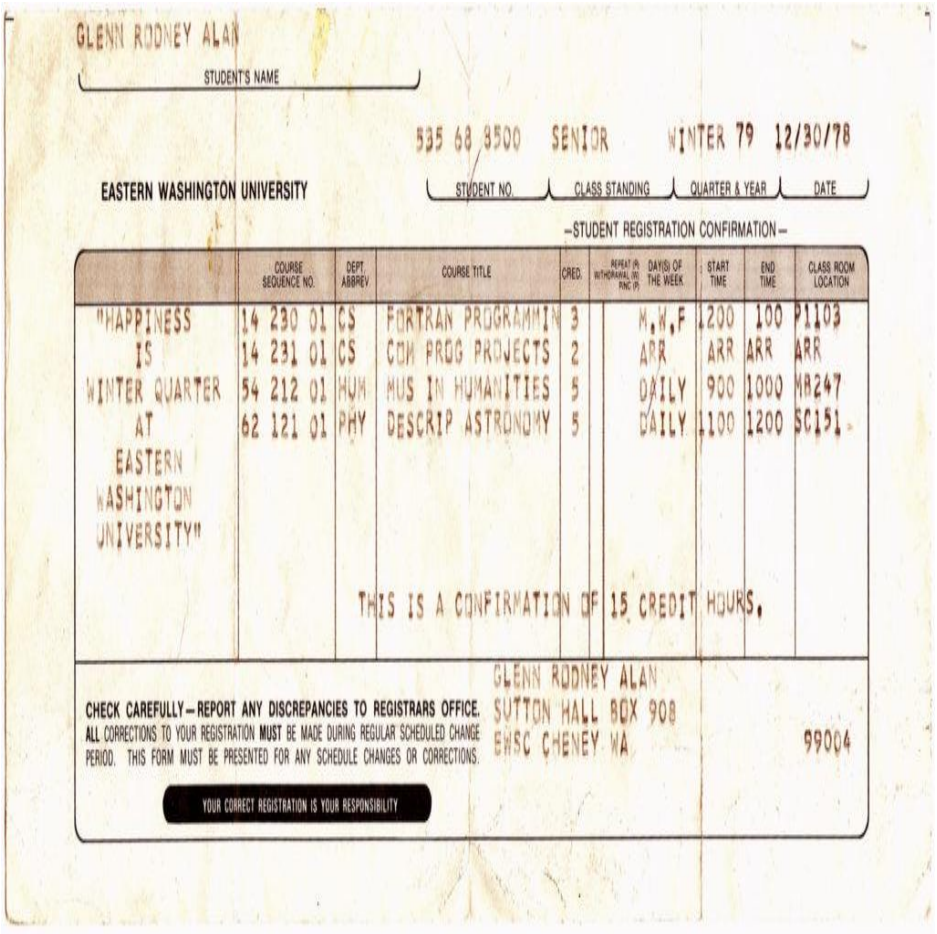
  Shortcode (or Short-order code), was the first High-Level Language (HLL) suggested by John McCauley in 1949. However, it was William Schmitt who implemented it for the BINAC computer the same year and for the UNIVAC in 1950.

- 1952: Autocode

  Autocode was a general term used for a family of programming languages. First developed by Alick Glennie for the Mark 1 computer at the University of Manchester, Autocode was the first-ever compiled language to be implemented meaning that it can be translated directly into machine code using a program called a compiler. Autocode was used on the Ferranti Pegasus and Sirius early computing machines in addition to the Mark 1.

- 1957: FORTRAN

  FORmula TRANslation or FORTRAN was created by John Backus and is considered to be the oldest programming language in use today. The programming language was created for high-level scientific, mathematical, and statistical computations. FORTRAN is still in use today in some of the world's most advanced supercomputers.



GLENN RODNEY ALAN
STUDENT'S NAME

535 68 8500 SENIOR WINTER 79 12/30/78

EASTERN WASHINGTON UNIVERSITY

STUDENT NO. | CLASS STANDING | QUARTER & YEAR | DATE

–STUDENT REGISTRATION CONFIRMATION–

	COURSE SEQUENCE NO.	DEPT. ABBREV	COURSE TITLE	CRED.	REPEAT (R) WITHDRAWAL (W) PNC (P)	DAY(S) OF THE WEEK	START TIME	END TIME	CLASS ROOM LOCATION
"HAPPINESS	14 230 01	CS	FORTRAN PROGRAMMIN	3		M,W,F	1200	100	P1103
IS	14 231 01	CS	COM PROG PROJECTS	2		ARR	ARR	ARR	ARR
WINTER QUARTER	54 212 01	HUM	MUS IN HUMANITIES	5		DAILY	900	1000	MB247
AT	62 121 01	PHY	DESCRIP ASTRONOMY	5		DAILY	1100	1200	SC151
EASTERN									
WASHINGTON									
UNIVERSITY"									

THIS IS A CONFIRMATION OF 15 CREDIT HOURS.

CHECK CAREFULLY – REPORT ANY DISCREPANCIES TO REGISTRARS OFFICE. ALL CORRECTIONS TO YOUR REGISTRATION MUST BE MADE DURING REGULAR SCHEDULED CHANGE PERIOD. THIS FORM MUST BE PRESENTED FOR ANY SCHEDULE CHANGES OR CORRECTIONS.

YOUR CORRECT REGISTRATION IS YOUR RESPONSIBILITY

GLENN RODNEY ALAN
SUTTON HALL BOX 908
EWSC CHENEY WA 99004

Source: https://www.flickr.com/photos/chrstphre/5408224597

- 1958: ALGOL (Algorithmic Language)

  Algorithmic language or ALGOL was created by a joint committee of American and European computer scientists. ALGOL served as the starting point for the development of some of the most important programming languages including Pascal, C, C++, and Java.

- 1958: LISP (List Processor)

  List processor or LISP was invented by John McCarthy at the Massachusetts Institue of Technology (MIT). Originally purposed for artificial intelligence, LISP is one of the oldest programming languages still in use today and can be used in the place of Ruby or Python. Companies such as Acceleration, Boeing, and Genworks are still using LISP in their tech stacks.

Source: https://upload.wikimedia.org/wikipedia/commons/7/78/Lisp-logo.jpg

- 1959: COBOL (Common Business Oriented Language)

  Common Business Oriented Language (COBOL), is the programming language behind many credit card processors, ATMs, telephone and cell calls, hospital signals, and traffic signals systems (just to name a few). The development of the language was led by Dr. Grace Murray Hopper and was designed so that it could run on all brands and types of computers. COBOL is still used to this day primarily for banking and gamification systems.

Source: https://pixabay.com/photos/atm-withdraw-cash-map-ec-card-1524870/

- 1964: BASIC (Beginner's All-Purpose Symbolic Instruction Code)

  Beginners All-Purpose Symbolic Instruction Code or BASIC was developed by a group of students at Dartmouth College. The language was written for students who did not have a strong understanding of mathematics or computers. The language was developed further by Microsoft founders Bill Gates and Paul Allen and became the first marketable product of the company.

- 1970: PASCAL

  Named after the French mathematician Blaise Pascal, Niklaus Wirth developed the programming language in his honor. It was developed as a learning tool for computer programming which meant it was easy to learn. It was favored by Apple in the company's early days, because of its ease of use and power.

Source: https://www.flickr.com/photos/blakespot/14520224626

- 1972: Smalltalk

  Developed at the Xerox Palo Alto Research Centre by Alan Kay, Adele Goldberg, and Dan Ingalls, Smalltalk allowed for computer programmers to modify code on the fly. It introduced a variety of programming language aspects that are visible languages of today such as Python, Java, and Ruby. Companies such as Leafly, Logitech, and CrowdStrike state they use Smalltalk in their tech stacks.

- 1972: C

  Developed by Dennis Ritchie at the Bell Telephone Laboratories for use with the Unix operating system. It was called C because it was based on an earlier language called 'B'. Many of the current leading languages are derivatives of C including; C#, Java, JavaScript, Perl, PHP, and Python. It also has been/still been used by huge companies like Google, Facebook, and Apple.

- 1972: SQL (SEQUEL at the time)

  SQL was first developed by IBM researchers Raymond Boyce and Donald Chamberlain. SEQUEL (as it was referred to at the time), is used for viewing and changing information that is stored in databases. Nowadays the language is an acronym – SQL, which stands for Structured Query Language. There are a plethora of companies that use SQL and some of them include Microsoft and Accenture.

- 1980/81: Ada

  Ada was originally designed by a team led by Jean Ichbiah of CUU Honeywell Bull under contract to the United States Department of Defense. Named after the mid-19th-century mathematician Ada Lovelace, Ada is a structured, statically typed, imperative, wide-spectrum, and object-oriented high-level programming language. Ada was extended from other popular programming languages at the time such as Pascal. Ada is used for air-traffic management systems in countries such as Australia, Belgium, and Germany as well as a host of other transport and space projects.

- 1983: C++

  Bjarne Stroustrup modified the C language at the Bell Labs, C++ is an extension of C with enhancements such as classes, virtual functions, and templates. It has been listed in the top 10 programming languages since 1986 and received Hall of Fame status in 2003. C++ is used in MS Office, Adobe Photoshop, game engines, and other high-performance software.

- 1983: Objective-C

  Developed by Brad Cox and Tom Love, Objective-C is the main programming language used to write software for macOS and iOS, Apple's operating systems.

Source: https://www.cleanpng.com/png-app-store-ios-12-computer-icons-ios-4465743/download-png.html

- 1987: Perl

  Perl was created by Larry Wall and is a general-purpose, high-level programming language. It was originally designed as a scripting language designed for text editing but nowadays it's widely used for many purposes such as CGI, database applications, system administration, network programming, and graphic programming.

```
#!/usr/bin/perl

# import $PATH/$PWD variables as array and scalar
use Env qw(@PATH PWD);

print "PWD: $PWD\n";
print "PATH:\n";

foreach (@PATH) {
  print "$_\n";
}
```

Source: https://www.flickr.com/photos/xmodulo/31127154470

- 1990: Haskell

  Haskell is a general-purpose programming language named after the American logician and mathematician Haskell Brooks Curry. It is a purely functional programming language meaning it's primarily mathematical. It's used across multiple industries particularly those that deal with complicated calculations, records, and number-crunching. Like many other programming languages from this era, it is not overly common to see Haskell in use for well-known applications. With that said, the programming language has been used to write a number of games one of which is Nikki and the Robots.

Source: https://upload.wikimedia.org/wikipedia/commons/0/0f/Haskell_Logo.jpg

- 1991: Python

  Named after the British comedy troupe 'Monty Python', Python was developed by Guido Van Rossum. It is a general-purpose, high-level programming language created to support a variety of programming styles and be fun to use (a number of the tutorials, samples, and instructions often contain Monty Python references). Python is, to this day, one of the most popular programming languages in the world is used by companies such as Google, Yahoo, and Spotify.

Source: https://www.flickr.com/photos/appleboy/8679381967

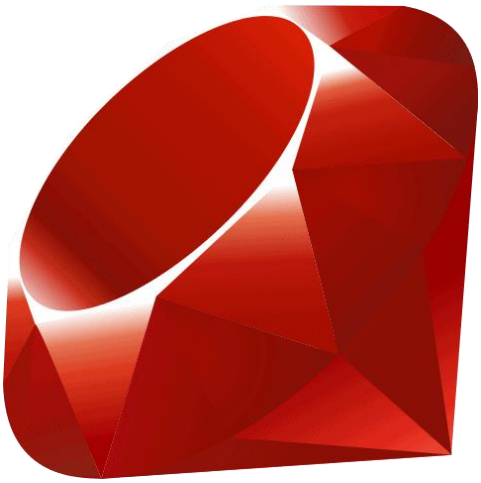
- 1991: Visual Basic

  Developed by Microsoft, Visual Basic allows programmers to utilize a drag-and-drop style of choosing and changing pre-selected chunks of code through a graphical user interface (GUI). The language is not overly used these days however Microsoft has used portions Visual Basic to a number of their applications like Word, Excel, and Access.

Source: https://pixabay.com/pl/illustrations/visual-basic-programowanie-j%C4%99zyk-906838/

- 1993: Ruby

  Created by Yukihiro Matsumoto, Ruby is an interpreted high-level programming language. A teaching language which was influenced by Perl, Ada, Lisp, and Smalltalk – among others. Ruby's primary uses are for web applications development and Ruby on Rails. Twitter, Hulu, and Groupon are some well-known examples of companies that use Ruby.



Source: https://upload.wikimedia.org/wikipedia/commons/thumb/7/73/Ruby_logo.svg/600px-Ruby_logo.svg.png

- 1995: Java

  Java is a general-purpose, high-level language created by James Gosling for an interactive TV project. It has cross-platform functionality and is consistently among the top of the world's most popular programming languages. Java can be found everywhere, from computers to smartphones to parking meters.

Source: https://upload.wikimedia.org/wikipedia/commons/thumb/e/e9/Java-Debugging-Tips-881x441.jpg/800px-Java-Debugging-Tips-881x441.jpg

- 1995: PHP

  Formerly known as 'Personal Home Page' which now stands for 'Hypertext Preprocessor', PHP was developed by Rasmus Lerdorf. Its primary uses include building and maintaining dynamic web pages, as well as server-side development. Some of the biggest companies from across the globe use PHP including Facebook, Wikipedia, Digg, WordPress, and Joomla.

- 1995: JavaScript

  JavaScript was created by Brendan Eich, this language is primarily used for dynamic web development, PDF documents, web browsers, and desktop widgets. Almost every major website uses JavaScript. Gmail, Adobe Photoshop, and Mozilla Firefox include some well-known examples.

- 2000: C#

  Developed at Microsoft with the hope of combining the computing ability of C++ with the simplicity of Visual Basic, C# is based on C++ and shares many similarities with Java. The language is used in almost all Microsoft products and is seen primarily in developing desktop applications.

- 2003: Scala

  Developed by Martin Odersky, Scala which combines mathematical functional programming and organized object-oriented programming. Scala's compatibility with Java makes it helpful with Android development. Linkedin, Twitter, Foursquare, and Netflix are just a few examples of the many companies that use Scala in their tech stacks.

- 2003: Groovy

  Derived from Java, Groovy was developed by James Strachan and Bob McWhirter. The language improves productivity because of its succinct and easy to learn. Some well-known companies that are using Groovy in their tech stacks are Starbucks, Transferwise, and Craftbase.

- 2009: Go

  Go was developed by Google to address issues that occur due to large software systems. Due to its simple and modern structure, Go has gained popularity among some of the largest tech companies around the world such as Google, Uber, Twitch, and Dropbox.

- 2014: Swift

  Developed by Apple as a replacement for C, C++, and Objective-C, Swift was developed with the intention to be easier than the aforementioned languages and allow less room for error. Swift's versatility means it can be used for desktop, mobile, and cloud applications. Leading language app 'Duolingo' launched a new app that was written in Swift.

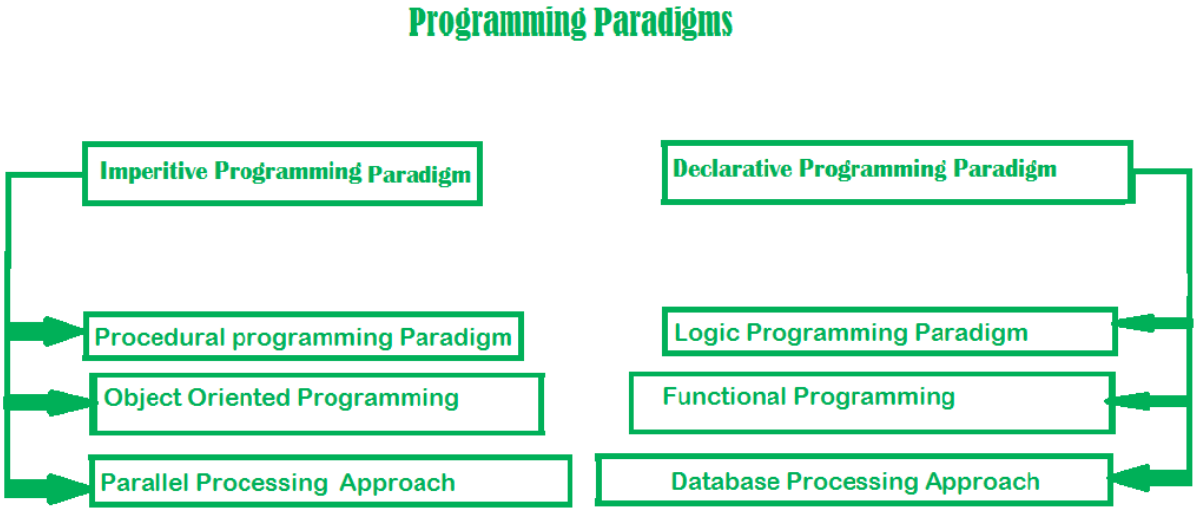
Programming Paradigms

Introduction to Programming Paradigms

The term programming paradigm refers to a style of programming. It does not refer to a specific language, but rather it refers to the way you program.

There are lots of programming languages that are well-known but all of them need to follow some strategy when they are implemented. And that strategy is a paradigm.

The Types of Programming Paradigms



Source: geeksforgeeks.org

1. Imperative Programming Paradigm

Imperative programming consists of sets of detailed instructions that are given to the computer to execute in a given order. It's called "imperative" because as programmers we dictate exactly what the computer has to do, in a very specific way.

Imperative programming focuses on describing how a program operates, step by step.

Say you want to bake a cake. Your imperative program to do this might look like this;

```
1- Pour flour in a bowl
2- Pour a couple eggs in the same bowl
3- Pour some milk in the same bowl
4- Mix the ingredients
5- Pour the mix in a mold
6- Cook for 35 minutes
7- Let chill
```

Using an actual code example, let's say we want to filter an array of numbers to only keep the elements bigger than 5. Our imperative code might look like this:

```
const nums = [1,4,3,6,7,8,9,2]
const result = []

for (let i = 0; i < nums.length; i++) {
    if (nums[i] > 5) result.push(nums[i])
}

console.log(result) // Output: [ 6, 7, 8, 9 ]
```

See that we're telling the program to iterate through each element in the array, compare the item value with 5, and if the item is bigger than 5, push it into an array.

1.1. Procedural Programming Paradigm

Procedural programming is a derivation of imperative programming, adding to it the feature of functions (also known as "procedures" or "subroutines").

In procedural programming, the user is encouraged to subdivide the program execution into functions, as a way of improving modularity and organization.

Following our cake example, procedural programming may look like this:

```
function pourIngredients() {
    - Pour flour in a bowl
    - Pour a couple eggs in the same bowl
    - Pour some milk in the same bowl
}

function mixAndTransferToMold() {
    - Mix the ingredients
    - Pour the mix in a mold
}

function cookAndLetChill() {
    - Cook for 35 minutes
    - Let chill
}

pourIngredients()
mixAndTransferToMold()
cookAndLetChill()
```

You can see that, thanks to the implementation of functions, we could just read the three function calls at the end of the file and get a good idea of what our program does.

Languages that support the procedural programming paradigm are:

- C, C++, Java, Pascal

Why should you consider learning the procedural programming paradigm?

- It's simple.
- An easier way to keep track of program flow.
- It has the ability to be strongly modular or structured.
- Needs less memory: It's efficient and effective.

1.2. Object-Oriented Programming

One of the most popular programming paradigms is object-oriented programming (OOP).

The core concept of OOP is to separate concerns into entities which are coded as objects. Each entity will group a given set of information (properties) and actions (methods) that can be performed by the entity.

OOP makes heavy usage of classes (which are a way of creating new objects starting out from a blueprint or boilerplate that the programmer sets). Objects that are created from a class are called instances.

Following our pseudo-code cooking example, now let's say in our bakery we have a main cook (called Frank) and an assistant cook (called Anthony) and each of them will have certain responsibilities in the baking process. If we used OOP, our program might look like this.

```
// Create the two classes corresponding to each entity
class Cook {
        constructor constructor (name) {
        this.name = name
    }

    mixAndBake() {
        - Mix the ingredients
        - Pour the mix in a mold
        - Cook for 35 minutes
    }
}

class AssistantCook {
    constructor (name) {
        this.name = name
    }

    pourIngredients() {
        - Pour flour in a bowl
        - Pour a couple eggs in the same bowl
        - Pour some milk in the same bowl
    }

    chillTheCake() {
        - Let chill
    }
}

// Instantiate an object from each class
const Frank = new Cook('Frank')
const Anthony = new AssistantCook('Anthony')

// Call the corresponding methods from each instance
Anthony.pourIngredients()
Frank.mixAndBake()
Anthony.chillTheCake()
```

Languages that support the object-oriented paradigm:

• Python, Ruby, Java, C++, Smalltalk

Why should you consider learning the object-oriented programming paradigm?

• Reuse of code through Inheritance
• Flexibility through Polymorphism
• Improved software development productivity
• Lower cost of development

1.3. Parallel Processing Approach

Parallel processing is the processing of program instructions by dividing them among multiple processors.

A parallel processing system allows many processors to run a program in less time by dividing them up.

Languages that support the Parallel processing approach:

• NESL (one of the oldest ones), C, C++

Why should you consider learning the parallel processing approach?

• Speeds up performance.
• Often used in Artificial Intelligence
• It makes it easy to solve problems since this approach seems to be like a divide and conquer method.

2. Declarative Programming Paradigm

Declarative programming is all about hiding away complexity and bringing programming languages closer to human language and thinking. It's the direct opposite of imperative programming in the sense that the programmer doesn't give instructions about how the computer should execute the task, but rather on what result is needed.

This will be much clearer with an example. Following the same array filtering story, a declarative approach might be:

```
const nums = [1,4,3,6,7,8,9,2]

console.log(nums.filter(num => num > 5)) // Output: [ 6, 7, 8, 9 ]
```

See that with the filter function, we're not explicitly telling the computer to iterate over the array or store the values in a separate array. We just say what we want ("filter") and the condition to be met ("num > 5").

2.1. Logic Programming Paradigm

The logic programming paradigm takes a declarative approach to problem-solving. It's based on formal logic.

The logic programming paradigm isn't made up of instructions - rather it's made up of facts and clauses. It uses everything it knows and tries to come up with the world where all of those facts and clauses are true.

Languages that support the logic programming paradigm:

- Prolog, Absys, ALF, Alice, Ciao

Why should you consider learning the logic programming paradigm?

- Easy to implement the code.
- Debugging is easy.
- Since it's structured using true/false statements, we can develop the programs quickly using logic programming.
- As it's based on thinking, expression and implementation, it can be applied in non-computational programs too.
- It supports special forms of knowledge such as meta-level or higher-order knowledge as it can be altered.

2.2. Functional Programming

Functional programming takes the concept of functions a little bit further.

In functional programming, functions are treated as first-class citizens, meaning that they can be assigned to variables, passed as arguments, and returned from other functions.

Another key concept is the idea of pure functions. A pure function is one that relies only on its inputs to generate its result. And given the same input, it will always produce the same result. Besides, it produces no side effects (any change outside the function's environment).

With these concepts in mind, functional programming encourages programs written mostly with functions (surprise 😮). It also defends the idea that code modularity and the absence of side effects makes it easier to identify and separate responsibilities within the codebase. This therefore improves the code maintainability.

Going back to the array filtering example, we can see that with the imperative paradigm we might use an external variable to store the function's result, which can be considered a side effect.

```
const nums = [1,4,3,6,7,8,9,2]
const result = [] // External variable

for (let i = 0; i < nums.length; i++) {
    if (nums[i] > 5) result.push(nums[i])
}

console.log(result) // Output: [ 6, 7, 8, 9 ]
```

To transform this into functional programming, we could do it like this:

```
const nums = [1,4,3,6,7,8,9,2]

function filterNums() {
    const result = [] // Internal variable

    for (let i = 0; i < nums.length; i++) {
        if (nums[i] > 5) result.push(nums[i])
    }

    return result
}

console.log(filterNums()) // Output: [ 6, 7, 8, 9 ]
```

It's almost the same code, but we wrap our iteration within a function, in which we also store the result array. In this way, we can assure the function doesn't modify anything outside its scope. It only creates a variable to process its own information, and once the execution is finished, the variable is gone too.

Languages that support functional programming paradigm:

- Haskell, Scala, JavaScript

Why should you consider learning the functional programming paradigm?

- Functions can be coded quickly and easily.
- General-purpose functions can be reusable which leads to rapid software development.
- Unit testing is easier.
- Debugging is easier.
- Overall application is less complex since functions are pretty straightforward.

2.3. Database Processing Approach

This programming methodology is based on data and its movement. Program statements are defined by data rather than hard-coding a series of steps.

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS)

To process the data and querying them, databases use tables. Data can then be easily accessed, managed, modified, updated, controlled and organized.

A good database processing approach is crucial to any company or organization. This is because the database stores all the pertinent details about the company such as employee records, transaction records and salary details.

Most databases use Structured Query Language (SQL) for writing and querying data.

Here's an example in database processing approach (SQL):

```sql
CREATE DATABASE personalDetails;
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

The PersonID column is of type int and will hold an integer. The LastName, FirstName, Address, and City columns are of type varchar and will hold characters, and the maximum length for these fields is 255 characters.

Database processing approach is often best used when:

- Working with databases to structure them.
- Accessing, modifying, updating data on the database.
- Communicating with servers.

Why are databases important and why should you consider learning database processing approach?

- Massive amount of data is handled by the database
- Accurate
- Easy to update data
- Data integrity