2021-22 | ISSUE 1

ASCII NEWSLETTER

ASSOCIATION OF STUDENTS OF COMPUTER SCIENCE FOR INFORMATION INTERCHANGE

WHAT'S INSIDE THIS ISSUE

WELCOME!

Language has been and still is the primary means of communication and human interaction. And when you look at a computer, you'll find it's not so much different. There are many pieces of hardware and software that need to communicate with each other. But a computer is a machine which understands nothing but 1s and 0s, the combination of which creates meaning. That's why the creation of programming languages was a revolutionary step that took this field to another level. And in this issue of ASCII, we go further to extend our understanding on Programming Languages.

Department of Computer Science and Engineering

Vision:

To be acclaimed internationally for excellence in teaching and research in Computer Science & Engineering, and in fostering a culture of creativity and innovation to responsibly harness state-of-the-art technologies for societal needs.

Mission:

Mission 1: To assist students in developing a strong foundation in Computer Science and Engineering by providing analytical, computational thinking and problem solving skills.

Mission 2: To inculcate entrepreneurial skills to develop solutions and products for interdisciplinary problems by cultivating curiosity, team spirit and spirit of innovation.

Mission 3: To provide opportunities for students to acquire knowledge of state-of-the-art in Computer Science and Engineering through industry internships, collaborative projects, and global exchange programmes with Institutions of international repute.

Mission 4: To develop life-long learning, ethics, moral values and spirit of service so as to contribute to the society through technology.

Mission 5: To be a premier research-intensive department by providing a stimulating environment for knowledge discovery and creation.

Programme Educational Objectives (PEOs)

The Computer Science & Engineering Program graduates will

PEO1: Strive on a global platform to pursue their professional career in Computer Science and Engineering.

PEO2: Contribute to product development as entrepreneurs in inter disciplinary fields of engineering and technology.

PEO3: Demonstrate high regard for professionalism,integrity and respect values in diverse culture, and have a concern for society and environment.

Programme Outcomes (PO's) and Programme Specific Outcomes (PSO's)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design and development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to Assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

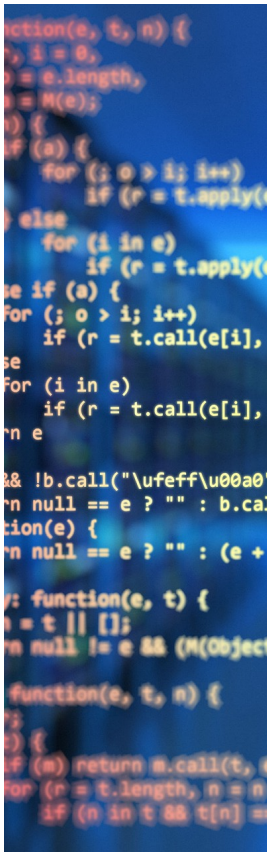
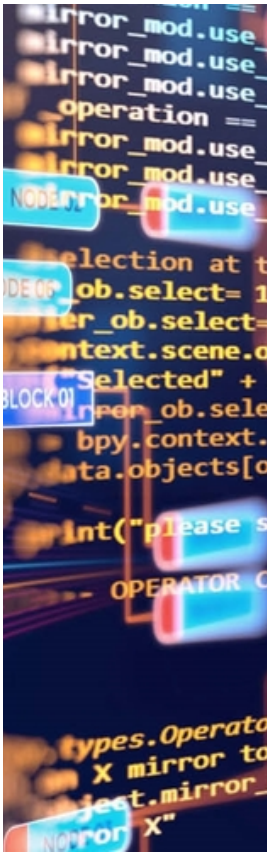
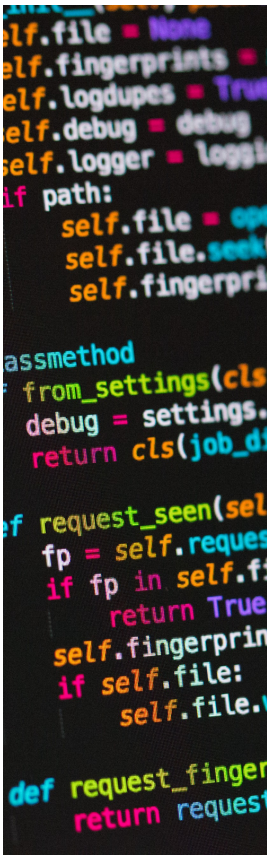
PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PSO1: Adopt Standard Practices: Ability to design and engineer, innovative, optimal and elegant computing solutions to interdisciplinary problems using standard practices, tools and technologies.

PSO2: Research and Innovation: Ability to learn emerging computing paradigms for research and innovation

CONTENTS


WHAT IS A PROGRAMMING LANGUAGE?

Compiled by Mihika Shrivastava

Computers today, are fundamentally the same machines that they were about 30 years ago. They are still electronic devices (on a majority basis, though some of them are non-electronic), primarily composed of transistors. Transistors are simply an on/off switch. So when hundreds/thousands of these on/off switches are combined, we get a computer.

Now, a computer uses Binary code (that's the 0s and 1s) to instruct these switches on whether they should turn on (1) or turn off (0). Each transistor receives a 0 or 1, and with thousands of them working at once, computing can be done.

But unfortunately, attempting to make an entire computer work by manually typing a number for each transistor would take an incredible amount of time, considering that even the smallest computer as of 2018 has in the order of 100,000 transistors, and one of the highest transistor-count machines out there today have about 400 trillion of them.

Now even as a novice, if one pays any attention to the technology world, they've undoubtedly heard the terms ‘coding’ and ‘programming’, dozens of times. This is where they come in.

Instead of having to manually type, to control each switch, we (humans; programmers) have developed high-level programming languages to help speed up the process. Instead of addressing each transistor individually, we address entire sections of them to perform a specific task.

So a programming language is essentially the vocabulary and the grammar for getting a computer or any other computing device for that matter, to perform a specific set of tasks. They are way more than just colourful text. There are about 700 different programming languages out there, some are still used, some are not.

Each language is different from the others in one way or another. They are suited for different purposes across different industries. While some may be used to solve problems or interpret data, others may be used to create apps or video games or do animation.

The possibilities of possible applications are many and so are the choices for the programming languages used to implement these.

References:

- careerkarma.com

DIFFERENT PARADIGMS OF PROGRAMMING LANGUAGES.

Compiled by B. Balanisha

Programming Paradigms!
Sounds very strange, isn't it!? The term refers to the style of programming. It is unspecific to any particular programming language, rather it tells us the way to program. There are quite a lot of programming languages that are well known, but all of them are supposed to follow some strategy. Apart from the varieties of existing programming languages, there are also lots of paradigms that are responsible to fulfil each and every demand.

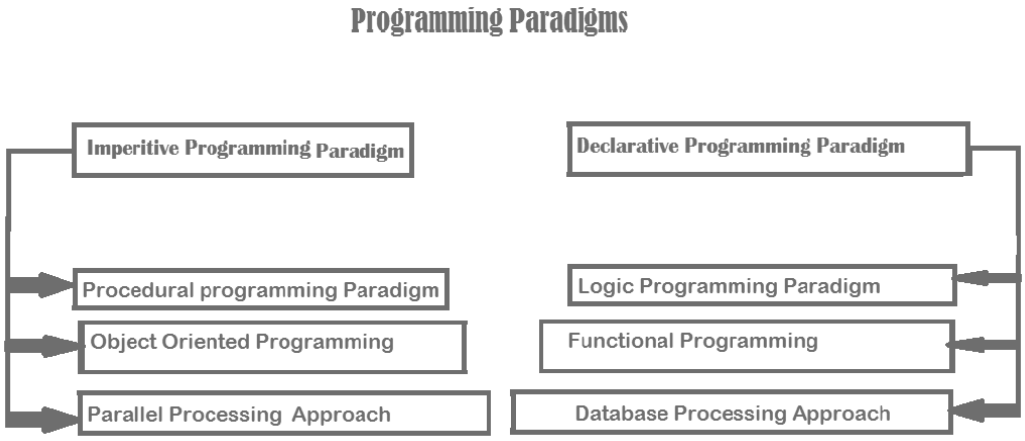
Programming paradigms are majorly classified into **Imperative** Programming paradigm and **Declarative** Programming paradigm. Imperative, being the oldest programming paradigm, works on the basis of changing the program state by using assignment statements. It follows a step-by-step procedure and its main goal is to achieve its goal. This paradigm usually contains several statements, and the result is stored after the execution of all the statements.

The declarative Programming paradigm refers to the style of building programs that express the logic of a computation without minding its control flow. The focus of this type is to know what has to be done rather than how.

Imperative Programming Paradigm:
Imperative Programming Paradigm can be broadly divided into three categories: Procedural, Object-Oriented and Parallel Processing.

1. **Procedural Programming Paradigm:**
This type of paradigm contains a series of computational steps to be carried out. Any procedure in this can be called at any point of the program's execution. The code can be reused whenever required.

2. **Object-Oriented Programming:**
Object-Oriented programming uses objects rather than functions and logic to organise and manipulate data. More emphasis is laid on data and this helps to handle many kinds of real-life problems in today's scenario.



3. **Parallel Processing Approach:**
This paradigm refers to the processing of programs by diving them into an indefinite amount of smaller programs and distributing them to multiple processors in order to reduce the run time.

Declarative Programming Paradigm:
The declarative programming paradigm can be broadly classified into three categories namely logic, functional and database.

1. **Logic Programming Paradigm:**
The programs in this paradigm are rather written in form of facts and rules to implement the logic. The execution of these programs would be as same as a mathematical proof.

2. **Functional Programming Paradigm:**
This paradigm mainly deals with mathematical functions and it is language independent. Execution of mathematical functions is the main principle.

3. **Database / Data-driven Programming Paradigm:**
This paradigm mainly deals with the storage and manipulation of data. A good approach towards this paradigm is very crucial to business organisations.

References:
- geeksforgeeks.org/introduction-of-programming-paradigms/
- cs.lmu.edu/~ray/notes/paradigms/
- freecodecamp.org/news/what-exactly-is-a-programming-paradigm/

THE FUNCTIONAL PROGRAMMING PARADIGM.

Compiled by A. Shivaani

Functional programming is a programming paradigm that attempts to bind each and everything in pure mathematical functions. It is a declarative type of programming style that aims at what to solve rather than how to solve. Clojure, Common Lisp, Erlang, Haskell, and Scala are some of the notable programming languages that follow the functional programming approach. The programming paradigm is based on Lambda Calculus.

LAMBDA CALCULUS

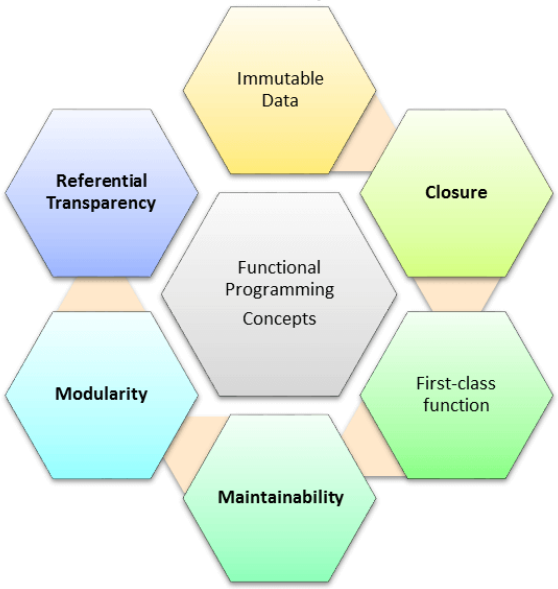
Lambda Calculus was developed by Alonzo Church in the 1930s as a part of his research into the foundation of mathematics. It is a framework for studying computations with functions. The lambda calculus makes use of expressions instead of statements. Unlike a statement executed to assign variables, the evaluation of an expression produces a value. Lambda calculus forms the basis of almost all of the functional programming languages.

Anything that can be solved using lambda calculus is computable. In terms of its computational ability, lambda calculus is similar to the Turing machine that laid the foundation for the imperative style of programming.

FUNCTIONAL PROGRAMMING CONCEPTS:

Pure Functions

- Are Immutable, wherein one always produces the same output with the same arguments disregarding other factors.
- Functions are deterministic. Pure functions give some output and do modify any argument or global variables i.e. they have no side-effects
- Since pure functions have no side effects or hidden I/O, programs built using functional paradigms are easy to debug. Moreover, pure functions make writing concurrent applications easier.
- When the code is written using the functional programming style, a capable compiler is able to:
  a) Memorize the results
  b) Parallelize the instructions
  c) Wait for evaluating results



Recursion

- In the functional programming paradigm, there are no loops. Instead, functional programming languages rely on recursion for iteration.
- Recursion is implemented using recursive functions, which repetitively call themselves until the base case is reached.

Referential Transparency

- Variable values once defined in a functional programming language shouldn't be changed throughout the execution of the program. It assures that the same language expression gives the same output.
- Functional programs don't have any assignment statements. For storing additional values in a program developed using functional programming, new variables must be defined, whose state is constant throughout the program.
- Referential transparency eliminates even the slightest chances of any undesired effects due to the fact that any variable can be replaced with its actual value during any point in the program execution.

First-Class and can be Higher-Order

- Functions in the functional programming style are treated as variables. Hence, they are first-class functions. These first-class functions are allowed to be passed to other functions as parameters or returned from functions or stored in data structures.
- Variables are immutable i.e. it isn't possible to modify a variable once it has been initialized. Though we can create a new variable, modifying existing variables is not allowed..·
- The immutable nature of variables in a functional programming language benefits in the form of preserving the state throughout the execution of a program.

ADVANTAGES

- Pure functions don't change any states and are entirely dependent on the input.
- The return value given by such functions is the same as the output they give.
- Due to the nature of pure functions to avoid changing variables or any data outside it, implementing concurrency becomes efficacious.
- Pure functions take arguments once and produce unchangeable output. Hence, they don't produce any hidden output. They use immutable values, making debugging and testing easier.

DISADVANTAGES

- Immutable values combined with recursion might lead to a reduction in performance.
- Though writing pure functions is easy, combining the same with the rest of the application as well as the I/O operations is tough.
- Writing programs in recursive style in place of using loops for the same can be a daunting task.

References:

- geeksforgeeks.com
- guru99.com

DOMAIN SPECIFIC LANGUAGES

Compiled by Likhitha S. & Mihika Shrivastava

DSL's are languages created to perform assigned tasks like solving programs, all within a specific domain.

Let's try to understand with an example:
Suppose you wish to open a capped bottle. Now, there are several ways how you could go about doing that. Maybe you could use your teeth, or maybe you could pry the cap off with a fork or a spoon. But these ways are hard and may not always work. But using the actual tool- a bottle opener is bound to work.

Similarly, when it comes to programming languages- Java, Visual Basic, C/C++ are all general programming languages (GPL), which can be used for a variety of purposes. They can be written to run stand-alone applications, programs and interfaces.

But there are situations where general programming language just won't work. Hence, we have DSLs. A DSL focuses on one task or is built to work on one platform.

Here are some examples for DSL's:

- SQL for databases.
- HTML for web layouts.
- CSS for web layout styling.
- DOT is for defining graphs.

When it comes to differentiation, the line between domain-specific languages and scripting languages is somewhat blurred. However, domain-specific languages often lack low-level functions for filesystem access, interprocess control, and other functions that characterize full-featured programming languages, scripting or otherwise. Many domain-specific languages do not compile to byte-code or executable code, but to various kinds of media objects. For example, GraphViz exports to PostScript, GIF, JPEG, etc.

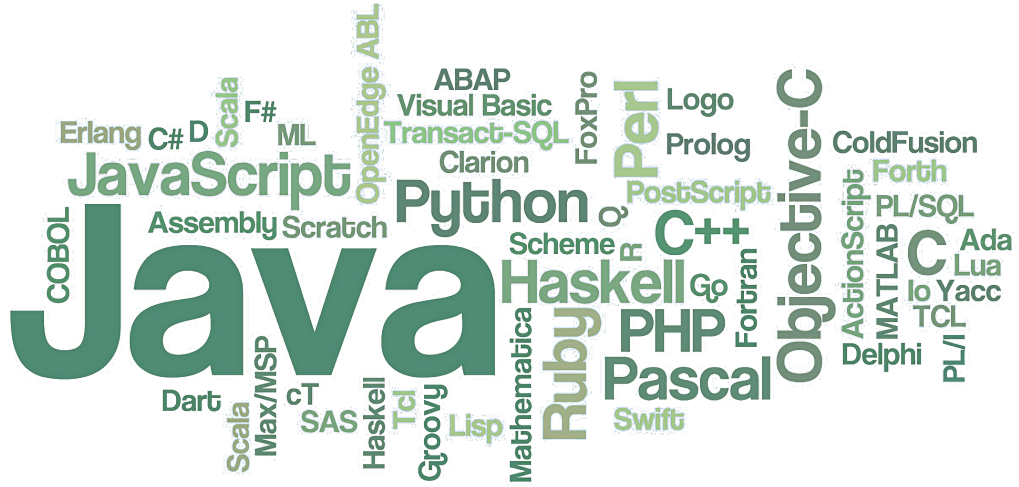
So all this does lead to the question. Why should one choose a DSL over a GPL?
Well, though a GPL is quite versatile in its uses, a DSL can be analyzed much better. DSLs often tend to be safer and whenever there are errors, those errors tend to be specific to the domain, so they are easier to understand.
This also means that interpreting DSLs is easier, so bringing them to a new platform would be quite easy.
But most importantly, DSLs only focus on the important concepts by enabling abstraction, hence we're left only with the information that truly matters.

References:

- tomassetti.me/domain-specific-languages/

CURRENTLY POPULAR PROGRAMMING LANGUAGES.

Compiled by Mihika Shrivastava



It is quite difficult to give a definite answer to the question 'Which is the most popular programming language out there today?' because popularity depends upon the usage, and the usage depends upon the context.

A language may occupy the greater number of programmer hours, or a different one may have more lines of code, or it may utilize the most CPU time, and so on. Further, there are some languages very popular for only particular kinds of applications, like Fortran in computational science and engineering and C in embedded applications and operating systems.

So there are various methods that have been proposed to measure the popularity of a language, each subject to a different bias. A few methods include:

- Counting the number of times the language name is mentioned in web searches and YouTube searches.
- Counting the number of job advertisements that mention the language.
- The number of books sold that teach or describe the language.
- Estimating of the number of existing lines of code written in the language – which may underestimate languages not often found in public searches.

So based on these methods, several indices have been published over the years. Here are a few:

TIOBE PROGRAMMING COMMUNITY INDEX

- Maintained by: TIOBE Company based in Eindhoven, Netherlands.
- The index is updated once a month.
- Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings.

- TIOBE index is not for determining the best programming language but to check whether one's programming skills are still up to date or not.
- The Top 5 Programming Languages as per the TIOBE index (as of April 2020) are:
    1. Java
    2. C
    3. Python
    4. C++
    5. C#

PYPL POPULARITY OF PROGRAMMING LANGUAGE

- Ranking by GitHub.
- It uses Google search activity to rank language popularity. It focuses on people searching for tutorials in the respective languages as a proxy for popularity.
- The Top 5 programming languages currently as per PYPL Popularity are:

1. Python
2. Java
3. Javascript
4. C#
5. PHP

IEEE SPECTRUM

- They rank the popularity by weighting & combining 11 metrics from 8 sources: CareerBuilder, Google, GitHub, Hacker News, the IEEE, Reddit, Stack overflow & Twitter
- The Top 5 programming languages as IEEE Spectrum currently are:

1. Python
2. Java
3. C
4. C++
5. JavaScript

Applications of a few of these languages:

1. **PYTHON**

- Develop different applications like web applications, graphic user interface based applications, software development applications, scientific and numeric applications
- Network programming.
- Games and 3D applications.

2. **C++**

- Allows procedural programming for intensive functions of CPU and provide control over hardware.
- This language is very fast because of which it is widely used in developing different games or in gaming engines.
- C++ is mainly used in developing the suites of a game tool.

3. **Java**

- Used in the distributed environment of the internet.
- It is the most popular programming language for Android applications and is also among the most favoured for the development of edge devices and the internet of things.

4. **JavaScript**

- Web Development & Web Applications
- Presentations & Server Applications
- Games, Art, Smartwatch Applications.

5. **PHP**

- Used to develop Static websites or Dynamic websites or Web applications.

References:

- their respective official websites
- wikipedia.org

UPCOMING PROGRAMMING LANGUAGES.

Contributed by Lavanya A.

If we consider the modern human civilization as a vehicle, then the software development industry is the engine and programming languages are the fuel for that engine.

The term “Modern programming language” is ambiguous. There is a common misconception that languages like Python and JavaScript are modern programming languages, whereas Java is an old one, considering in reality, all of these languages emerged in the same period: the 1990s.

RUST

Graydon Hoare first developed Rust in 2010. Rust is an open-source language whose development is currently being led by Mozilla along with many other companies and communities. It has three primary design goals: Safety, Speed, and Concurrency. Rust also has several distinguishing features: The borrow checker, composition over inheritance and fabulous tooling.

JULIA

Julia is a dynamic, high-level programming language that offers first-class support for Concurrent, Parallel and Distributed Computing.

In 2018 the first stable version of Julia got released and grabbed the attention of industries. Julia and Python compete with each other in many areas.

KOTLIN

Google has declared Kotlin as a first-class language to develop Android and boosted Kotlin’s acceptance in the community. Furthermore, the popular Java Enterprise framework- Spring has started to support Kotlin in the Spring eco-system since 2017, which further boosted its popularity. Kotlin’s characteristics include Conciseness, Versatile Safe, Interoperable and Tool-enabled features.

Kotlin 1.0, was released in 2016. Nearly 5 years after its first initial release in 2011. The latest stable release is the Kotlin 1.5.31, released in September 2021. Kotlin is currently used in over ten products at JetBrains and a variety of other companies like Amex, NBC digital, Expedia, and Gradle.

DART

In the list made by Google, Dart stands in second place. Dart is a general-purpose programming language that supports objected-oriented programming. Dart can also comply with JavaScript.

ELIXIR

Elixir is another new language that has many similarities to the Ruby ecosystem. It focuses on creating high-availability, low-latency systems, unlike the language Rails which had issues on the same front. Elixir runs on Erlang VM which helps it to achieve the above performance boosts. Erlang VM has strong performance in the telecom industry, built over 25 years.

ELM

Elm is a usability-focused functional programming language that compiles high-performance JavaScript. This can be used with or without JavaScript to build a web page. Elm's primary benefits compared to JavaScript are reliability, maintainability, and programmers delight.

References:

- techbeacon.com/app-dev-testing/5-emerging-programming-languages-bright-future
- towardsdatascience.com/top-7-modern-programming-language-to-learn-now-156863bd1eec

THE FUTURE OF PROGRAMMING LANGUAGES.

Contributed by Rohit Sharma T.

We have progressed a lot in the development of programming languages from low-level to high-level and we have achieved program portability, platform independence, multi-threaded application development, database handling, network programming, web-based application development, enterprise application development, etc. We have also developed many programming paradigms like structured programming, object-based and object-oriented programming and many others. Now the question lies what would the future of programming languages look like ?

Next generation programming languages may contain an adaptive behavior and self-repairs features, in which it can adapt itself to any environment or platforms or problem (while loop that automatically changes into for loop when needed, etc.), whereas self-repairs refers to a programming language that learns to improve itself further without the need of human interference (debugging itself and retrieve errors, where repairs can be perform automatically), respectively.

That will open up a whole new level of programming language where developer can spend much more time developing rather than spending a tedious time on debugging.
A thread in computer science is short for a thread of execution. Threads are a way for a program to divide (termed "split") itself into two or more simultaneously (or pseudo-simultaneously) running tasks. Each language have their own way to plan user's threads execution, the future programming language should allow the programmer create kernel's threads.

Future programming languages will be model driven and less machine based. Model-driven attempts to capture knowledge and derive decisions through explicit representation and rules. For example, in a model-driven world, a cat would be explicitly represented as a four-legged animal, with two eyes, a nose and a mouth that is furry (except when not) and that is relatively small (except when not), etc. Support for meta-data(data that provides information about other data), static and dynamic reasoning will become standard.

References:
- techrepublic.com/article/the-future-of-programming-languages-what-to-expect-in-this-new-infrastructure-as-code-world/
- francescolelli.info/programming/the-present-the-past-and-the-future-of-programming-languages-a-historical-perspective/

SOME INSANE PROGRAMMING LANGUAGES.

Contributed by Rohit Sharma T.

LOLCODE

LOLCODE is made up of lolspeak, the ‘language’ used by lolcats. The language was designed by Adam Lindsay in 2007, a researcher at Lancaster University’s Computing Department. The language isn’t as complete as traditional ones, with syntax and operator priorities not clearly defined but there are functioning compliers for that available out there. The hilarity and cuteness of the language more that makes up for this though. Just take a look at the ‘Hello World!’ code. ----->

```
HAI
CAN HAS STDIO?
VISIBLE "Hello World!"
KTHXBYE
```

BEFUNGE

Befunge was developed by Chris Pressey in 1993, with the aim of creating a language that would be as hard to compile as possible. He does this by implementing self-modifying code and having the same instruction being executed in four different ways, not to mention the instruction set itself. However, a number of compilers were eventually created. Just take a look at the ‘Hello World!’ code. ----->

```
>                  v
v  ,,,,,"Hello"48*,
v,,,,,,"World!"25*,@
```

ARNOLDC

Here is a programming language made entirely out of one-liners from movies featuring Arnold Schwarzenegge, classics such as Terminator, Predator and Total Recall. ArnoldC was created by Lauri Hartikka, who swapped out standard commands with their equivalent Arnold one-liner. Example includes False and True, which becomes "I LIED" and "NO PROBLEMO", respectively. Here’s how a "Hello World!" code would look like: ----->

```
IT'S SHOWTIME
TALK TO THE HAND "Hello World!"
YOU HAVE BEEN TERMINATED
```

SHAKESPEARE

This is inspired by the play of Shakespeare, the Hamlet . Just take a look at the ‘Hello World!’ code. ----->

```
The Infamous Hello World Program.

Romeo, a young man with a remarkable patience.
Juliet, a likewise young woman of remarkable grace.
Ophelia, a remarkable woman much in dispute with Hamlet.
Hamlet, the flatterer of Andersen Insulting A/S.


                    Act I: Hamlet's insults and flattery.

                    Scene I: The insulting of Romeo.

[Enter Hamlet and Romeo]

Hamlet:
 You lying stupid fatherless big smelly half-witted coward!
 You are as stupid as the difference between a handsome rich brave
 hero and thyself! Speak your mind!

 You are as brave as the sum of your fat little stuffed misused dusty
 old rotten codpiece and a beautiful fair warm peaceful sunny summer's
 day. You are as healthy as the difference between the sum of the
 sweetest reddest rose and my father and yourself! Speak your mind!

 You are as cowardly as the sum of yourself and the difference
 between a big mighty proud kingdom and a horse. Speak your mind.

 Speak your mind!

[Exit Romeo]
```

Sources:
https://www.hongkiat.com/blog/bizarre-insane-programming-languages/

AUGUST 2021

National Assessment and Accreditation Council (NAAC)

Amrita Vishwa Vidyapeetham was accredited (Cycle-3) with an A++ the highest grade by the National Assessment and Accreditation Council (NAAC).
This put Amrita Vishwa Vidyapeetham under Category 1 Autonomy Higher Education Institution in India.

AMRITA
VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

RE-ACCREDITED WITH THE
HIGHEST **NAAC** GRADE OF

A++

Congratulations & thanks

to Faculty, Staff, Students, Alumni
Parents, Employers
& Partners.

SEPTEMBER 2021

National Institutional Ranking Framework (NIRF)

Amrita Vishwa Vidyapeetham emerged as the fifth best university in the National Institutional Ranking Framework (NIRF) Ranking 2021 for Indian Universities. Amrita has been adjudged as one of the "Top 10 Universities in India" for the fifth consecutive year. In the overall category, Amrita secured 12th place moving up one place from last year's position.

The list of the best universities is prepared based on various parameters such as teaching, learning & resources, research & professional practices, graduation outcomes, outreach & inclusivity, placement records and perception.

NEW FACULTY IN THE DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Mr. Chakravartula Raghavachari is an Assistant Professor in the Department of Computer Science and Engineering at Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore.

He did his Masters in Remote Sensing and Wireless Sensor Networks from the same university. He obtained his B.E in Computer Science and Engineering from RMD Engineering College, Anna University, Chennai.

He is currently pursuing PhD from the Center for Computational Engineering and Networking (CEN), Amrita Vishwa Vidyapeetham, Coimbatore.
He is working in the field of Computer Vision and Robotics.
Before joining Amrita as Assistant Professor, he had been in AMuDa lab at Amrita School of Engineering involved as Junior Research Fellow for a DST funded project.

www.amrita.edu/faculty/chakravartula-raghavachari

OUTSTANDING PROJECT AWARDS

COMPUTER SCIENCE ENGINEERING MTECH 2019-2021

Reg. No.: CB.EN.P2CSE19018
Name: Raghul S
Guide Name: Dr.G.Jeyakumar
Title: Investigations on implementing distributed evolutionary algorithm framework with a fault-tolerance mechanism in cloud infrastructure

Reg. No.: CB.EN.P2CSE19022
Name: Sharmila S
Guide Name: Mr. Sabarish B
Title: A Efficient Fuzzy Trajectory Clustering algorithm (EFTCA) for electing optimum clusters

Reg. No.: CB.EN.P2CSE19007
Name: Chanchal M
Guide Name: Ms. Malathi P
Title: Image data hiding scheme using deep neural network

LANGUAGE MODELLING USING DEEP LEARNING

A TALK CONDUCTED BY CSE DEPARTMENT & THE ALUMNI COORDINATION TEAM

On 25th September 2021, The Department of Computer Science and Engineering, along with the Alumni Coordination Team, hosted a talk in the domain of NLP- **Language Modelling using Deep Learning.**

The Speaker was an alumnus of our university, Ms Anjali Ragupathi (CSE, B.Tech 2017-2021, CB.EN.U4CSE17307).
During her undergraduate studies, she was a research intern at the 'Namaskar with Love Foundation', and a teaching assistant for undergraduate lab courses.
Her areas of interest include Natural Language Processing (NLP) and Computational Linguistics.

The talk focused on the Evolution of Natural Language Processing from basic statistical models to neural language models.
Various deep learning models such as RNN, LSTM, Encoder-Decoder, and Transformers were discussed in great detail. The talk also pointed out the ethical issues with respect to neural language models.

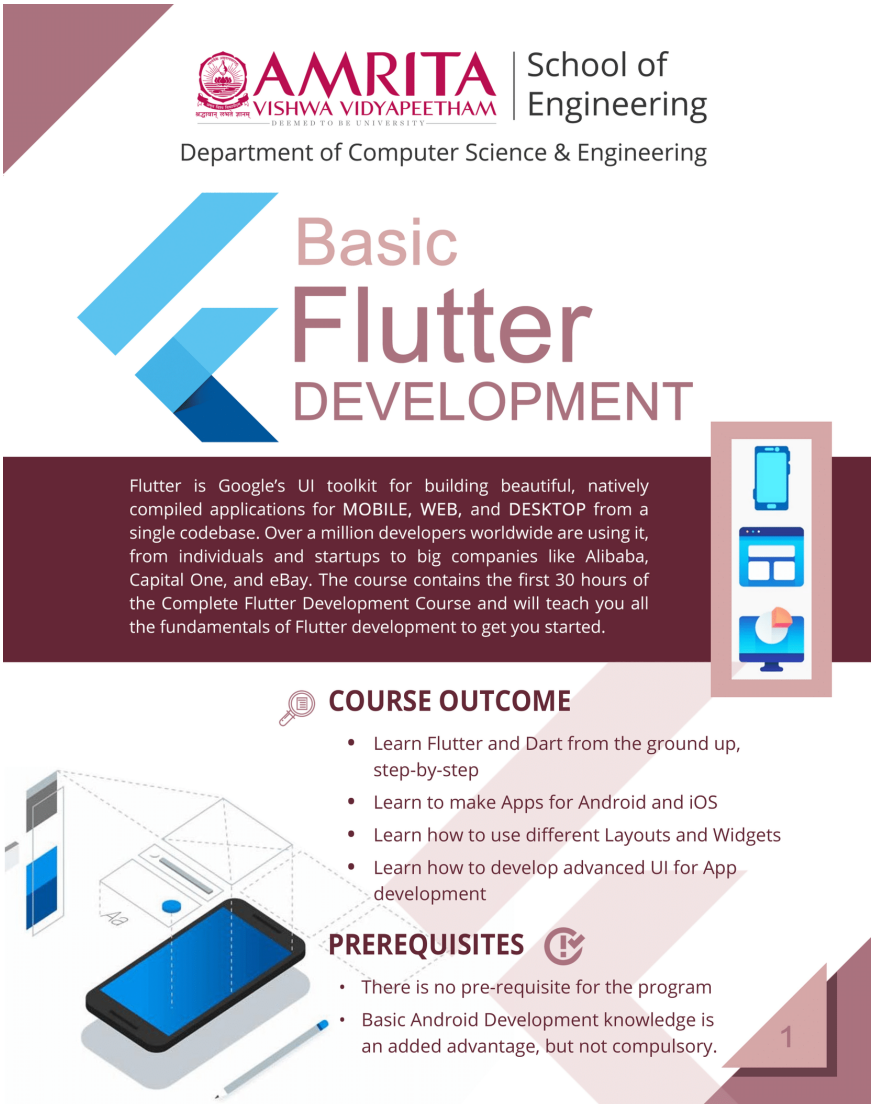
The talk was attended by UG-PG students, research scholars and faculty members of the CSE department.
Mr Aswin K proposed the vote of thanks.

Coordinators:
Dr Manu Madhavan
Ms Abirami K

Links to the talk:
Part-1
Part-2
Presentation link

BASIC FLUTTER DEVELOPMENT

COURSE OVERVIEW



Faculty: Dhanya N.M.
www.amrita.edu/faculty/nm-dhanya

The course started on the 7th of June 2021, right after the Endsem exam of the higher semester students.
The course was open for CSE students of 2nd, 3rd and final years.
Around 170 students registered for the course among which two students were from other departments who registered purely based on their interests.
The course was offered on all weekdays (Monday to Friday) from 6.00 PM to 7.00 PM.

30 lecture hours were conducted which covered all basic concepts and widgets of flutter. The course was conducted in the pattern of regular pop-quizzes based on the topics covered and weekend assignments with the concepts covered and some topics to explore.
From the feedback, we have gathered that the students immensely enjoyed the weekend assignments, which encouraged them and provided them with an opportunity to further explore Flutter's widgets and their properties.

THE WORLD OF DATA SCIENCE

POEM

(Hey there, software world.)
Lead and innovate and ponder more
Compute, analyse, program, score
Pandas, Python or any such brute
Supervised or unsupervised, I'll be there for you.

(Hey there, bankers!)
Modelling, predicting, managing and more
Cost and revenue, allocate both
Segment, recommend or develop for you
Data driving you bonkers? I'll be there for you.

(Hey there, hospitals.)
Process, recognise or interpret your image
Analyse, correlate or infer from your lineage
Predict and interpolate, for a healthier you
IoT or Hadoop, I'll be there for you.

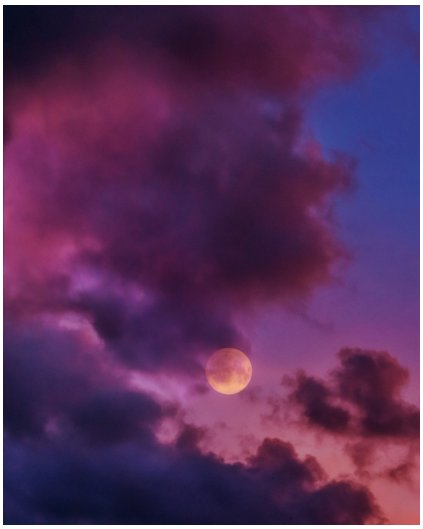
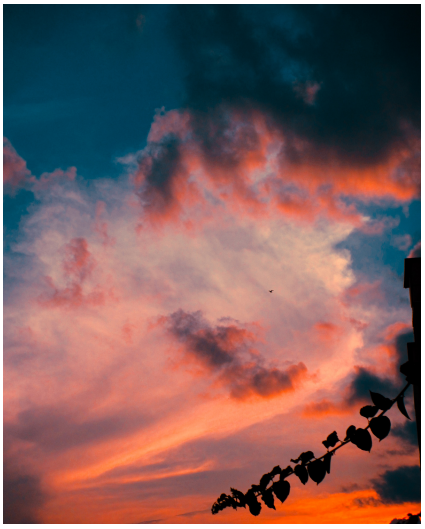
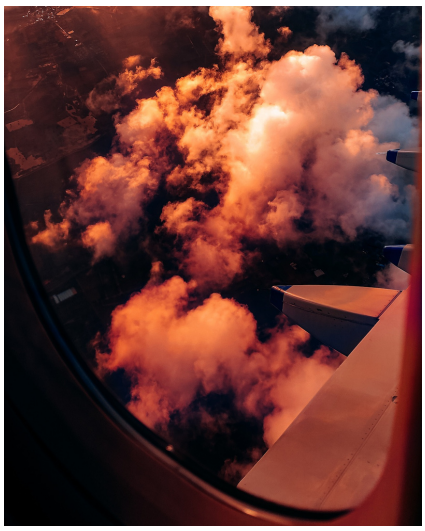
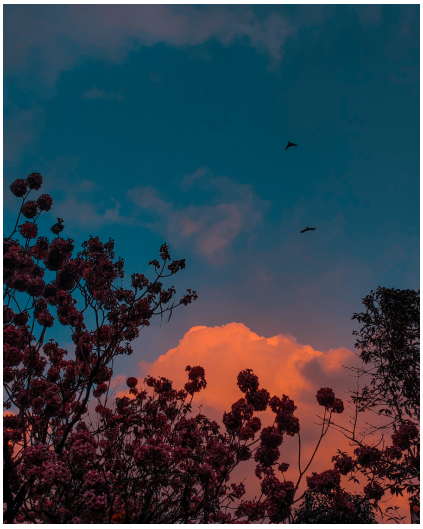
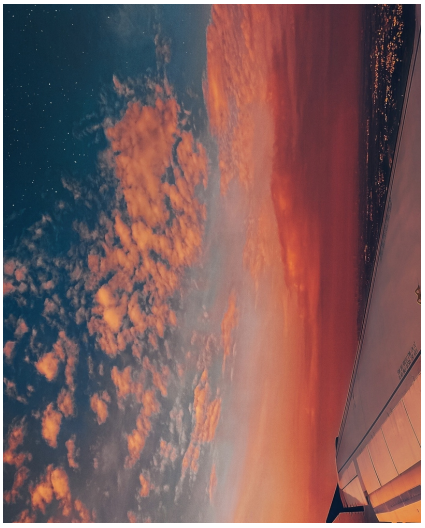
Hey there, automation!
Decision, prediction, processing and more
Analyse, Visualise and plenty such in store
PCA, K-Mean, Cluster, Neural, NLP
As a Data Scientist, I'll always be there for you!

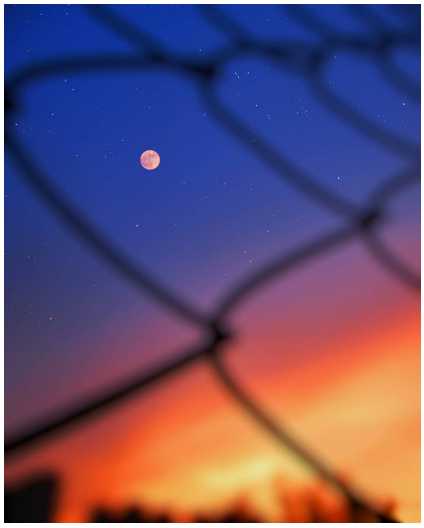
Poem by:
Pragya Ananth
(CB.SC.I5DAS18027)
Fourth year, Integrated M.Sc. Data Science

PHOTOGRAPHY

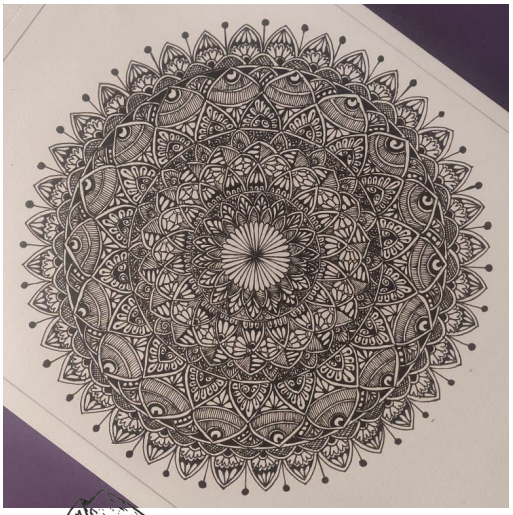
by Mihika Shrivastava



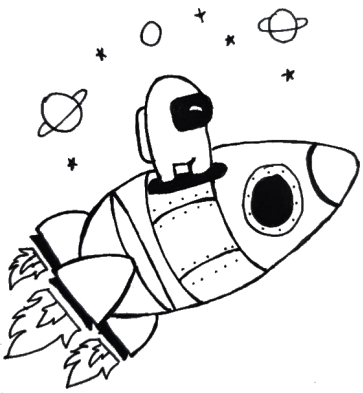


@_mihikas

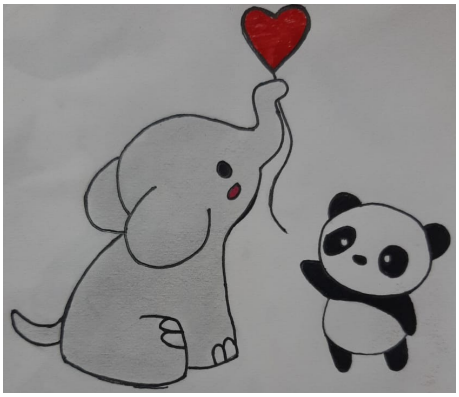


ART CORNER

SUBMITTED BY:
MIHIKA SHRIVASTAVA
B BALANISHA
A SHIVAANI
LAVANYA A.
LIKHITHA S



nned with CamScanner



TEAM AT WORK

FACULTY COORDINATORS:

DR. D. VENKATARAMAN
MS. T. BAGYAMMAL
MS. JEENA

ASCII EXECUTIVE MEMBERS:

SAI PRIYADARSHINI,
CB.EN.U4CSE17250
ADITHI NARAYANAN,
CB.EN.U4CSE18205
ROOPA VIDHYA,
CB.EN.U4CSE18143
NIRMAL K,
CB.EN.U4CSE19038
SUMITHRA S,
CB.EN.U4CSE19247
MAHIMA LOLLA,
CB.EN.U4CSE19128
VARADHARAJAN K,
CB.EN.U4CSE19257

EDITOR:

MIHIKA SHRIVASTAVA,
CB.EN.U4CSE19439

INPUTS & SUGGESTIONS:

ADITHI NARAYAN,
CB.EN.U4CSE18205

CONTRIBUTORS:

(CSE E)

A. SHIVAANI,
CB.EN.U4CSE19401

B. BALANISHA,
CB.EN.U4CSE19413

LAVANYA A.,
CB.EN.U4CSE19434

MIHIKA SHRIVASTAVA,
CB.EN.U4CSE19439

LIKHITHA S.,
CB.EN.U4CSE19460

ROHIT SHARMA T.,
CB.EN.U4CSE19461

THANK YOU.