Maratha Vidya Prasarak Samaj's
Karmaveer Adv. Baburao Ganpatrao Thakare College of Engineering
Udoji Maratha Boarding Campus, Near Pumping Station, Gangapur Road, Nashik
Permanently Affiliated to Savitribai Phule Pune University Vide Letter No.:CA/1542 &
Approved by AICTE, New Delhi- Vide Letter No.: 740-89-32(E) ET/98 AISHE Code-C-41622

COMPUTER DEPARTMENT

TECHNICAL MAGAZINE

VOL 1 • ISSUE 2 • 2018-19

Computer Department

Vision

To be the center for excellence for training the world-class engineers to work with multi-disciplinary domain based on the state-of-the-art of technology enabled academic system blended with industrial and business practices.

Mission

To educate and train undergraduate students in Computer Engineering by instilling excellence to fulfill professional and social requirements in business and industry on the platform of scientifically designed academic processes.

Editorial Team

Mr. Pushkar P. Shinde
Editor-in-chief

Piyush Sonar
SE Computer
Editor

Program Educational Objectives

1.To inculcate computational and programming skills in the field of Computer Engineering.
2.To prepare the graduates to fulfill professional requirements in industry.
3.To motivate students to solve problems related to society.

I CAN'T USE LOGIC IN PROGRAMMING WHAT SHOULD I DO?

Are you a new developer and recently stepped into programming? You might be getting frustrated when you see that you don't get logics in programming and you don't know how to start solving a specific problem. You see other developers are good at using their brains in programming and solving the coding question very quickly. the question is....why you don't get the logic in programming and how to get better at programming logic?

The problem that most of the beginners are failed to understand is..." until or unless your brain won't do a lot of practice it's impossible to get the logic in programming.". Take an example of a typist or footballer. The reason why a typist is good at typing or footballer is good in his game is because of practice.

When you practice enough for something the response time is reduced for your brain and you eventually get the logics or solution for the problem. Practicing enough for the problems in programming makes you experienced and the more experience you will have the better programmer you'll become. Also, you need to be patient enough not to leave it by thinking it is not your cup of tea.

Understand that there is no shortcut to get better at programming but there are some techniques and tips to get better at logic in programming. We are going to discuss those techniques but remeber that all the techniques requires practice, practice, and practice. and definitely patience.

1. Solve New Problems Every Day.

The first advice is once you solve a specific problem don't repeat it for more than three to four times. Three to four times is ok but then move to the next problem and face new challenges. Let's say you are practicing for printing different kind of patterns, once you practice enough for two or three times, move to the next coding challenge.

Keep moving on and try to face a new problem every single day. Your brain has to prepare itself for the new challenge to reduce the response time and get the logic in programming. Solving a problem to print even and odd numbers multiple times for a couple of days won't help you in programming. Learn new things every day, this will also give you exposure to real-life problems and it will help you to write code for new challenging situations or problems.

2. Keep Moving On Level By Level

When you start doing programming start with the easy problem from some resources and then move to the next level. Practice enough for a variety of questions at the easy level, then move to some complex program (intermediate level questions) and try to solve a variety of questions for this level. Again move to the next level or more complex problem (hard level questions) and solve alot of problems. A lot of websites are there like GeeksforGeeks, HackerRank, Codewars,CodinGameto practice and improve the logics in programming level by level.

3. Divide Problems in Smaller Chunks

When you are given a problem firstly try to understand the complete problem and find out what exactly needs to be done. Think about the problem carefully and write down on paper what steps you need to take in order to solve a specific problem. Think about all the case scenarios, steps and according to that write down the input or variables that you need to take in order to solve the problem.

For example, you need to write a program to perform the addition of two numbers. Now break down this problem into smaller chunks...

Step 1: By reading it you got to know you need 2 numbers and both need to be stored somewhere (in memory).

Step 2: To add those two numbers you need operand ("+").

Step 3: To store the addition result you need some memory.

Step 4: You need to display the result to the user or you need to use it somewhere else in the program.

Writing smaller steps will help you to map complex programs into smaller manageable chunks. These smaller chunks can be solved individually and then it can be merged to get the final output or actual solution.

4. Check Other People's Code

One of the best things to get better at programming logic is...keep checking the code that others have written. Check code written by other developers on Stackoverflow (largest community for developers) GitHub, Bitbucket or other open-source libraries. Check some great projects on GitHub and learn from it.
Check how people are writing the codes and how people are solving some programming problems.

When you look at other people's code and use some method or piece of code from there adjusting in your own code to get the solution, you eventually need to think over it and use your brain or logic to solve a problem and get the correct solution. Checking other people's code also helps you to find out the easier solution or various methods for the same problem.

5. Make Projects

One of the most important thing that a beginner or experienced person should follow to get better at programming logic is to make projects. Working on some real-life projects gives you more exposure and experience to become better at programming. You can choose any kind of project to build such as a web app, android app or iOS app. Make calculators, eCommerce projects, personal portfolio or anything that you love to build. You can make any small application or if you are experienced you can build some complex or big project. You learn how the workflow of building a project goes on. When you work on some project you need to solve a big problem by breaking it into smaller steps. You need to think over it carefully and solve these smaller chunks in order to build the complete project.

You use the programming syntax and logic to write down some piece of code in order to solve these smaller chunks, you also move some code here and there to get the correct result, you merge some piece of code, you use the implementation of one function into another one and you face a lot of challenging situations. When you build a project, you go through a lot of difficulties and you debug a lot of problems that help in building the logic in programming.

Tips

1. Don't skip the question while solving some exercises. Most of the beginners make a common mistake that they skip some questions and move to the next chapter. Suppose if there are 10 questions and you solved 7 questions (rest 3 questions you skipped because you think it's easy), out of that you solved 4 questions on your own and you checked the answers for 3 questions from somewhere else. Now when you move to the 2nd chapter and start solving the exercises from this chapter you face difficulty because your brain is not trained to handle the variety of questions for the 2nd chapter (this is all because you skipped some variety of question from the previous chapter that was helpful in training your mind to handle the question from 2nd chapter). Do as many questions as you can and train your mind to improve the logic in programming.

2. When someone is teaching programming, don't just understand the concept and think that you don't need to solve the problems on your own if you understood everything whatever the next person is teaching you. You really need to get your hands dirty in code. You can't get better at building logics in programming until or unless you don't practice on your own.

3. Do not check the solution immediately. Check the solution when you spent enough amount of time and still unable to solve the problem. Try to solve the problems on your own first and have patience. Once you solved it, check the solution written by other developers.

4. When you make some program and you feel that you need to read some theoretical concept in order to solve the problem then please go through the theoretical concept first. The theoretical concept builds the basic foundation and helps in solving the problem.

5. Be consistent. Don't leave a gap, it's really important. Practice for the programming questions every day. Practicing for three days and leaving it for two days breaks the rhythm and doesn't help in getting better at programming (especially for beginners).

-MINAL MAIN
FE COMPUTER

KUBERNETES :: KEY TO AUTOMATION

INTRODUCTION

Kubernetes (also known as k8s or "kube") is an open source container orchestration platform that automates many of the manual processes involved in deploying, managing, and scaling containerized applications. In other words, you can cluster together groups of hosts running Linux containers, and Kubernetes helps you easily and efficiently manage those clusters. Kubernetes clusters can span hosts across on-premise, public, private, or hybrid clouds. Kubernetes was originally developed and designed by engineers at Google. Google was one of the early contributors to Linux container technology and has talked publicly about how everything at Google runs in containers.

KUBERNETES CLUSTER

A Kubernetes cluster is a set of node machines for running containerized applications. If you're running Kubernetes, you're running a cluster. At a minimum, a cluster contains a control plane and one or more compute machines, or nodes. The control plane is responsible for maintaining the desired state of the cluster, such as which applications are running and which container images they use.

Nodes actually run the applications and workloads. The cluster is the heart of Kubernetes' key advantage: the ability to schedule and run containers across a group of machines, be they physical or virtual, on premises or in the cloud. Kubernetes containers aren't tied to individual machines. Rather, they're abstracted across the cluster.

UNDERSTANDING KUBERNETES

Some of the more common terms to help you better understand Kubernetes.

- **Control plane:** The collection of processes that control Kubernetes nodes.
- **Nodes:** These machines perform the requested tasks assigned by the control plane.
- **Pod:** A group of one or more containers deployed to a single node. All containers in a pod share an IP address, IPC, hostname, and other resources.
- **Replication controller:** This controls how many identical copies of a pod should be running somewhere on the cluster.
- **Kubelet:** This service runs on nodes, reads the container manifests, and ensures the defined containers are started and running.
- **kubectl:** The command line configuration tool for Kubernetes.

HOW DOES KUBERNETES WORK?

A working Kubernetes deployment is called a cluster. You can visualize a Kubernetes cluster as two parts: the control plane and the compute machines, or nodes. Each node is its own Linux environment, and could be either a physical or virtual machine. Each node runs pods, which are made up of containers. The control plane is responsible for maintaining the desired state of the cluster, such as which applications are running and which container images they use. Compute machines actually run the applications and workloads. Kubernetes runs on top of an operating system

and interacts with pods of containers running on the nodes. The desired state of a Kubernetes cluster defines which applications or other workloads should be running, along with which images they use, which resources should be made available to them, and other such configuration details.

From an infrastructure point of view, there is little change to how you manage containers. Your control over containers just happens at a higher level, giving you better control without the need to micromanage each separate container or node.

NEED OF KUBERNETES

Kubernetes can help you deliver and manage containerized, legacy, and cloud-native apps, as well as those being refactored into microservices. In order to meet changing business needs, your development team needs to be able to rapidly build new applications and services. Cloud-native development starts with microservices in containers, which enables faster development and makes it easier to transform and optimize existing applications. Kubernetes fixes a lot of common problems with container proliferation by sorting containers together into "pods." Pods add a layer of abstraction to grouped containers, which helps you schedule workloads and provide necessary services like networking and storage to those containers. Other parts of Kubernetes help you balance loads across these pods and ensure you have the right number of containers running to support your workloads.

APPLICATIONS

• **GitLab:** A single application for all stages of the DevOps lifecycle.

• **CloudBees:** CI/CD automation engine for growing organizations.

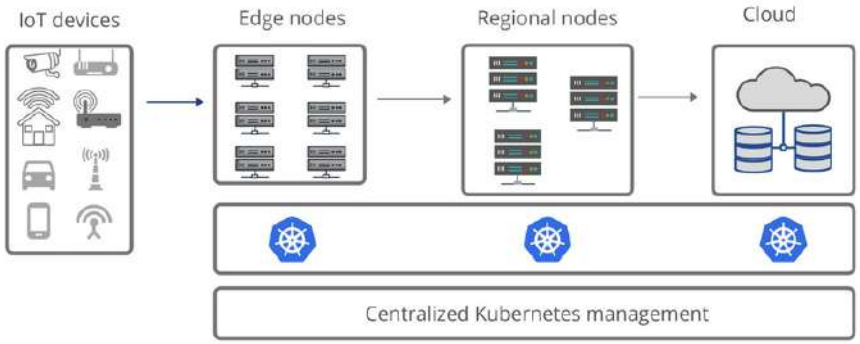
• **Neo4j:** Graph database management system. • **Seldon:** Machine-learning deployment for Kubernetes. • **WordPress:** Web publishing platform for websites and blogs.

WHO USES KUBERNETES?

✓ Google
✓ Udemy
✓ Slack
✓ Robinhood
✓ StackShare
✓ The New York Times

KUBERNETES INTEGRATIONS

✓ Docker
✓ Microsoft Azure
✓ Google Compute
✓ Rancher
✓ Helm
✓ Istio
✓ Google Kubernetes



Kubernetes at the edge

CONCLUSION

Kubernetes is one of the most commonly-used container management systems in the tech space. One important advantage of Kubernetes is quick deployment and application update features that bring improved productivity for developers. It allows the developer to focus on logically building the application.

REFERENCES

✓ https://www.redhat.com/en/topics/containers/what-is-kubernetes

✓ https://stackshare.io/kubernetes

✓ https://kubernetes.io/docs/home/

✓ "Kubernetes 1.21: Power to the Community". Kubernetes. Retrieved 2020-04-09.

Neeraj Nawale
FE Computer

HOW EXPLAINABLE ARTIFICIAL INTELLIGENCE CAN HELP HUMANS INNOVATE

As artificial intelligence becomes an increasing part of our daily lives, from the image and facial recognition systems popping up in all manner of applications to machine learning-powered predictive analytics, conversational applications, autonomous machines, and hyperpersonalized systems, we are finding that the need to trust these AI based systems with all manner of decision making and predictions is paramount. AI is finding its way into a broad range of industries such as education, construction, healthcare, manufacturing, law enforcement, and finance. The sorts of decisions and predictions being made by AI-enabled systems is becoming much more profound, and in many cases, critical to life, death, and personal wellness.

This is especially true for AI systems used in healthcare, driverless cars or even drones being deployed during war.

However most of us have little visibility and knowledge on how AI systems make the decisions they do, and as a result, how the results are being applied in the various fields that AI and machine learning is being applied.

Many of the algorithms used for machine learning are not able to be examined after the fact to understand specifically how and why a decision has been made. This is especially true of the most popular algorithms currently in use – specifically, deep learning neural network approaches. As humans, we must be able to fully understand how decisions are being made so that we can trust the decisions of AI systems. The lack of explainability and trust hampers our ability to fully trust AI systems. We want computer systems to work as expected and produce transparent explanations and reasons for decisions they make. This is known as Explainable AI (XAI).

Therefore, AI researchers are now turning their efforts towards developing AI algorithms that can explain themselves in a manner that humans can understand. If we can do this, then AI will be able to uncover and teach people new facts about the world that have not yet been discovered, leading to new innovations.

Learning from experience –

One field of AI, called reinforcement learning, studies how computers can learn from their own experiences. In reinforcement learning, an AI explores the world, receiving positive or negative feedback based on its actions.

This approach has led to algorithms that have independently learned to play chess at a superhuman level and prove mathematical theorems without any human guidance. AI researcher, use reinforcement learning to create AI algorithms that learn how to solve puzzles such as the Rubik's Cube.

Through reinforcement learning, AIs are independently learning to solve problems that even humans struggle to figure out. This has got many researchers thinking less about what AI can learn and more about what humans can learn from AI. A computer that can solve the Rubik's Cube should be able to teach people how to solve it, too.

Making the black box of AI transparent with Explainable AI (XAI) -

Explainable AI (XAI) is an emerging field in machine learning that aims to address how black box decisions of AI systems are made. This area inspects and tries to understand the steps and models involved in making decisions. XAI is thus expected by most of the owners, operators and users to answer some hot questions like: Why did the AI system make a specific prediction or decision? Why didn't the AI system do something else? When did the AI system succeed and when did it fail?

When do AI systems give enough confidence in the decision that you can trust it, and how can the AI system correct errors that arise?

One way to gain explainability in AI systems is to use machine learning algorithms that are inherently explainable. For example, simpler forms of machine learning such as decision trees, Bayesian classifiers, and other algorithms that have certain amounts of traceability and transparency in their decision making can provide the visibility needed for critical AI systems without sacrificing too much performance or accuracy. More complicated, but also potentially more powerful algorithms such as neural networks, ensemble methods including random forests, and other similar algorithms sacrifice transparency and explainability for power, performance, and accuracy.

AI explainability can be described in three parts which include: prediction accuracy which means models will explain how conclusions are reached to improve future decision making, decision understanding and trust from human users and operators, as well as inspection and traceability of actions undertaken by the AI systems. Traceability will enable humans to get into AI decision loops and have the ability to stop or control its tasks whenever need arises.

An AI system is not only expected to perform a certain task or impose decisions but also have a model with the ability to give a transparent report of why it took specific conclusions.

Peering into the black box -

Unfortunately, the minds of superhuman AIs are currently out of reach to us humans. AIs make terrible teachers and are what we in the computer science world call “black boxes.”

A black-box AI simply spits out solutions without giving reasons for its solutions. Computer scientists have been trying for decades to open this black box, and recent research has shown that many AI algorithms actually do think in ways that are similar to humans. For example, a computer trained to recognize animals will learn about different types of eyes and ears and will put this information together to correctly identify the animal.

The effort to open up the black box is called explainable AI. To understand this, we will see an example of solving Rubik’s Cube. The Rubik’s Cube is basically a pathfinding problem: Find a path from point A – a scrambled Rubik’s Cube – to point B – a solved Rubik’s Cube. Other pathfinding problems include navigation, theorem proving and chemical synthesis.

Solutions to the Rubik's Cube can be broken down into a few generalized steps – the first step, for example, could be to form a cross while the second step could be to put the corner pieces in place. While the Rubik's Cube itself has over 10 to the 19th power possible combinations, a generalized step-by-step guide is very easy to remember and is applicable in many different scenarios.

Approaching a problem by breaking it down into steps is often the default manner in which people explain things to one another. The Rubik's Cube naturally fits into this step-by-step framework, which gives us the opportunity to open the black box of our algorithm more easily. Creating AI algorithms that have this ability could allow people to collaborate with AI and break down a wide variety of complex problems into easy-to-understand steps.

Collaboration leads to innovation -

The process starts with using one's own intuition to define a step-by-step plan thought to potentially solve a complex problem. The algorithm then looks at each individual step and gives feedback about which steps are possible, which are impossible and ways the plan could be improved. The human then refines the initial plan using the advice from the AI, and the process repeats until the problem is solved. The hope is that the person and the AI will eventually converge to a kind of mutual understanding.

Considering the above example of Rubik's cube, the algorithm is able to consider a human plan for solving the Rubik's Cube, suggest improvements to the plan, recognize plans that do not work and find alternatives that do. In doing so, it gives feedback that leads to a step-by-step plan for solving the Rubik's Cube that a person can understand. The next step is to build an intuitive interface that will allow the algorithm to teach people how to solve the Rubik's Cube. Hope is to generalize this approach to a wide range of pathfinding problems.

People are intuitive in a way unmatched by any AI, but machines are far better in their computational power and algorithmic rigor. This back and forth between man and machine utilizes the strengths from both. This type of collaboration will shed light on previously unsolved problems in everything from chemistry to mathematics, leading to new solutions, intuitions and innovations that may have, otherwise, been out of reach.

Reference : (1) theconversation.com

By : Isha Suhas Kulkarni
FE Computer

INTERNET OF THINGS – A CHANGING WORLD

There used to be a time when Man was amazed by sci-fi movies. Now man lives a world immersed in science. Technology is evolving rapidly. IoT or the Internet of Things has become a lot familiar in our lives. It refers to the network of physical things like home appliances or security systems connected to the internet for effortless utility. IoT uses internet, data storage, analytics, and sensors to become better efficient to meet our daily needs.

The IoT provides a lot of convenience and comfort in the modern, fast-paced living. One cannot imagine going on a holiday without internet. This change has presented the modern-day consumer with an unprecedented amount of convenience and comfort through smarter living. This has further raised consumer expectations, and technology innovators are constantly striving to meet these ever-increasing demands.

Smart Homes and Electronic Gadgets

Home automation or smart home systems allows the owners to control their appliances, house temperature, lighting remotely through smartphones or laptops, which gets connected to a user interface.

Home security and alarm systems are used in many houses to monitor young children or in general for safety reasons. It is safe energy, that is, lights getting switched off when one is not using it.

Smart Cars

Smart cars can be a big boon for an easy commute. Connected cars with "talk to each other" low of technology where cars and drivers work together for creating an easy flow of traffic. Companies like Tesla, Volvo and Mercedes are trying to make smart cars a reality for the ordinary consumer. Such technology can avoid traffic jams, parking space management and make the commute in general much more comfortable. Driverless cars such as Connected Autonomous Vehicles (CAVs) and Autonomous Electric Vehicles (AEVs) can significantly help in management smooth traffic by reducing human-errors. These vehicles are a significant advantage to the elderly and disabled persons, and it can also cut the carbon emissions as it is powered electrically.

Animal Farming:

IoT has developed enough to serve for animal safety. Cattle grazing gets monitored through portable sensors, which the animals should wear.

Calorie intake, potential hazards, interior humidity and temperature monitoring, is possible through IoT. The sensors can help the farmers understand the optimal time for breeding by analyzing animal fertility.

Smart Agriculture

Climate change and lack of farmworkers are the two factors affecting the farming industry today. IoTs can analyze the best time to irrigate the crops, check temperature soil humidity, water level by collecting the data through sensors. The lack of farmworkers gets balanced through farm machines and internet-enabled gadgets. Smart farming has changed the agricultural management system by optimizing labour needs and observing weather patterns like high wind, heavy rain or extreme frost. IoT for agriculture and animal farming is getting adopted in many western countries as part of high-tech farming.

Healthcare:

IoT enabled devices can provide better healthcare and safety for patient care. Physicians can monitor the patient's vitals remotely and provide superlative care.

IoT also has devices collecting data on patient's blood pressure variation, exercise check, diabetes, calorie count and so on. Its transformation is visible in healthcare applications, hospital management, and even insurance companies.

E-Commerce:

E-Commerce is the shopping magnet of the twenty-first century. From safety-pins to furniture sets, you get everything you need. But IoT enabled retail has a lot of other benefits, like better tracking and logistics, better customer service and maintenance, and the much-known personalized data delivery. It has taken over the E-Commerce world in the last couple of decades. The supermarkets have automated check-ins and check-outs with the help of IoT. The shoppers can walk in, do the shopping and walk-out as their smartphones cover the bills.

Smart Cities:

Smart cities are not mere talks, not possible ventures with the help of IoT. Connected cities can change the life of its citizens, by providing better traffic management, connected cars for an easy commute, smart garbage for waste management, and even to improve air quality.

Hence, IoT is a considerable advantage for modern society. But there are security and safety issues as well. IoT for smart cities, home appliances, smart cars require shared data. Hackers can easily decipher personal details through smartphone applications. Such a lack of proper security needs clearing. Thus,there are wide range applications of IoT in our daily life which is making our work easy.

References:

1.https://www.researchgate.net/publication/329672903_Review_on_Applications_of_Internet_of_ThingIoT
2.https://www.sas.com/en_in/insights/big-data/internet-of-things.html
3.https://www.wsj.com/articles/the-internet-of-things-is-changing-the-world-01578689806

WHY FLUTTER IS THE FUTURE OF MOBILE APP DEVELOPMENT ?

What's common between Google Ads, Alibaba, Reflectly, and Birch Finance? They're all developed using Flutter. According to the Flutter community, more than 100,000 apps have been shipped to hundreds of million devices. Why is Flutter so popular, and why do experts believe it to be the future of cross-platform mobile app development? Let's look at what makes Flutter unique, why it's the future, and how it can benefit your developers.

What is Flutter?

Flutter is an open-source UI software development kit by Google that allows you to develop applications for multiple platforms.

From a single codebase, you can develop applications for iOS, Android, Windows, Mac, Linux, web, and Google Fuchsia. The UI toolkit is written in C, C++, and Dart and helps you create beautiful, natively-compiled applications.

What Makes Flutter Unique?

One essential feature that makes Flutter unique is that it allows you to create apps for multiple platforms using a single codebase. You don't have to absolve yourself in writing new codes for each platform.Additionally, the UI is exactly the same on Android and iOS. Hence, when you hire Flutter developers, they don't have to wrestle with styling issues specific to each platform. The Flutter framework guarantees that everything will look the same.

Why Is Flutter the Future of Cross-Platform Mobile App Development?

1. Simple Setup with Excellent Documentation 2. Single codebase for every platform
3. Least Coding Required
4. Completely customizable widgets
5. Faster application development
6. Faster Time to Market
7. Less Development Cost
8. Performance Boost
9. Wide number of open source package
10. Comes with great learning source
11. Firebase Integration

How Does Flutter Benefit Developers?

1. Large Community Support
2. Ideal for Advanced UI
3. Excellent Developer Experience
4. Open-Source Packages
5. Integration with Existing Apps

How Will Flutter for Mobile Development Continue to Get better in 2021?

- Dart's null safety will be introduced – Migration of the package ecosystem and plugin will be shepherded to null safety in Flutter.
- The ergonomics and performance of embedding Flutter in the existing iOS and Android application will be improved
- Production-quality support will be given for macOS, Web, Linux, and Windows
- Overall Flutter app quality will be improved through dedicated efforts on runtime performance, application download size overhead, memory usage, battery usage, etc.

Wrapping Up

There are several reasons why experts believe Flutter will be the indisputable king of mobile apps in the near future. The business benefits of Flutter are substantial. Companies can get their products to the market quickly, reduce costs, and
target multiple platforms, and hence more customers, at once.

From the developers' standpoint, Flutter is easier to use and allows faster app development. The growing community ensures that any bugs are promptly addressed.

-MINAL MAIN
FE COMPUTER