LitCode

Where ideas make waves.

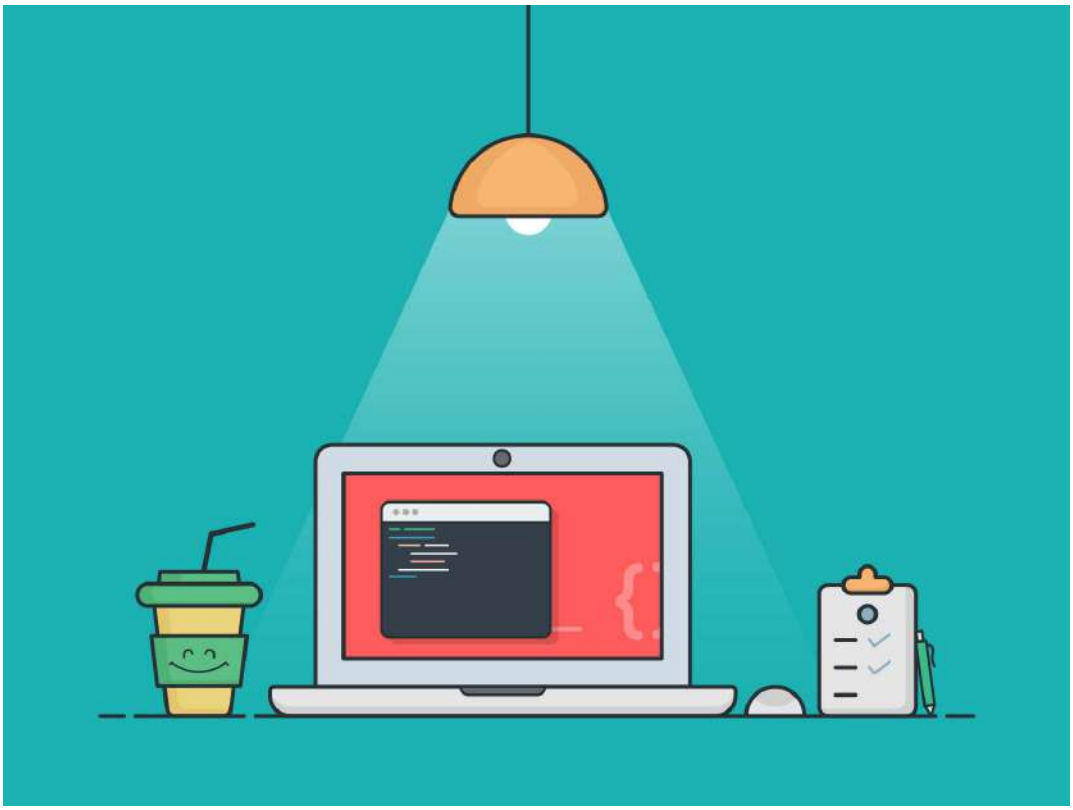
Posts published on July 17, 2025, from litcode.net. Printed on July 17, 2025 using Print My Blog

Why should everyone learn to code?

February 2, 2022

Categories: Let's learn, Let's read

Tags: 100DaysOfCode, Coding, JustCode, LitCode, ProgrammersLife



Why should everyone learn to code? This is the question I get asked the most. We live in the world created and ruled by technology. Learning to code helps you understand this chaotic world better.

In this article, I will go through the main reasons that will encourage you to learn to code. The resources listed I used when I was a beginner too.

You don't have to be a genius to do it.

Remember:

> "Always walk through life as if you have something new to learn and you will."
>
> Vernon Howard

What is coding?

Coding is writing instructions to the computer on what and how to do something. You write those instructions in your chosen programming language. When you learn to code you have the skills to create programs the computer will execute. Those programs have the elements of art, design, math and engineering. The coder is the person that does exactly this.

How is it done?

Coder writes programs in programming language, sometimes more than one. Every programming language has specific syntax and rules that apply to it. Part of learning to code is learning those rules. There are different types of programming languages:

- Python
- C++
- Java
- JavaScript
- Dart

and many more!

After learning to code in one programming language, learning another one is easy!

5 reasons why everyone should learn to code

I chose 5, to me, most important reasons that will motivate you to learn coding. Each of these reasons still motivate me to code every day.

Let’s jump right into them!

1. Coding teaches you how to think

> “Everyone in this country should learn how to program because it teaches you how to think”
>
> Steve Jobs

Whether you support Steve Jobs or not, he has a point.
Coding teaches you how to think more logically. One of the most important things coding thought me is that mistakes aren’t failure. When you start, every-day problems will have simpler and more creative solutions. Coding teaches you to try again until you find that right solution. After you start approaching your problems with open-mindedness.

While coding I realized there is more than one right answer.

2. Enhances problem solving skills

According to the definition, problem solving is the act of defining a problem. In other words:

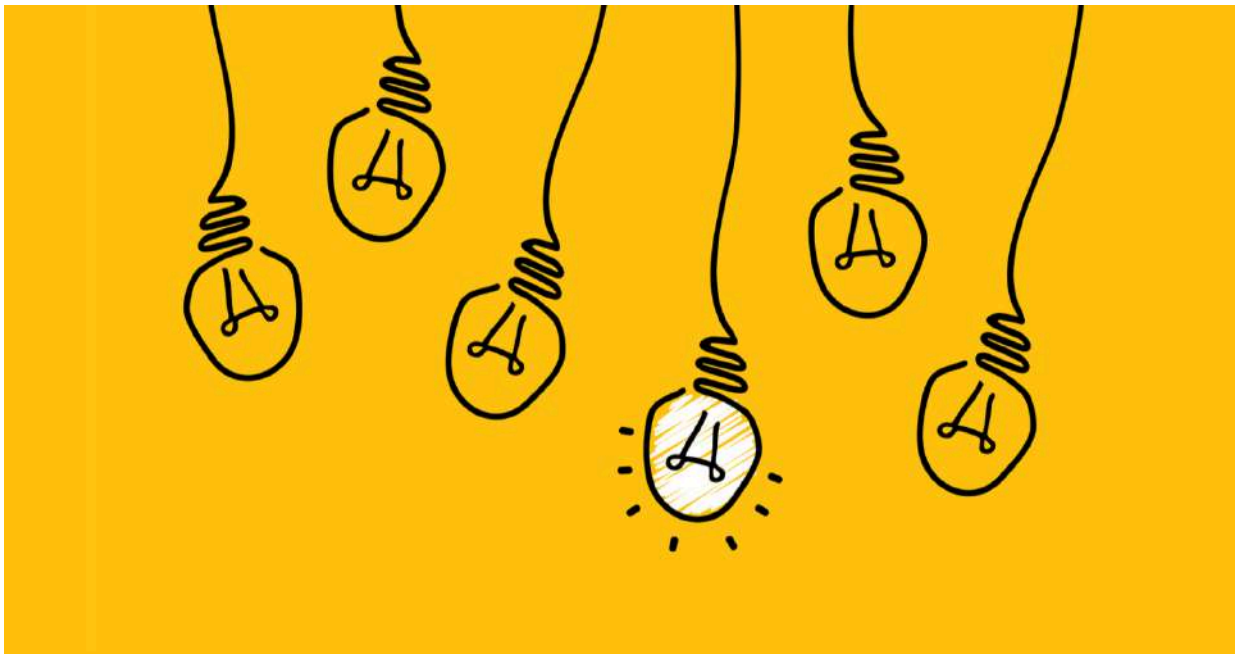
- Determining the cause of the problem
- Identifying steps to the solution
- Implementing the solution

So how does coding improve this?

Coding gives you a new perspective of the problem. It pushes you to approach the problem from different angles, to think outside the box. Problems never stop, solving them makes us happier. Whether you are solving a problem at work or in every-day life, coding helps.
You gain the skills to create more innovative solutions and expand your horizons.

After I started coding, I gained the skills to create numerous solutions.

3. Opportunity to create anything you imagine



When Sir Francis Bacon published his work, Meditationes Sacrae (1597), he said : "knowledge itself is power".

Learning new skills expands your knowledge. It helps you achieve your goals and gives you the motivation to do it.

When you learn to code, you gain the knowledge to create anything you can imagine in your head. You gain the power to transform your ideas into reality and show them to the world. Opportunities are limitless, from creating mobile apps, web apps, video games...everything!

With knowledge comes power, with power come opportunities.

4. Job opportunities and earning potential

Every job in the world has a connection with technology. IT jobs are positions in diverse fields. Such as computer software, hardware, computer support and more. Tech industry offers many high paying jobs and it is the fastest growing industry in the world. Whether you live in San Francisco or in Moscow you can find a job in tech. You can work remote or even start freelancing. It offers opportunities for recognition and working on diverse projects.

Learning to code opens this new world of possibilities to you. Feel free to say Hello.

```
#include <iostream>

int main()
{
    std::cout<<"Hello World";
    return 0;
}
```

The highest paying jobs in Tech for 2022:

- Product Manager.
- Artificial Intelligence (AI) Engineer.
- Full-Stack Developer.
- Cloud Architect.
- DevOps Engineer.

- Blockchain Engineer.
- Software Architect.

After you learn to code, you can create anything and be anyone.

5. Boosts your confidence

Learning to code, helps you improve yourself and jump outside the box. It made me jump out of molds pre-made for me.

Coding opens so many opportunities that will change the course of your life. Trying new things, creates new opportunities and introduces you to new people. All these things boost your confidence.

When learning to code, don't be afraid to make mistakes, "fail" or ask questions. Set yourself small goals during the process to keep you motivated.

Remember: success is not the outcome it is the process. Doing is success.

Resources

I want to provide you some resources to help you get started. Most of these I used as a beginner.

- **Codecademy**

This is a great website for beginners and the ones looking for some practice. It offers you fr courses in diverse subjects of coding. Codecademy is easy, free and offers you high-quality

content.

- **freeCodeCamp**

Another amazing free platform for learning. It offers quality courses suitable for beginners and the ones beyond. More than 2,000 campers have gotten developer jobs after joining Free Code Camp's open source community.

- **Code.org**

The main goal of Code.org is to motivate students to learn computer science. Because of their mission this is one of the best platforms to learn to code.

- **Khan Academy**

Khan Academy does it all. Similar to Code.org their mission is to create online tools that help educate students. It has great video lectures on different subjects including coding.

- **Udemy**

Udemy is an online learning and teaching marketplace with over 183000 courses and 40 million students. Here you can learn anything you want. Offers great courses on diverse topics and fields in programming.

Summary

In this article I talked about:

- **What is coding?**
- **How is it done?**
- **5 reasons why everyone should learn to code**

From teaching you to think, boosting your confidence, job opportunities and much more!

- **Resources to get you started**

I hope this information was useful and that it motivated you.

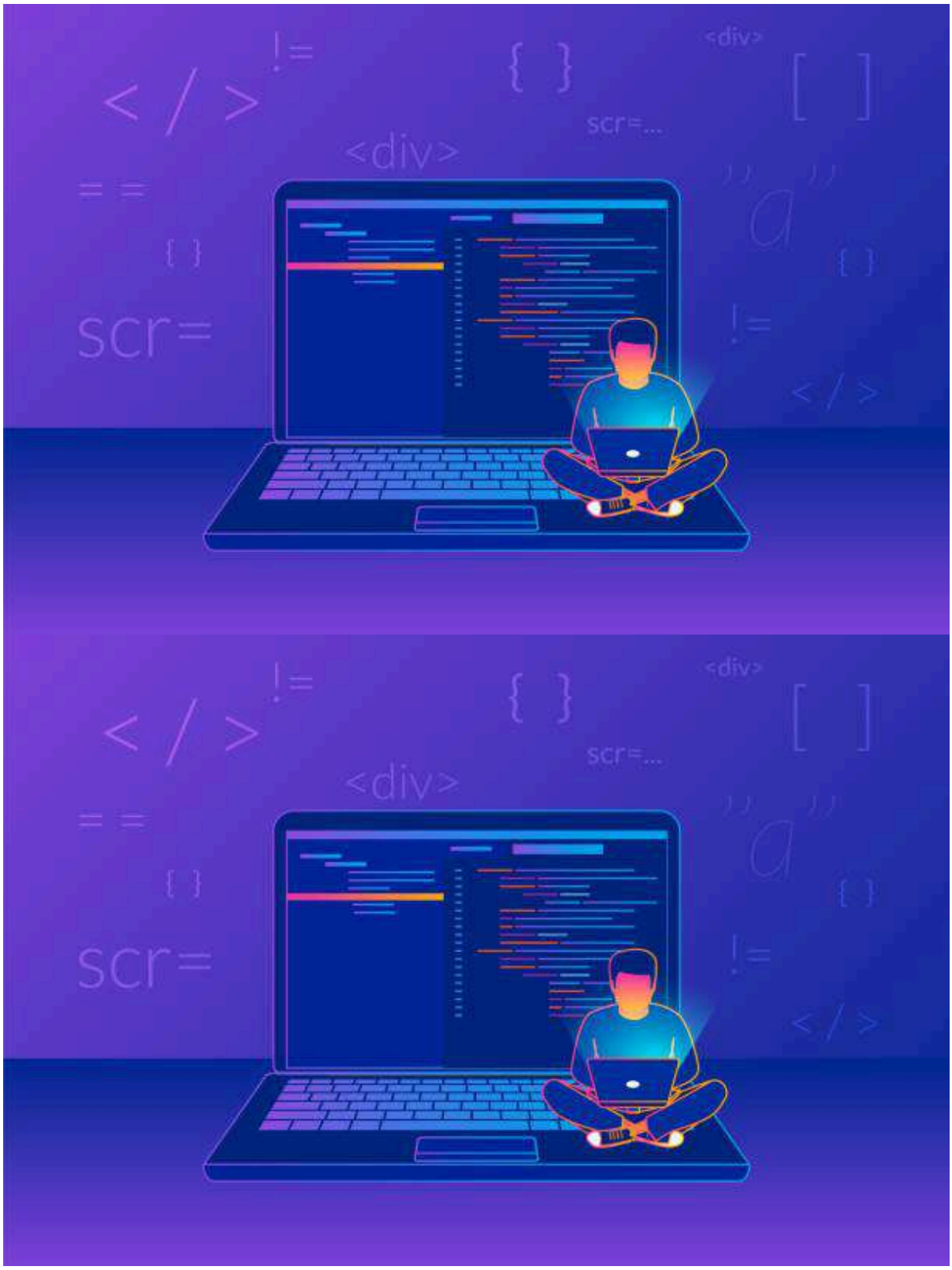
Happy creating – anonymous of LitCode!

How to be a better programmer?

February 4, 2022

Categories: Let's learn, Let's read

Tags: 100DaysOfCode, Coding, JustCode, LitCode, Programmer, ProgrammersLife



Every programmer has asked this question. Why?

When someone is passionate about their work, they will do everything to be better. It might

because of competition or just a desire to improve ourselves every day. I'm here to make this journey a little bit easier for you.

In today's article, I'll share some tips on how to improve as a programmer.
These tips helped me stay motivated and improve as a programmer on my journey.

Remember:

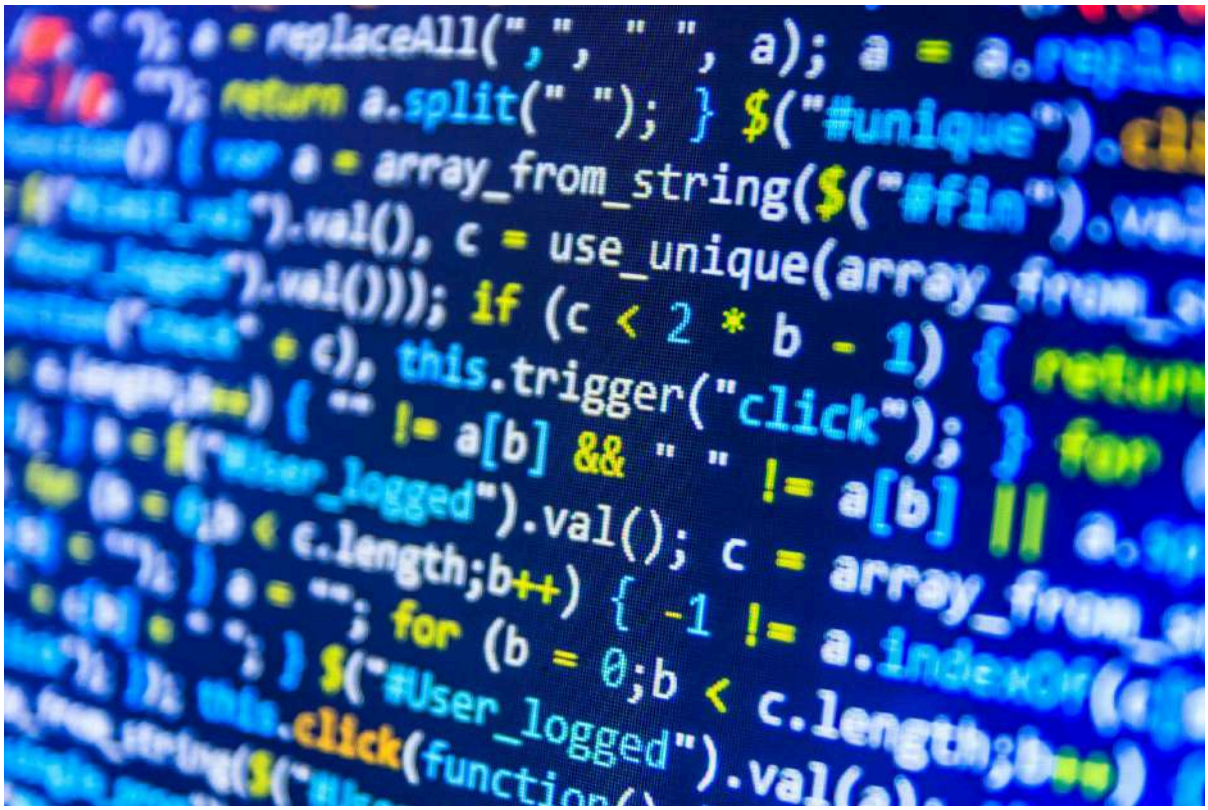
> "Every great dream begins with a dreamer. Always remember, you have within you the strength, the patience, and the passion to reach for the stars to change the world."
>
> Harriet Tubman

Never stop learning

We learn until we die. Every day is full of opportunities to grow and expand our knowledge. That is where the saying: "Seize the day." comes from.
As programmers, we work in the fastest-growing industry in the world.
Hundreds of new inventions are invented each day.
To be a better programmer, you should know that learning does not end after you graduate from school. **In order to improve you must have the desire to learn.**

We learn our whole lives. By learning, we can overcome challenges and solve problems more easily.

Write quality code you understand



Writing better code is a part of improving as a programmer. **Your code is the reflection of you.**

To write quality code, first, you must understand it. What is your end goal, how should the program behave? To understand your code you must learn. Don't start until you know what to do.

Remember: rushed code + distractions = bad code.

After you understand what you want to accomplish, then you start coding. And that is not the end. Every code can be improved. Be creative. Great programmers work on improving every day.

Here are some tips for writing quality code:

- Create descriptive names
- Comment your work
- Don't repeat yourself
- Solve errors efficiently
- Split your code into smaller units

Challenge your past self

Your only competition is you. That concept is hard to grasp, but it is the only truth. We all want to take down our competition believing that will improve us. But that doesn't make us better. Of course, healthy competition is often good for us, but when it becomes our only source of motivation it comes with side effects.

To be a better programmer your goal should be to compete with yourself. When we decide that, we approach things differently, we are more open-minded and creative.

- **A study found that mentally challenging ourselves can help us feel less anxious and sad.**

Set new goals, be honest with yourself and just start. There is only one way to go. Forward.

Work on diverse projects

Whether in school or self-thought, you should work on projects that aren't just assignments**.** **Practice is the key**. Set goals for yourself to develop as a programmer, but most importantly as a person.

Great programmers don't stick to only one thing. **They try different things, learn and develop diversity in their projects**. To be a better programmer, you must find what you enjoy and work on it. Setting priorities is a crucial part of growth. That doesn't mean that you should be interested in only one thing.

You can't find your niche without exploring. You can't become great if you do the same as everyone else.

- **Jump out of the box! Work, set goals, try new things, set priorities and have fun! The process is the success.**

As programmers, we have an opportunity to work on fantastic projects with amazing people. So don't be shy to show your work to the world!

Connect with other developers

> Everyone you will ever meet knows something you don't.
>
> Bill Nye

The one thing I know for sure is that two heads are better than one. Teamwork is a fundamental part of everything. We all have something new to learn from each other. That is how we grow.

To work with others means to interact, collaborate and cooperate to complete tasks and achieve goals.

Great programmers know the importance of connecting with other developers. That is a unique opportunity to share knowledge, experience and expertise. **To be a better programmer, you must create connections with others.**

While working with others, you learn to:

1. Communicate better
2. Be responsible
3. Be honest
4. What teamwork is
5. To listen

Resources

Google and Books are your best friends.

Clean Code by Robert Cecil Martin

- Write quality code you understand

The Pragmatic Programmer by David Thomas, Andrew Hunt

- Write quality code you understand
- Never stop learning

Show your work by Austin Kleon

- Connect with other developers
- Challenge your past self
- Work on diverse projects

Summary

In this article, I showed you some tips **to become a better programmer.**
All of these tips I use when I struggle and they keep me motivated. I hope the resources listed will make your journey easier.
You will struggle along the way, and that is the fun part! It will shape you as a person, and it will be fun to look at how far you've come.

Enjoy the process. Happy creating – anonymous of LitCode.

Learn C++ Programming

February 7, 2022

Categories: Let's create, Let's learn

Tags: 100DaysOfCode, C++, Coding, LitCode, ProgrammersLife

In today's article, I will be introducing you to the basics of C++.

The crucial part for beginners is C++ Syntax and The Basics. In the end, I listed Resources and References that helped me when I started learning it.

If you are new to the idea of programming, I recommend that you don't start with C++ cours ght

away. It is one of the most difficult languages for students to learn because it requires them to think a lot.

Before starting with the courses and coding, read this article as an introduction to your C++ journey.

What is C++?

> "C++ is designed to allow you to express ideas, but if you don't have ideas or don't have any clue about how to express them, C++ doesn't offer much help."
>
> **Bjarne Stroustrup**, C++ Programming Language

C++ is a cross-platform programming language developed by **Bjarne Stroustrup.** It was developed as an add-on to the C programming language. In its earlier days, it was called "C with classes".

Bjarne Stroustrup started working on C++ at Bell Laboratories in 1979.
In 1985, C++ was made publicly available outside of Bell Laboratories.

Since then, C++ has grown in popularity as one of the most used programming languages.

What is C++ used for?

With C++ you can create anything!

For over forty years, it's been used for game programming, software engineering, data structures, designing browsers, operating systems, applications and more. In my early days of programming, I used it as a contestant on many Competition Programming Challenges.

You'll find C++ in some way or another practically everywhere you look!

C++ vs other programming languages

C++ is a programming language that is powerful, efficient, and quick.

Here are some reasons why you should choose C++:

- C++ is super fast!

- This is a general-purpose programming language that can be used on multiple devices.
- C++ is cross-platform.
- It has a big, fast-growing community!
- Because of its popularity, C++ has a wide job market.

Whether you just started programming or not, learning the basics of C++ will be useful for your coding journey.

Before we dive in...

There are a few things to bear in mind before we get to the basics.

Make sure that you are learning a new skill, especially a new programming language, for the right reasons.

1. Learn the fundamentals of the language
2. Read the documentation
3. Set your learning goal
4. Understand your code
5. Develop projects outside the assignments for the course
6. Practice is the key

This is something that my article "How to Become a Better Programmer" will help you with.

Learn the Syntax

The rules that define a language's structure are referred to as **syntax.**
The easiest way to get introduced to C++ syntax is with the **Hello World** example. I will explain to you every line of this simple program.

```
#include <iostream>
using namespace std;

int main()
{
    cout<<"Hello World";

    return 0;
}
```

First line: `#include <iostream>` is a library of header files that allows you to interact with input and output objects. Header files add more functionalities to the program.
Line 2: `using namespace std` allows you to access names from the standard library for objects and variables.
Line 4: `int main()` always appears in a C++ program. This is referred to as a function.
Line 6: `cout` is a text output/printing object that is used in combination with the insertion operator (<<). It will print "Hello World" in this example.
Line 8: `return 0` ends the main() function.

Blank lines; C++ ignores white spaces, they're used to make code easier to read.
Every C++ statement ends with a semicolon (;).

Learn the Basics

Comments

Comment is an explanation or remark in a computer program's source code. Purpose of comments is to make the code more understandable and readable for the one reading it. Comments are ignored by compilers and interpreters.

In C++ they are declared with two forward slashes //.

```
//this is a comment in C++
```

Variables in C++

Variables are used to hold data in a computer program that may be accessed and changed.

The syntax for creating a variable in C++ is simple. **Variables have a data type, name, value assigned to it.**

```
data_type variable_name = value;
```

Here are the examples of variable data types in C++:

- `int` – a variable data type that stores whole numbers. They're called integers.
- `float` – a variable data type that stores floating-point numbers.
- `double` – a variable data type used to store floating-point numbers. **Difference between ble and float is that double is more precise and stores larger data.**

`char` – a variable data type that stores single characters. Such as ‘a’ or ‘B’. The character is inserted between single quotes (‘ ‘).

`bool` – a variable data type used to store true or false values.

`string` – a variable data type that stores text, such as “Hello World”. The text is inserted between double quotes (” “).

```
#include <iostream>

int main(){
  int x = 5; //data type is (int), name is (x) and value is (5)
  float b = 5.96; //data type is (float), name is (b) and value is
(5.96)
  double c = 7.88; //data type is (double), name is (c) and value is
(7.88)
  char y = 'a'; //data type is (char), name is (y) and value is (a)
  string s = "Hello World"; //data type is (string), name is (s) and
value is (Hello World)
  bool t = true; //data type is (bool), name is (t) and value is set to
(true)

  return 0;
}
```

Input and Output

`cin` – In C++, the cin (See in) object belongs to the iostream class. It is used to accept the input from the keyboard. It is used with the insertion operator (>>).

`cout` – In C++, the cout (See out) object belongs to the iostream class. It displays the output to a typical output device, such as a monitor. It is used with the extraction operator (<<)

```
#include <iostream>
using namespace std;

int main(){
  string name; //variable of (string) data type called name
  cin >> name; //enter input from the keyboard and store it in our
variable (name)
```

```
  cout << name; //after input, print variable value

  return 0;
}
```

Without the Line 2 we would have to type (std::) before our cin, cout and string because they belong to the standard library.

```
#include <iostream>

int main(){
  std::string name; //variable of (string) data type called name
  std::cin >> name; //enter input from the keyboard and store it in our variable (name)
  std::cout << name; //after input, print variable value

  return 0;
}
```

Conditions

Conditions are statements written by programmers that evaluate actions in the program and determine whether they are true or false.

If statement

We use If statement to specify code to be executed if condition is true.

The syntax is simple. **We type if keyword, in the parentheses we insert our condition, after curly braces we tell program what to do if our condition is true.**

```
#include <iostream>
using namespace std;

int main(){
  if(10>2){ //our condition checks if 10 is greater than 2
    cout << "10 is greater"; //if the condition is true print this
```

```
  }
}
```

Else statement

We use else statement to specify code to be executed if condition is false.

```
#include <iostream>
using namespace std;

int main(){
  int x = 1; // variable of type (int), named (x), with value of 1

  if(x>2){ //our condition checks if the value of (x) is greater than 2
    cout << "X is greater than 2."; //if the condition is true print
this
  }else{
    cout << "X is less than 2"; //if the condition is false print this
  }
}
```

Else if

We use else if statement to specify another condition if our first condition specified is false.

```
#include <iostream>
using namespace std;

int main(){
  int x = 0; // variable of type (int), named (x), with value of 1

  if(x>2){ //our condition checks if the value of (x) is greater than 2
    cout << "X is greater than 2."; //if the condition is true print
this
  }else if(x==0){ //our 2nd condition checks if the value of x equals to
0
    cout << "X is 0"; //if the first condition is false, print this
  }else{
```

```
    cout << "X is less than 2"; //if the condition is false print this
  }
}
```

Loops

Loop is a sequence of instruction s that is continually repeated until a certain condition is reached.

There are different types of loops: for loops, while loops, do-while loops.

For loop

When the number of iterations is known before entering the loop, for loops are commonly used. While loops that increment and test a loop variable might be used as shorthand for (for loops).

```
for (bool increment = true; increment = true; number = number+50 ) {
    cout << number;
  //while our variable increment is true variable number will be
incremented by 50
}
```

While loop

A While loop is used to repeat a block of code an unknown number of times until it meets a condition.

```
int i = 0;
 while (i < 5) { //while the current number is less than 5
   cout << i; //while condition is true print i
   i=i+1; //Increment the value of i with each iteration
 }
 // output will look like this - 1234
```

Do-while loop

The Do-while loop is a control flow statement that executes a block of code at least once before either repeating it or stopping it, based on a boolean condition at the end of the block.

```
int i = 0;
do {
  cout << i; //while condition is true print i
  i=i+1; //every time increment i by 1
}
while (i < 5); //while current i is less than 5

// output will look like this - 1234
```

Arrays

Array is a data structure that can hold a fixed number of elements of the same data type in a fixed size. A collection of data is stored in an array.

The syntax for arrays in C++ is: data type, array name, number of elements in the array and array values.

```
int my_array = ;
//my_array stores 4 values (1, 2, 3, 4) and can only store the values of
the same data type
//in this case data type of the array is int

string my_second_array = ;
//my_second_array stores 2 values (Hello and World) and can only store
the values of the same data type
//in this case data type of the array is string
```

Functions

Functions are “self-contained” code modules that perform tasks. Data is often “taken in,” processed, and a result is “returned” by the function. Our custom functions are created outside the main() function.

Syntax for declaring a function looks like this: data type, name of the function, parentheses, curly braces, inside curly braces we specify what our functionality.

If the data type is (int, string, double etc.) our function must return the same data type.

But if the data type is specified as void our function can return any data type or just print something.

```
int incrementFunction(){
  //functionality
  //must return int
}

void getName(){
  //functionality
}
```

In order for our function to be executed we must call it inside our main function.

```
#include <iostream>
using namespace std;

void greetUser(){
  cout << "Hello User";
}

int main(){
  greetUser();
  return 0;
}
```

Functions can also have input parameters.

```
#include <iostream>
using namespace std;

int myAge(int age){ //input parameter
  if(age>5){ //if age is greater than 5 return age from the function
    return age;
  }else{ //if condition is false return 0
    return 0;
  }
```

```
}

int main(){
  int my_age = 6;

  //when we are calling a function with return type we must print it
  cout << myAge(age); //if the function has input parameters we provide
them when we call it
  return 0;
}
```

IMPORTANT NOTE: in this article I listed just the basics to get you introduced to the language. This is useful for beginners to understand C++ better.

Put your knowledge into practice

In order to learn to code in a new programming language, it is crucial to **practice**. That is the key to success. Without practice, you will never be able to understand your code. And without understanding, you won't be able to code in that language.

If you are taking a C++ course, build projects that aren't just assignments.
That way you can practice what you've already covered and also learn new concepts.

Resources

Udemy C++ Courses:

- Beginning C++ Programming – From Beginner to Beyond
- Learn C++ Programming -Beginner to Advance- Deep Dive in C++
- The C++20 Masterclass: From Fundamentals to Advanced

Youtube C++ Courses:

- C++ Tutorial for Beginners
- C++ FULL COURSE For Beginners

Online platforms:

- Codecademy C++
- Learn C++
- W3Schools C++

Documentation:

- DevDocs C++ Documentation

Books:

- C++ Primer
- A Tour of C++
- The C++ Programming Language
- Think Like a Programmer: An Introduction to Creative Problem Solving

Summary

In today's article, I introduced you to the basics of C++.
This article is not a complete C++ course, just an introduction to the basics.
There are still more components of C++ like Classes and Objects, Pointers, Vectors, Polymorphism and much more! **Opportunities with C++ are limitless. You can create anything you want.**
I listed the resources to put you on the right part for your C++ journey. I hope you find the topics covered in this article helpful and that I motivated you to start!

Happy creating! – anonymous of LitCode

What are good coding goals?

February 9, 2022

Categories: Let's learn, Let's read

Tags: 100DaysOfCode, Coding, Dreams, Goals, LitCode

In today's article, I will talk about accomplishing your coding goals. But all of this information can be applied to any goal you have in life. Most of the people I encounter don't have goals or don't know what goals are.

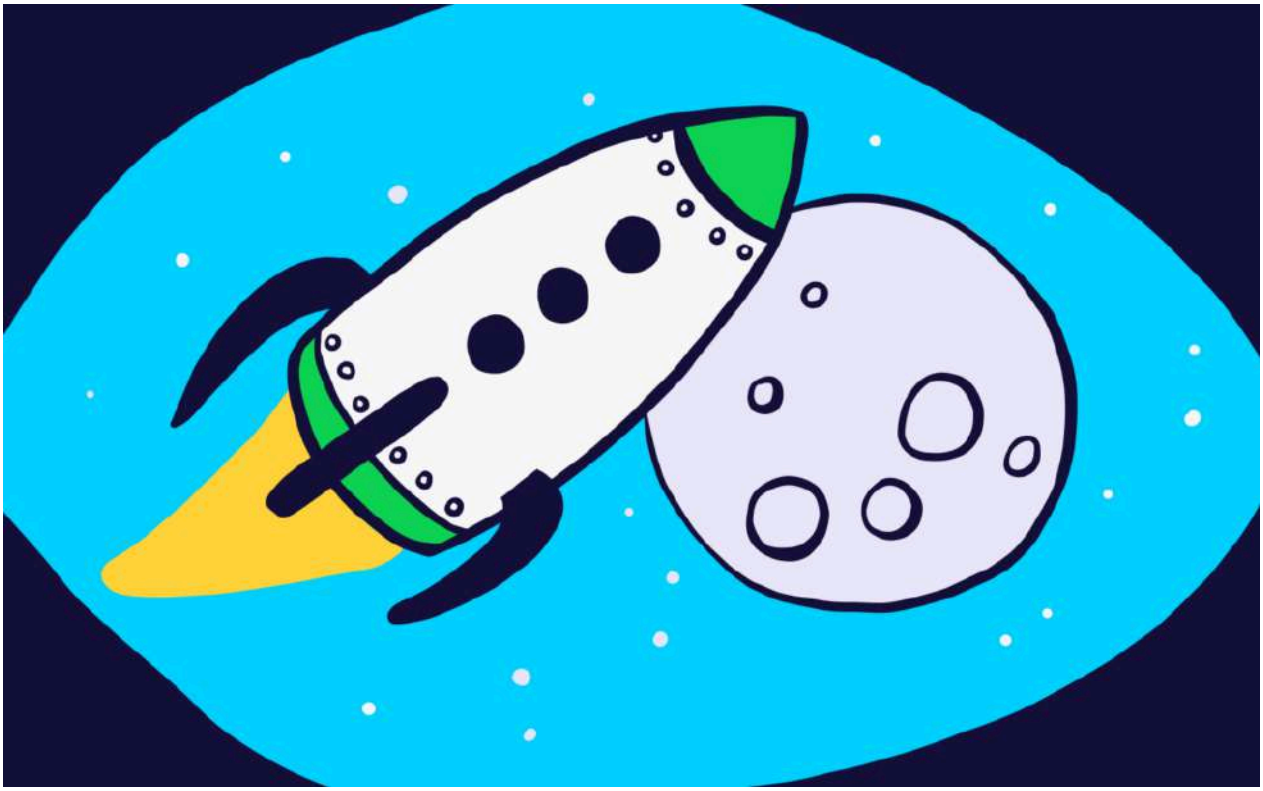
Only 8 % of people accomplish their goals because of this.

I will help you not to be one of that 92 %.

Here I share tips and resources to get you started. I hope the information in this article will put you on the right path on your coding and life journey.

Let's jump right in!

What are Goals?



By definition, a goal is a vision of the future or the desired outcome that someone or a group of people envisions, plans, and devotes to achieving. Usually, in the process of goal setting, we set deadlines. If you want a better life, set goals. They are truly your best friends.

You might not even notice, but every day you set goals.

A goal can be waking up on time, going to school, getting that promotion, reading a new book etc. Or just simply doing something new. For each of these goals, you must have a plan for them to become real.

There are three types of goals:

- **Outcome goals** – goals that focus on achieving the desired end goal.

Example: winning the basketball championship

- **Process goals** – goals that focus on actions.

Example: working out for 2 hours

- **Performance goals** – based on your standards.

Example: getting an A from Math test

In your life, you set at least one of these goals without calling them by what they actually are. They are GOALS.

What is the importance of Goal Setting?

> "If you're bored with life – you don't get up every morning with a burning desire to do things – you don't have enough goals."
>
> Lou Holtz

Goals give you purpose.
They set us in the right direction for our life. Once you start setting goals your whole life will change.

How?

Our goals define our values, and our values define us. Anything you do throughout the day is a part of who you are. How you react to certain situations or how you think. Goals set priorities and push you to act towards them.

If your goal is to get a promotion, your goal is pushing you to get to work on time, improve your performance etc.

All of your life decision will be affected by your goal. This means that you are in control of your future and that is the most liberating feeling.
Goals are not just the outcome. Even do getting to that finish line is amazing. The process makes it even better. Knowing that you struggled, sacrificed and fought for your goals is what makes it better.

Goal Fuel – Passion

Passion is defined as a strong feeling of enthusiasm or excitement for something or about doing something. This is the crucial part of goal setting.

Everything you do, you must do with passion.

A passionate person has very strong feelings about something or a strong belief in something. That something should be your goal.
Passion is the goal fuel. Without it, you will never achieve any of your goals.

If you don't have a reason to do something why would you do it?

When it comes to goal setting, we are lucky to be human. We are able to feel emotions. If you get emotionally attached to something it is more likely you will succeed. Even if you fail, you are closer to the goal more than yesterday.

How to find your passion? Put yourself out there.
In order to find it, you must try different things. Finding passion is a process just like any other. Don't beat yourself up if your first coding class is not what you really imagined.

There is passion (purpose) for everyone.

How to accomplish (coding) goals?

Consider your passions

Remember: Passion = Goal Fuel.
When setting goals, focus on what you're passionate about. If you do something you love, you are more open to the sacrifice that the process brings. That will motivate you to get up every morning.

You can have multiple passions.

Like I do, my passions are coding and writing, and my goal was to make a coding/motivatior og.
So I did exactly that. I combined my passions into a bigger passion.

Develop a plan

Make a plan. **A good plan will set you in the right direction towards your goals.**

The two best ways to commit to a goal are:

- Writing it down.
- Sharing it with others.

Writing it down puts the goal in the physical form. By sharing it with others you commit to accomplishing it. But by sharing it, I don't mean telling your whole plan and ideas to others. Just tell them what you're focusing on at the moment. If you notice your plan isn't working out how you imagined. Adjust it.

Put in the work

> "Pain nourishes courage. You can't be brave if you've only had wonderful things happen to you."
>
> Mary Tyler Moore

After you make your plan, just start.

There is no other way to achieve something. Like I said process matters the most.

The only way to get to the end is to start. The thing that helps me in hard times is to visualize the end goal. It makes the process much easier and more endurable.

Pain is worth it, trust me.

You will fail

How will you appreciate the wins if you've never failed?
Failing is a part of life, it makes victory sweeter. When you get emotionally invested, failing will hurt.

But what if we look at it this way;
You haven't actually failed. This is all a part of the process. The work you put in, the actions you take, failures you experience.

All called process.

Failures are opportunities to learn, change and they're the biggest source of motivation. If you don't fail, you're not doing anything.

Resources

Books:

- The Subtle Art of Not Giving a F*ck
- Goals!
- Your Best Year Ever
- Algorithms to Live By: The Computer Science of Human Decisions

Videos:

- How to design your life
- How To Stay Motivated When Learning To Code
- The Easy Way to Achieve Your Goals
- My Programming Goals For 2022!

Check out: How to be a better programmer

Summary

In today's article, I shared with you tips on how to get started with reaching your (coding) goals.
I hope you find the information in this article useful and that it motivated you to start.
Remember: Passion = Goal Fuel.

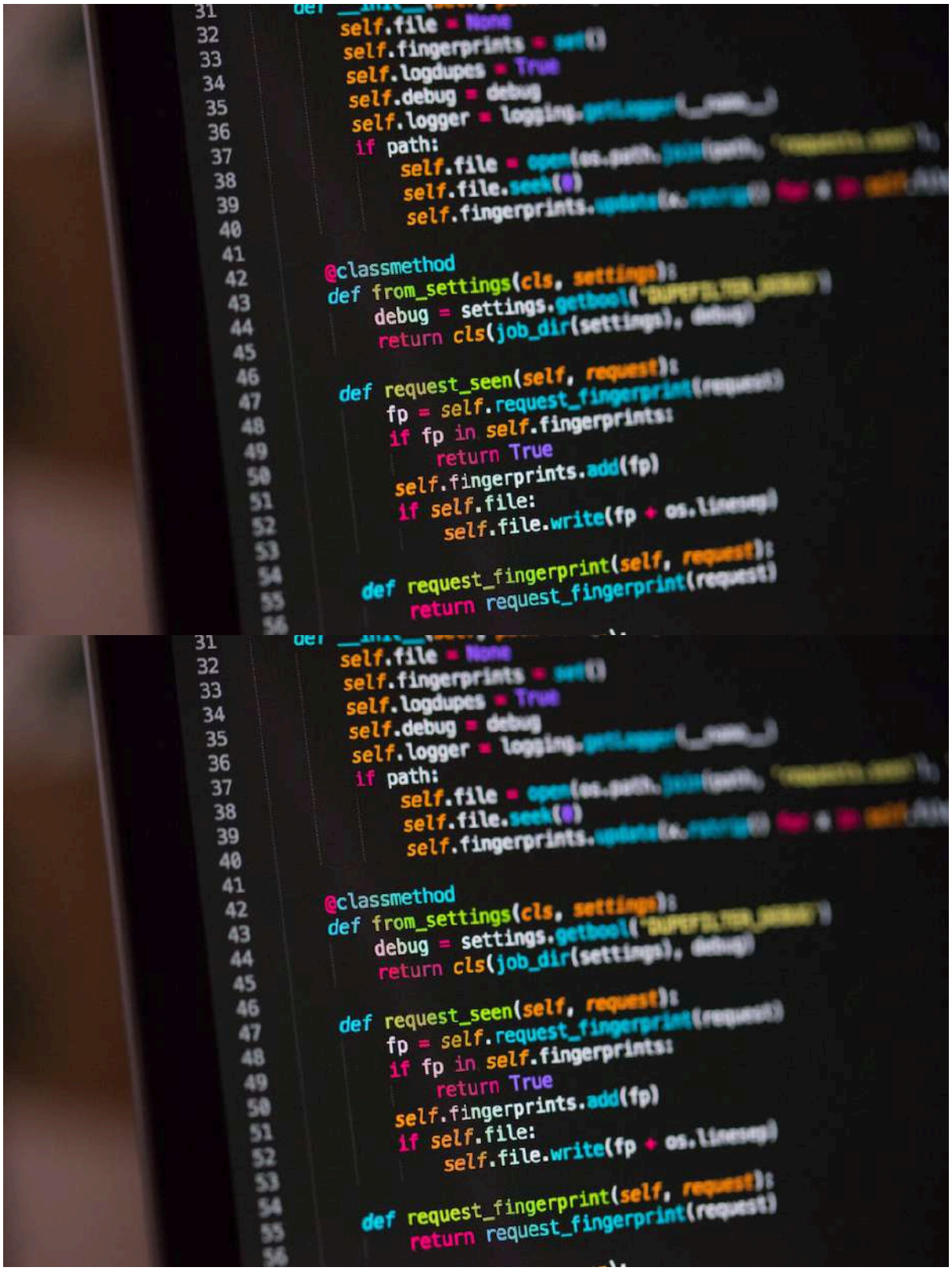
Enjoy the process. Happy creating! – anonymous of LitCode

Python Quiz

February 12, 2022

Categories: Quiz

Tags: 100DaysOfCode, Coding, LitCode, Programmer, Python, Quiz



Do you want to test your knowledge of Python?

Welcome to LitCode's first Quiz! 🚀

Today's Quiz is an opportunity to test your knowledge of Python and learn new things.
We will cover the basics but also more complex questions and answers.
In the end, you will be able to check your results to see where you are on your Python journey! 🐍

> "Education without application is just entertainment."
>
> Tim Sanders

Even if your scores are not what you expected, every day is an opportunity to learn. By taking this Quiz you are closer to your coding goals.
Let's jump right into it!

Resources

Here I listed the resources to get you started on your Python journey (or help you practice).

Remember: practice is the key to success.

- Python Official Site
- Learn Python Programming
- Python YouTube Course
- Python Tutorial
- Automate the boring stuff with Python
- Learn Python the Hard Way

Python is one of the best programming languages for beginners to learn.
It is undoubtedly considered a top programming language. Python has a large and supportive community and a great job marketplace.

With Python, you can build anything!

My articles that can help you:

- **How to be a better programmer?**
- **Reaching your (coding) Goals?**

Summary

In today's post, I helped you check your knowledge of Python and listed helpful resources t
you started. I hope the Quiz was engaging and you learned something new! Keep practicing and

show off your work.

So *until next time*, remember:

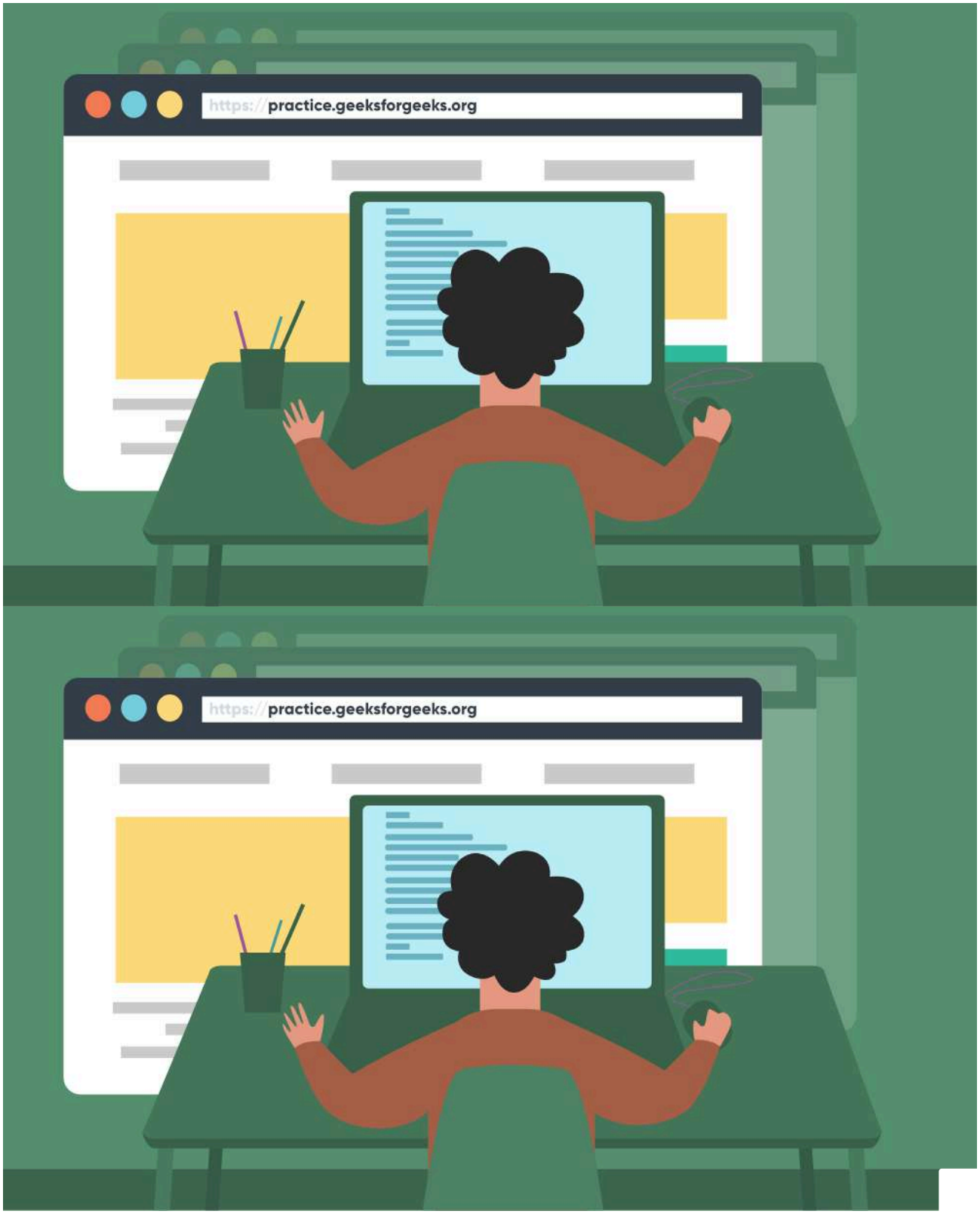
Enjoy the process. – anonymous of LitCode

A Guide to Competitive Programming

February 16, 2022

Categories: Let's learn, Let's read

Tags: 100DaysOfCode, Coding, Competitive, Goals, LitCode, Programming



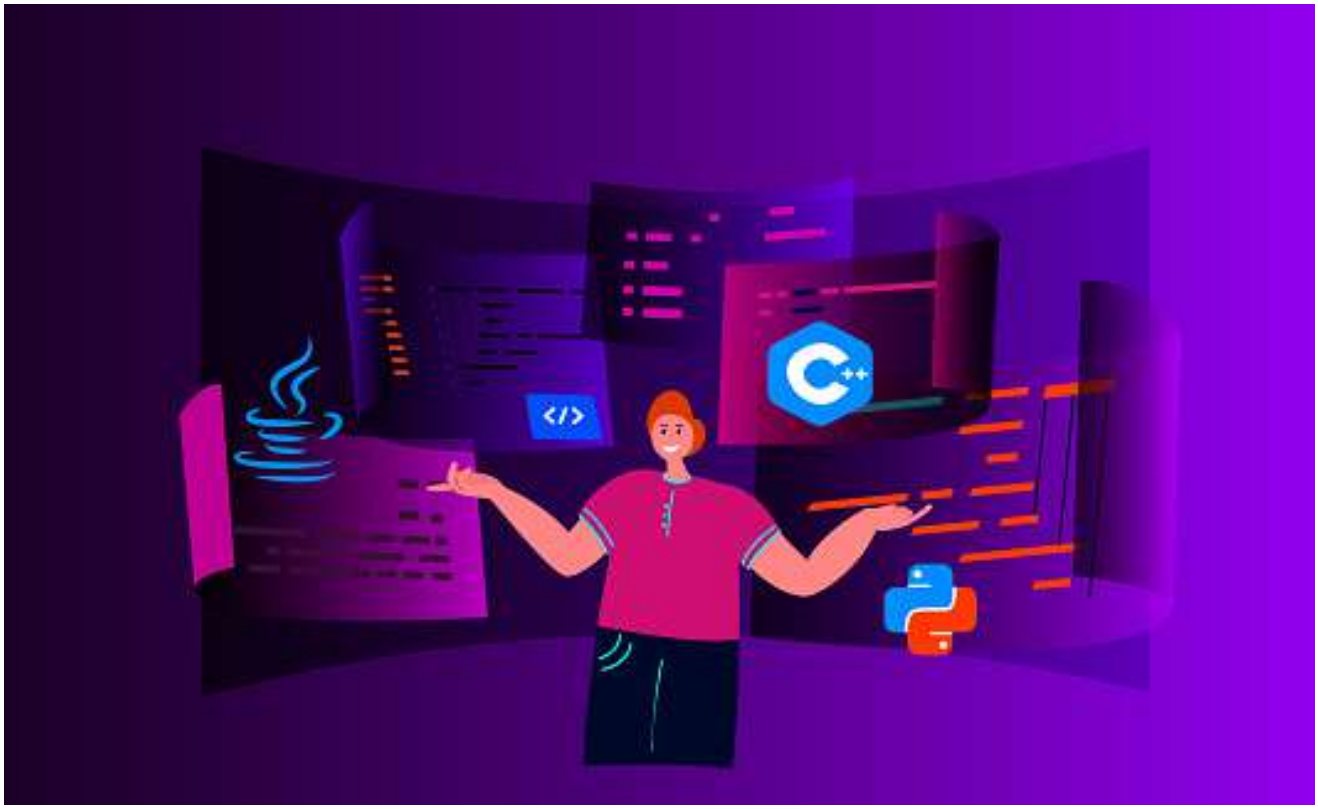
What is Competitive Programming? I'm sure every programmer has heard at least once about competitive programming.

In today's article, I will explain to you what that means.
Competitive Programming is my big passion. It was one of the first things I pursued on my coding journey. I've been on many competitions, online and in-person, and I'll try my best to bring you closer to this part of programming.

This article is your first step if you are interested in becoming a sport programmer. Competitive Programming has many benefits and is not as complicated as others see it.

Let's get started!

What is Competitive Programming?



> "Talk is cheap. Show me the code."
>
> **Linus Torvalds**

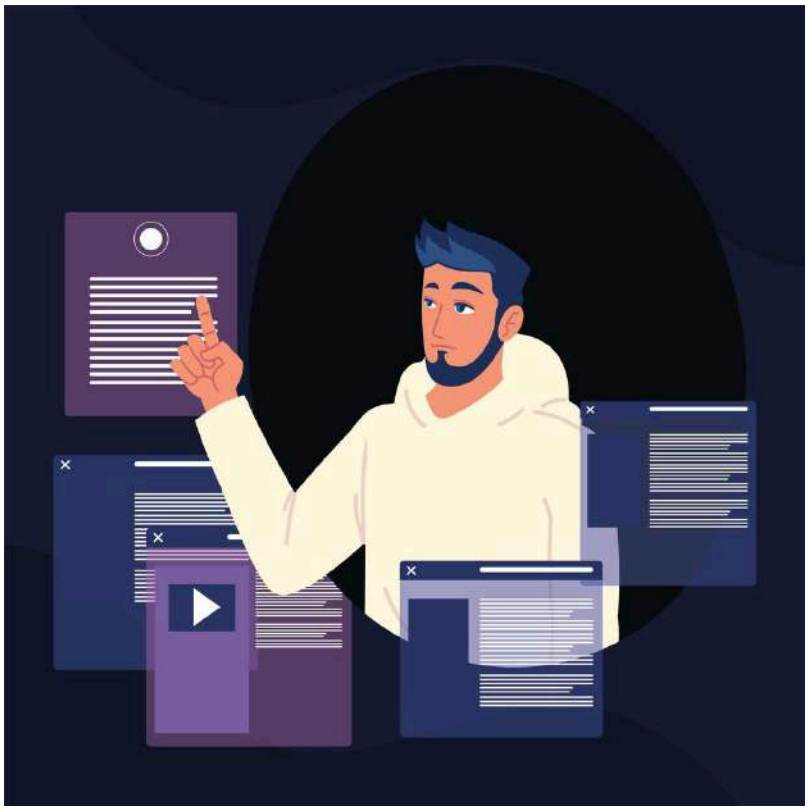
Competitive programming is a mind sport in which participants attempt to program according to given requirements over the Internet or a local network. **The contestants are called Sport Programmers.**

Several major software and Internet corporations have acknowledged and promoted compe e programming. Such as Google and Facebook.

Competitive programming is complex, and many famous software engineers have started this way. **Even though it is complex, it is not as hard as it seems. Two things you need to succeed in CP are desire and practice!**

CP is a huge opportunity to learn, connect and build a bright future for yourself in the Tech world.

What are competitions really like?



In a programming competition, the host presents the candidates with a collection of logical or mathematical challenges, commonly known as puzzles. Contestants build computer programs to solve each problem.

Devoting your time in CP is one of the best decisions you will make in your tech career. It challenges you to think logically. You learn fundamentals of programming that will 100% be useful for your career.

Committing to CP is the best way to start your programming journey. From the start, it teaches you how to think and solve problems like a programmer.

If you start your programming journey poorly, fast, and you don't understand your work, it will be harder for you to make up for those mistakes.

Types of competitions

Short-term and long-term competition formats are the two types of competitions. Each round of the short-term competition lasts anything between one and five hours. Long-term competitions might last anywhere from a few days and several months.

Most famous contests

Contests for Competition Programming you can find anywhere. **You can choose to compete online or to compete in person.**

- International Collegiate Programming Contest (ICPC) – one of the oldest competitions, for students of universities in groups of 3 persons each.
- International Olympiad in Informatics (IOI) – one of the oldest competitions, for secondary school students.
- Codeforces Round – typically two-hour contest, held every week.
- Facebook Hacker Cup – a competition held from 2011, provided and sponsored by Facebook.
- HackerRank – multiple competitions.
- Halite – An AI programming challenge sponsored by Two Sigma, Cornell Tech, and Google.

Competitive Programming Criterium

The judging in most contests is done automatically by host machines, often known as judges. Every contestant's solution is tested by the judge against a series of test cases.

There is also an option for online judging. **Online judges are sites where you can submit a solution to a problem in your preferred language by writing code in online IDE's or by writing code in your IDE installed on your system.**

Every competition has its criterium for judging. In some, it might be hidden test cases, sub-test cases or the whole problem or nothing. It all depends on the host.

Qualities of a Sport Programmer

To succeed in CP or the programming industry in general, you have to work on your skills. You have to practice, learn new things and have the desire to learn.

1. **You have to be patient and not give up**. When you get stuck on a problem, you need limitless patience. Trust me, you will get stuck. Prove your worth by excelling in hard times.
2. **You need to have the desire to learn**. We are only able to improve if we learn new things. By doing exactly that, we fix our mistakes, get a different look at the problem and prepare for the future.
3. **You have to be creative.** Think outside the box. Approach your problems from a different perspective and you will find the solution. There are always many solutions to the same problem.
4. **Passion**. As I discussed in my previous articles. Passion is Goal Fuel.

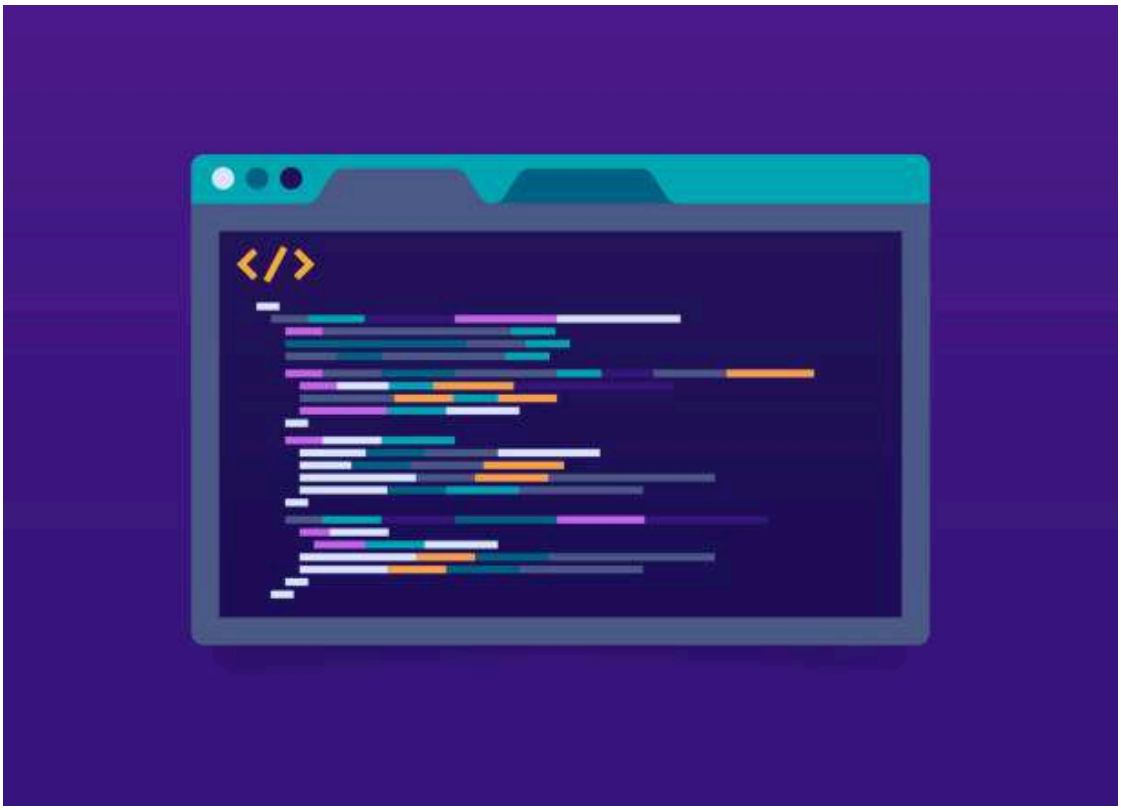
> "I'm not a great programmer; I'm just a good programmer with great habits."
>
> **Kent Beck**

Check out:

How to be a better programmer

Reaching your (coding) Goals to learn more.

Benefits of Competitive Programming



The benefits of Competitive Programming are endless. **If you are ambitious and passionate about programming, it will only help you with your career.**

If you think that after pursuing CP you are not using the concepts learnt in your everyday work, think again. Even though you are not using the same algorithm you used in CP, logic, error handling and problem-solving skills you are.

Here are some benefits:

- The ability to evaluate and appraise one's work more accurately.
- New approaches to problem-solving, as well as the ability to tackle increasingly difficult issues.
- Under pressure, the ability to provide outcomes.

- Advancement and preparation for a career.
- Work of higher caliber.

Resources

Udemy:

- Competitive Programming Essentials
- Competitive Programming
- Algorithms for Job Interviews and Competitive Programming

YouTube:

- Starting Competitive Programming – Steps and Mistakes
- What is Competitive Programming?

Books:

- Guide to Competitive Programming
- Cracking the Coding Interview
- Competitive Programming Handbook

My Articles:

- How to be a better programmer
- Reaching your (coding) Goals

Summary

In today's article, I introduced you to Competitive Programming. The purpose of this post is to introduce you to the concept of CP, motivate you to start and show you the benefits of this programming field.

As a programmer who attended CP Competitions, online and in-person, I recommend that you read this article before jumping into courses. It will prepare you for this bumpy but Goal worth fighting for.

This is part 1 of my Competitive Programming series, stay tuned for more!

Enjoy the process. – anonymous of LitCode.

Is Flutter easy to learn?

February 18, 2022

Categories: Flutter Beginner, Let's create

Tags: 100DaysOfCode, Coding, Flutter, Goals, LitCode, Mobile App Development

In today's article, I will be introducing you to Flutter and Dart!

This is the first post in my Flutter series.

I will explain to you what is Flutter and what is it used for. **But also we will be starting to build a simple, beginner-friendly Inspire App!** This is a great introductory project for beginners. We will be working on developing this project (and many more) throughout this series.

This article is PART 1 and the introduction to Flutter Widgets.

Mobile App Development is my big passion, and my goal is to teach you more about it. **Let's get started!**

What is Flutter?

Flutter is a set of tools that lets us make stunning apps for iOS, Android, the web, and desktop. Flutter's mission is to make developing for all devices easier.

It is:

- Free
- Open-source
- Backed by and originated at Google
- Fast because it compiles to truly native apps that don't use crutches
- Written in one place and compiled to a web app, iOS app and Android app

Flutter has a great community of ambitious developers ready to help whenever needed!

What is Flutter used for?

Google's mission with Flutter is...

> To build a better way to develop for mobile.
>
> Google

Flutter is an open-source framework for developing mobile applications for Android and iOS. It gives you a simple, powerful, efficient, and straightforward SDK for writing mobile apps in Dart, Google's programming language.

Check out: Flutter.

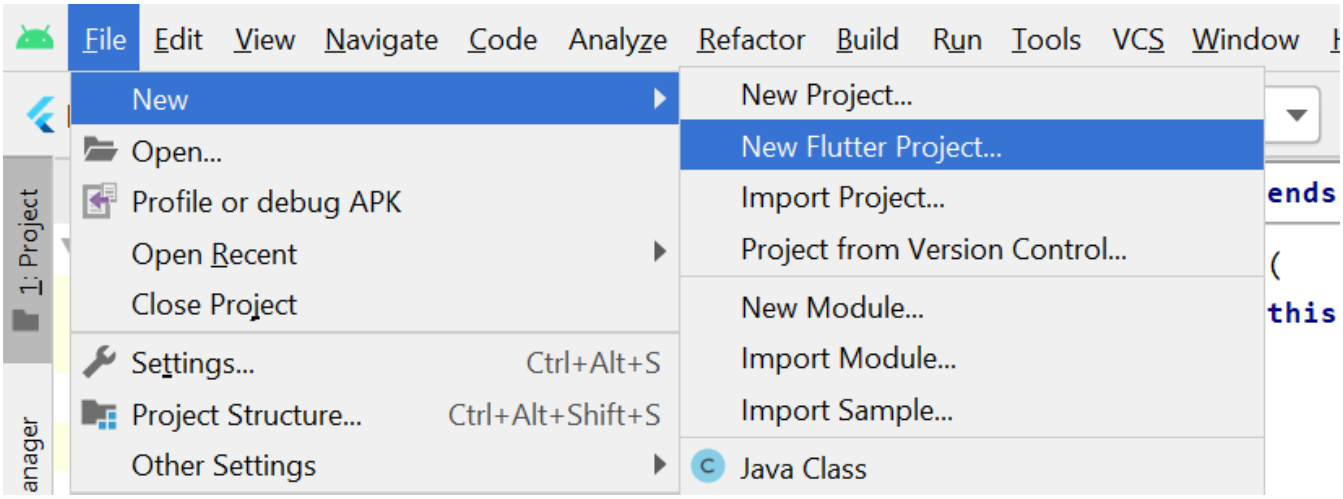
Make your first Flutter App

Let's start with your first Flutter Project – Inspire.
Inspire – app generates new quotes every time you press the **Get inspired** button. This is a great starter project for your Flutter journey. You'll learn the basics of Widgets, State and work on your logic skills!

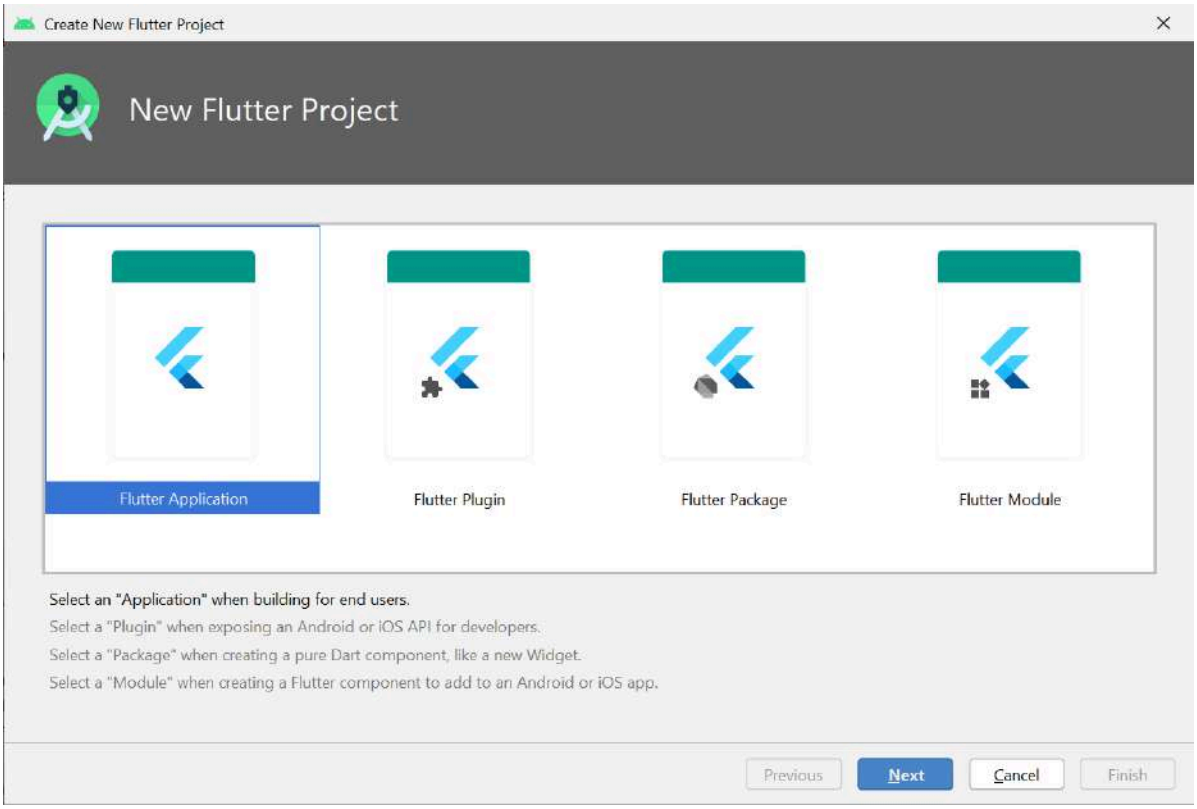
In this article we will create and customize our App's Widgets.

1. Creating a Flutter Project

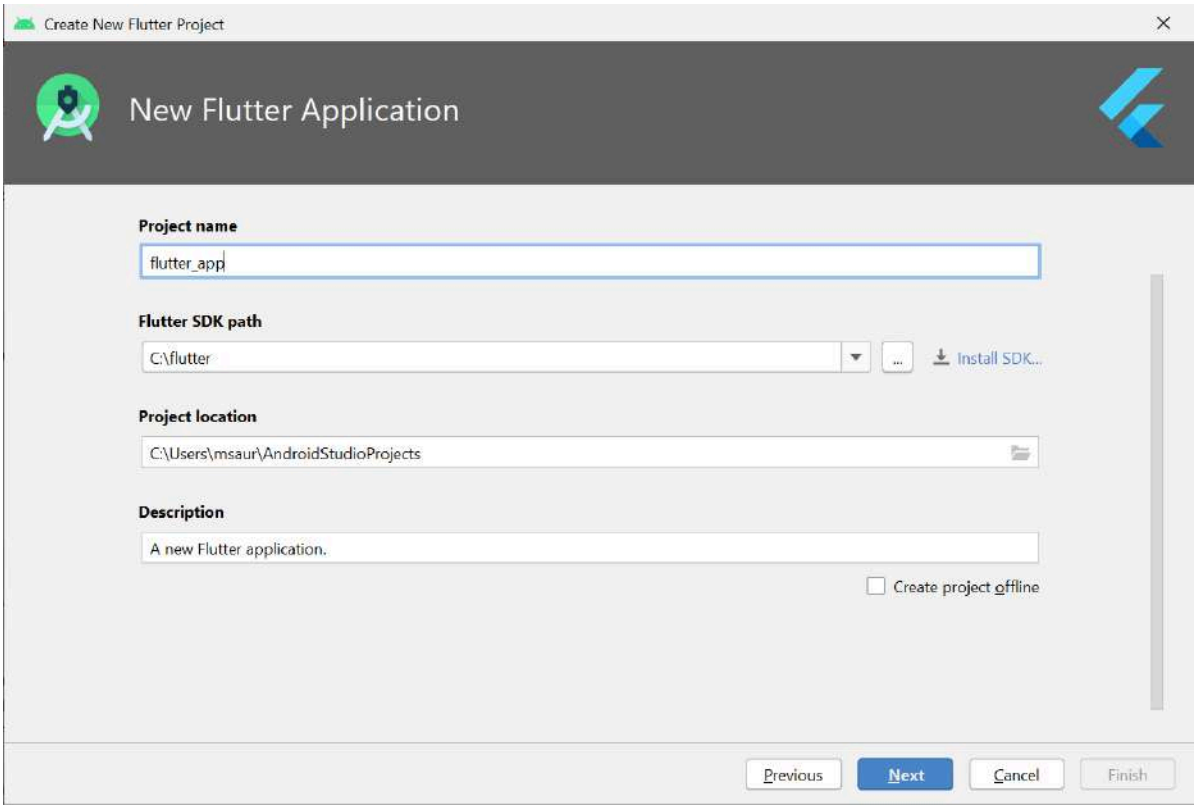
Go to File -> New -> New Flutter Project



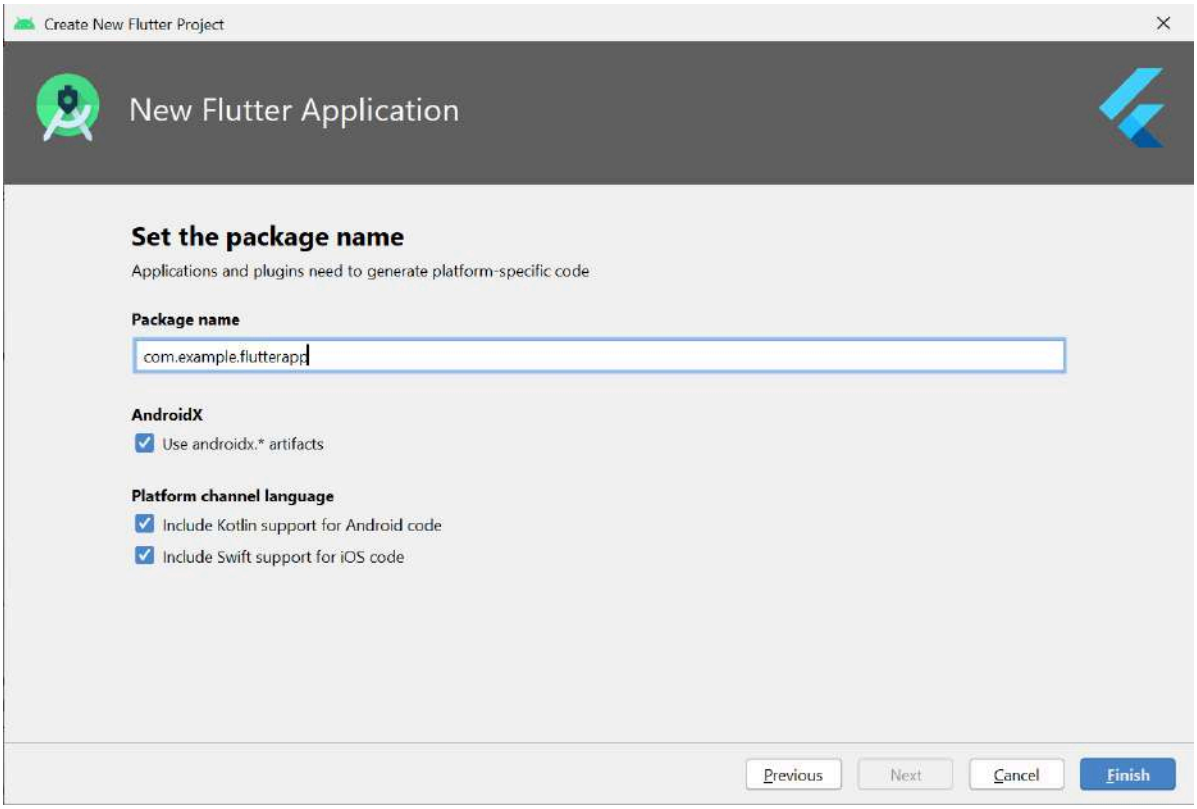
Select Flutter Application



Name your project whatever you want (flutter_app, inspire_app etc.)



In the package name field, you can type your organization name or just yourname.com



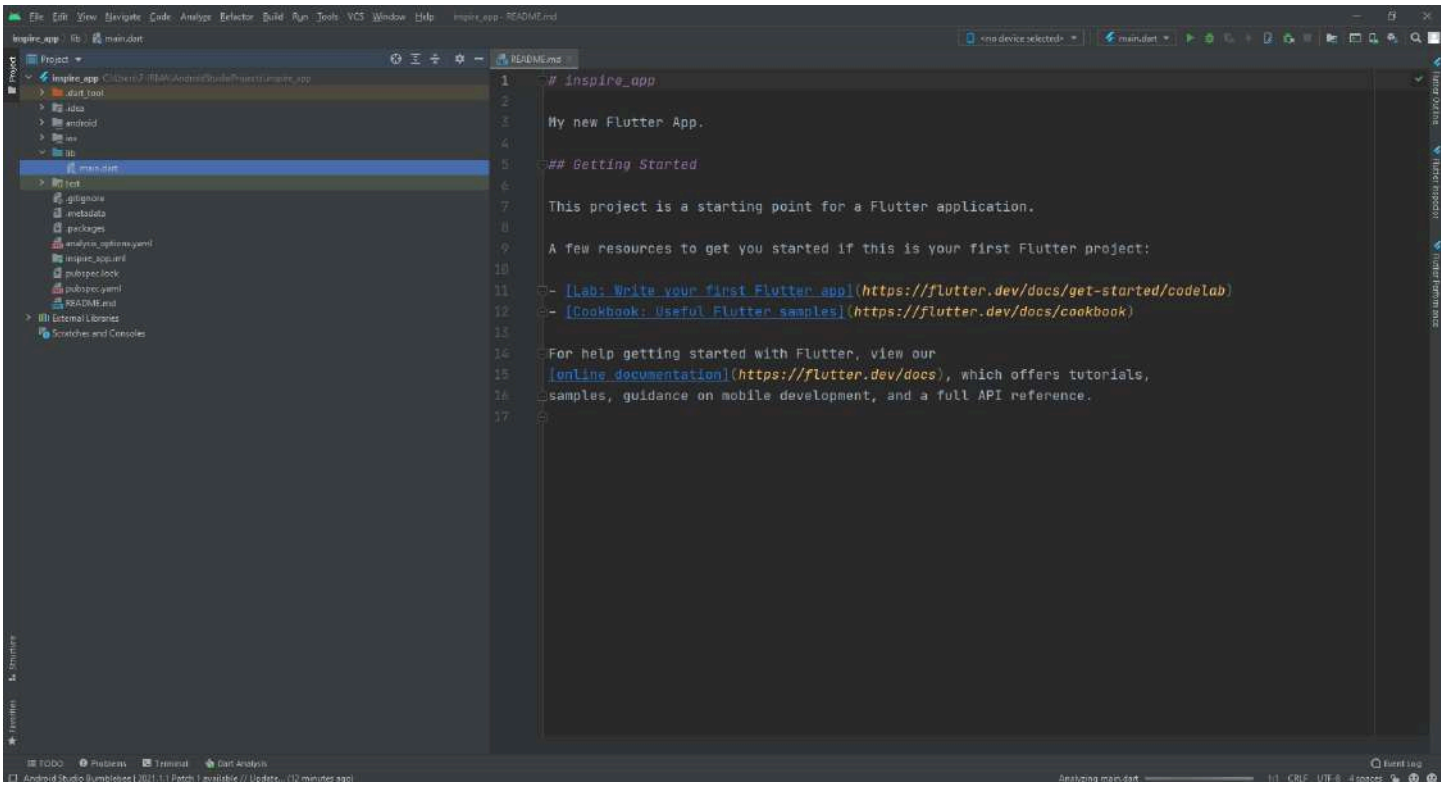
2. Project Structure

After creating a fresh Flutter project you will have these folders:

- android and ios – platform-specific parts of the project.
- lib – home of the dart source code. Here you build the app hierarchy.
- test – if you have unit tests.

And these files:

- pubspec.yaml – the project file for dart projects. Here we set project name, description, dependencies etc.
- .gitignore and README.md – important for developers that use git and GitHub.
- .metadata and .packages – configuration files you'll never open!



3. Running our App

There are multiple ways of running our App.

- You can hit the green “Play” button in Android Studio.
- You can also do it from the Command Line using “flutter run”.

Any time you save a change to your code, it is recompiled and a new version is loaded. **That is called hot reloading**. To hot reload you can press the lightning icon next to the “Play” button.

4. Main.dart and main() function

After creating your project, open main.dart located in lib folder.
You will have something like this:

```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // Try running your application with "flutter run". You'll see the
        // application has a blue toolbar. Then, without quitting the app, try
        // changing the primarySwatch below to Colors.green and then invoke
        // "hot reload" (press "r" in the console where you ran "flutter run",
        // or simply save your changes to "hot reload" in a Flutter IDE).
        // Notice that the counter didn't reset back to zero; the application
        // is not restarted.
        primarySwatch: Colors.blue,
      ),  // ThemeData
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );  // MaterialApp
  }
}

class MyHomePage extends StatefulWidget {
```

This is Flutter's Counter App project. **To start building our Inspire App, delete everything up to void main() as I selected it too.**

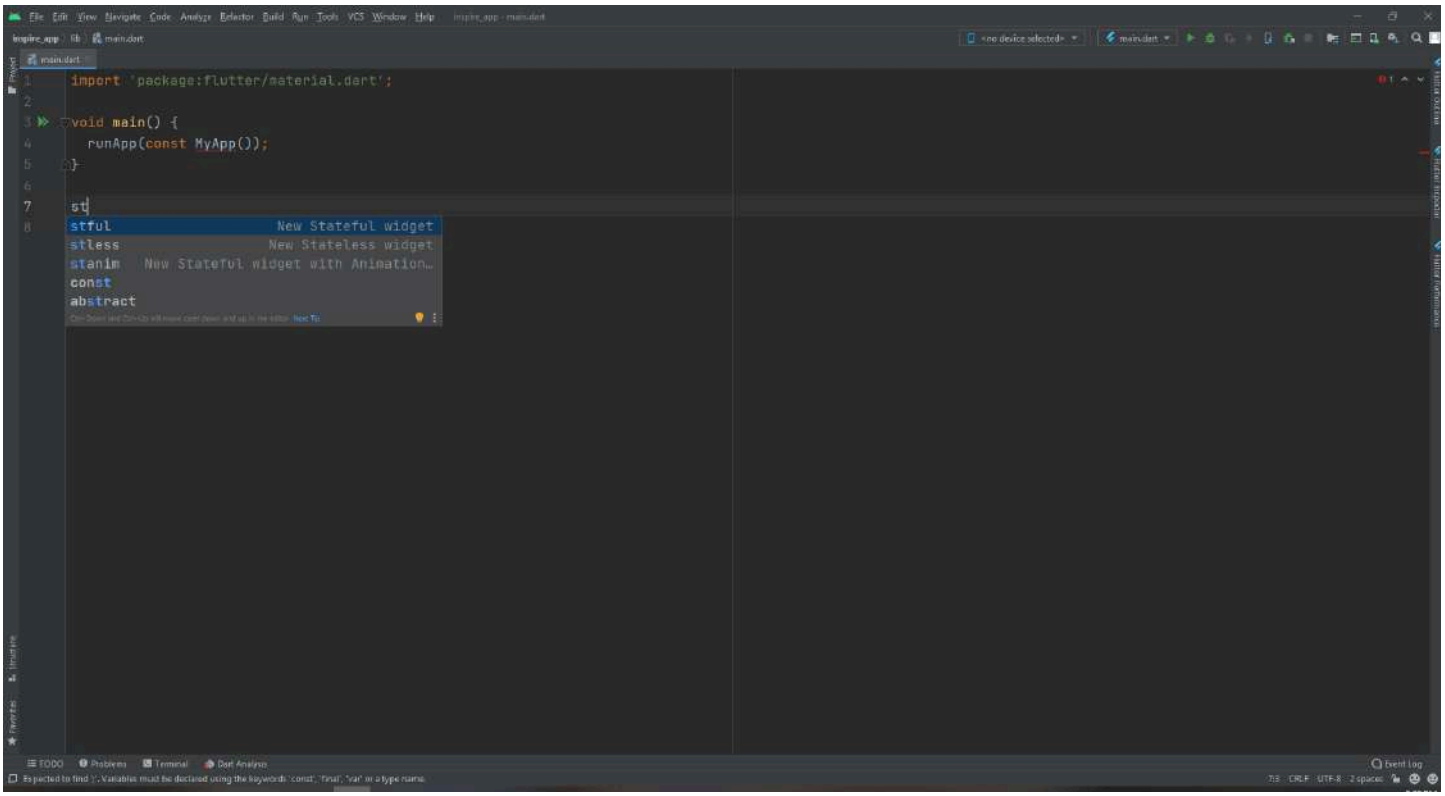
A main function is the foundation of a Flutter app. The function runApp() will be invoked by main(). The Root Widget is passed into runApp(). In the next part of this series, we'll go over this in more detail. **The most important thing to remember is that main.dart is your project's starting file, and main() function executes your whole app.**

5. Creating Stateful/Stateless Widget

Widgets in Flutter are building blocks of everything we create. There are different types of Widgets (Value, Layout, Navigation, etc.)

Stateless widget is the one that doesn't maintain it's State. A Stateful Widget does. State refers to data within the widget that can change.
For example, State can be changing the text displayed, colours or any change to the UI (what our user sees).

After deleting the Flutter Counter App, below main() type stful.
You're going to get a dropdown like this:



Press Enter and type the name of your Stateful Widget.

After entering your custom Widget name, also replace it with MyApp in main() function above.

```
import 'package:flutter/material.dart';

void main() {
  runApp(const InspireApp());
}

class InspireApp extends StatefulWidget {
  const InspireApp({Key? key}) : super(key: key);

  @override
  _InspireAppState createState() => _InspireAppState();
}

class _InspireAppState extends State<InspireApp> {
  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```

6. Material App Widget

The MaterialApp Widget generates your app's outer framework. Material App Widget is never visible to the user because its elements are hidden. It wraps the entirety of your app.

The inner-visible framework is created by the Scaffold widget. It lays out your app's visible architecture. (AppBar, body section, navbar, floating button, and so on.)
Scaffold can never be used without MaterialApp Widget!

```
import 'package:flutter/material.dart';

void main() {
  //replace InspireApp with MaterialApp()
  //Add InspireApp as home property of MaterialApp Widget
  runApp(const MaterialApp(home: InspireApp()));
}

class InspireApp extends StatefulWidget {
  const InspireApp({Key? key}) : super(key: key);

  @override
  _InspireAppState createState() => _InspireAppState();
}

class _InspireAppState extends State<InspireApp> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(); //replace Container() with Scaffold() Widget
  }
}
```

- We replaced Container Widget with Scaffold
- In runApp we replaced InspireApp with MaterialApp Widget and added InspireApp as home property of MaterialApp

7. Scaffold properties

Scaffold Widget has many properties; background color, appBar, body, floatingActionButton etc.

- To change the background colour of our app we use the background color property.
- To use an appBar we add appBar property, to use body add body property, etc.

```
import 'package:flutter/material.dart';

void main() {
  runApp(const MaterialApp(
    home: InspireApp(),
  ));
}

class InspireApp extends StatefulWidget {
  const InspireApp({Key? key}) : super(key: key);

  @override
  _InspireAppState createState() => _InspireAppState();
}

class _InspireAppState extends State<InspireApp> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.red, //property for changing the color
      appBar: AppBar(), //adding an AppBar widget
      body: Center(), //adding the widgets displayed in the body of App
    );
  }
}
```

8. Building the Body

- **Center()** Widgets centers it’s child Widget.
- **AppBar()** Widget creates an AppBar.

- **Container()** Widget is a parent widget that can hold several child widgets and manage them using width, height, padding, background colour, and other properties.
- **Column()** is a widget that displays its children in a vertical array.
- **CupertinoButton()** is a widget in flutter used to display iOS style button.
- **Text()** Widget allows us to display a string of text with a single line in app.

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';

void main() {
  runApp(const MaterialApp(
    home: InspireApp(),
  ));
}

class InspireApp extends StatefulWidget {
  const InspireApp({Key? key}) : super(key: key);

  @override
  _InspireAppState createState() => _InspireAppState();
}

class _InspireAppState extends State<InspireApp> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.red,
      appBar: AppBar(),
      body: Center( //centers its child
        child: Container(
          child: Column( //vertical array of widgets
            children: ,
          ),
        ),
      ),
    );
```

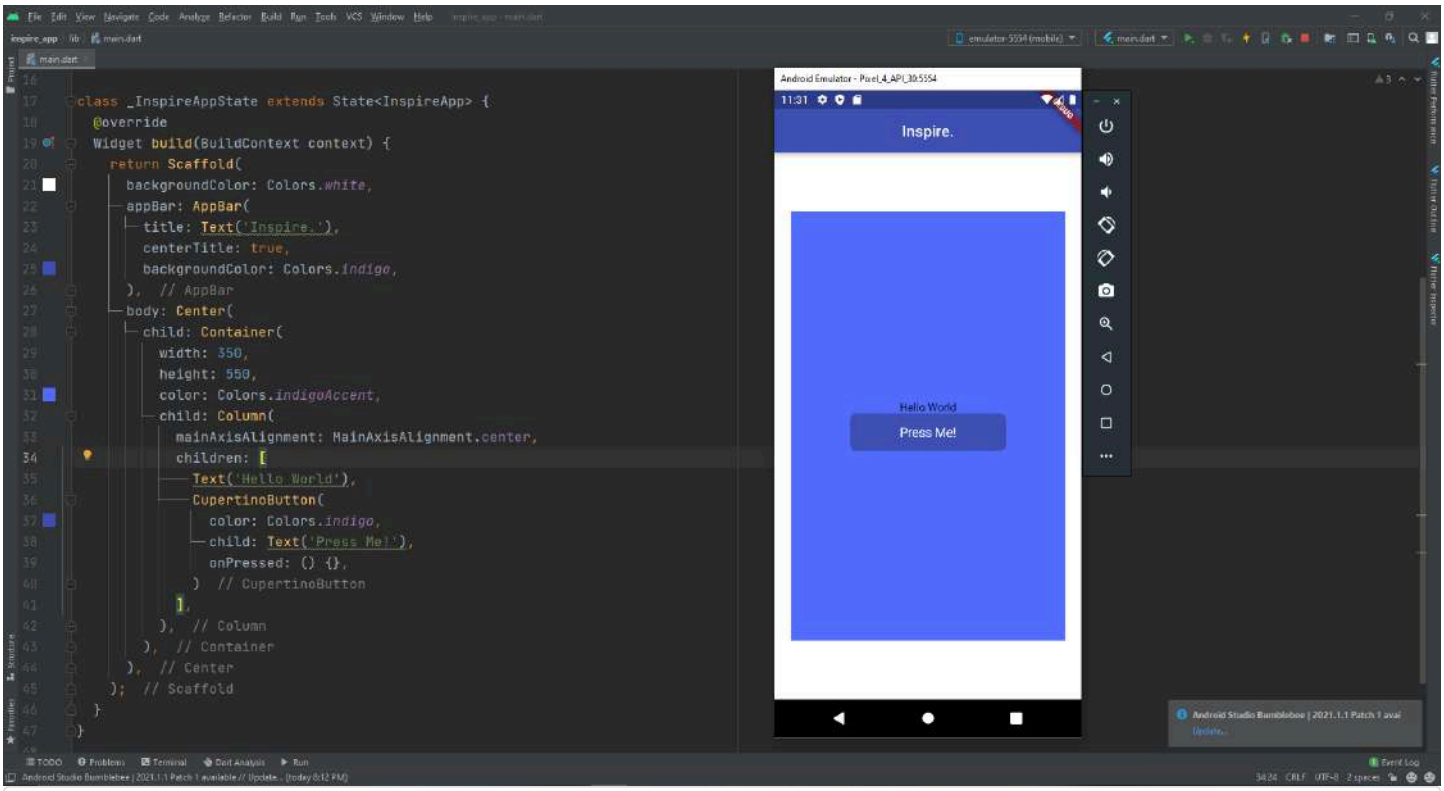
```
  }
}
```

9. Widget style properties

You notice that your App is not going to run, or change UI, by just adding different Widgets to it.

To style and customize your Widgets, we use Widget properties.

- **width** – set width of your Widget (in our case Container)
- **height** – set height of your Widget (in our case Container)
- **mainAxisAlignment** – align elements in your Column (or Row) in our case used to center elements of the Column Widget.
- **onPressed()** – a function that our CupertinoButton() widget has, when you press the button, this function will do something you tell it.
- **title** – set title of your AppBar (takes in Text Widget)




```
return Scaffold(
     backgroundColor: Colors.white, //from red to white
     appBar: AppBar(
        title: Text('Inspire.'),
        centerTitle: true, //center AppBar title
        backgroundColor: Colors.indigo,
      ),
```

```
     body: Center(
        child: Container(
          width: 350,
          height: 550,
          color: Colors.indigoAccent,
          child: Column(
            //all children of the Column will be aligned to the center
            mainAxisAlignment: MainAxisAlignment.center,
            children: ,
          ),
        ),
      ),
    );
  }
}
```

- To separate two Widgets you can use **SizedBox Widget.**
- It has **height** and **width** properties that you can use to accomplish this.

```
children: ,
```

10. Custom Colors (HEX)

There are not so many default Colors in Flutter. To add custom Colors we can use Color() widget and pass in our color.

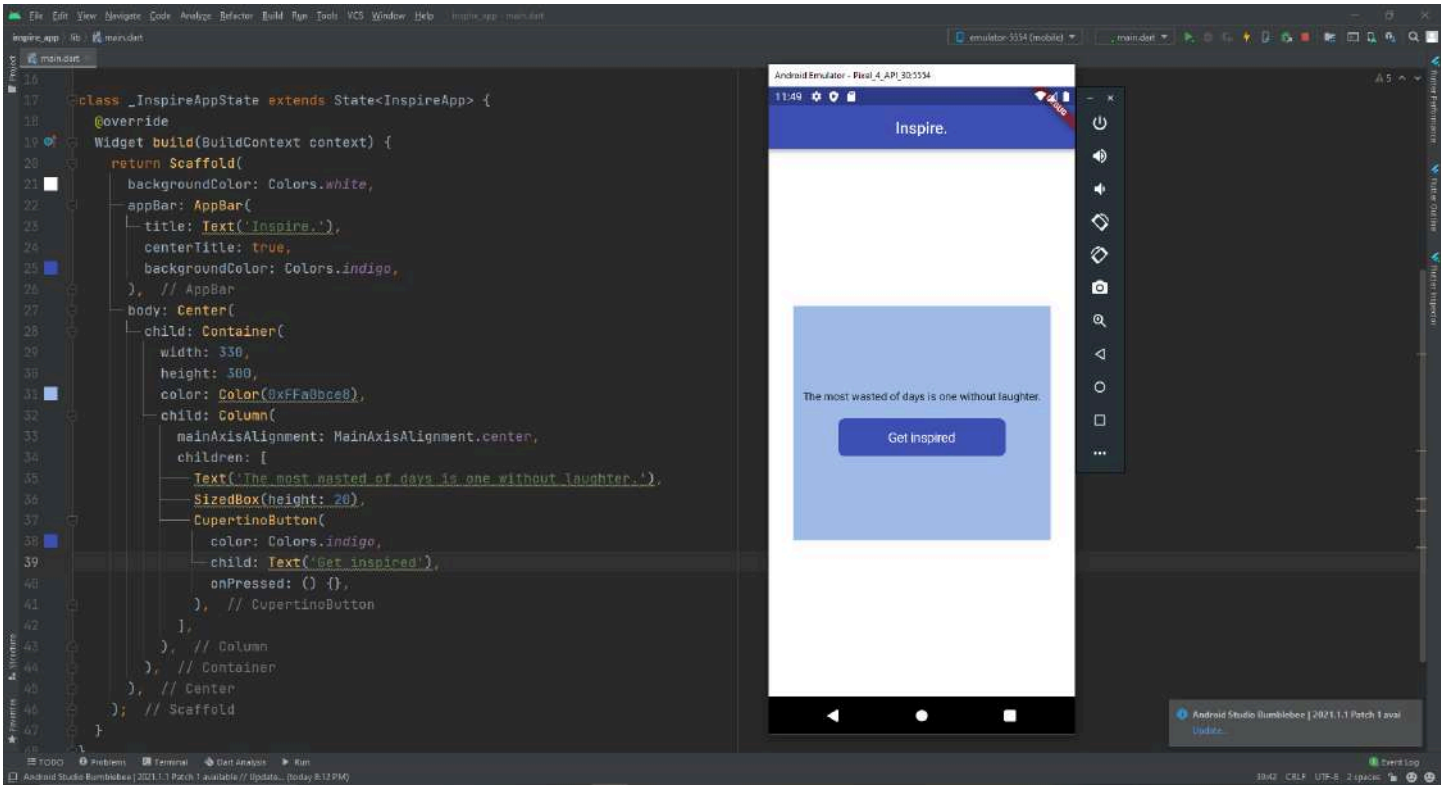
When adding HEX colors in your Color() Widget we pass in 0xFF (Color(0xFF)) and then your custom HEX code.

```
Center(
  child: Container(
    width: 330, //changed size
    height: 300, //changed size
    color: Color(0xFFa0bce8), //color using 0xFF and our HEX code
(a0bce8)
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: ,
```

```
      ),
    ),
  ),
```

Changes:

- Changed width and height values
- Added TEST quote instead of displaying Hello World
- Changed button message to Get inspired
- Added custom HEX color



Summary

In this article, we started building our Inspire App. We covered what is Flutter and it's usage but also the Basics of Flutter Widgets.

This is PART 1 in the Flutter series – Introduction to Widgets.
We explored Widget properties and made a fun, simple design for Inspire.
Currently, our App has no logic implemented and when our Button is pressed quotes don't change.

We'll tackle this problem and more in PART 2 of the Flutter series!
I hope this post was helpful for anyone diving into the Flutter World and World of App Development!

Happy creating! – anonymous of LitCode

5 Books that changed my Life

February 21, 2022

Categories: Let's read

Tags: 100DaysOfCode, Books, JustCode, LitCode, Motivation, Productivity

In today's article, I'll share with you 5 books that changed my life. I want to show you the true benefits of reading books. Not only did they change my everyday life, but also they enhanc y

productivity, changed my habits and thinking. **Books are powerful way to improve your programming skills too**.

Reading books is the no.1 habit of the most successful people in the world. You have the opportunity to become one of them! So why don't you?

Remember:

> "Books are a uniquely portable magic."
>
> Stephen King

Let's get started!

Why should you read books?

Once, I could barely motivate myself to read a page of a book. I would fall asleep after a first paragraph. **But after I pushed myself through that second paragraph, a new world opened for me.** If I still haven't convinced you to start reading books, let's look at the benefits of reading.

- **Gain knowledge** – this is the obvious one. No matter what kind of book you are reading, you have something new to learn. It might be a new word, term or a different way of thinking.
- **Improve communication skills** – by reading books, you grow your vocabulary. You will never find yourself in those awkward situations when you can't think of a word. You'll be able to communicate with anyone, anytime.
- **Enhances memory and focus** – keeping our brain fit is as important as having that summer body. One way to do it is to read books!
- **Be more creative** – reading opens a new world of possibilities for you. Soon after you start reading, ideas will start popping in your head!

How can books change your life?

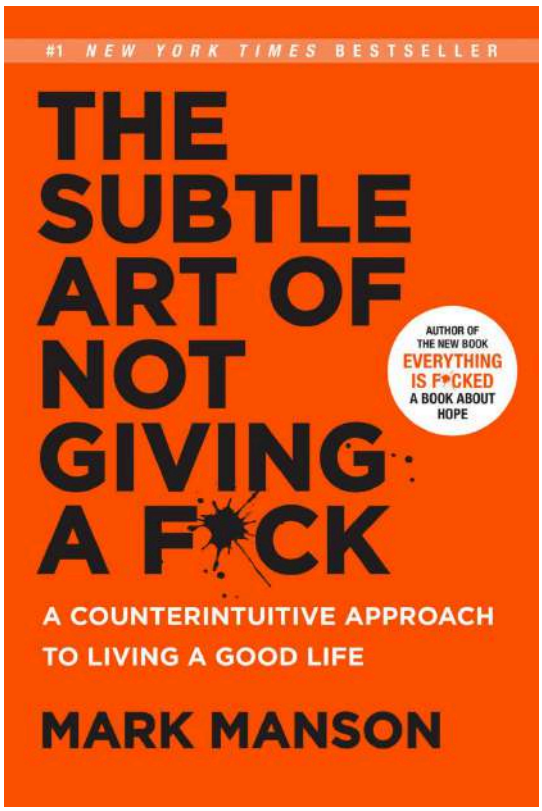
After watching a great movie, you start behaving like the main character. The same thing happens after reading a great book. Not only are books better than movies, but they also change the way you think, make you more open-minded and enhance your creativity!

After reading a great book, you become a new person. Now you are not being LIKE the main character, you ARE the main character in your own story. But this main character is better, it has a new way of thinking and the way it looks at the world.

5 Books that changed my life

Now let's dive into Five books that changed my life. These are not the only ones that changed my life. But they currently have the most influence on me.

1. The Subtle Art of Not Giving a F*ck



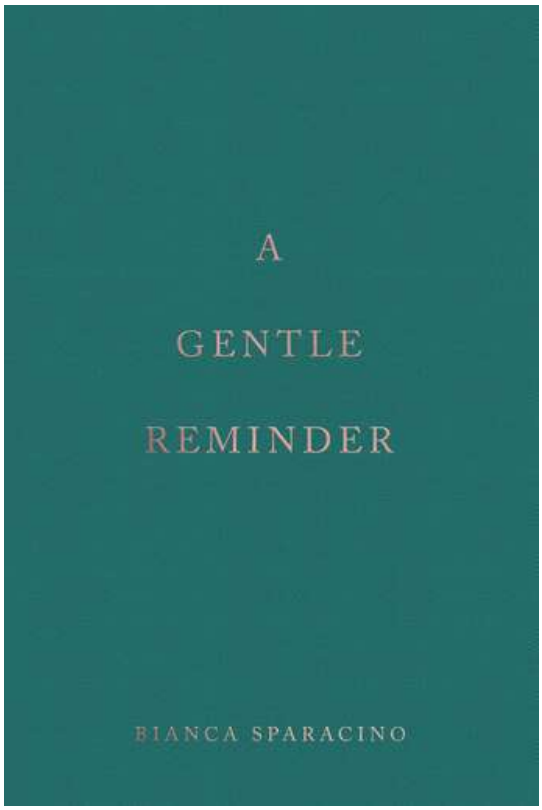
- **ABOUT THE BOOK:**

The Subtle Art of Not Giving a F*ck by Mark Manson is a book that contradicts self-help books by encouraging you not to try, to say no frequently, and to embrace negative thinking. **Being okay with being yourself and caring about something more important than pain is what it means to not give a f*ck.**

- **MY EXPERIENCE:**

I bought this book at the perfect time in my life. It opened my eyes and called me out in many ways. At first, I was skeptical, but reading it was definitely worth it. It changed my every day life ar thinking but also my work experience.

2. A Gentle Reminder



- **ABOUT THE BOOK:**

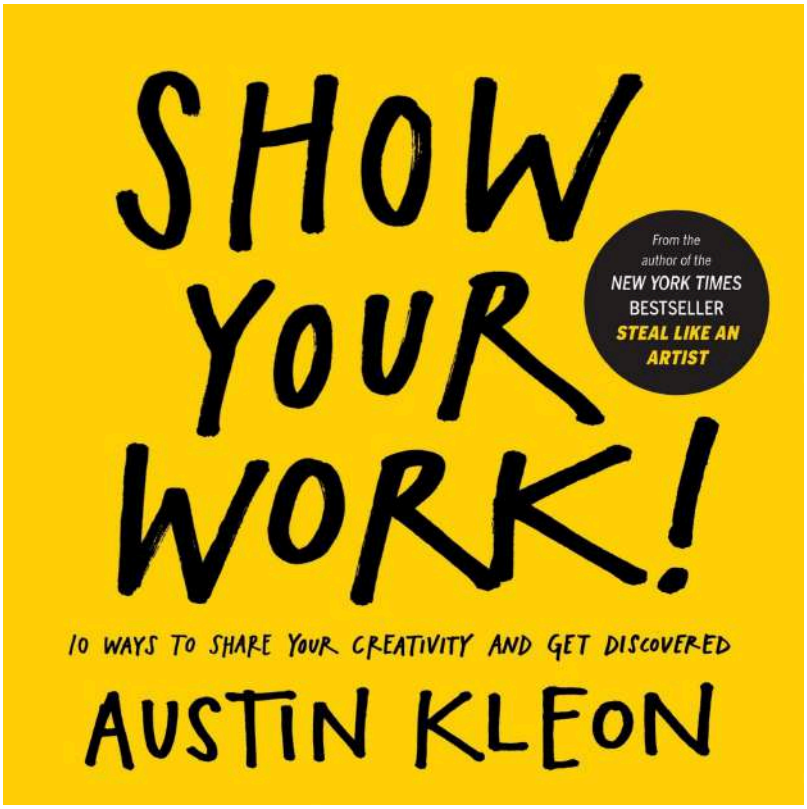
A gentle reminder, on the days when you feel light in this world and when the sun rises a little later. For when your heart is overflowing with hope and you're learning how to repair it. A gentle reminder for when you start to believe in goodness, and you need words to reassemble your shattered parts.

And more...

- **MY EXPERIENCE:**

This book will heal you in many ways. It made me more open-minded about life's experiences and challenges. **This book is all your past, current and future thoughts on paper.** A must-read!

3. Show Your Work



- **ABOUT THE BOOK:**

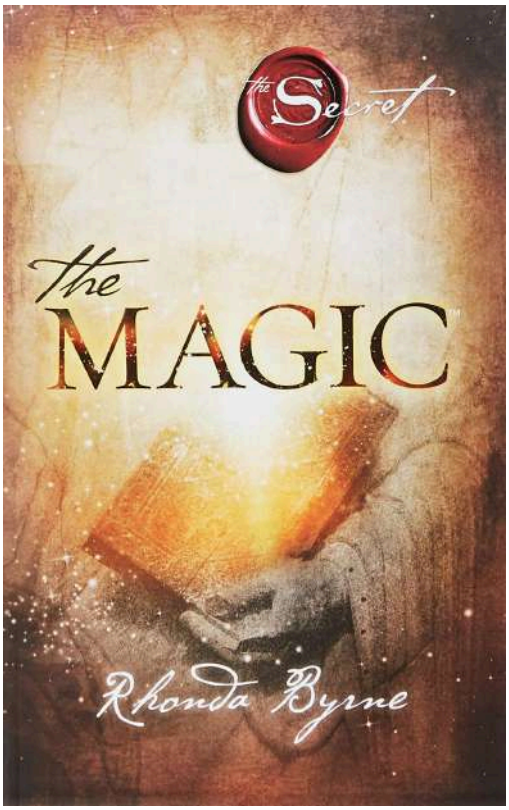
To learn, discover your audience, and develop your work, this book encourages you to share your creative work and process. It offers tips on documenting your process, training others, and improving your storytelling skills so you may share the work you love with your community.

- **MY EXPERIENCE:**

This book motivated me to start my Blog! Not only did it share amazing advice, but it showed me the benefits of showing my programming work. I read it just before I started creating my Blog.

It changed my professional and day-to-day life for the better!

4. The Magic



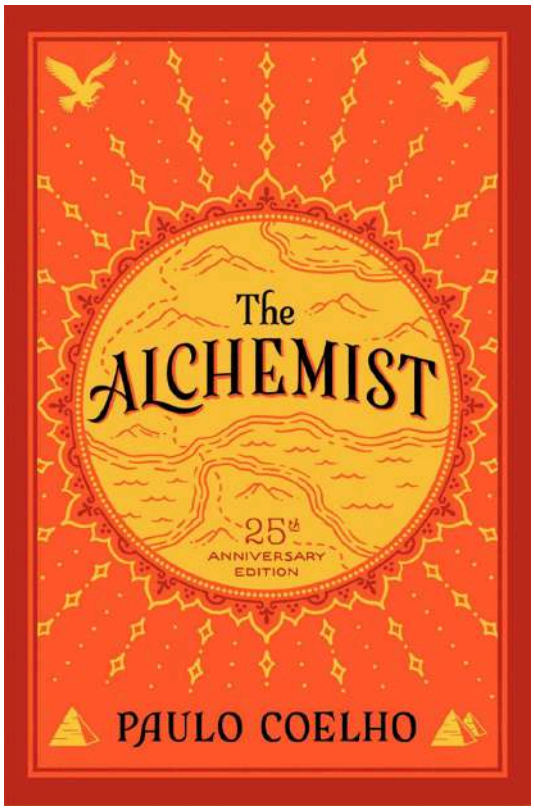
- **ABOUT THE BOOK:**

Rhonda Byrne offers 28 magical techniques that will teach you how to utilize gratitude to enhance your health, money, career, and relationships, as well as how to manifest your tiniest requests and greatest goals.

- **MY EXPERIENCE:**

The first time I read this book was a couple of years ago. But this is the book that needs to be read over and over again! **It showed me that the only way to receive in the future is to be thankful for what we have in the present.**

5. The Alchemist



- **ABOUT THE BOOK:**

Santiago, an Andalusian shepherd boy, travels from Spain to the Egyptian desert in search of a treasure buried in the Pyramids. He's a plain sheepherder who also happens to be an adventurous explorer. He is guided by his dreams as well as "signs" from his surroundings.

- **MY EXPERIENCE:**

This is a classic, but that doesn't make it less worth it. **It motivated me to start and focus on my goals and dreams. But also calmed me and made me believe.** A must-read!

Summary

In this article, I shared with you 5 books that changed my life. I hope these books will also change your life for the better.

Believe me, once you start reading, you will start living in a new world, like a new, improved person. I shared why you should start reading and how books can change your life.

Happy reading. Enjoy the process – anonymous of LitCode.

Flutter Development – part II

February 23, 2022

Categories: Flutter Beginner, Let's create

Tags: 100DaysOfCode, Coding, Flutter, JustCode, LitCode

In today's article, we will continue with developing our Inspire App.

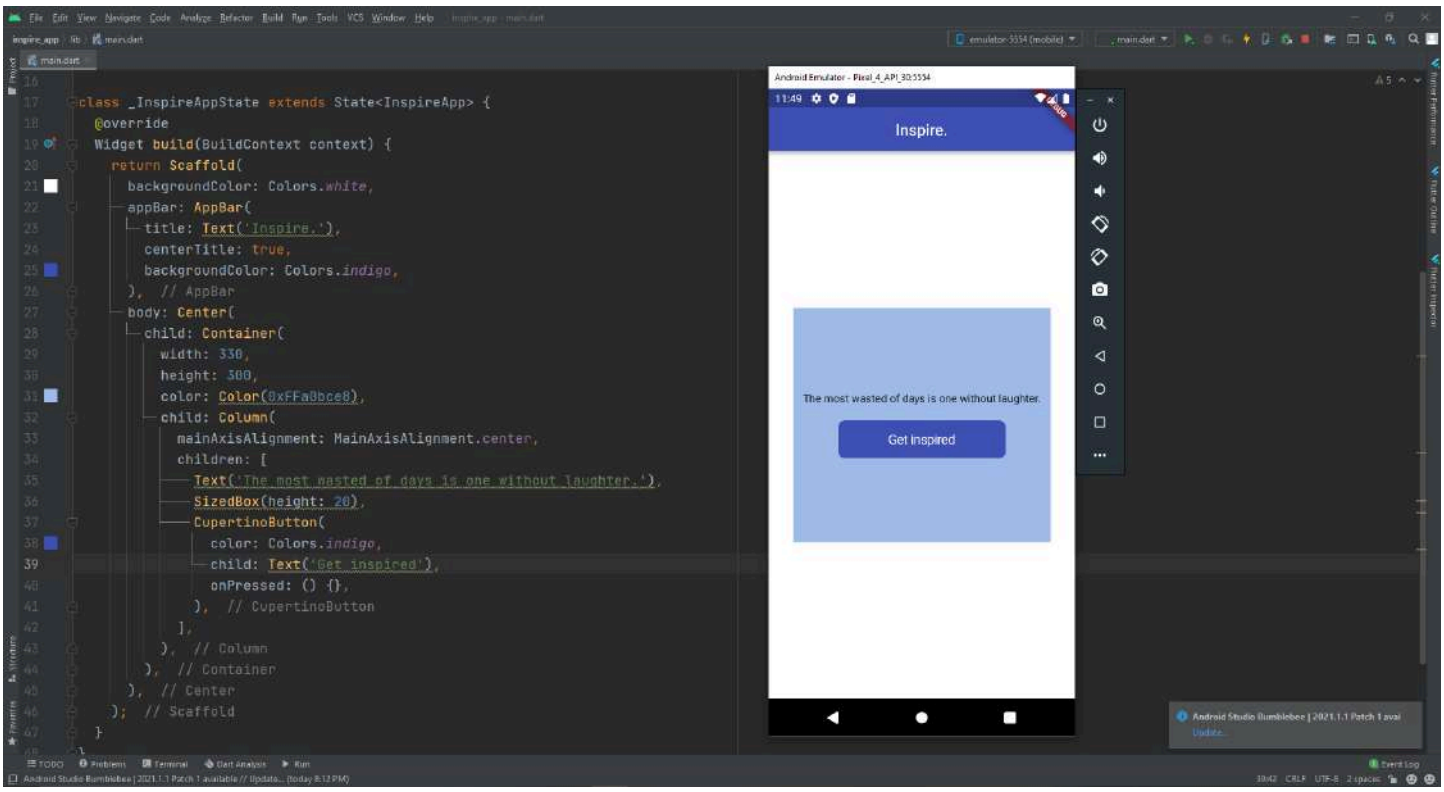
This is part II of LitCode's Flutter series. Make sure to check out the part I at: Introductio Flutter.

After finishing this tutorial, you are done with Level 1 of Flutter Development! In the next part, we will be starting with Level 2, introducing many new concepts as we move on.

Today we are building the simple logic of Inspire.

Remember: every part of our App will evolve during this Flutter series, these are not the final touch-ups on Inspire App!!

What did we build?



Final screenshot of Inspire. in part I

In the last part, we made the skeleton of our app and got introduced to Flutter Widgets.

Discussed topics:

1. What is Flutter?
2. Creating a Flutter Project
3. Project Structure
4. Main.dart and main function
5. Stateful and Stateless Widgets
6. Material App
7. Scaffold
8. Building the body
9. Styling properties
10. Custom Colors (HEX)

What are we building today?

Our primary focus for today is building the main logic of our App and introducing new Flutter concepts.

We will cover:

1. Variables in Dart
2. Lists in Dart
3. Libraries – Dart math
4. setState
5. Changing the Text displayed

1. Adding a quoteNumber variable

A variable is a "named space in memory" where data is stored. It serves as a container for program data.

The syntax for creating Variables in Dart looks like this:

```
datatype variable_name = value;
```

There are different **data types** in Dart:

- **Int, double, num** – for storing whole numbers and floating-point numbers.
- **String** – for storing characters.
- **Booleans** – holding true or false values.
- **Lists** – ordered group of objects.
- **Maps** – set of key-value pairs.

*you can also use widgets to specify which kind of content will be stored, more on this later in the series.

Now let's create quoteNumber variable. This variable holds our current quote number.
If the quote number equals to 1 that means our current quote displayed is at second position in our quotes List. (created in the second step).

Why second why not first?
In almost every programming language, counting starts from 0. That means that the first

element is at position 0.

```
class InspireApp extends StatefulWidget {
  const InspireApp({Key? key}) : super(key: key);

  @override
  _InspireAppState createState() => _InspireAppState();
}

class _InspireAppState extends State<InspireApp> {
int quoteNumber = 0;
//we set our quoteNumber variable to 0 so it starts counting from the
//first element in our quote list
```

2. Adding a simple List of Quotes

An array is a type of collection that is often used in programming. Arrays are represented in Dart as List objects. **A List is essentially a collection of objects that are arranged in a certain order.**

The syntax for adding Lists in Flutter looks like this:

```
List<datatype> list_name = [];
//or
List<datatype> list_name = [];
```

When creating Lists in Dart, we can specify the total number of elements this list will have or let Dart do that.

Example:

```
List<String> my_list = ;

List<String> my_second_list = ;
```

Now, let's add List of Quotes to our main.dart file. You can add as much quotes as you wish. My quotes List has 7 elements.

```
class InspireApp extends StatefulWidget {
  const InspireApp({Key? key}) : super(key: key);

  @override
  _InspireAppState createState() => _InspireAppState();
}

class _InspireAppState extends State<InspireApp> {
  int quoteNumber = 0; //variable that will hold current quote number

  //let's create a list of quotes type String
  List<String> quotes = ;
```

3. Libraries in Flutter – dart math library

Many developers have written some tools that you can use in Flutter apps. **They are referred to as Libraries, and all we have to do is import them and use them in our code.**

Today we will use dart math library. It contains different mathematical constants, mathematical functions and a few classes.

It is important to us because it allows us to show a random quote every time a button is pressed.

```
//1. math library
import 'dart:math';
//2. provides widgets used for iOS development
import 'package:flutter/cupertino.dart';
//3. provides widgets from Material App Design
import 'package:flutter/material.dart';
```

After importing the math library now we need to create an object to use it's Random() class**.** **Random() class generates random bool, int, or double value.**

To use it, we create a variable that will hold the class. After we can use that variable to access the Random() class properties. Such as nextBool(), nextDouble() and more.

*more on classes and objects in Level 2. The only thing you need to understand is that: to use the function that enables us to generate a random number, we must create a Random() class object and store it in a variable.

```
class InspireApp extends StatefulWidget {
  const InspireApp({Key? key}) : super(key: key);

  @override
  _InspireAppState createState() => _InspireAppState();
}

class _InspireAppState extends State<InspireApp> {
  int quoteNumber = 0; //variable that will hold current quote number
  Random randomNumber = Random(); //var that holds the object of Random
class

  //let's create a list of quotes type String
  List<String> quotes = ;
```

4. setState()

setState() sets properties and makes changes in the UI. To use setState() our widget must be Stateful. When you know your UI will change (text displayed, colours, images... etc.) use setState.

In our App, we want our text to change every time a button is pressed. To accomplish that we use setState.

```
child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: ,
          ),
```

To change the value of the quoteNumber to a random every time a button is pressed we use the nextInt() function. In the parenthesis of this function, we pass in the max number. The random numbers generated won't be larger than the max.

```
child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
```

```
            children: ,
          ),
```

To access the nextInt() function we use randomNumber variable.

5. Changing the Text displayed

One last step:
In the Text Widget, we need to display our quote List element at quoteNumber index. So to access a specific element of the List we use [].

We need to display an element of the quote List created at the random index. To accomplish that we type:

```
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: }'), //changing the text displayed
              SizedBox(height: 20),
              CupertinoButton(
                  color: Colors.indigoAccent,
                  child: Text('Get inspired'),
                  onPressed: () {
                    setState(() {
                      quoteNumber = randomNumber.nextInt(7); //random
logic
                    });
                  })
            ],
          ),
```

Our App looks like this:

https://litcode.net/wp-content/uploads/2022/02/inspire_app.mp4

Summary

Today we worked on the primary logic of our app. We got introduced to lists, variables, data types, libraries…and more! **Our main focus was finishing the app functionality.**

I hope this Post thought you more about Flutter!

In part III, we are starting Level 2 of Flutter Development.

We will cover:

- Classes and Objects
- Functions (Methods)
- Code refactoring
- Images in Flutter
- Introducing new Widgets
- …

Check out: Introduction to Flutter.

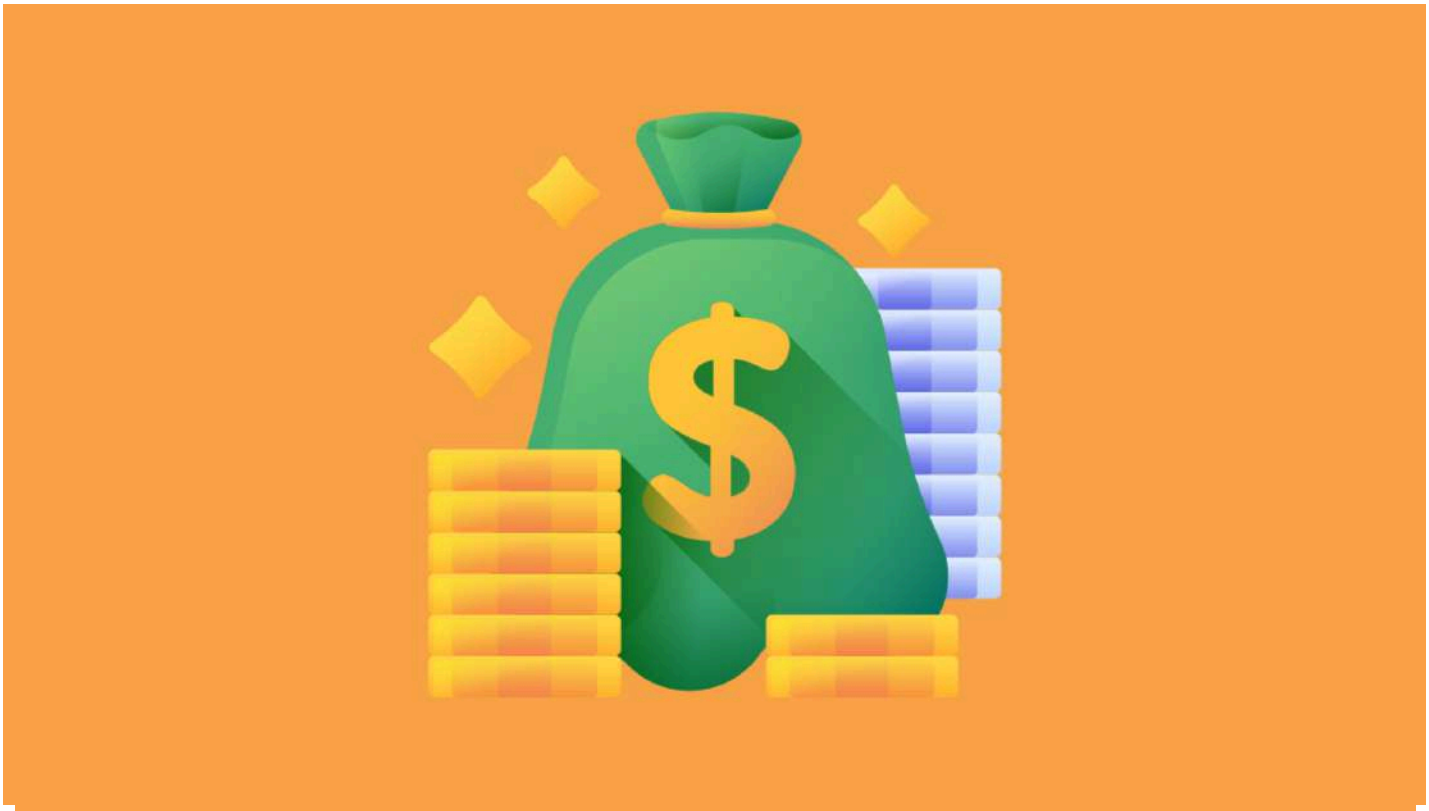
Stay tuned for more! Happy creating. – anonymous of LitCode.

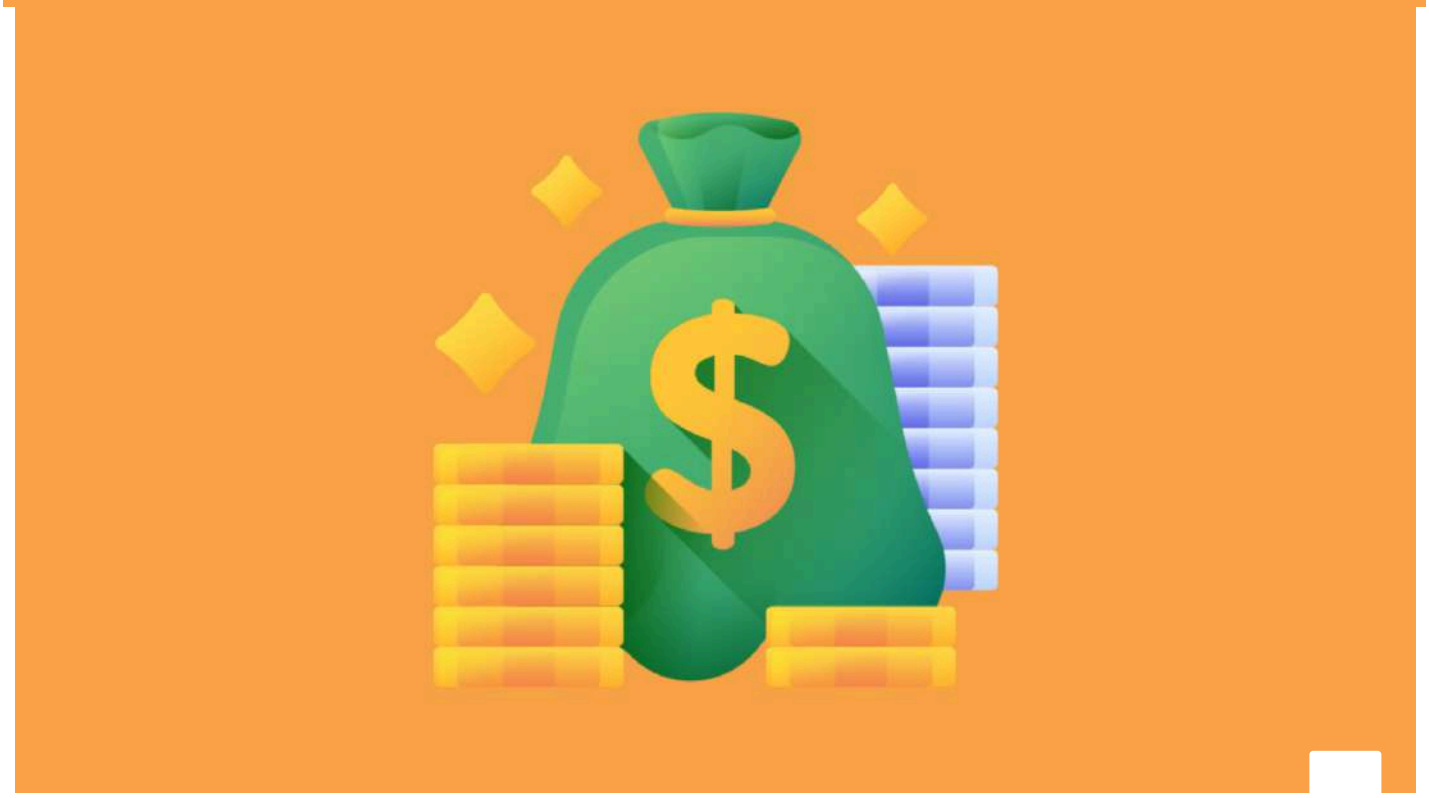
Is it hard to earn money as a programmer?

February 28, 2022

Categories: Let's learn, Let's read

Tags: 100DaysOfCode, Coding, LitCode, Programmer, ProgrammersLife





How to earn money as a programmer? You are not the only one who has asked this question. We are all trying to find ways to excel in our jobs. One way we feel appreciated is by actually earning money.

In today's article, I'll share 5 ways to earn money as a programmer. I hope the information in this article will make your journey easier. In the programming world, besides working at your full-time job, developers use one of these 5 ways to make a side income.

Let's get started!

Why is Showing your Work important?

There are many benefits of sharing your work with the world. To get recognized, you must start from somewhere.

Recognition boosts your confidence and motivates you. After sharing your work for the first time, you will be open to learn from your mistakes.

Trust me, you'll make mistakes, but that is all part of the process. By sharing your work, you get the opportunity to connect with other developers. But also with influential people in the tech world.

To succeed, you must put yourself out in the world.

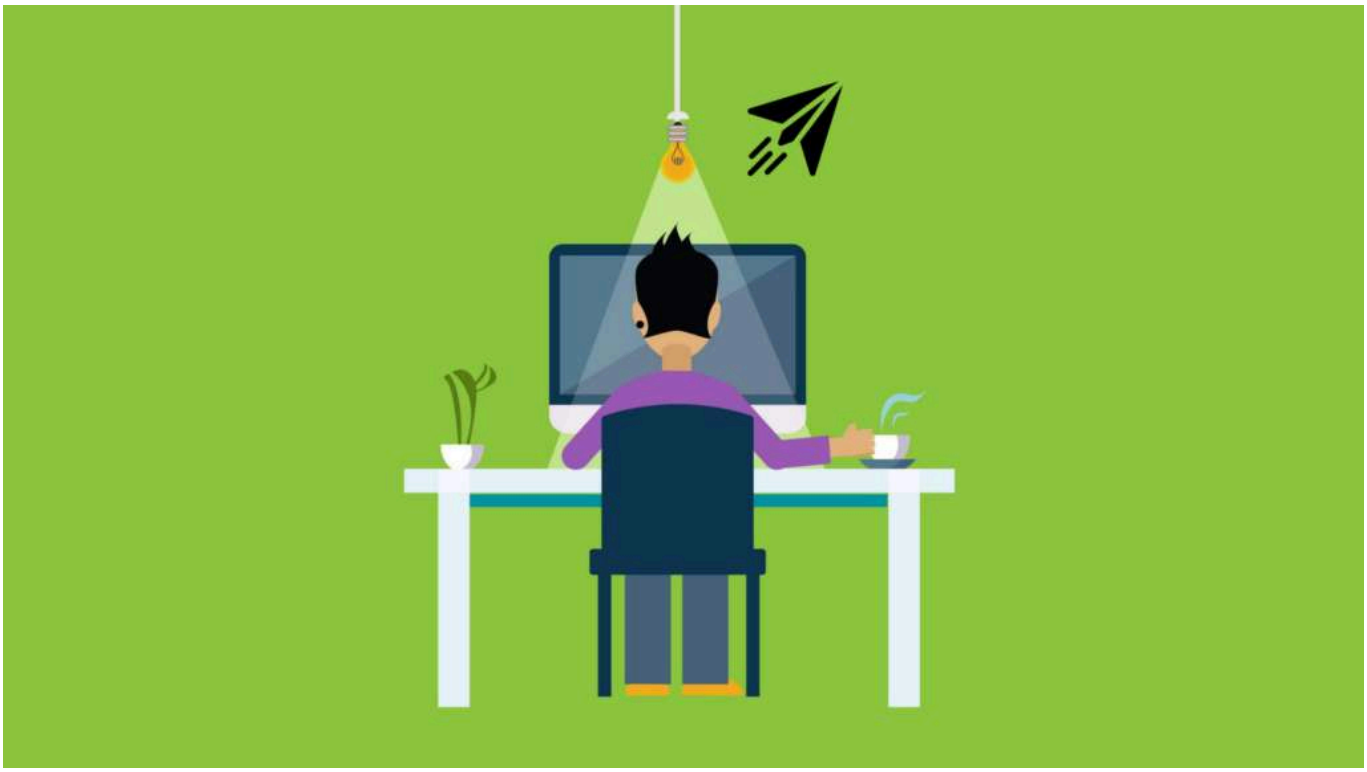
5 ways to Earn Money as a Programmer

Programming can be tough. After you start a new project or search for a new job it might feel impossible to succeed. But it is not. It is crucial to stick even in the hard times.

Remember:

> Many of life's failures are people who did not realize how close they were to success when they gave up.
>
> Thomas A. Edison

1. Start Freelancing

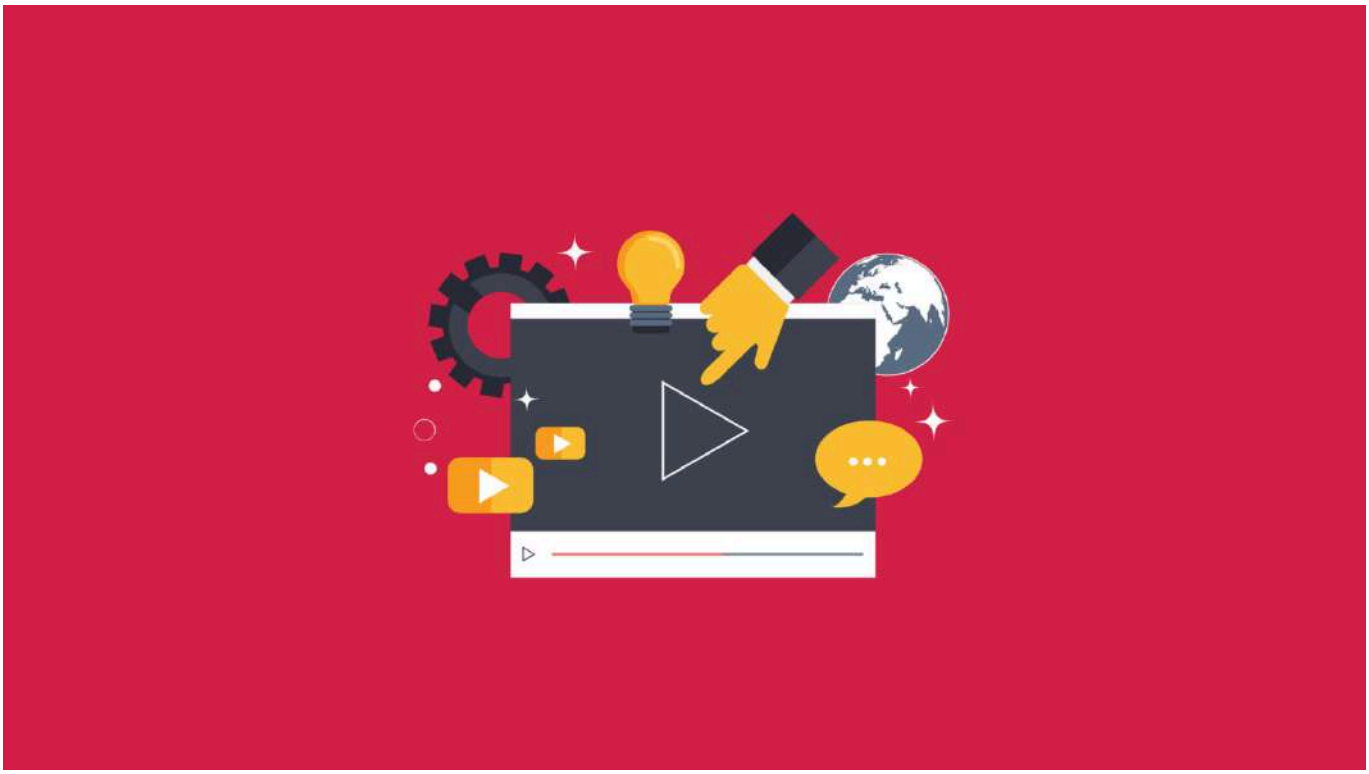


A freelancer is a self-employed worker who is paid on a per-job or per-task basis, usually for short-term projects. You get freedom, flexible work hours and a better work balance.

There are so many developers working as freelancers. But it's also common to work on your full-time job and still have a freelance side hustle. **It is all about organizing your time.**

Freelancers start by applying to sites like Fiverr or Gun.io and present their expertise to attract customers.

2. Create a YouTube Channel



Starting a YouTube Channel is a great way to earn income. Creating your YouTube Channel is completely free.

Whether you are thinking about making YouTube your full-time job or a side hustle, the benefits are endless. It pushes you outside of your **comfort zone** and you can become **a source of inspiration** for many.

By teaching people about your work you become a master at what you do. When you learn a concept to teach someone about it you excel in it.

3. Start a Tech Blog

Starting a Blog is one source of passive income. You can earn money only by blogging, but also you can attract your customers by showing your work on your Blog.

In the process, you **learn new things** and **improve your writing skills**. Writing, explaining your projects and coding concepts makes you understand your work better.

You connect to other developers, gain an audience and get the opportunity to become an expert at what you do.

If you enjoy programming, why not write about it?

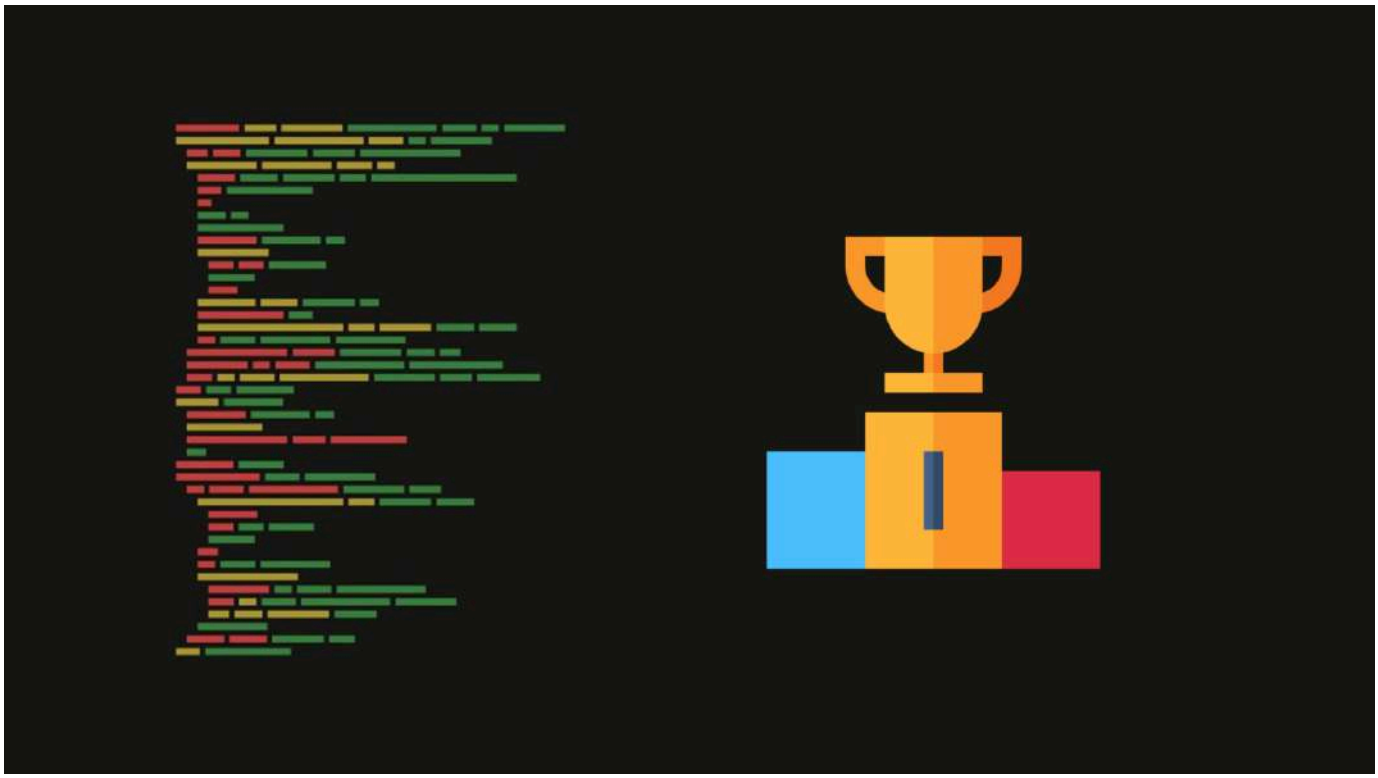
4. Develop Mobile Apps

You can create mobile apps by yourself or with a group of developers. Either way, it is a great way to improve your coding skills and has great money earning potential.

It pushes you to be **creative** and **think outside the box**.
You must understand that not all of your apps will have Instagram's or Twitter's success. But that is not a reason not to try at all.

Even though this post teaches you how to earn money, **money is not the only reason people start new projects and risk everything.**

5. Start competing



Coding contests such as the ACM ICPC, Monthly Cook-off challenges, and others are a terrific way to make money. **If you rank at the top, you get the opportunity to get recognized and win great prizes.**

Competing is complex and requires you to understand your code to every single detail. **But each of the ways I listed today requires the same.**

You can learn more about Competitive Programming here.

Resources

Freelancing sites:

- Fiverr
- Toptal
- Gun.io
- WeWorkRemotely

You can check out my Posts:

My Post on CP: Competition Programming

My Posts where I develop in Flutter: Introduction to Flutter & Developing with Flutter II

Summary

Today we discussed 5 ways to earn money as a programmer:

- Freelancing
- Creating a YouTube Channel
- Starting a Tech Blog
- Developing Mobile Apps
- Trying out at Competitions

I hope this article was helpful and that it showed you **the value of becoming a programmer**. You actually can work from anywhere.

Check out the related post: Why should everyone learn to code?

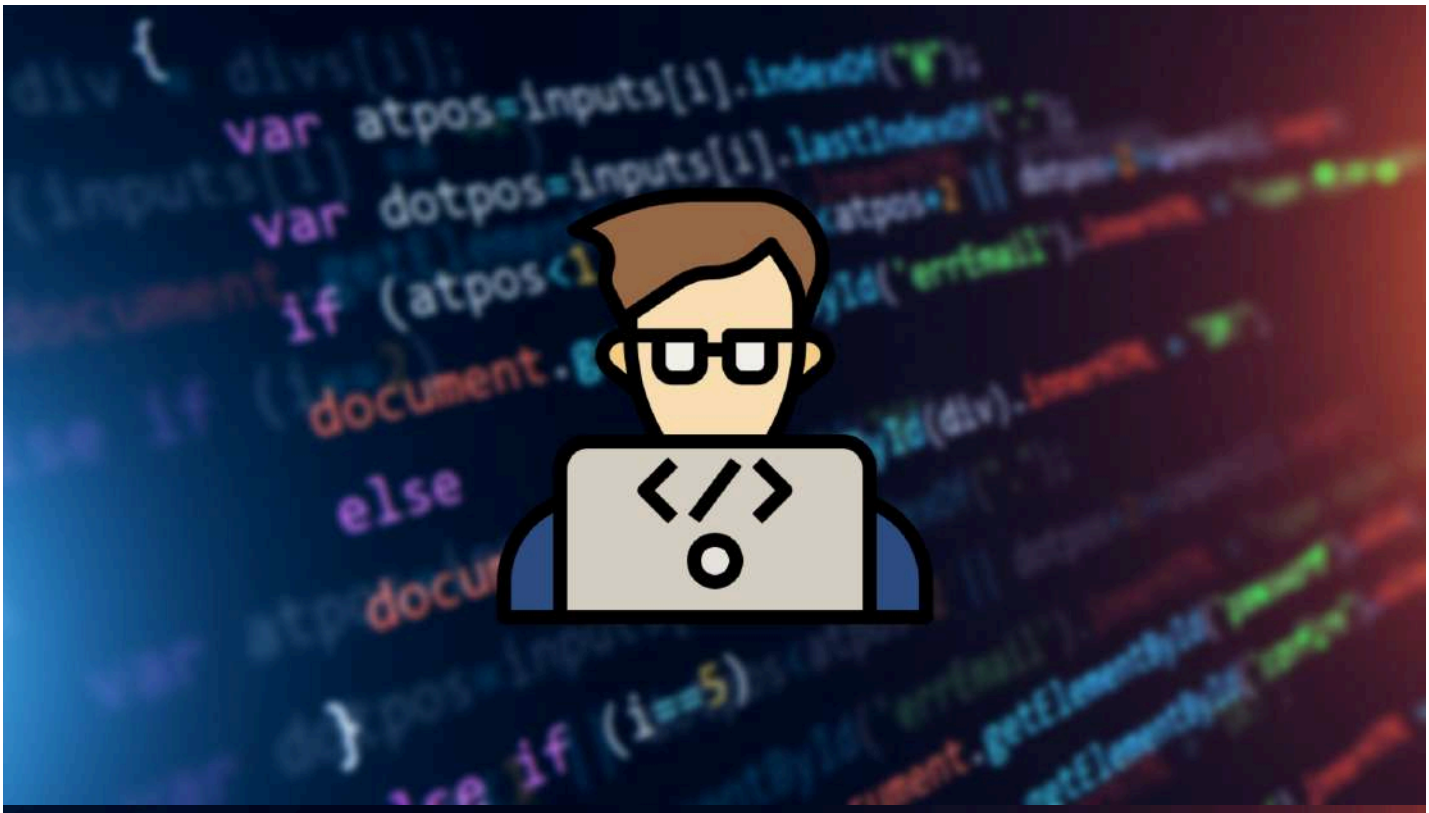
Happy creating! – anonymous of LitCode

Coding Contests for Beginners

March 3, 2022

Categories: Let's learn, Let's read

Tags: 100DaysOfCode, Coding, JustCode, LitCode, ProgrammersLife



Welcome to Part II of LitCode's Competitive Programming series. In today's article, I'll discuss 7 best ways to prepare for CP Competitions.

Without preparation and practice, there is no way anyone can succeed in this field. CP is all about consistency and overcoming challenges, more on this later in this article.

If you haven't already, check out the Part I – What is Competitive Programming article on my Blog.

Now, let's get started!

7 ways to prepare for a coding contest

Let's talk:

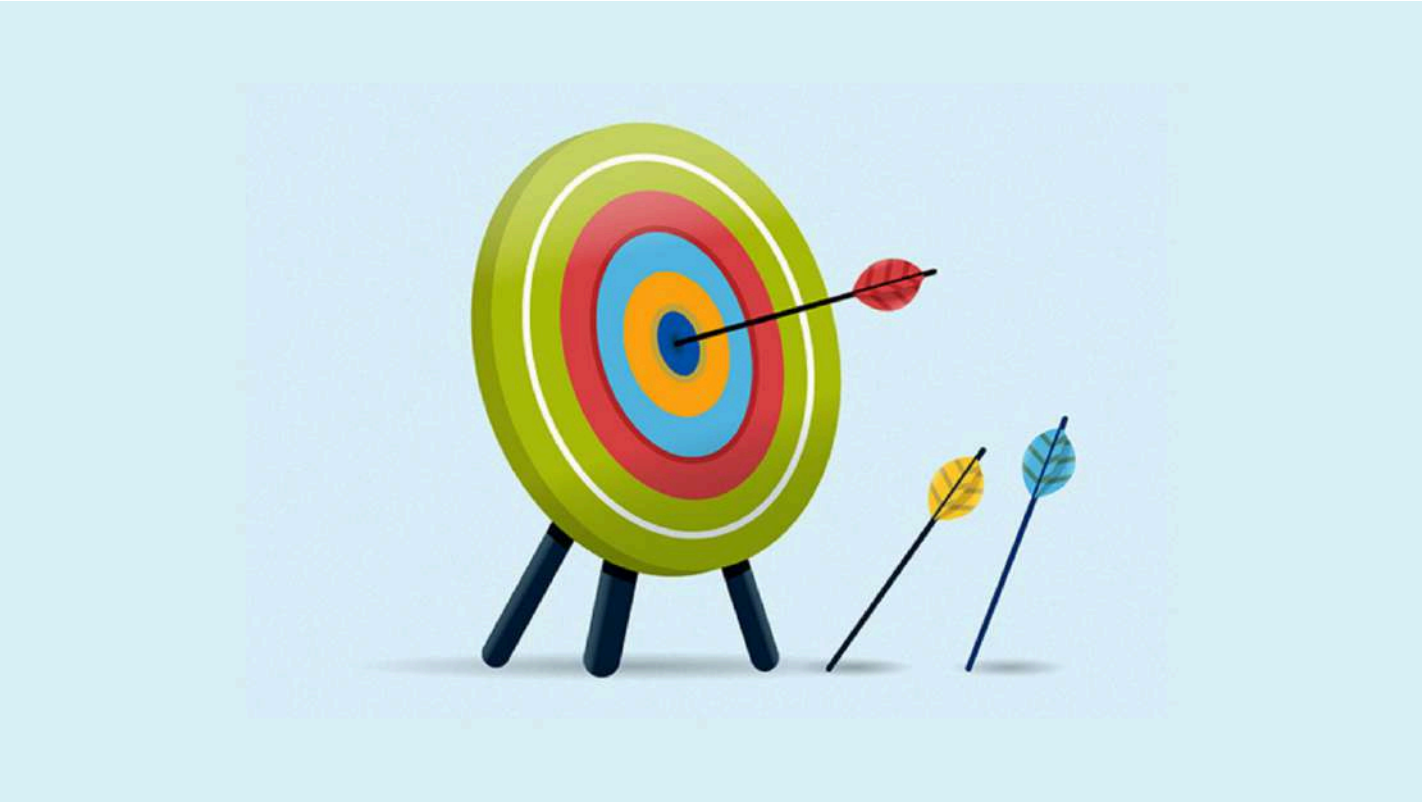
- Mastering the basics
- Practice
- Learning new concepts
- Dividing your problem
- The correct way of solving problems
- Practicing online and connecting
- Asking for help

1. Master the basics

Why is mastering the basics important?
Whether we discuss chess, football or coding, mastering the basics sets you for success. Without a good foundation, there is **no progress** and **no improvement**. Later success is determined by your ability to set a good foundation for your work.

The best way to compete is by selecting a specific programming language and completely focusing on it. You learn the syntax, data structures and basic algorithms. If you learn an algorithm in Python, you will know it in C++.

2. Practice is the key



As I mentioned in my previous posts, **practice is the key to success.**
Being a programmer requires practicing your skills every day for several hours. When you commit to practicing, you set yourself for success. Practice helps you learn faster and makes you more confident in your work.

In Competitive Programming, that means going through tons and tons of exercises. Going through different kinds of problems, different difficulty levels and applying your knowledge.
Talent is nothing without practice.

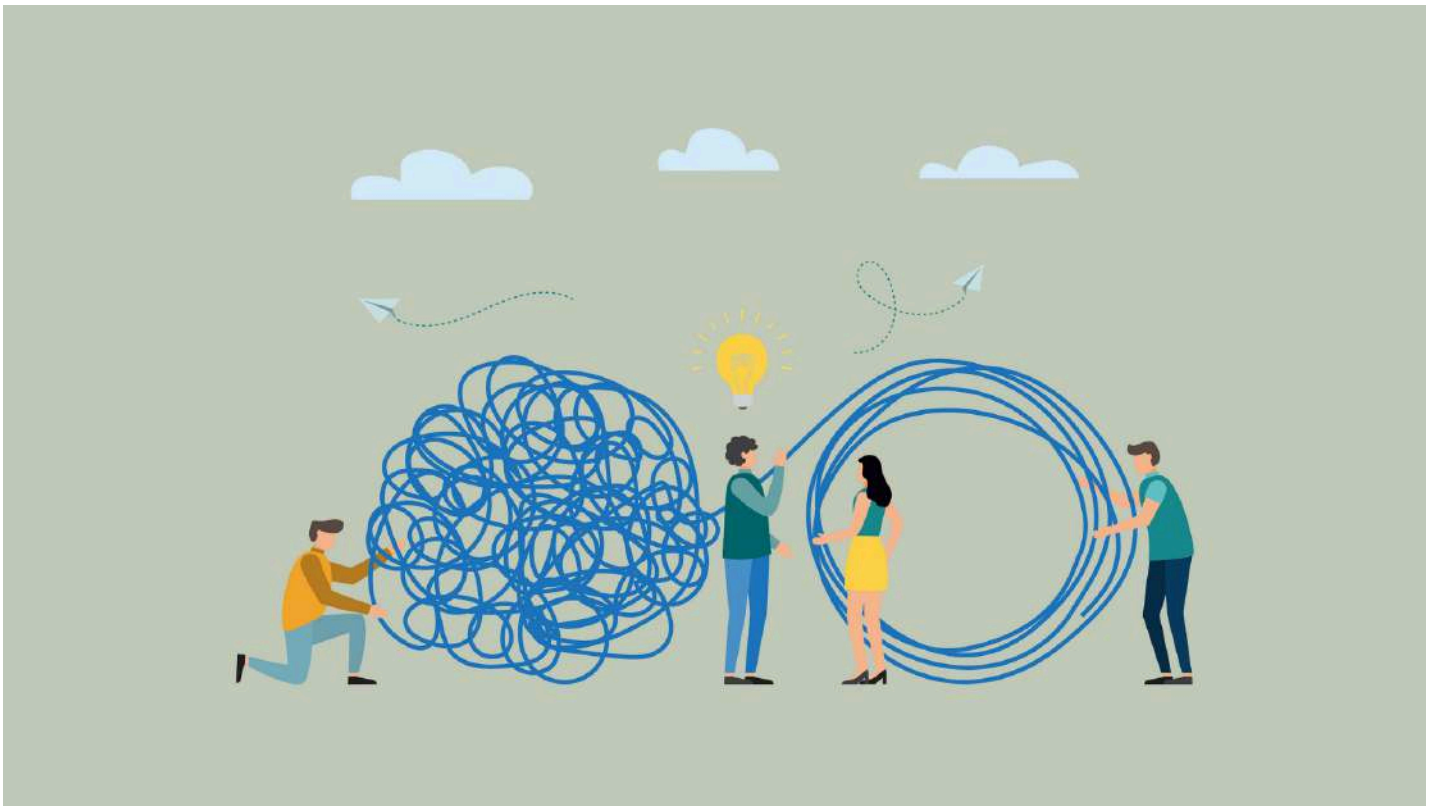
One other form of practice is by actually competing in competitions. You apply your knowledge, learn new things and gain more experience.

3. Learn new concepts every day

If you're stuck on a specific problem for hours, days, weeks; move on.
Move on to the other things, learn new concepts and solve different kinds of exercises. Later, you can come back to your "unsolvable" problem. By the time you come back, you'll have more experience and knowledge that makes the solving process easier.

Programming is all about problem-solving, but don't make additional problems with your stubbornness. **Failing to solve a specific exercise does not equal failure in CP.**

4. Split the problem into smaller chunks



Splitting your problem into smaller chunks is called the **divide and conquer algorithm**.

A divide-and-conquer algorithm iteratively divides a problem into two or more sub-problems of the same or related type until they are simple enough to be solved directly. **The sub-problems' solutions are then combined to solve the original problem.**

This is the best way to solve any problem you have, especially the ones in programming. **Divide and conquer is a technique (a certain way of solving problems) but also an algorithm used in programming. It works the same in both scenarios.**

Simple example:

1. **Write a program that asks the user for their name and greets them with their name.**

To solve this problem we must know:

- how to ask a user for input
- storing the value to a variable
- how to print and combine 2 strings

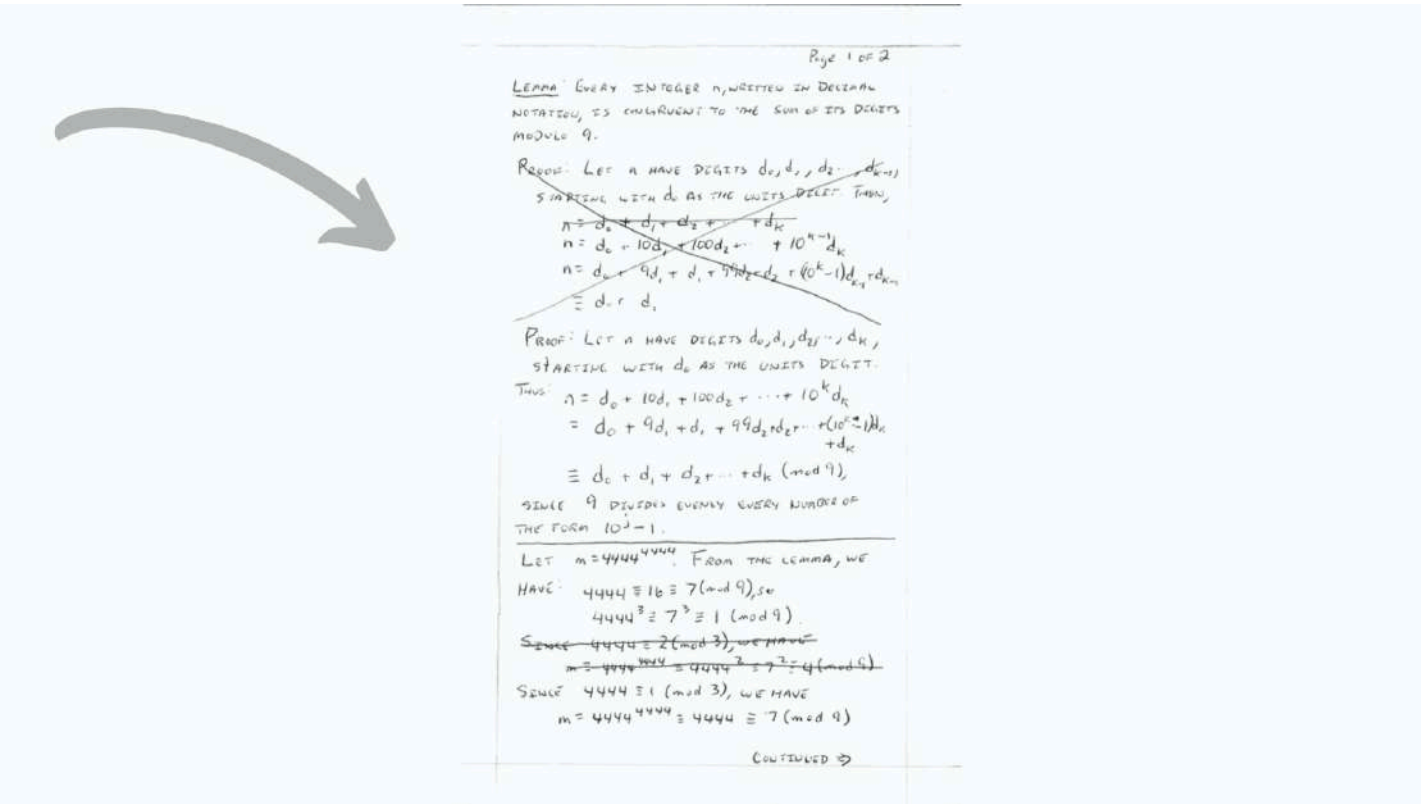
These are subproblems that lead to achieving the end goal; printing the input name.

5. Solve the problem on paper then write the code

When you're solving a more complicated problem, a problem you never came across, it's crucial to build the ideal solution and steps to achieve this.

When you write your problem on paper, **you are not required to write the exact code** you'll use. On paper, you go through problem requirements, inputs, outputs and different examples.

Example of working out your problem:



Page 1 of 2

LEMMA: EVERY INTEGER n, WRITTEN IN DECIMAL NOTATION, IS CONGRUENT TO THE SUM OF ITS DIGITS MODULO 9.

~~PROOF: LET n HAVE DIGITS $d_0, d_1, d_2, \ldots, d_{k-1}$, STARTING WITH d_0 AS THE UNITS DIGIT. THEN,~~

$$\cancel{n = d_0 + d_1 + d_2 + \cdots + d_k}$$

$$\cancel{n = d_0 + 10d_1 + 100d_2 + \cdots + 10^{k-1}d_k}$$

$$\cancel{n = d_0 + 9d_1 + d_1 + 99d_2 + d_2 + (10^k - 1)d_{k-1} + d_{k-1}}$$

$$\cancel{\equiv d_0 + d_1}$$

PROOF: LET n HAVE DIGITS $d_0, d_1, d_2, \ldots, d_k$, STARTING WITH d_0 AS THE UNITS DIGIT.

THUS:

$$n = d_0 + 10d_1 + 100d_2 + \cdots + 10^k d_k$$

$$= d_0 + 9d_1 + d_1 + 99d_2 + d_2 + \cdots + (10^k - 1)d_k + d_k$$

$$\equiv d_0 + d_1 + d_2 + \cdots + d_k \pmod 9,$$

SINCE 9 DIVIDES EVENLY EVERY NUMBER OF THE FORM $10^j - 1$.

LET $m = 4444^{4444}$. FROM THE LEMMA, WE HAVE: $4444 \equiv 16 \equiv 7 \pmod 9$, SO

$$4444^3 \equiv 7^3 \equiv 1 \pmod 9.$$

~~SINCE $4444 \equiv 2 \pmod 3$, WE HAVE~~

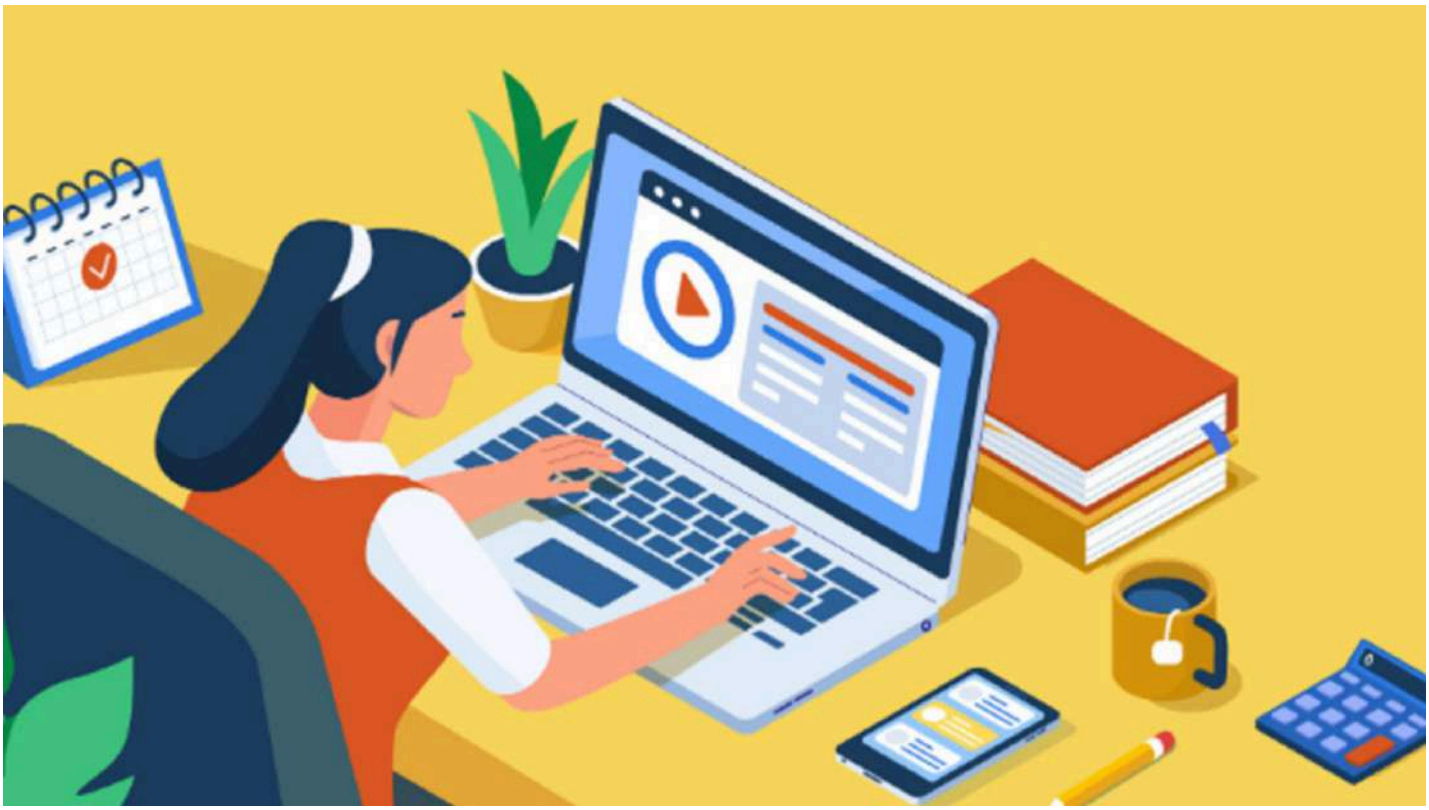
~~$m = 4444^{4444} \equiv 4444^2 \equiv 7^2 \equiv 4 \pmod 9$~~

SINCE $4444 \equiv 1 \pmod 3$, WE HAVE

$$m = 4444^{4444} \equiv 4444 \equiv 7 \pmod 9$$

CONTINUED ⇨

6. Practice online and connect with other developers



Once you try online coding competitions, you'll never go back.
That way, you get to practice your knowledge, learn from your mistakes and test yourself. You start thinking outside the box and work on your creativity. In the resources I listed the sites that will help you practice and compete. But most importantly, you become a part of the community!

You connect with developers with the same or similar goals as yours. Get to share knowledge, gain experience and learn from others. **That is the true beauty of the programming world.**

7. Don't be afraid to ask for help

Asking for help strengthens connections by allowing others to offer their knowledge and resources, showing that you value and trust their opinions. **Your questions are not stupid, especially when you're a beginner.** This is how we build our foundations.

One of the best ways we learn is from others. You might learn a new approach, new way of thinking or get an idea.

Resources

Sites to practice online:

- CodinGame
- HackerRank
- Project Euler
- Codewars
- Topcoder
- Codeforces

Part I: What is Competitive Programming?

Summary

Today we dived deeper into the Competitive Programming world.
We discussed the basics, key concepts and useful tricks that will help you prepare more for your next Competition. **I hope you find the information in this article useful and that it helped you.**

Remember:

> "The more you practice the better you'll be, the harder you train the great in you they'll see."
>
> **Alcurtis Turner**

Flutter Development – part III

March 7, 2022

Categories: Flutter Beginner, Let's create

Tags: 100DaysOfCode, Coding, Flutter, LitCode, Mobile App Development

Welcome to Developing with Flutter – part III. Today, we'll continue building our Inspire App. If you haven't already, make sure to check out part I and part II of this series.

After finishing this tutorial, you are done with the first part of Level 2 in Flutter Development! In this post, we are working on both the logic and design of Inspire.

Remember: every part of our App will evolve during this Flutter series, these are not the final touch-ups on Inspire App!!

What did we build?

https://litcode.net/wp-content/uploads/2022/03/inspire_app.mp4

Final changes on Inspire – part II

In the last part, our primary focus was building the logic of Inspire.

Discussed topics:

1. Data types in Dart
2. Variables in Dart
3. Lists in Dart
4. Libraries
5. Dart math library

6. setState()
7. Dynamically changing the displayed text

What are we building today?

Today I'll introduce you to new Flutter Widgets that will update the look of our App. But also, we will set up everything for refactoring our code and work more on the logic.

We will cover:

1. Classes and Class Constructors
2. Objects
3. Functions
4. Images in Flutter
5. Stack
6. Padding
7. New styling properties

1. Classes and Constructors

According to Wikipedia:

- A class is an expandable program-code template for constructing objects in object-oriented programming, giving starting values for state (member variables) and behaviour implementations (member functions or methods).

These are some big words. All of this becomes clearer on a simple example:

```
class ClassName {
  //syntax for creating a class
  //after the brackets you build the body
}
```

As already said above; a class can have its members(variables, methods) and a class has a Constructor. Through a Constructor we can access some class properties.

```
class Animals {
  String dog;
```

```
  String cat;   //class properties (member variables)
  String rabbit;

  //this is how a class constructor is created
  Animals({this.dog, this.rabbit, this.cat})
}
```

In your Flutter project, in lib folder create a new file called **quote.dart.** Here we are creating our Quote class.

```
class Quote {
  String authorName; //the name of the quote author
  String quoteText; //the quote content
  String imageURL; //name of image file with author's image

  //Quote class Constructor
  Quote(
      {this.quoteText,
       this.imageURL,
       this.authorName});
}
```

2. Adding images to our Project

To display the image of our quote author, we must add an image folder to our project. After that, in pubspec.yaml file, we tell Flutter that we added some assets (in our case images).

The images folder must be created in the project level. Like this:

Then you just add the images you want to use. In my project I have 7 images. After that you open pubspec.yaml file, uncomment the assets part and enter your folder name.

Now you're ready to use your images!

3. Creating the Quote Brain

Let's create another file **quotebrain.dart** in our lib folder.
Here we're storing the quoted text, author name and author images. But also building some methods to make our lives easier.

```
class QuoteBrain {


}
```

Now let's import our quote.dart file. In this file, we will create a list of quotes, authors and image URLs.

1. We will create the quoteNum variable in this file and set it to 0
2. Import dart math library and create a randomNum variable that holds the Random() object
3. Create a list and store Quote Class objects
4. Build 3 functions that return quoteText, author and image URL
5. Create a function that updates the quoteNum

```
import 'dart:math';
import 'package:inspire_app/quote.dart';
```

```
// Steps 1. and 2.
class QuoteBrain {
  int quoteNum = 0;
  Random random = Random();

}
```

What are Objects?

An object is an abstract data type defined by a developer in object-oriented programming (OOP). It may include additional objects as well as various attributes and functions.
We already used Random() object. In our case, it allowed us to use the Random Class function(method) nextInt. With this function, we're able to generate a random number.

```
import 'dart:math';
import 'package:inspire_app/quote.dart';



class QuoteBrain {
  //Step 1.
  int quoteNum = 0;
  //Step 2.
  Random random = Random();
  //Step 3.
  List<Quote> quoteBrain = ;
}
```

Creating our functions/methods

Functions are "self-contained" code modules that perform tasks. Data is often "taken in," processed, and a result is "returned" by the function. Our custom functions are created outside the main() function.

Syntax for declaring a function looks like this:

```
datatype function_name(){
  //function body
}
```

Inside QuoteBrain class let's create:

- **getText** function that returns the quote text at quoteNum
- **getAuthor** function that returns author name at quoteNum
- **getImage** function that returns image URL at quoteNum

To access the properties of any class, we use the dot operator.

```
//Step 4.
String getText() {
    return quoteBrain.quoteText; //access the text
 }

String getAuthor() {
    return quoteBrain.authorName; //access the author name
 }

String getImage() {
    return quoteBrain.imageURL; //access the image URL
}
```

Last let's create a nextQuote method. Here we'll set quoteNum to a random value and increase it by one.

```
//this function type is void because it only sets quoteNum to
//a random value; it doesn't return anything
//Step 5.
void nextQuote() {
  _quoteNum = _random.nextInt(7);
}
```

4. Stack, Padding and style properties

Now, let's work on the design. Before that from main.dart we can remove our quotesList, quoteNum and randomNum variable.

- **Stack Widget** – a built-in widget in flutter SDK which allows us to make a layer of wid by putting them on top of each other.

- **Padding Widget** – a widget that insets its child by the given padding.

As our Container Widget child, we'll add a Stack Widget.

```
      body: Center(
        child: Container(
          alignment: Alignment.center,
          child: Stack( //new Stack widget
            alignment: Alignment.topCenter,
            children: ,
                ),
              ),
```

Now let's add one more Container where we will display our author image.
We use the **AssetImage Widget** and pass in image URL through **getImage** function. Set this image to cover the widget through **fit property.**

```
Container(
  width: 100,
  height: 100,
  decoration: BoxDecoration(
  color: Colors.white,
  borderRadius: BorderRadius.all(Radius.circular(50.0)),
  border: Border.all(width: 1.2),
  image: DecorationImage(
       image: AssetImage(quoteBrain.getImage()), //displays the Image
       fit: BoxFit.cover, //covers the widget
    ),
),
),
```

Let's also move the **Button** inside the Stack. In **onPressed** property, we **setState** and call **nextQuote** function through **QuoteBrain** property.

```
CupertinoButton(
  color: Colors.indigoAccent,
  child: Text('Get inspired'),
  onPressed: () {
```

```
    setState(() {
      //accessing nextQuote method through the object
       quoteBrain.nextQuote();
  });
});
```

The Final product so far…

https://litcode.net/wp-content/uploads/2022/03/inspireIII.mp4

Last shot – Inspire part III

Summary

Today we worked on Inspire design and logic. We got introduced to Classes, Constructors, Objects, Stacks…and more!

I hope this Post thought you more about Flutter!

In part IV we'll cover some **code refactoring** and **I'll introduce you to Firebase**…

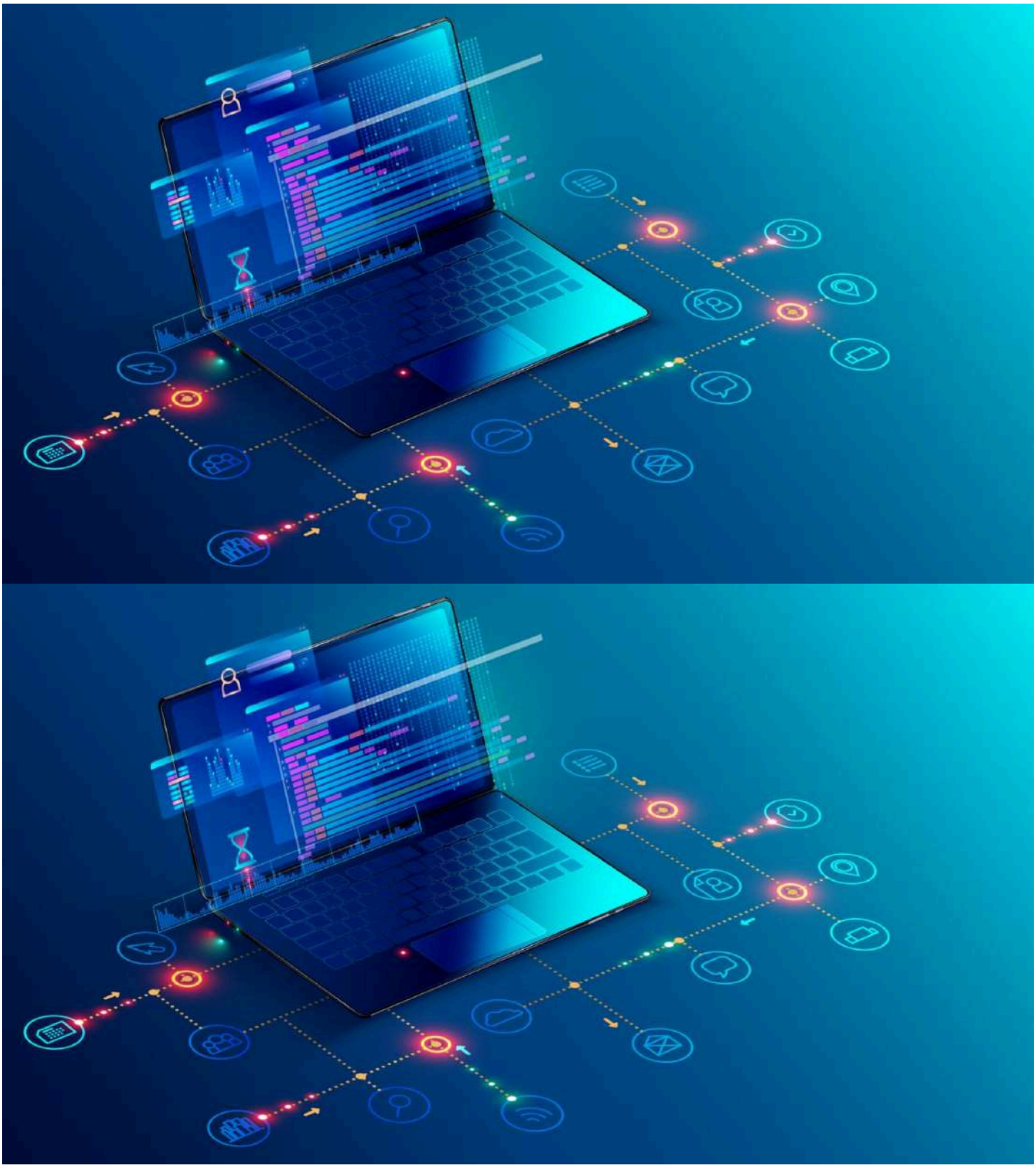
Stay tuned for more! Happy creating. – anonymous of LitCode.

Is programming for me?

March 11, 2022

Categories: Quiz

Tags: 100DaysOfCode, Coding, LitCode, ProgrammersLife, Quiz



Is programming really for you? 🤷‍♀️

Welcome to LitCodes second Quiz!

If you're indecisive about pursuing a programming career, this Quiz is here to clear your mind. At the end of the Quiz, you'll be able to check your answers and get your most honest review.

> "Learning to code is learning to create and innovate."
>
> *Enda Kenny*

Don't give up on coding if your scores aren't great! Check out Resources to improve your skills and excel in your programming journey!

Take the Quiz!

Resources

Even if your scores aren't as great as you thought they would be, check this out:

- Why should everyone learn to code?
- Books that changed my life
- Reaching your (coding) goals
- How to be a better programmer?

How to start coding?

March 14, 2022

Categories: Let's learn, Let's read

Tags: 100DaysOfCode, Coding, Goals, LitCode

If you're not sure how to start your coding journey, this post is for you!

In today's article, I'll share with you some tips that I wish I knew when I started programming. We'll cover setting jour coding goals, choosing your coding language, building logical thinking and more!

As a beginner, you might feel intimidated by the whole idea of programming. You shouldn't be. **Programming is a skill like any other and can be mastered by practice and putting in the work.** I hope this post makes your journey easier!

Let's get started!

Why should everyone learn to code?

As I discussed in my first LitCode post, everyone should learn to code. **But this doesn't mean that coding is for everyone.** This is the question I get asked the most. We live in a world created and ruled by technology. Learning to code helps you understand this chaotic world better.

In that article, I talked about 5 reasons why everyone should learn to code.

Feel free to check it out: Why should everyone learn to code?

Why should YOU learn to code?

Before moving on with this article you need to do something first. You need to set your coding goals right away.

- **Why do you want to learn to code?**
- **What do you expect from your coding journey?**
- **What do you want to accomplish?**

After answering these questions your head will be clearer. You'll have the hardest part already done.

> "If you don't know where you are going, you will probably end up somewhere else."
>
> Lawrence J. Peter

1. Choose your language

You might be interested in Game Development, then C++ or C# is the right fit for you. Or Web Development, Competitive Programming, Mobile App Development and more!

Your field determines your programming language.

But we can all agree, with Python you can never go wrong. I believe that starting with a language like C++ can't harm you in any way too. **It's complicated for beginners, but after you learn how C++ works, you can toggle between any languages you want.**

Popular programming languages:

1. Python
2. C++
3. Java
4. JavaScript
5. C#
6. Swift
7. Dart (with Flutter)

...and more!

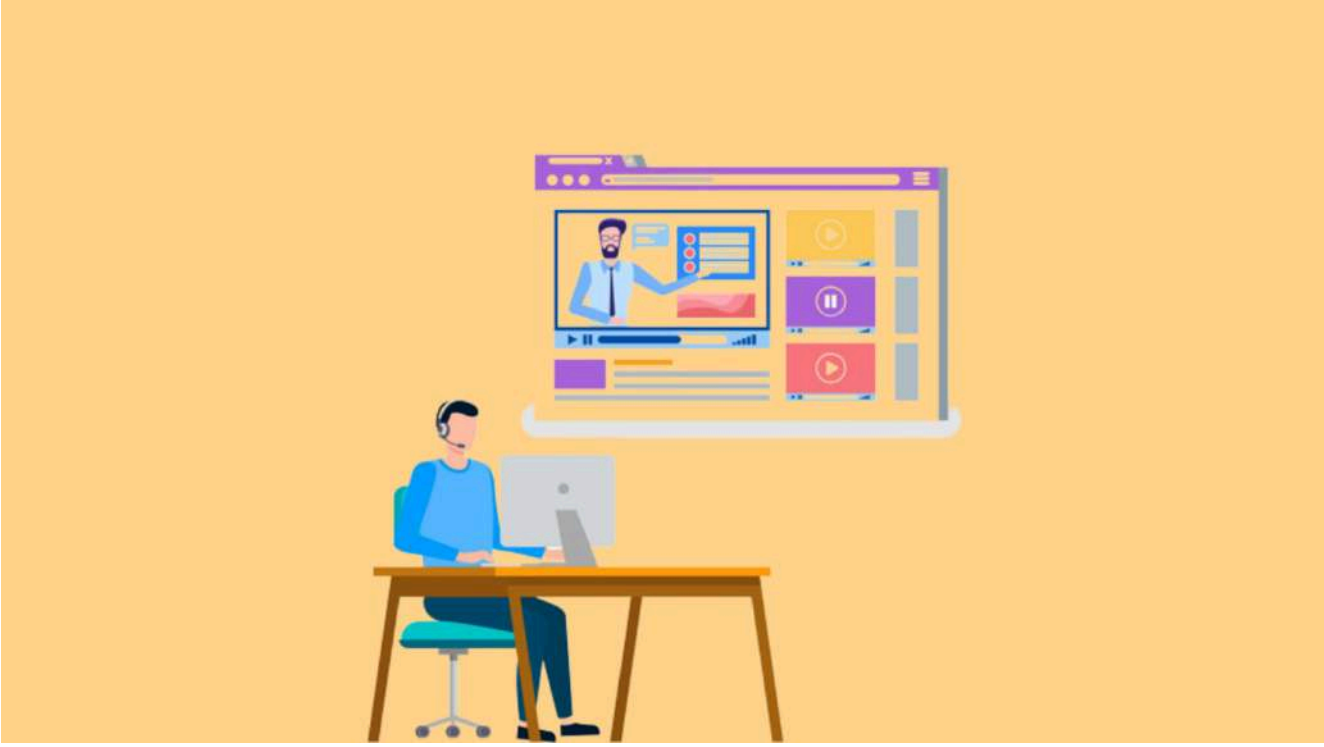
2. Build logic skills

Logical thinking is a concept that **CAN** be learned. People are not born with fantastic problem-solving skills.

There are numerous ways to master the art of logical thinking.

- **The most important one is to learn new things every day.**
- **Check code written by other developers in order to learn how certain things are achieved.**
- **Ask questions, solve problems with pen and paper.**
- **Be consistent with coding because it is not a thing that is mastered overnight.**
- **Make mistakes!**

3. Enroll in a Course



I believe that following a Course is the best way to start programming. **You get a tutor, video lectures and tons of exercises/examples that will put you on the right path.**

Nowadays, everyone is making courses and tutorials. If you like learning with video lectures, consider some courses on Udemy or YouTube.

It is crucial not to skip through the course material. The tutor made that material for a reason. You might not understand it yet, but trust the tutor!

4. Read books

> "Read, read, read."
>
> William Faulkner

If you're not into video lectures, reading programming books is a great way to start programming too. But also, while taking a course, it is common for a tutor to recommend you the reading material. You can find books on any subject you want. Every programming language has great Documentation and Cookbooks you can learn from.

Start with these:

1. The Self-Taught Programmer: The Definitive Guide to Programming Professionally
2. The Pragmatic Programmer: From Journeyman to Master
3. Python Programming: An Introduction to Computer Science
4. Think like a programmer: An introduction to creative problem solving
5. Learning to Program

5. Join a coding community

Whether you're just starting or you're an experienced programmer having a coding community is important! In a community you are able to ask questions, solve problems together and meet amazing people.

You meet developers that have similar ideas to you and get a chance to work on some amazing projects! **When you join a coding community you get a chance to inspire and motivate someone.**

Things to keep in mind

There are still some things to keep in mind before starting to code. If you don't get the **BASICS** right, you're not getting anywhere. A bad **START** equals a bad **JOURNEY**. You have to be **PATIENT** on this road. As I said, coding is not something you learn overnight, it takes time.

Set your **GOALS**, build **PROJECTS** and don't give up!

If you're still indecisive about Is programming for you, check out my Quiz.

Summary

Today we discussed:

- Setting your programming goals
- Choosing your programming language
- Building logic skills
- Enrolling in a programming course
- Reading books
- Joining a coding community
- Things to keep in mind before starting

I hope this article was helpful and put you on the right path for your coding journey. Check out the related post: Why should everyone learn to code?

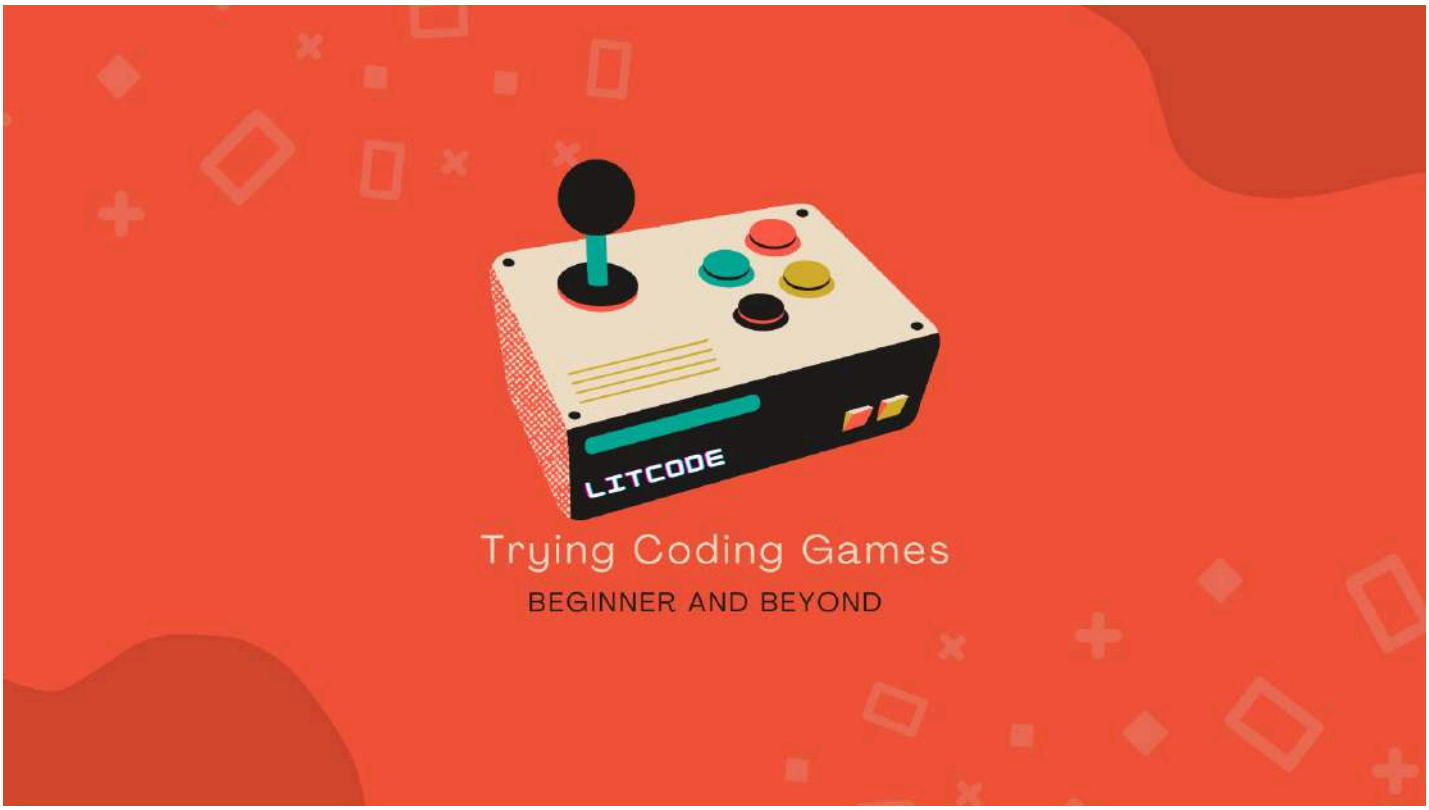
Happy coding! Enjoy the process. – Irma

What games teach you to code?

March 16, 2022

Categories: Let's create

Tags: 100DaysOfCode, Coding, Games, LitCode, Online



Let’s put some online coding game sites to the test! In today’s article, I’ll show you 5 coding game sites that I coded in.

Playing coding games is a fun and useful way to learn to code! You get a chance to learn new concepts and practice them on real examples. Also, you have the opportunity to connect to other developers/gamers on your coding platform! These coding sites are suitable both for beginners and more experienced developers.

If you're in the mood to challenge yourself...let's get started!

1. CodinGame

CodinGame is a fun way for programmers to improve their coding abilities by playing games.

On CodinGame, you can code anything you want. **While solving challenges, you can get hired for the skills shown.** This is more of an intermediate coding game site, but it can't hurt ambitious beginners too!
To get started with CodinGame create your free account and start competing!

Let me show you an example of CodinGame challenge:

On CodinGame, you can solve the challenges in specific categories. Each challenge shows you what you'll learn and how that benefits your coding skills! **This is how the editor looks:**

After solving the challenge, play test cases and check your results!

https://litcode.net/wp-content/uploads/2022/03/CodinGame.mp4

2. CodeCombat

CodeCombat is a video game that teaches computer programming topics and languages.

This game is suitable for kids aged 9 to 16. Students learn to type in coding languages such as JavaScript, Python, HTML, and CoffeeScript, as well as computer science principles.

Let’s code in Python:

After solving challenges, you earn new equipment and can change your characters!

Get the instructions, your coding goals and just code! This is a super fun way to practice your coding skills and it’s definitely suitable for beginners.

3. Elevator Saga

Elevator Saga is a game where you control a set of elevators using JavaScript.

Your goal is to write a JavaScript program to program the movement of elevators. **The idea is to get people where they need to be as quickly as possible.** You can move through the more tough challenges depending on how well you perform it.

https://litcode.net/wp-content/uploads/2022/03/ElevatorSaga.mp4

Check the Documentation and Help Section for explanation. Good luck!

4. SQL Murder Mystery

In SQL City, there's been a murder! **The SQL Murder Mystery is intended to serve as both a self-directed learning tool for SQL principles and procedures, as well as a fun game for experienced SQL users to solve a mystery.**

(SQL) is a shorthand for Structured Query Language. Data structures in relational databases are constructed using SQL, and data stocks based on them are searched.

5. CSS Diner

CSS Diner is a game where you may learn the fundamentals of CSS by completing a series of 32 challenges. This is an excellent approach to become better acquainted with the language while also having fun learning it.

As you progress and solve more challenges, you move to more complex CSS Concepts.

Resources

Here are the independent links to all of these sites:

1. CodinGame
2. CodeCombat
3. Elevator Saga
4. SQL Murder Mystery
5. CSS Diner

If you're a beginner, check these out:

- How to start your coding journey?
- Reaching your (coding) Goals
- How to be a better programmer?

Summary

In today's article, I shared with you 5 coding game sites. On these, you can practice and put your coding skills to the test! This is a great way to challenge yourself as a programmer.

I hope the information in this article was helpful and that it motivated you to code.
Remember:

> "The only impossible journey is the one you never begin."
>
> Tony Robbins

Happy coding! – Irma

Flutter Development – Quiz

March 21, 2022

Categories: Flutter Beginner, Quiz

Tags: 100DaysOfCode, Coding, LitCode, Mobile App Development, Quiz

Welcome to Developing with Flutter – Quiz! In todays quiz, you get the opportunity to test your knowledge of Flutter. To get the best results, check out the previous parts of the LitCodes Developing with Flutter series.

If you're passionate about Mobile App Development, this quiz is for you! Even if your scores aren't what you expected, that only means that you should practice more. **Practice is the key to success.**

> "Work hard and be kind and amazing things will happen."
>
> Conan O'Brien

Take the Quiz!

Resources

Check out the Flutter documentation.

Developing with Flutter – Quiz

LitCodes Flutter series:

- Introduction to Flutter
- Developing with Flutter – part II
- Developing with Flutter – part III

This might help you:

- How to be a better programmer?
- How to earn money as a programmer?
- Reaching your (coding) Goals?

Summary

Today we covered:

1. What is Flutter?
2. Which programming language does Flutter use?
3. In pubspec.yaml we…
4. What is a Stateless Widget?
5. Column Widget…
6. When adding HEX Color codes we pass in 0xEE.
7. What does setState() do?

8. Why did we use getImage() in QuoteBrain for?
9. What is Stack?
10. In Dart the datatypes are:

I hope this Quiz helped you understand your level of knowledge in Flutter.

Happy coding. – Irma

Sieve of Eratosthenes

March 23, 2022

Categories: Let's solve

Tags: 100DaysOfCode, Coding, Competition, LitCode, Problem

Welcome to the first Post of the Let’s solve category!

Today I'll explain to you what Sieve of Eratosthenes really is. It is an easy and useful algorithm for programmers. Especially for the ones interested in Competitive Programming.

If you haven't already, make sure to check out LitCodes Competitive Programming series. **Code Implementation in todays post is in C++.**

Check out:

- What is Competitive Programming?
- Getting ready for Coding competitions
- Getting started with C++

What is Sieve of Eratosthenes?

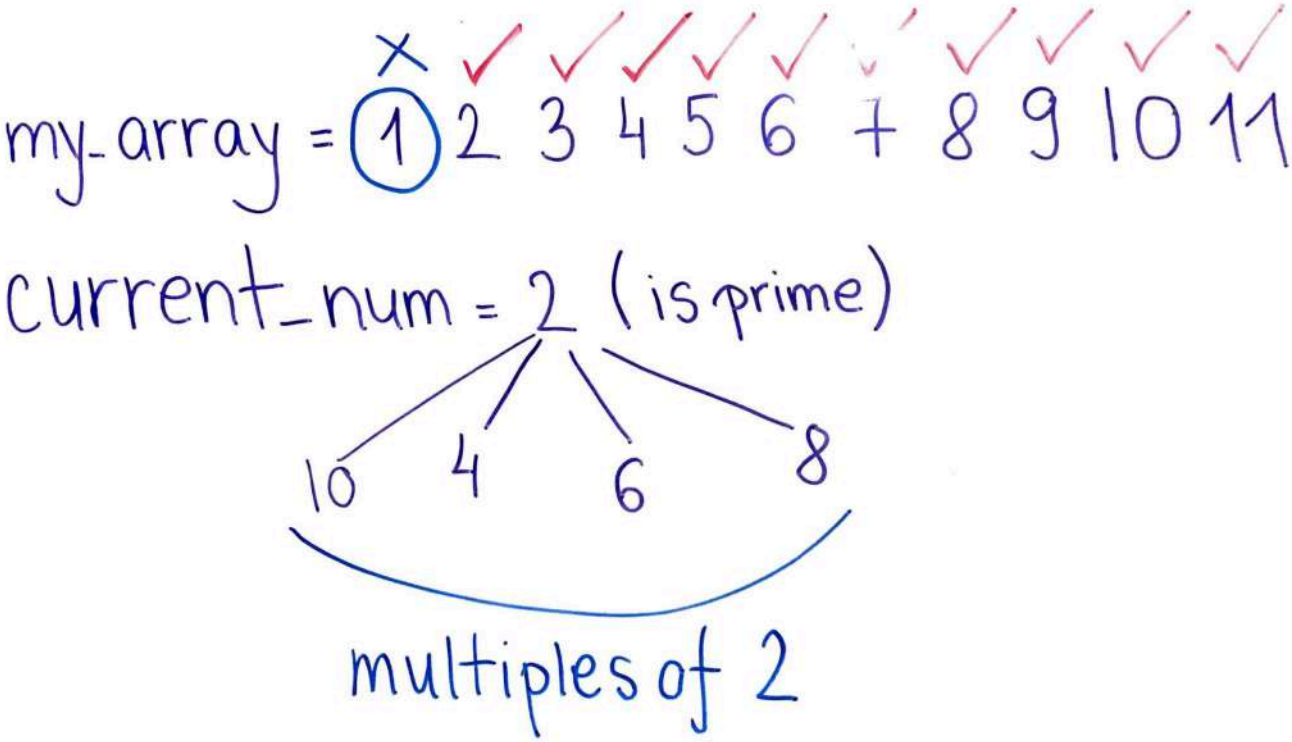
In mathematics, the **Sieve of Eratosthenes** is an ancient algorithm for finding all prime numbers up to any given limit. When is small, this approach works effectively, and we may use it to determine if any natural number smaller than or equal to is prime or composite.

The Idea

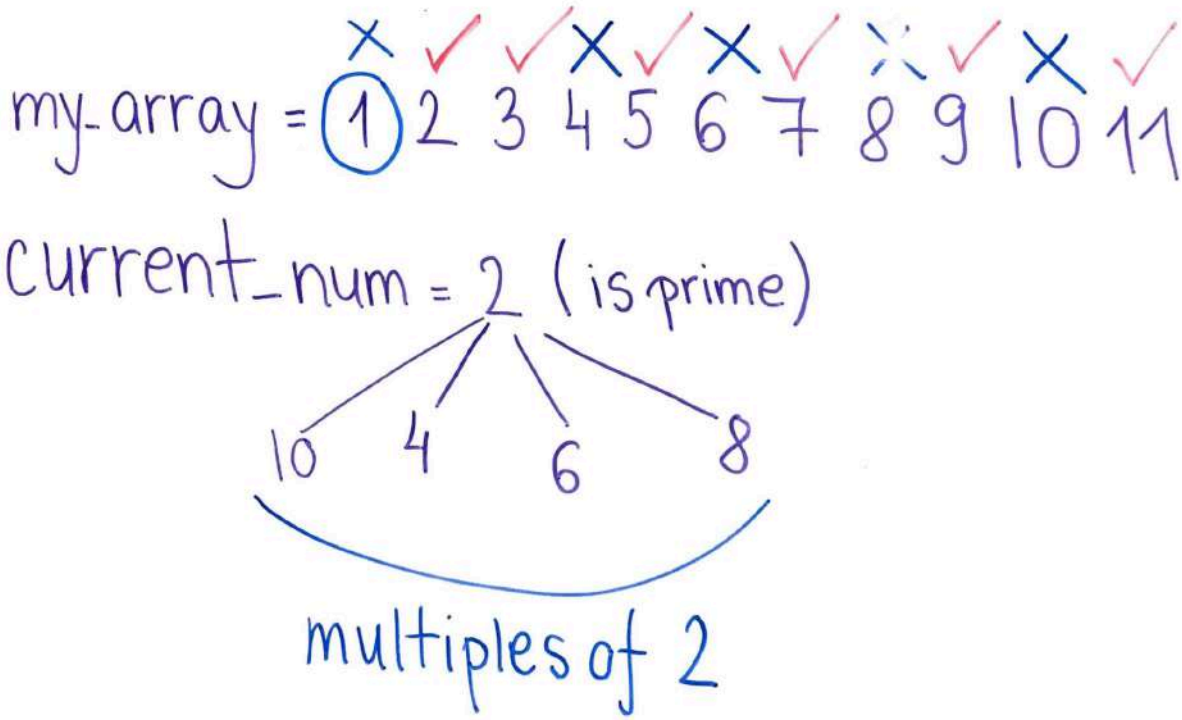
In the beginning, all the numbers in the array (except 1) are **prime**. We iterate through the array, if the **current number is marked as prime**, all of his **multiples** are marked as **not prime**. We repeat the same process until we iterate through the whole array of numbers.

Example 1

We have an array called **my_array** that holds numbers from 1 up to 11.
As I said before, 1 is not prime and all other array members are prime.



Now, let's imagine that our current number is 2.
2 is marked as prime (true), so every multiple of 2 will be marked as not prime (false). In our array multiples of 2 are 4, 6, 8 and 10.



So when we find multiples of our current number (2), we mark them as not prime (false).

Code Implementation

This is the code idea:

```
global: isPrime[100]
void sieve_func(int n){
 for(i=2; i<=n; i++)
  isPrime[i] = true;

 for(i=2; i<=n/2; i++)
  if(isPrime[i])
   for(j=i*2; j<=n; j+=i)
    isPrime[j] = false;
}
```

1. We define our array here called isPrime
2. We define our sieve_func that takes in number (n) where (n) is the limit
3. Through the first for loop, we set every value from 2 of our isPrime array to true (as in the numbers are prime)
4. Through the second for loop from 2 to n/2 we check if the current number is prime
5. Inside the if statement body we create another for loop that will set every multiple of our current number to false

- **Why is the second loop up to n/2?**

Because all of the multiples of the current number will be greater than (i/2). If we take (n) as the upper bound, our (j) will be greater than the limit (j is every multiple). We don't want that to happen.

- **How to assign every multiple to false?**

Using a for loop with (j) that equals (i*2) we assign every number in the steps of (j+i) to false. This is a trick used to go through all the multiples.

The C++ code:

```
#include <iostream>
using namespace std;

const int Nmax = 100001;
bool isPrime;

void sieve(int n){
    for(int i = 2; i<=n; i++){
        isPrime=true;
    }

    for(int i = 2; i<=n/2; i++){
        if(isPrime){
            for(int j=i*2; j<=n; j+=i){
                isPrime = false;
            }
        }
    }

}

int main(){
    sieve(100);
    for(int i = 1; i<=100; i++){
        if(isPrime) cout<<i<<" ";
    }
    return 0;
}
```

Summary

In today's post, we covered the Sieve of Eratosthenes algorithm.

I hope this motivated you to start your Competitive Programming journey. This is just one of many algorithms and problems we'll cover in the Let's solve LitCode series.

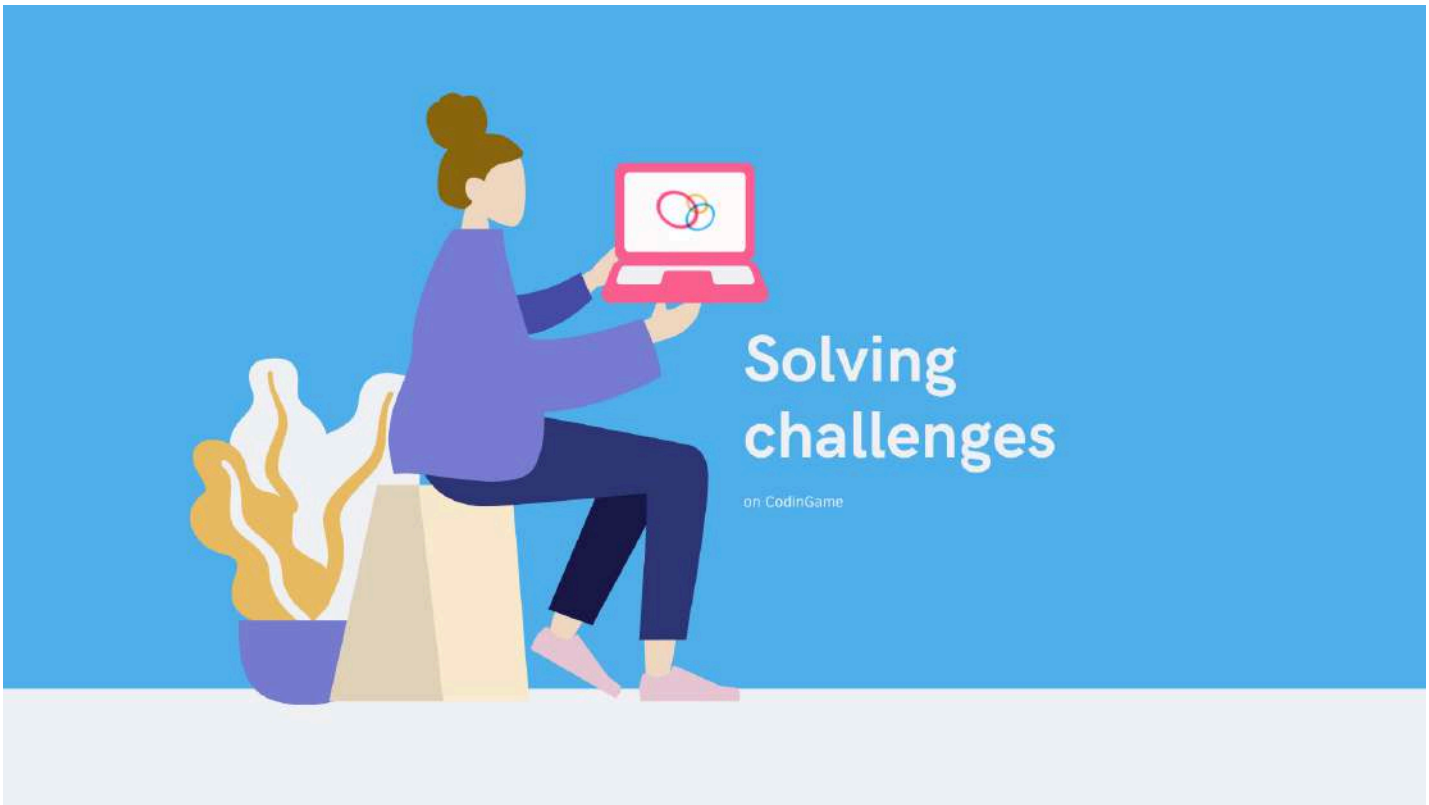
Happy coding. – Irma.

Solving challenges on CodinGame

March 27, 2022

Categories: Let's learn, Let's solve

Tags: 100DaysOfCode, Coding, LitCode, Solving



In today's post, we're getting introduced to CodinGame! **By solving challenges on CodinGame, you get to improve your coding skills and connect to other developers.** With me, you're ing 2 introduction challenges to get started with this fun site!

Don't get intimidated or quit because you failed to solve a coding challenge. You must practice coding in order to master it.

Now, let's jump right into the post!

What is CodinGame?

CodinGame is a fun way for programmers to improve their coding abilities by playing games. On CodinGame, you can code anything you want. While solving challenges you can get hired for the skills shown.

To learn more about what CodinGame actually is check out my post: Putting coding games to the test.

How to approach a coding problem?

My favorite coding saying is:

> First solve the problem, then write the code.

I couldn't agree more. I believe that solving a problem with pen and paper or on a whiteboard is the best way to solve it. You organize your thoughts by writing them down.

The best way to solve a coding problem is:

- Write the solution on paper
- Break the problem into sub-problems
- Solve the sub-problems
- Write the code
- Optimize and refactor if possible

I don't know how to code the solution...

You are not alone. This is a problem that almost every programmer I know has encountered. Especially when you're just starting. You're getting familiar with the programming language and you can't expect to know everything.

The most important thing: you must continue to put in the work. The coding tricks will [illegible] along. No one is born with the ultimate knowledge of the programming language.

Problem 1: The Descent

Problem statement:
Destroy the mountains before your starship collides with one of them. For that, shoot the highest mountain on your path.

Rules:
At the start of each game turn, you are given the height of the 8 mountains from left to right.
By the end of the game turn, you must fire on the highest mountain by outputting its index (from 0 to 7).

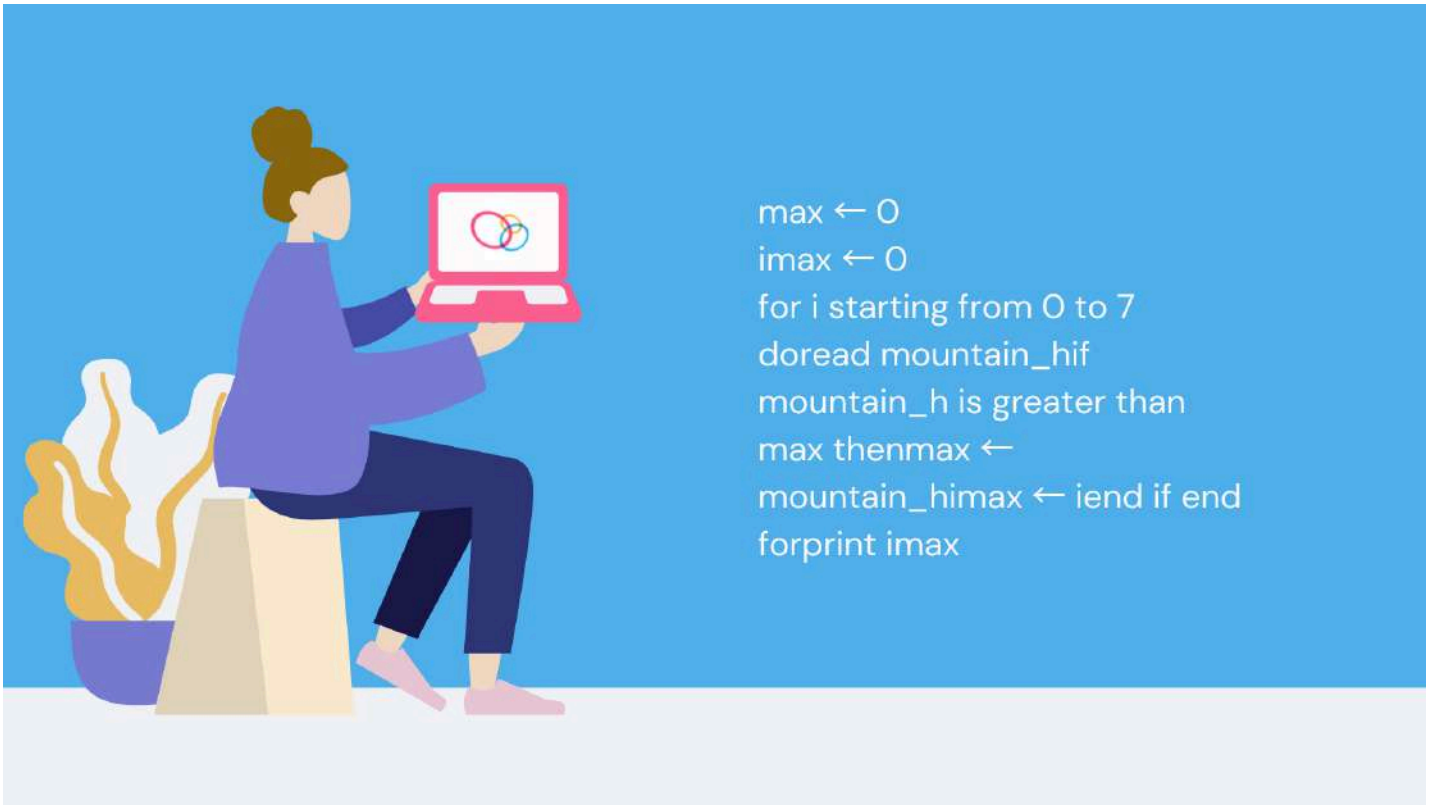
Firing on a mountain will only destroy part of it, reducing its height. Your ship descends after each pass.

Creating the solution

- When a computer program begins execution, standard streams are preconnected input and output communication channels between the program and its environment. The three input/output (I/O) connections are standard input (stdin), standard output (stdout), and standard error (stderr) (stderr).
- The supplied answer must be sent to stdout by your function.
- In the default code, you'll see an unending while loop. It's all part of the game's cycle.
- Each iteration of the while loop represents a gaming round in which your code must ta put data from stdin and reply to it using stdout.

- The game loop is not infinite: each problem has a maximum number of turns that may be solved.
- To discover the tallest, you must compare the heights of the mountains (provided in stdin) each round.

Pseudocode:



Code Implementation:

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

using namespace std;

/**
 * The while loop represents the game.
 * Each iteration represents a turn of the game
 * where you are given inputs (the heights of the mountains)
 * and where you have to print an output (the index of the mountain to
fire on)
 * The inputs you are given are automatically updated according to
```

```
last actions.
 **/

int main()
{

    // game loop
    while (1) {
        int max = 0;
        int res = 0;
        for (int i = 0; i < 8; i++) {
            int mountain_h; // represents the height of one mountain.
            cin >> mountain_h; cin.ignore();
            if(mountain_h > max){
                max = mountain_h;
                res = i;
            }
        }

        // Write an action using cout. DON'T FORGET THE "<< endl"
        // To debug: cerr << "Debug messages..." << endl;

        cout << res << endl; // The index of the mountain to fire on.
    }
}
```

Problem 2: Power of Thor

Problem statement:

Your program must allow Thor to reach the light of power.

Rules:

Thor moves on a map that is 40 wide by 18 high. Note that the coordinates (X and Y) start at the top left! This means the most top-left cell has the coordinates “X=0, Y=0” and the bottom right one has the coordinates “X=39, Y=17”.

Creating the solution

- You must compare the location of Thor and the position of the light of power.
- The referee program does not update Thor’s coordinates (they are only during the first turn). As a result, every time you send a move command in stdout, or every turn, you must update them.
- You may examine the four primary directions and use concatenation for the remaining four instead of having one instance for each conceivable direction (8 if conditions).
- For instance, if Thor needs to go north (“N”) and east (“E”), the proper command is “N”+”E”=”NE.”

Pseudocode:

Code Implementation

```
int main()
{
    int light_x; // the X position of the light of power
    int light_y; // the Y position of the light of power
    int initial_tx; // Thor's starting X position
    int initial_ty; // Thor's starting Y position
    cin >> light_x >> light_y >> initial_tx >> initial_ty; cin.ignore();

    // game loop
    while (1) {
        int remaining_turns; // The remaining amount of turns Thor can
move. Do not remove this line.
        cin >> remaining_turns; cin.ignore();
        string res;



        string directionX = "";
        string directionY = "";
```

```
        if (initial_tx > light_x) {
            directionX = "W";
            initial_tx--;
        } else if (initial_tx < light_x) {
            directionX = "E";
        }

        if (initial_ty > light_y) {
            directionY = "N";
        } else if (initial_ty < light_y) {
            directionY = "S";
            initial_ty++;
        }

        cout << directionY << directionX << endl;
    }
}
```

Resources

Check out: CodinGame.

Learn more about:

- Putting coding games to the test
- Getting ready for Coding Competitions
- How to be a better programmer

Summary

In today's post, I introduced you to CodinGame. We covered:

1. **The Descent – sorting and loops**
2. **The Power of Thor – conditions**

I hope this article motivated you to try out CodinGame.

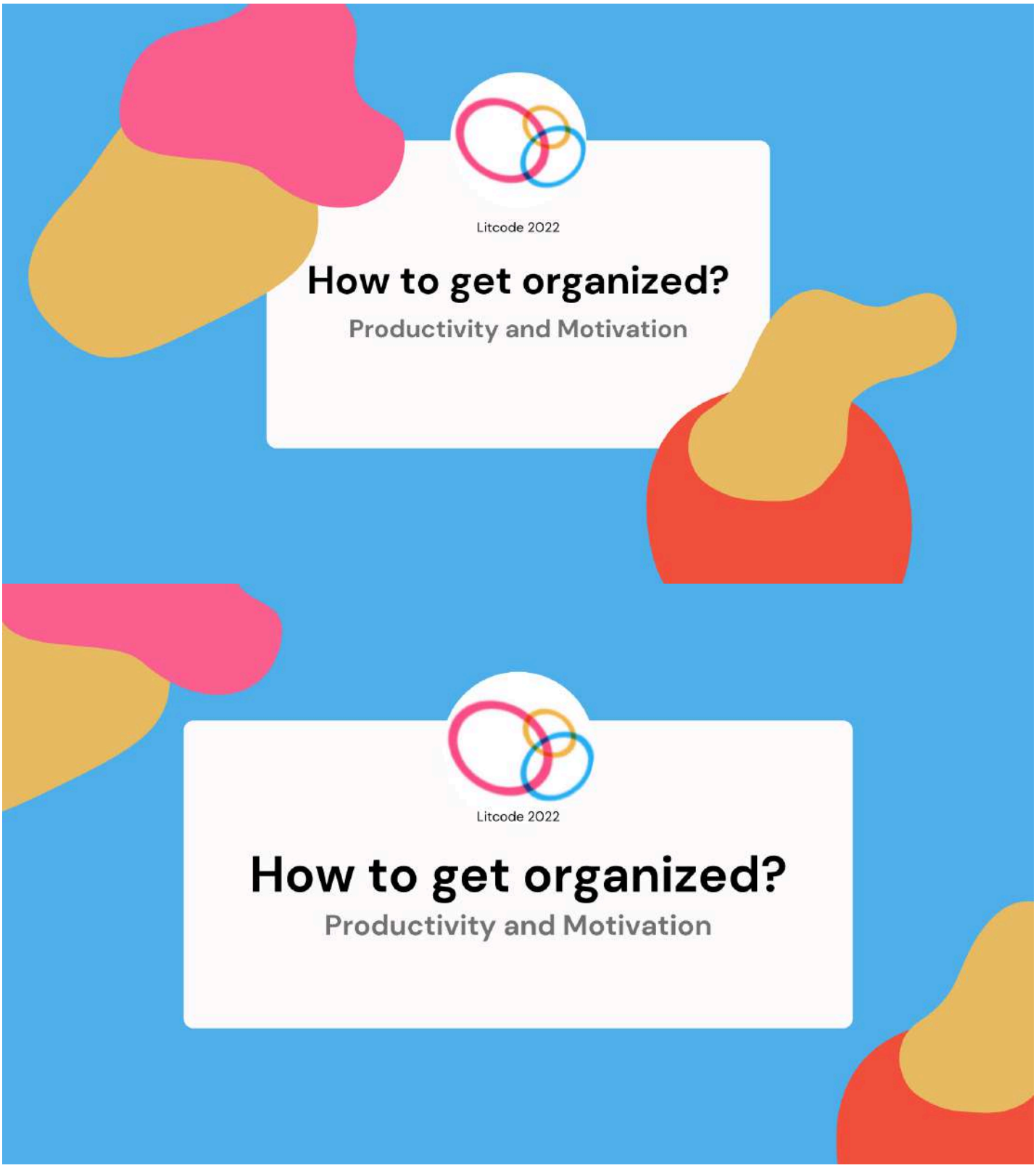
Until the next time. Happy coding. – Irma

How to get organized?

April 1, 2022

Categories: Let's read

Tags: 100DaysOfCode, Coding, Goals, LitCode, Motivation, Productivity



In today's article, I'll share with you tips on how to get organized!

These tips you can apply to any area of your life. Whether we're talking about programming, tech or getting your everyday life in order. I believe that besides putting in the work, organization and planning are primary factors that will set you up for success. After following my advice and tips, your professional/everyday life will get much easier.

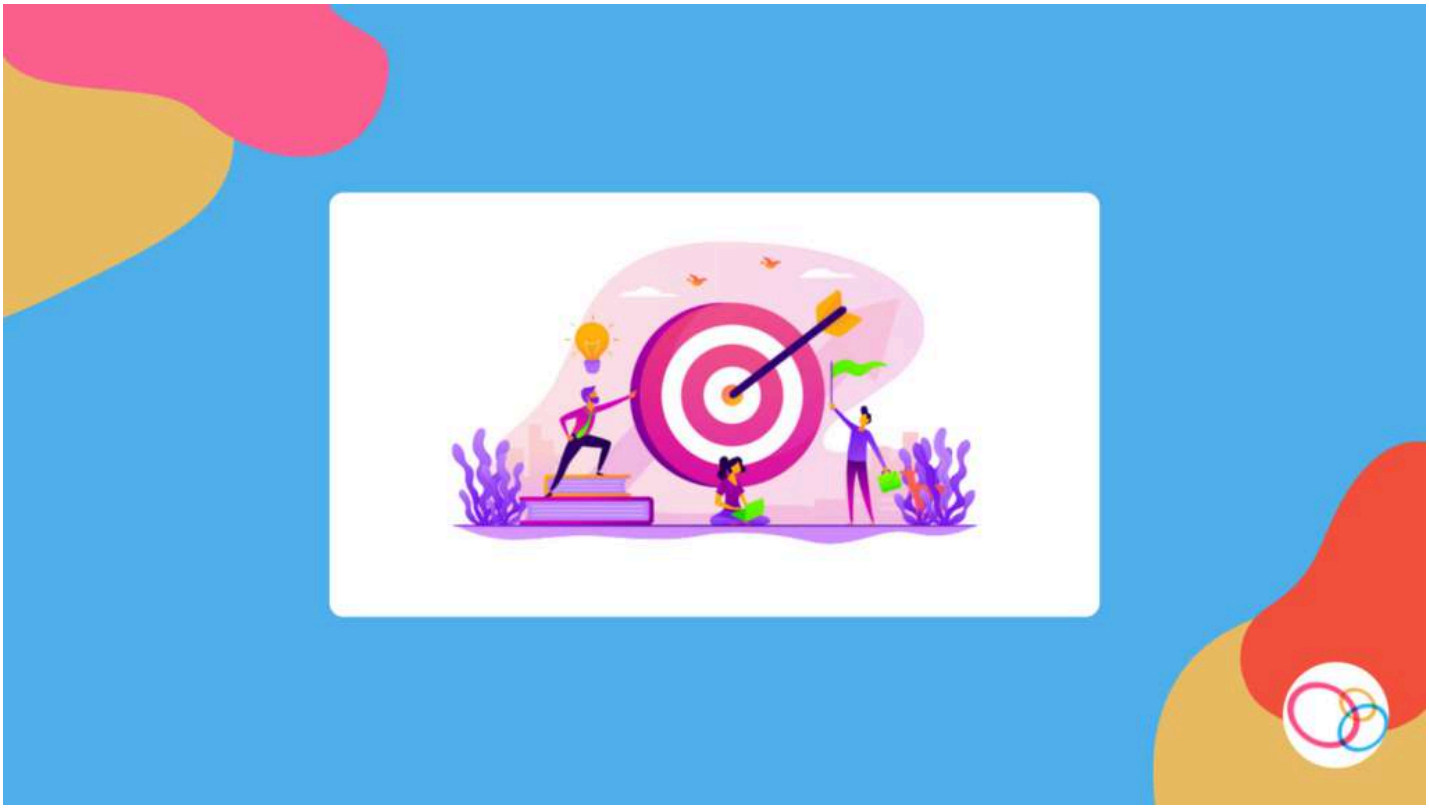
I hope you find the information in the article useful. Now, let's start!

Why is organization important?

When you learn how to get organized you learn to prioritize your everyday tasks, activities and achieve goals. When you're organized you're not stressed, everything you need to do has specific steps that lead to its completion. After you get organized, you start balancing more things at once, saving time and increasing productivity.

The best example is working in a team. You'll never accomplish anything if your work is not organized. Without a plan, deadlines, cutting off distractions and putting in the work there is no success.

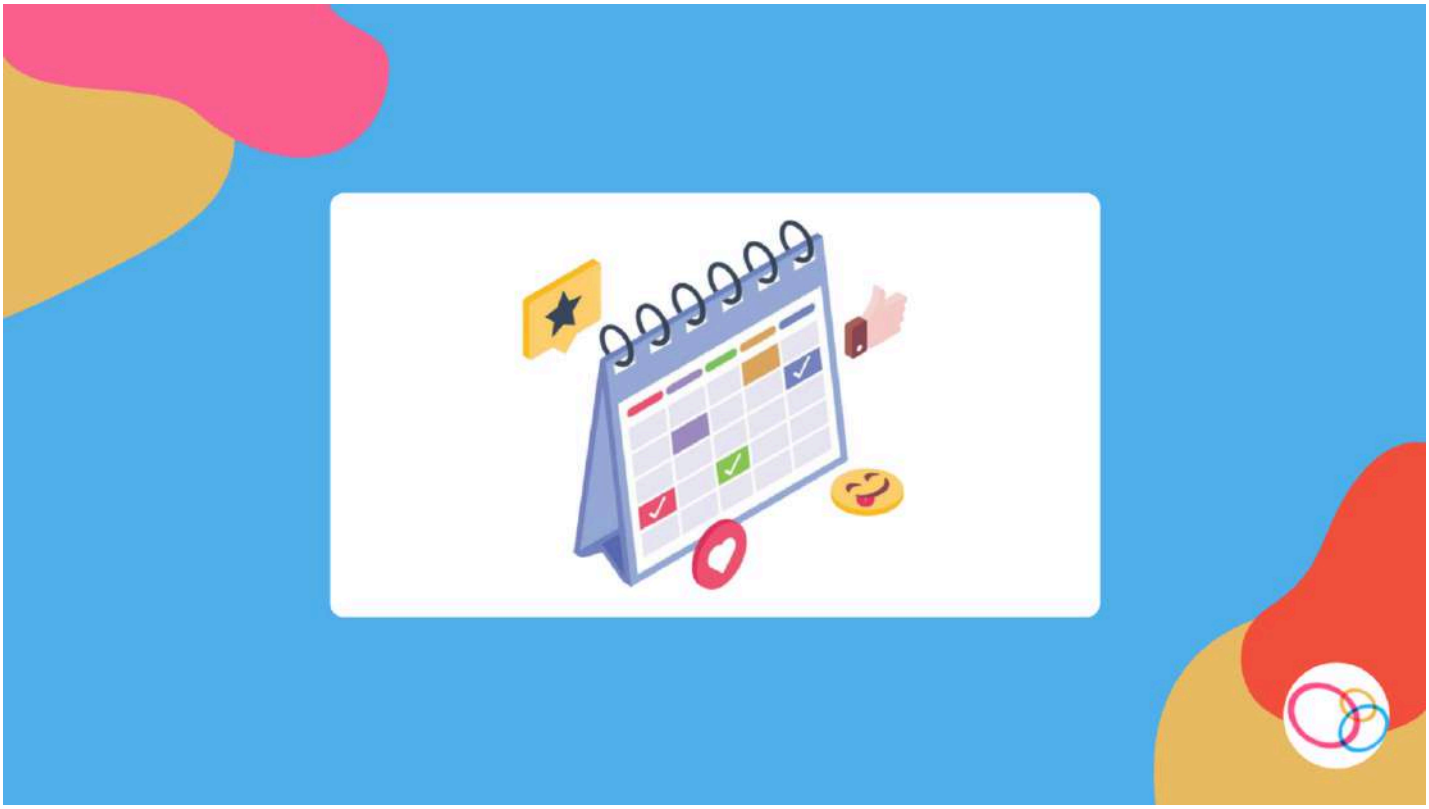
1. Making a plan



Having a plan is the first step to an organized life. A great plan is a plan that describes your end goal. If you write something down, you are more likely to accomplish it.

Your plan needs time for realization. When you set a certain time for focusing on a task you are already organizing. Take notes on your actions and thoughts. Go over your previous steps and spend time planning new ones. **And finally, just start. You can have an amazing plan but without taking the first step you're not getting anywhere.**

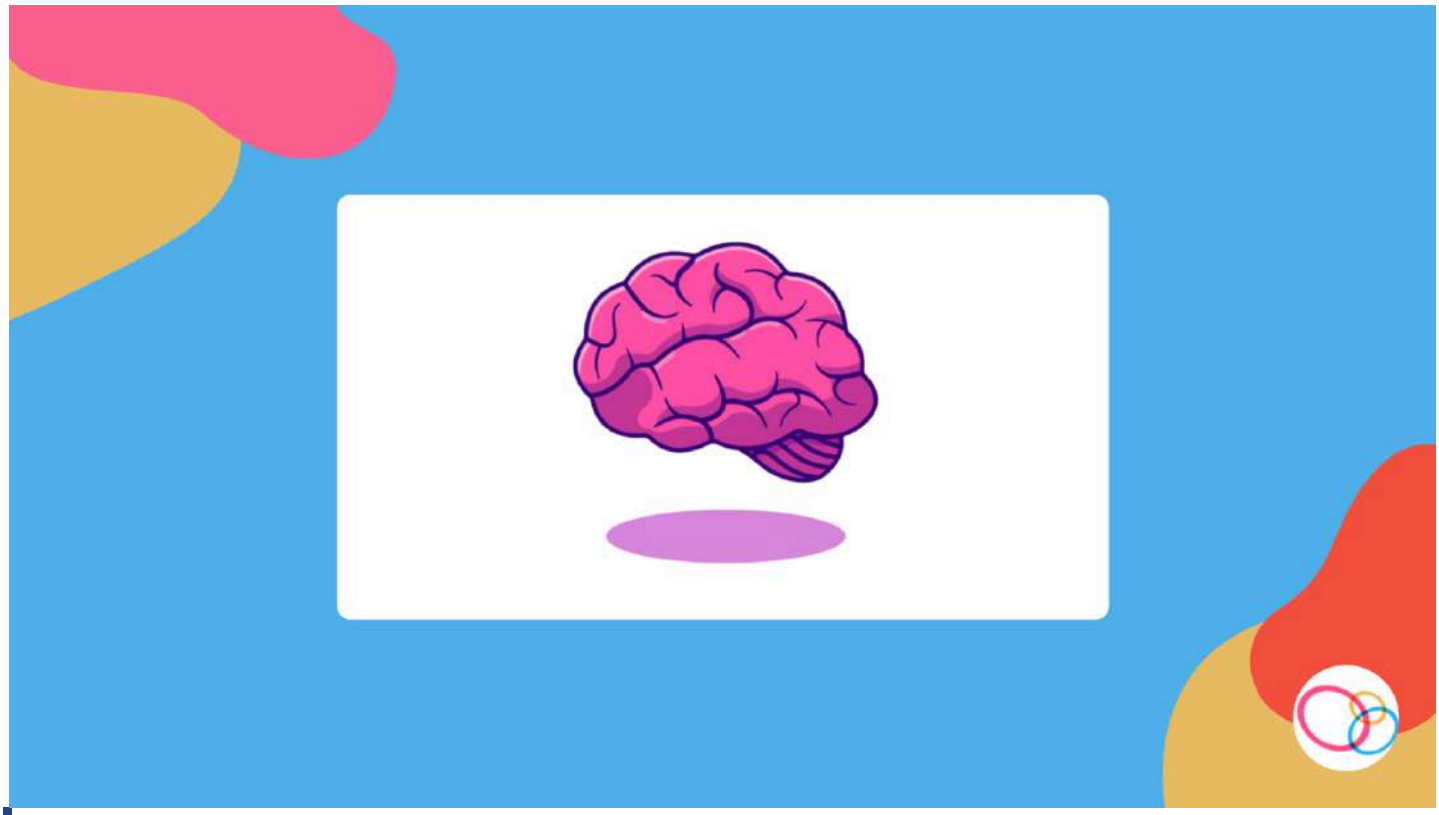
2. Setting deadlines



Deadlines put you through a lot of pressure. **But they are the best way to get organized and start work.** After meeting them, you get rewarded with the feeling of completion. You are more likely to complete your long-term and short-term goals. Being able to meet a deadline is very important when working in a team but also by yourself. That shows your the discipline and determination to achieve your end goal.

Some people like the pressure, they like setting deadlines. You can view deadlines as big challenges.

3. Declutter your mind



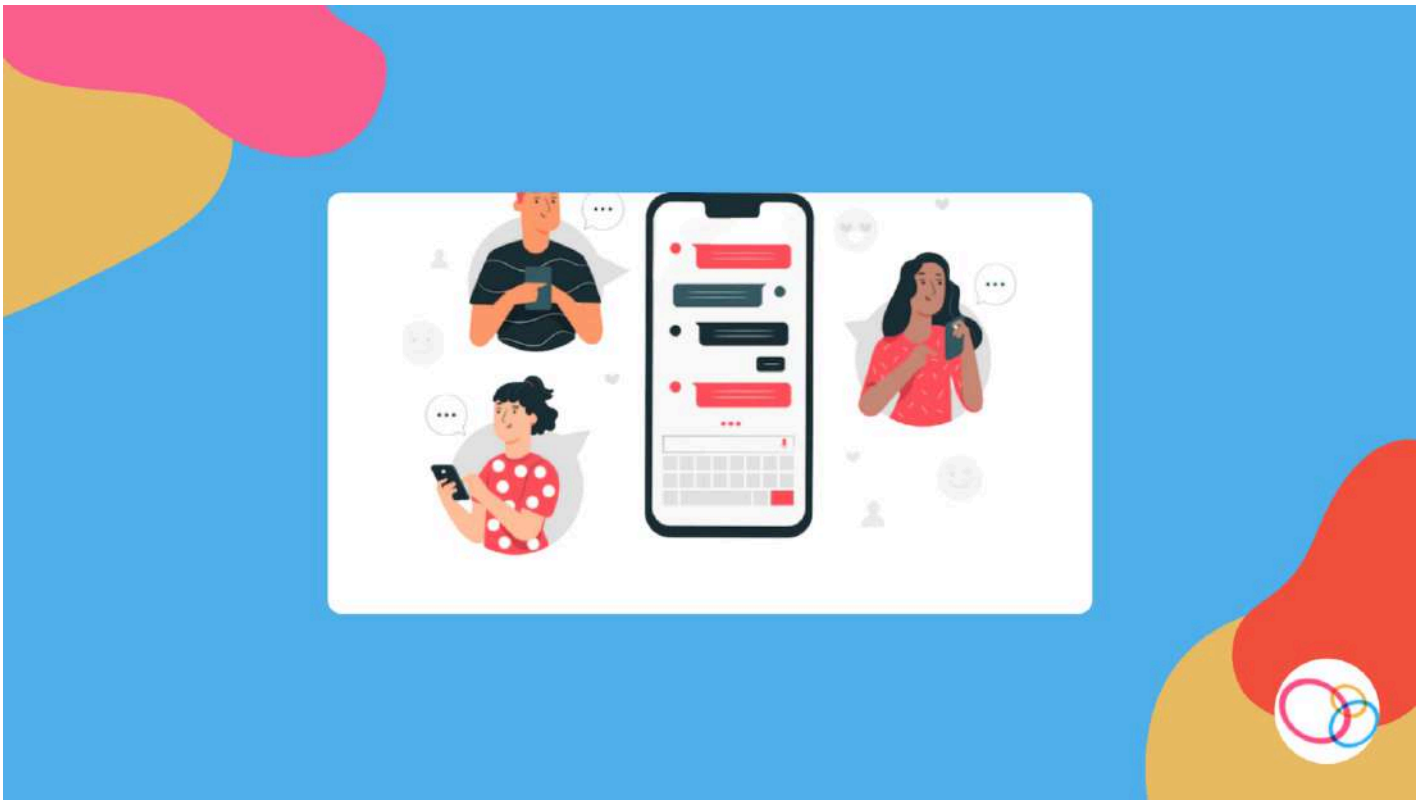
> **Clarity helps you find focus and direction**.

To organize the world you live in, you must organize your mind first. Even though pressure can be good after a long time it clutters your mind. Soon after that, you notice that you're underperforming or unhealthy.

To clear your mind means to empty your head of all thoughts in order to see things more clearly. This is a common practice after spending several hours or days solving a particular problem.

Without organization, you're not able to clear your mind.

4. Cut off distractions



After developing a plan, it's time to reduce distractions to increase focus.

Common distractions for people working from home are family, roommates and pets. Also, the human mind wanders, and we're more excited to complete non-work-related activities**. If you struggle with this, Neil Gaimans technique might help you.** Start by sitting in front of your computer. The main rule is: "You can do nothing or work.". Soon you'll realize that typing anything is much more interesting than staring out the window for 30 minutes straight.

After a long work session, take a break. Breaks are important.

5. Working smart (but putting in the work)

By definition working smart means finding the few activities that matter the most and going all-in on those to create the best possible results.

But this doesn't mean finding all possible shorter ways to actually complete a task or to meet a deadline. **Sometimes those shorter paths lead us to more work and sloppy results.** To work smart means to follow all the steps described above.

Create a plan, set a deadline, clear your mind and cut off distractions.

Resources

I have already written some articles that talk about productivity and motivation. I hope you find them useful too. **Check them out:**

- Books that changed my life
- Reaching your (coding) goals
- Why should everyone learn to code
- How to be a better programmer

Summary

Today we talked about...

1. Why is organization important?
2. Making a plan
3. Setting deadlines
4. Decluttering the mind
5. Cutting off distractions
6. Working smart

I hope you find the information in this article helpful.

Check out my Instagram: @thelitcode

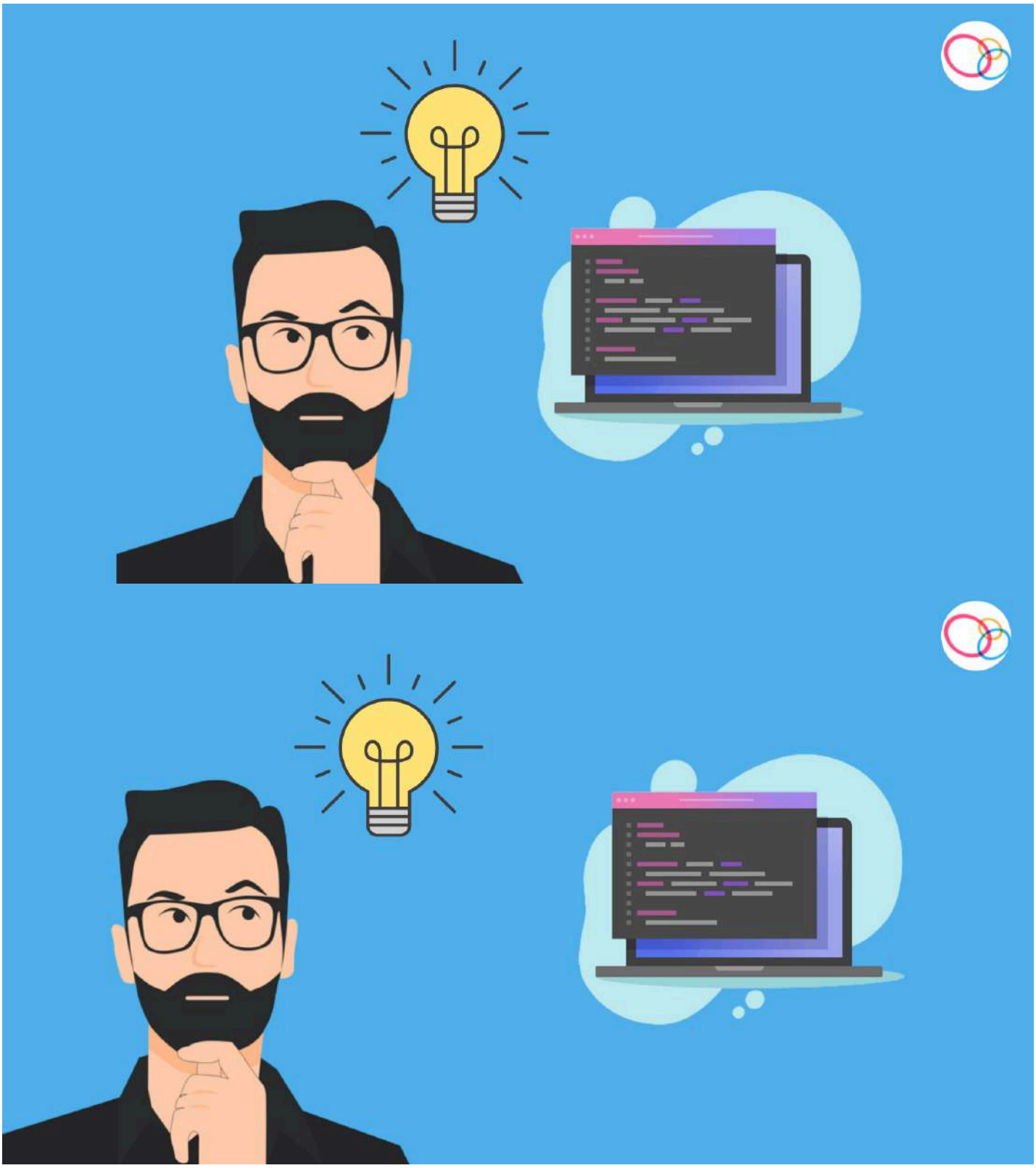
Happy coding! – Irma

How to think like a programmer?

April 12, 2022

Categories: Let's learn, Let's solve

Tags: 100DaysOfCode, Coding, Goals, LitCode, ProblemSolving



In today's article, we'll cover how to Think like a programmer. This is the question I asked myself all the time when I first started coding. And this question is completely valid. Learnin w to think like a programmer is crucial for your success. Not all of us have started the right way and

we're definitely not born as amazing problem solvers. On this life journey, we learn how to solve problems.

Today I'm here to make your programming journey easier. **I'll talk about the definition of problem-solving and we'll cover one of my favorite riddles! Most of the knowledge I'm sharing with you today I learned from the amazing book Think like a Programmer by V. Anton Spraul.**

Now, let's start!

The struggle is real

I'm going to be honest with you, you will struggle. The sooner you realize that and accept it the better your life will be. In programming problems and challenges are what make it interesting. If you're not creative and not willing to put in the work you'll struggle more.

In the beginning, after just learning the syntax you'll expect to magically know how to code for NASA. That is just not possible. **Learning the syntax is not learning how to code.** It is just one part of it. The real beauty of coding is the creative, problem-solving process.

Programming is about creativity, putting in the work and sticking through the hard times. But in the end, isn't everything in life just like that?

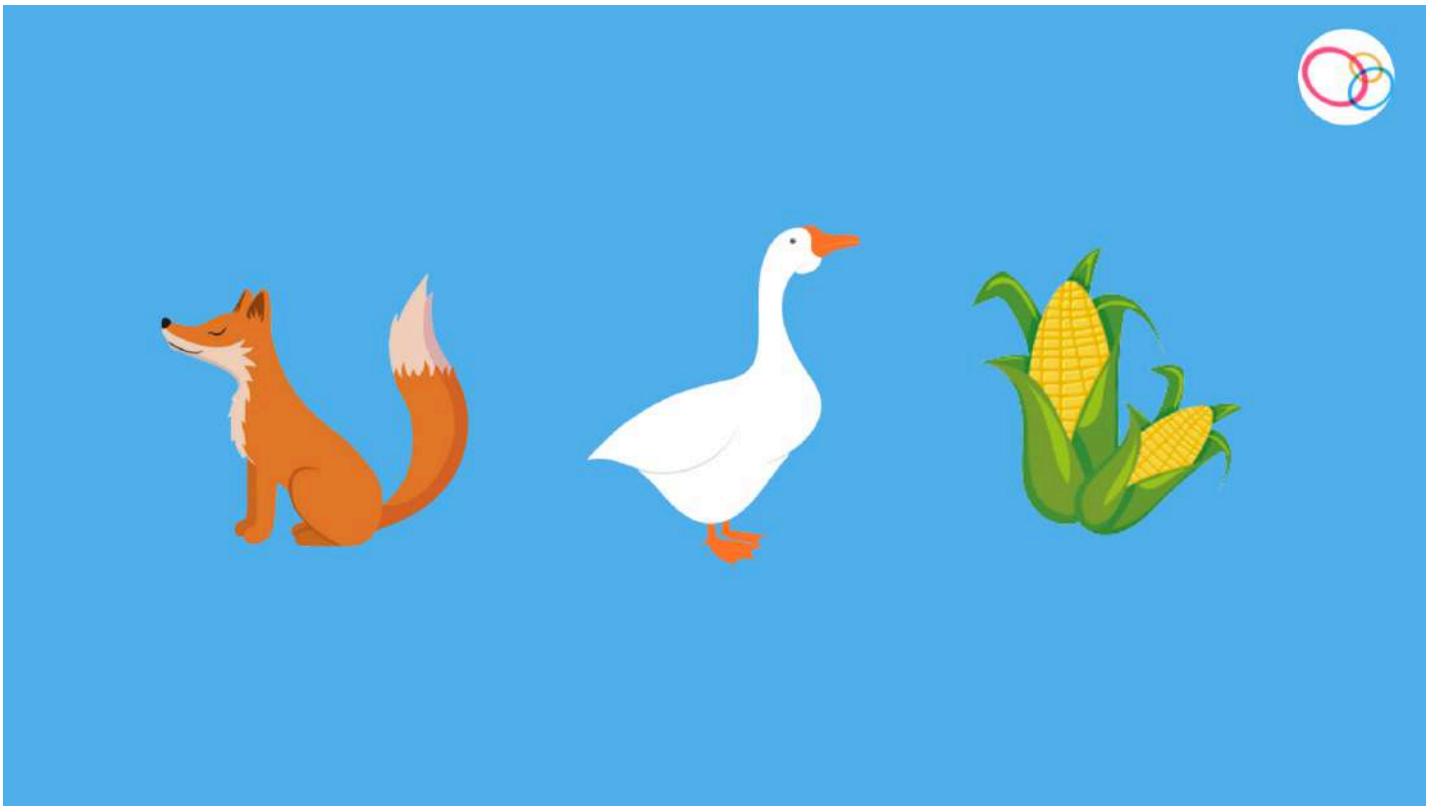
Definition of Problem Solving

Before we dive into the riddle, we need to understand problem-solving.

Problem-solving may be defined as the creation of an original program that executes a certain set of tasks while following all stated limitations. In the words of V. Anton Spraul, a program that does only the first part is a Kobayashi Maru.

This is a reference to Star Trek, but in summary Captain Kirk one of the characters in the trilogy did not solve the dilemma of Kobayashi Maru; he avoided it. **Avoiding the problem is not solving it.** It can work for you a couple of times, but it causes earthquakes later.

The Fox, the Goose and the sack of corn



Problem statement

A farmer with a fox, a goose, and a sack of corn needs to cross a river. The farmer has a rowboat, but there
is room for only the farmer and one of his three items. Unfortunately, both the fox and the goose are
hungry. The fox cannot be left alone with the goose, or the fox will eat the goose. Likewise, the goose
cannot be left alone with the sack of corn, or the goose will eat the corn. How does the farmer get everything across the river?

Possible outcomes

Because the farmer can only carry one object at a time, he'll have to make repeated journeys to the distant shore to get everything. If the farmer takes the fox on the first trip, the goose will be left with the sack of corn, which the goose will eat. Similarly, if the farmer took the corn sack on the first trip, the fox would be left with the goose, which the fox would eat.

So is there a solution?

Steps that lead to solution

From now on, remember this to solve problems.

If you're unaware of all the possible actions you could take, you may be unable to solve the problem. All of this is going to become clearer in a second.

LIST THE CONSTRAINTS – don't be Captain Kirk, focus on the problem.

In our problem the constraints are:

- The Farmer can only take 1 item in the boat at a time
- The Fox and the Goose can't be left alone on the same shore
- The Goose and the corn can't be left alone on the same shore

LIST THE OPERATIONS – what you need to do in order to form a solution. Actions = Operations.

- Row the boat from one shore to another
- If the boat is empty, load an item from the shore
- If the boat is not empty, unload an item to the shore

Solution

Steps:

1. Get the duck to the FAR shore
2. Come back to the NEAR shore and take the fox to the FAR shore
3. Take the duck back to the NEAR shore and get the corn to the FAR shore
4. Come back for the duck

Most of the people failed to solve this riddle because of this last step. They were simply unaware that you CAN take an item back.

Resources

One of the best recommendations I'll ever give you on LitCode is Sprauls book. Check out:

- Think like a programmer – C++ Edition
- Think Like a Programmer – Python Edition

Also, you can check out my articles where I cover problem-solving:

- Solving challenges on CodinGame
- Sieve of Eratosthenes
- Putting coding games to the test

Summary

Today we covered:

- Programming struggles
- Definition of Problem-Solving
- The Fox, the Goose and the sack of corn
- Steps that lead to a solution

I hope you find this article helpful, and I can't wait for you to read my next post!

Happy creating. – Irma

Why is creativity important?

April 28, 2022

Categories: Let's read

Tags: 100DaysOfCode, Coding, Creativity, LitCode, Passion

Welcome to Why is creativity important! After a break, I am back with my regular posts.

In today's post, I'll show you the true value of working on your creative skills. You'll see that programming is more similar to design and art than maths. This is the most common misconception that I heard people say about programming.

Now let's get started!

What is creativity?

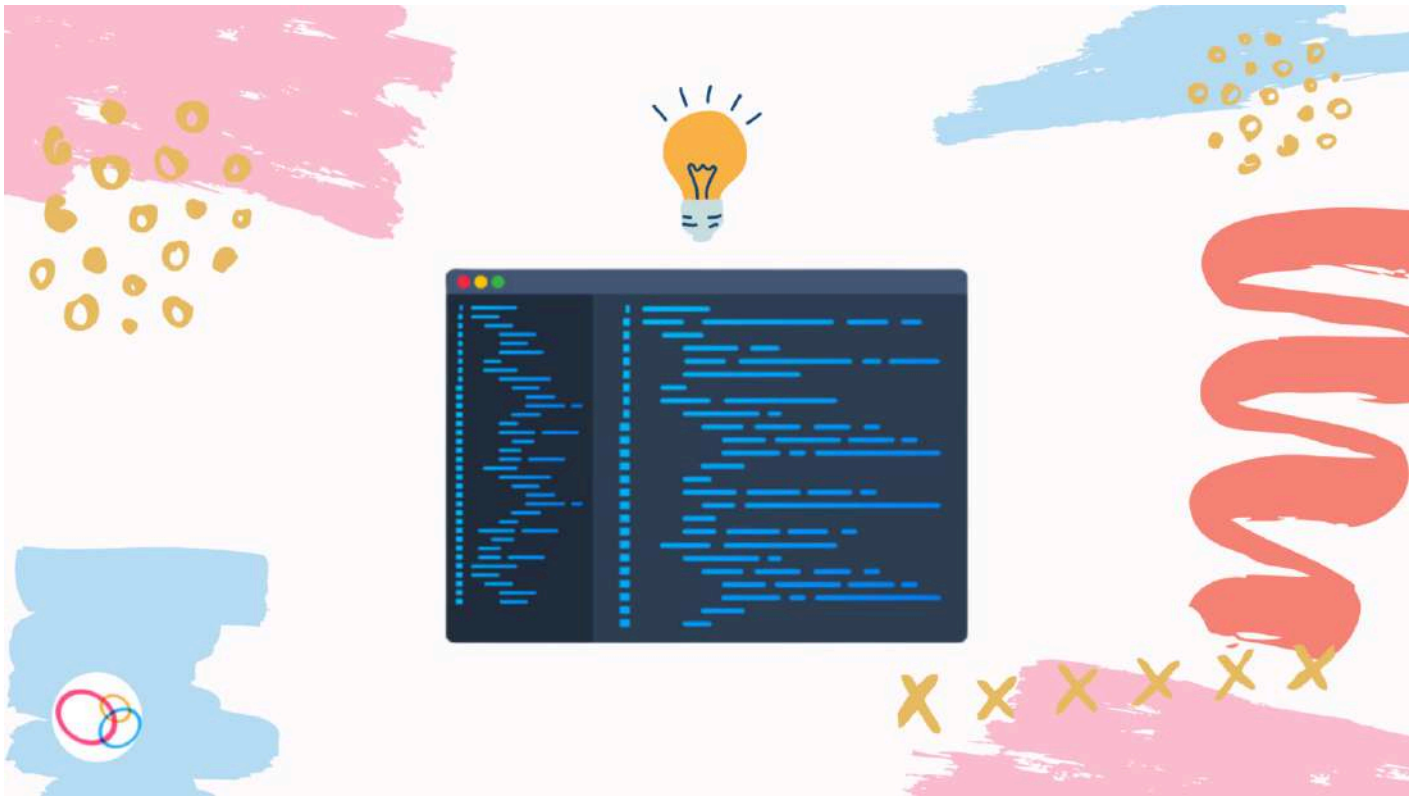
By definition, creativity is:

- **the use of imagination or original ideas to create something; inventiveness.**

The keyword here is CREATE. And this is the best and the simplest way to describe what it is. Creativity is all about expressing yourself. Creativity is the ability to come up with new and exciting ideas. It is not strictly painting or drawing.

You are **CREATING** and **EXPRESSING** even while programming. That is done by solving problems, creating mobile apps, games and much more!

What does it mean to be creative?



The ability to develop or identify ideas, alternatives, or possibilities that may be beneficial in solving issues, interacting with others or amusing ourselves and others are referred to as **creativity**.

All of these skills are very important if you are considering a career in tech. Programmers are not robots that solve problems mechanically. **Every part of programming is creative.**

Creativity is considered something you are just born with. This is just not true. **Everything in life requires work to be mastered. Developing your creative skills will benefit you in every area of life.**

Programming and Creativity

Now that we cleared that out, let's get into the fun part.

Creativity motivates you

Let me give you two examples:

- You were just brainstorming for your new Mobile App development project and found the right idea. Immediately you'll feel the adrenaline boost, happiness boost and the rush. **This idea will motivate you to start working on this new project right away.**

- Your mood is down and you don't know what to do to lift it up. Music isn't helping, talking with friends and all sorts of things that usually do. What are you going to do? **Simple activities that challenge your creativity and push you to be creative lift your mood up instantly.**

Helps with problem-solving

> "You can't use up creativity. The more you use the more you have."
>
> Maya Angelou

This is the most crucial skill you can have as a programmer. And I am not referring to mathematical problems. I am referring to real-life and work challenges.

You can't solve a problem without being creative about it. By developing and working on your creative skills, you are also developing your problem-solving skills. By doing simple riddles and common challenges, you are boosting both parts.

Through creativity you express yourself

Everyone is facing stress everyday. One way to destress or pause and think is by doing creative activities. You don’t need to be Picasso to be creative. **There is no one right way to do it.**

You can do simple activities like:

- doodling
- reading
- cooking
- journaling
- writing
- brainstorming
- music

etc.

Clarifies thoughts

By doing the activities mentioned above, you are expressing yourself. Expressing yourself through something you enjoy clarifies your thoughts and organizes them. And there are endless benefits to this.

After getting your mind organized, you are making better decisions, being more productive and feeling better in general.

While working on a big project or solving a problem for several hours my best advice is to pause. Stop and do something you enjoy. Do something that develops your creativity.

Resources

If you liked this article, check out my similar ones:

- How to get organized?
- Books that changed my life
- Reaching your (coding) goals.

Summary

Today we covered:

1. What is creativity?
2. What does it mean to be creative?
3. Programming and creativity
4. Creativity and motivation
5. How creativity helps with problem-solving
6. Expressing yourself through creativity
7. Clarifying your thoughts

I hope you find this article helpful, and I can't wait for you to read my next post!

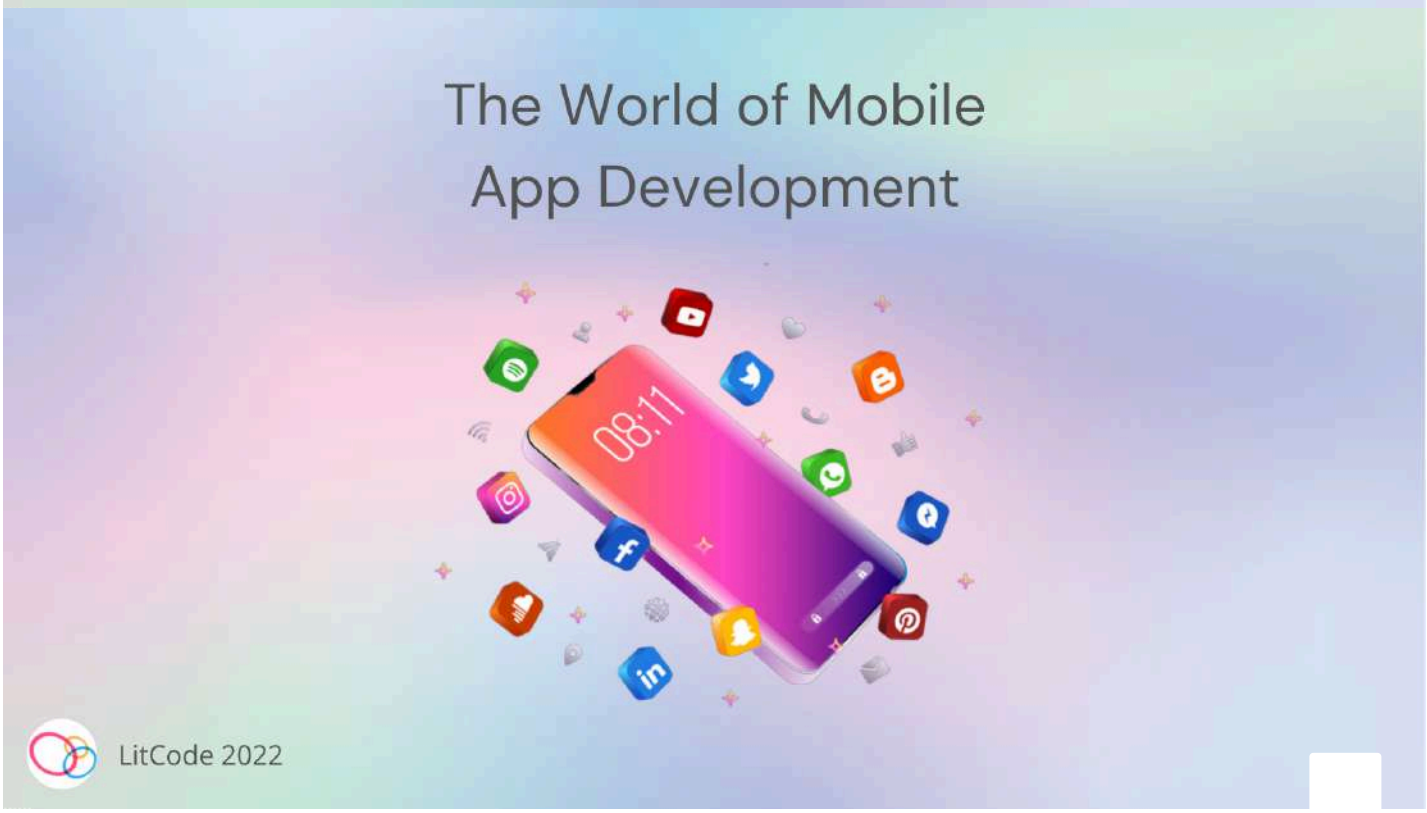
Happy creating. – Irma

Mobile Apps to develop this Summer

May 6, 2022

Categories: Let's create

Tags: 100DaysOfCode, Coding, Goals, LitCode, Mobile App Development



Welcome to **Mobile Apps to develop this Summer!** In todays article, we'll dive deeper into the Mobile App Development World. **I'll give you 7 app ideas for you to try out.**

These aren't unique App ideas, you can already find hundreds of Apps just like the ones mentioned in this article. But the most important thing about these projects is, that it pushes you to practice. In programming, the theory doesn't matter. **You only learn by actively doing.** These projects are a fun way to test your skills and learn something new!

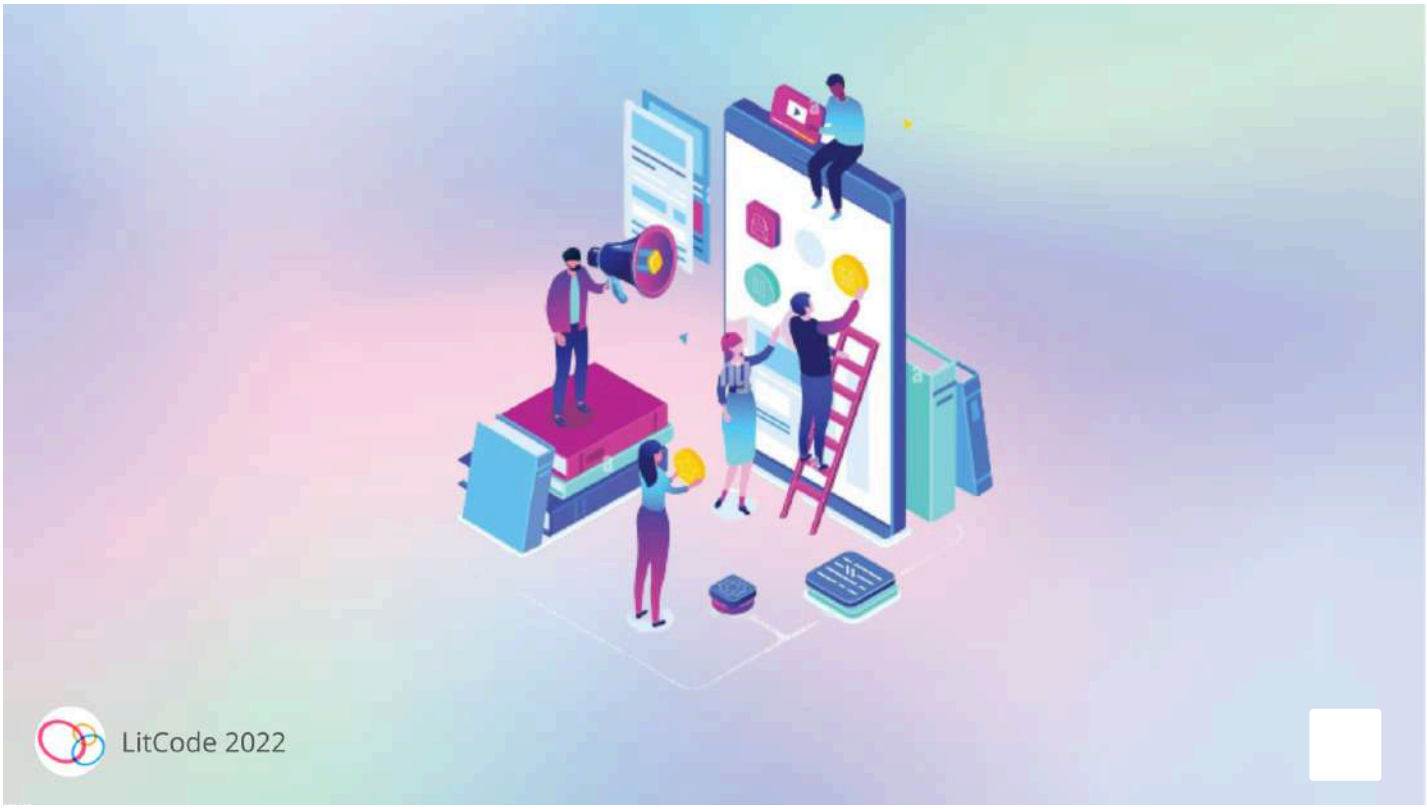
Now let's get started...

Mobile App Development world

By definition Mobile App Development is...
The process of developing software applications for mobile devices is known as mobile application development.

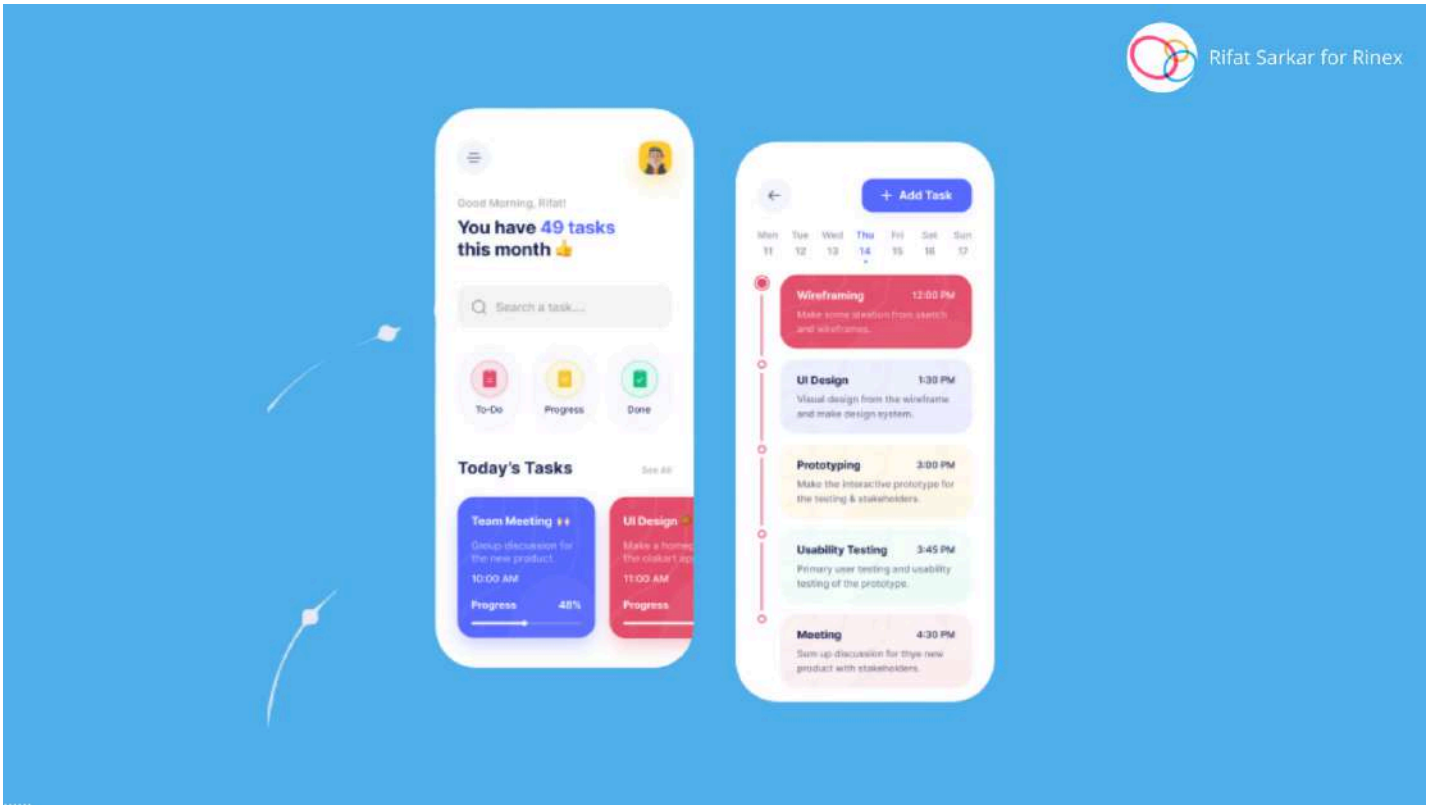
A Mobile App Developer does exactly this. But the world of app development is much more complex. The best part is that, even if you're working for someone you are using your creative side fully.
Nowadays, the job market for this role is huge.

7 App ideas for this summer



Now, I'm introducing you to 7 app projects that I believe are a tremendous way to test your skills. **While working on these projects, you get to revise concepts already learnt and learn new ones.** The apps mentioned below are apps you definitely already have been introduced to.

1. ToDo App



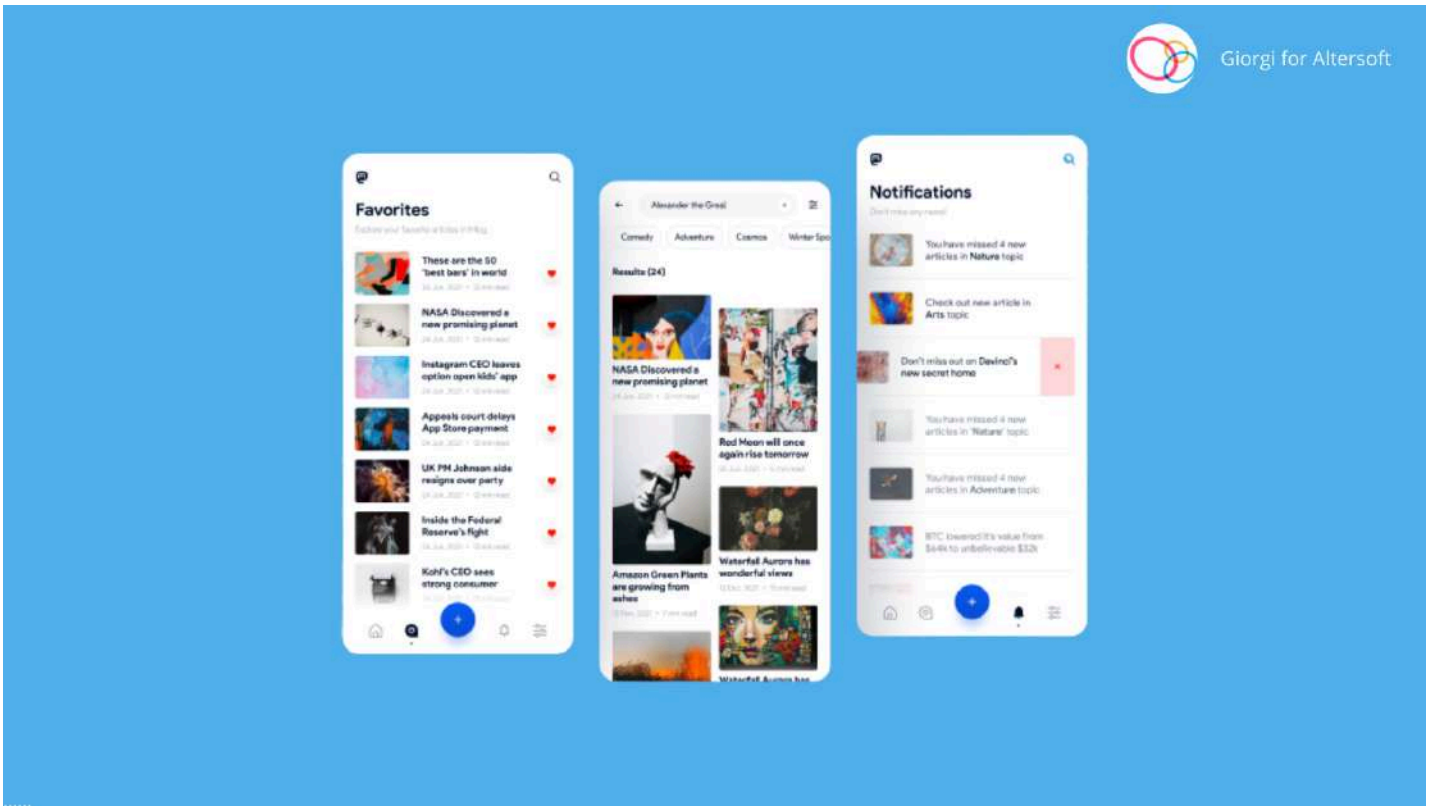
PROJECT OUTLINE:

- Maintains our day-to-day tasks or list everything that we have to do, with the crucial tasks at the top of the to-do list, and the least important tasks at the bottom. It plans our daily schedule.

FEATURES:

1. Authentication flow (email, google, apple, facebook)
2. Profile page
3. My Calendar
4. Todays tasks
5. Upcoming tasks
6. Urgent
7. Trash

2. Blog App



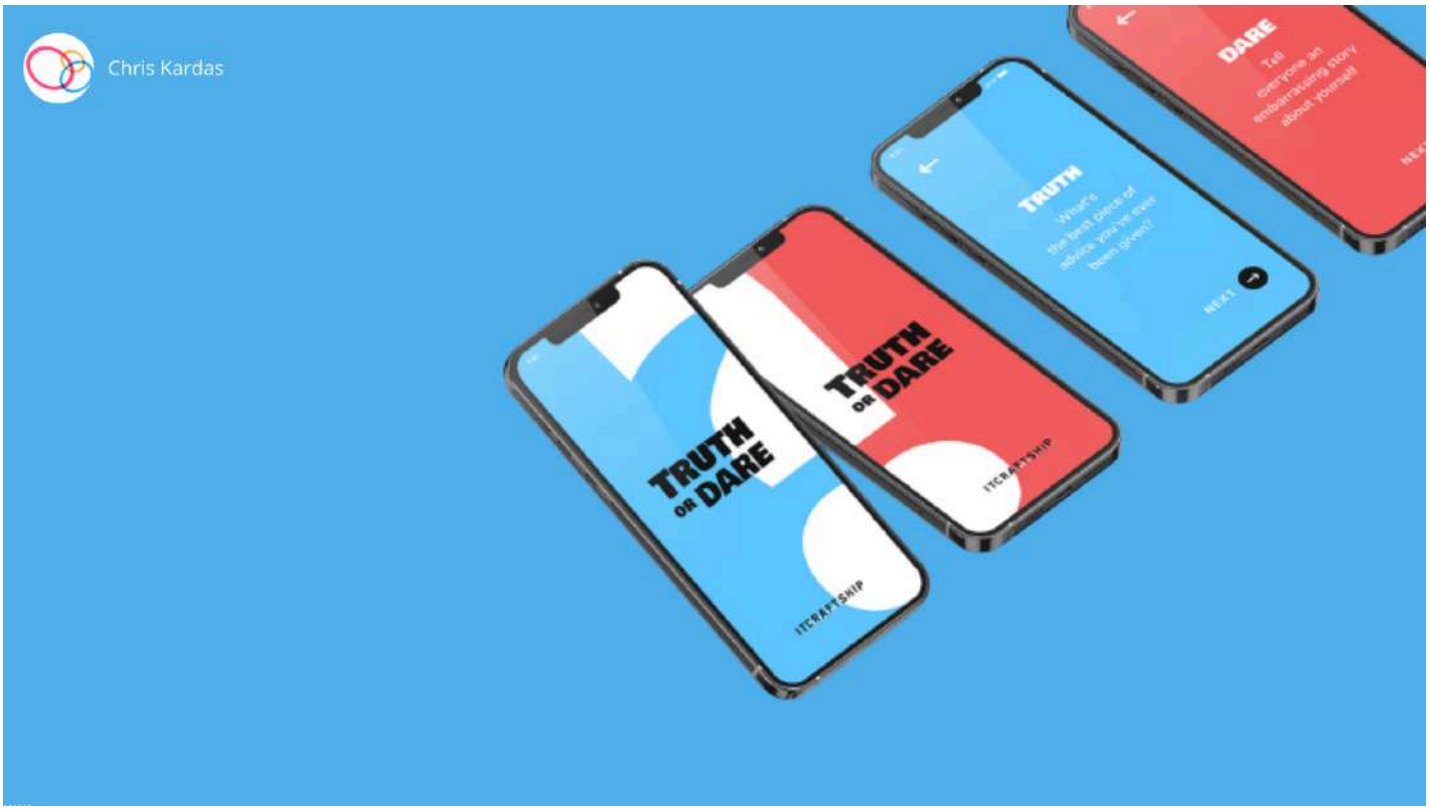
PROJECT OUTLINE:

- Used to publish and discover blog posts on diverse subjects. The project can also be converted to a specific theme. User can follow accounts and their posts. Search bar for discovering new users and posts.

FEATURES:

1. Authentication flow (google, email, facebook, apple)
2. Profile Page – displays My Posts
3. Search
4. Explore Page
5. My Feed – displays posts from followed accounts

3. Truth or Dare/Spin the bottle



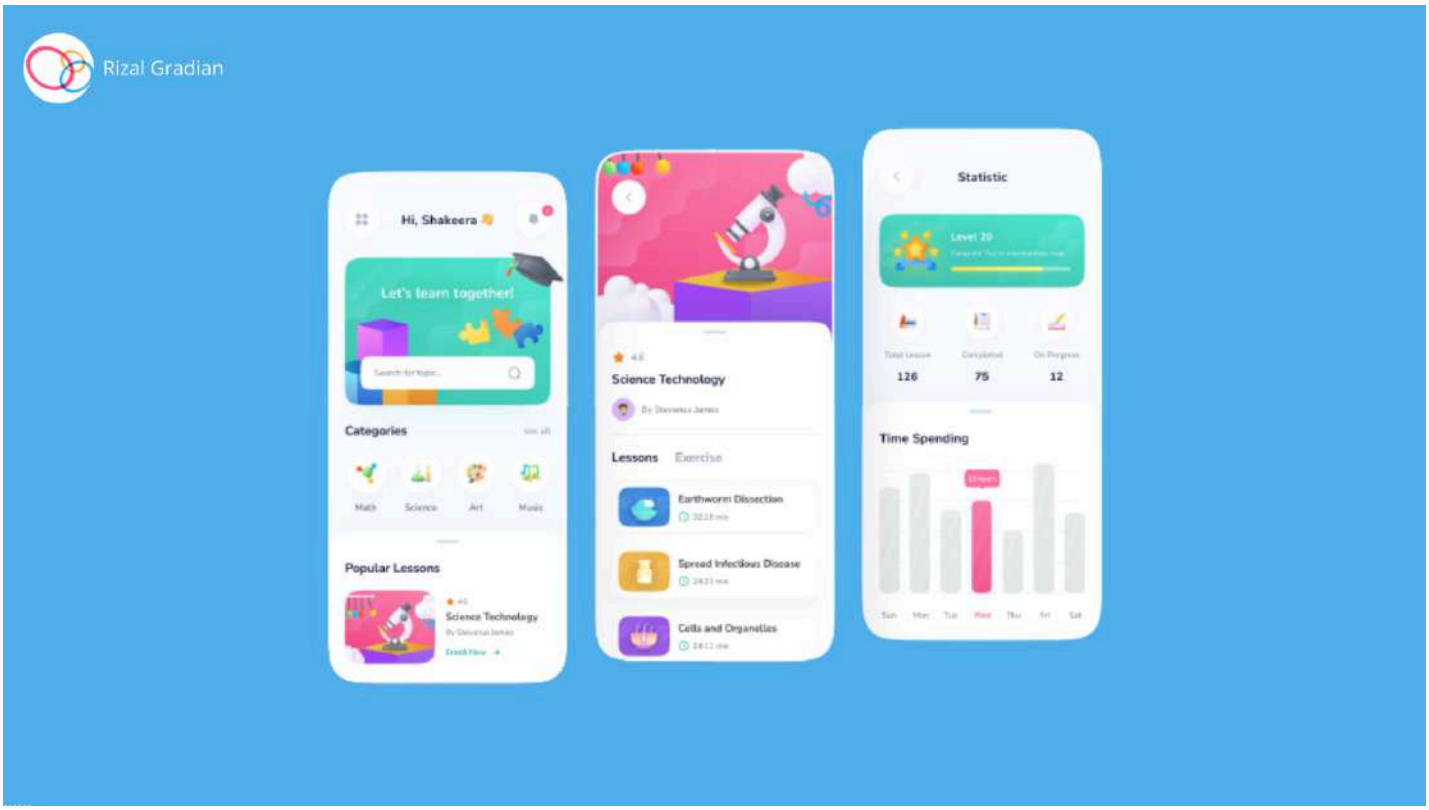
PROJECT OUTLINE:

- A fun way to play spin the bottle/truth or dare. The user creates the account, can create its own questions and share them with the world. Also separate page for the card groups that come by default. Explore and Search to discover new sets of cards.

FEATURES:

1. Authentication flow (email, google, facebook, apple)
2. Profile Page – displays your Question Cards
3. Card Groups available Page
4. Explore Page
5. Search

4. Quiz App



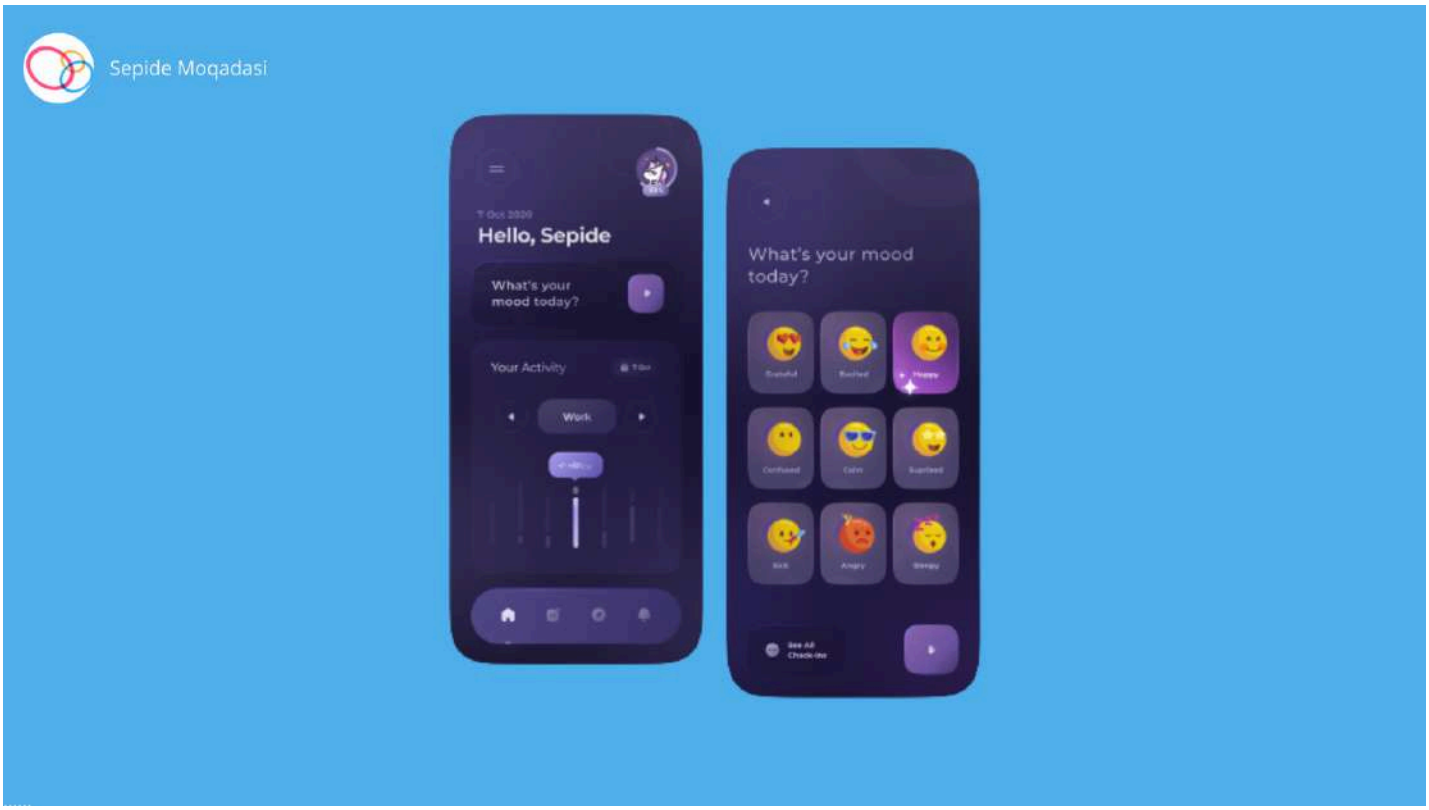
PROJECT OUTLINE:

- The user creates an account and can play and compete with other users and friends. Can create its quizzes and share them with the world. Profile page which displays the progress on already played quizzes and ranking.

FEATURES:

1. Authentication flow (email, google, facebook, apple)
2. Profile Page – displays your Progress on already played quizzes
3. My Quizzes – created custom
4. Explore Page
5. Search

5. Mood Log App



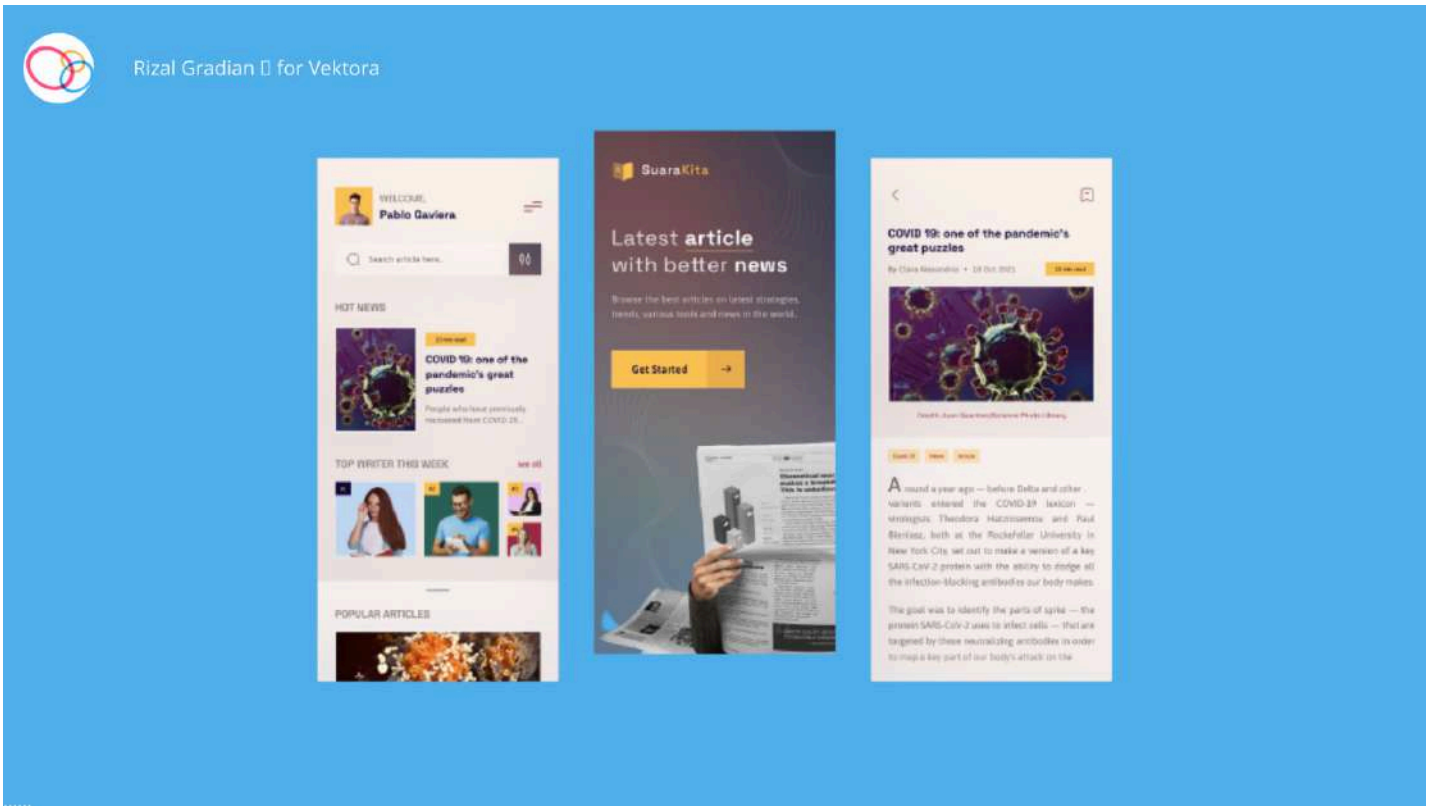
PROJECT OUTLINE:

- An App used to track your mood. Can be used also as a mobile journal for mood tracking. After entering data and answering the props user jumps to the advice section.

FEATURES:

1. Authentication flow (email, google, facebook, apple)
2. Profile Page – displays today's mantra
3. My Calendar – displays all your entered data
4. Todays mood tracking page
5. Advice section – Breathing exercises

6. News App



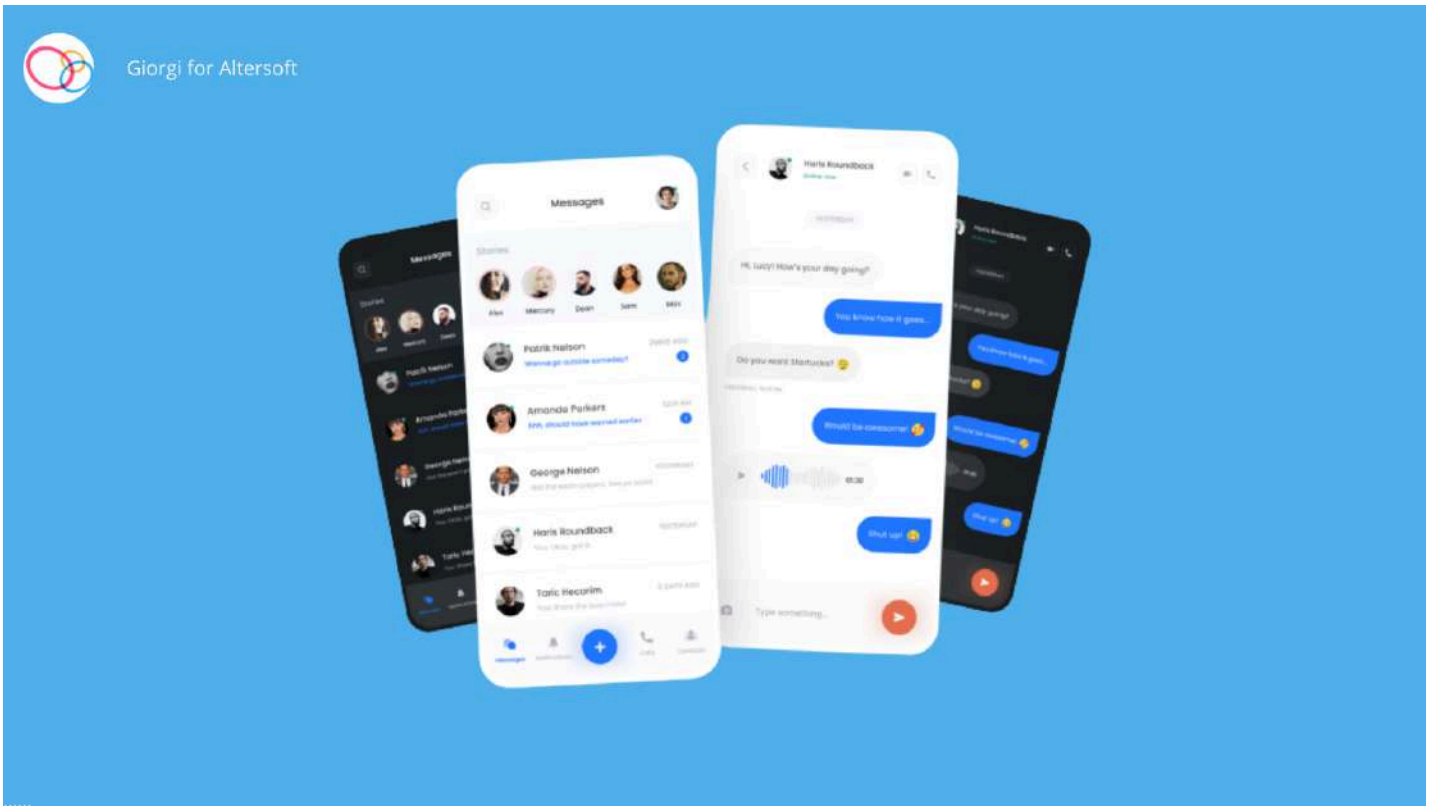
PROJECT OUTLINE:

- A regular news app that shows updated information every day. Users can follow other accounts that post relevant information. Explore and Search Page to discover the content users like.

FEATURES:

1. Authentication flow (email, google, facebook, apple)
2. Profile Page – displays users' information
3. My Feed – shows articles from the accounts followed
4. Explore Page
5. Search

7. Messaging App



PROJECT OUTLINE:

- A chat application or platform that enables users to instant message and connects through their mobile devices. For inspiration, you can use Messenger or WhatsApp.

FEATURES:

1. Authentication flow (email, google, facebook, apple)
2. Profile Page – displays users' information
3. Recent Page – your recent chat history
4. Adding new groups functionality
5. Audio Calls option

Resources

The beautiful app designs shown in this article were found on Dribble. To learn more check out Dribble official website.

If you liked this article you might also like:

- Why is creativity important?

- Introduction to Flutter
- Developing with Flutter – part II
- Developing with Flutter – part III

Summary

Today we covered:

1. Mobile App Development World
2. Presenting 7 App Ideas for this summer
3. ToDo App outline
4. Blog App outline
5. Truth or Dare App outline
6. Quiz App outline
7. Mood Log App outline
8. News App outline
9. Messaging App outline

I hope you find this article helpful, and I can't wait for you to read my next post!

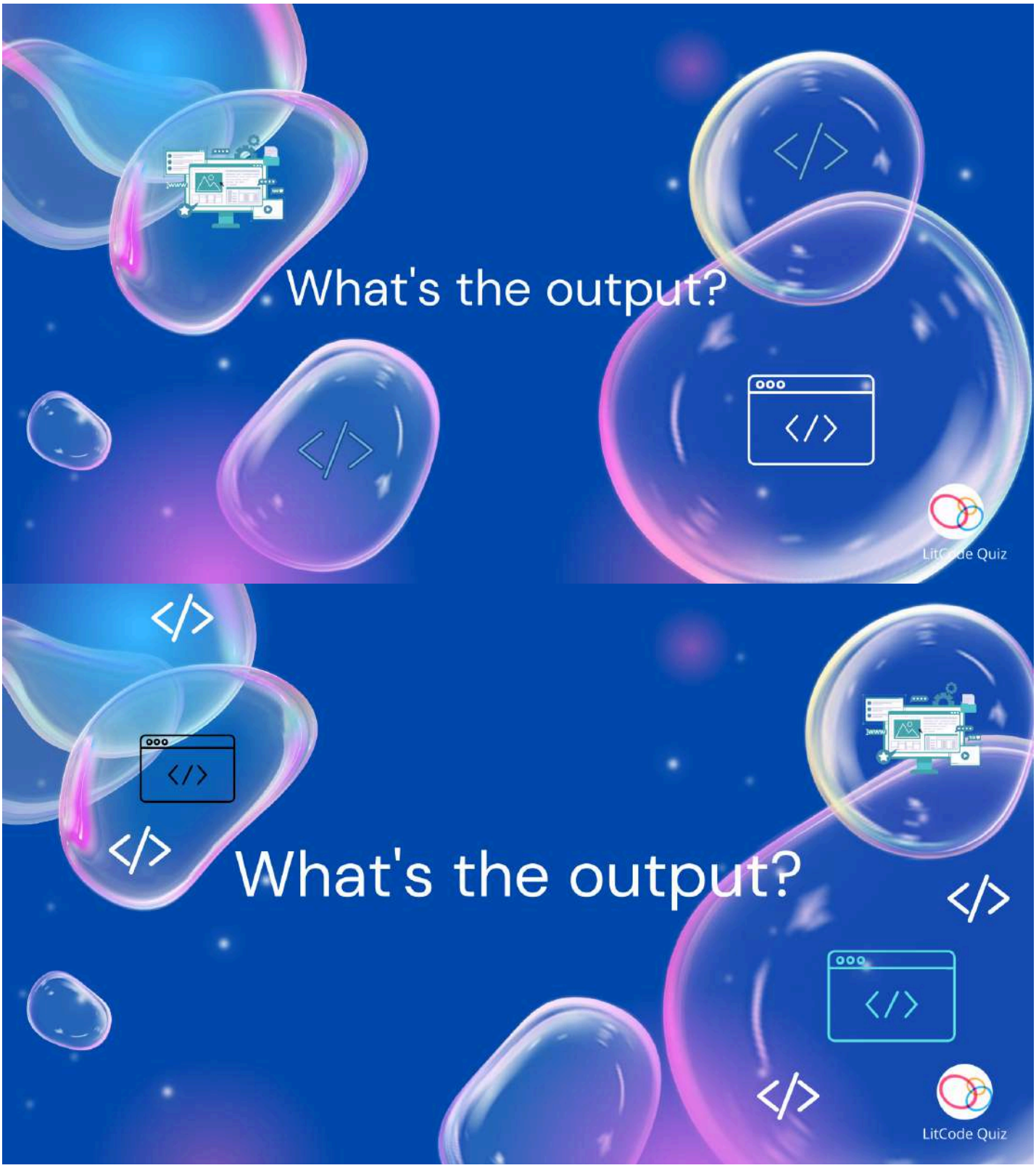
Happy creating. – Irma

C++ Quiz!

May 13, 2022

Categories: Quiz

Tags: 100DaysOfCode, Goals, LitCode, Quiz



Welcome to What's the output? – Quiz. In today's quiz, we'll cover some of the most common concepts in C++. Besides simple questions and code blocks, there will be more challenging s to test yourself. After completing the quiz you'll be able to check your results!

The most important things about this quiz are **the revision** and **just practice.**

You can learn all the theories but without actual coding, you'll never learn to code! The best way to master programming is to cover theory, check code blocks and make your programs.

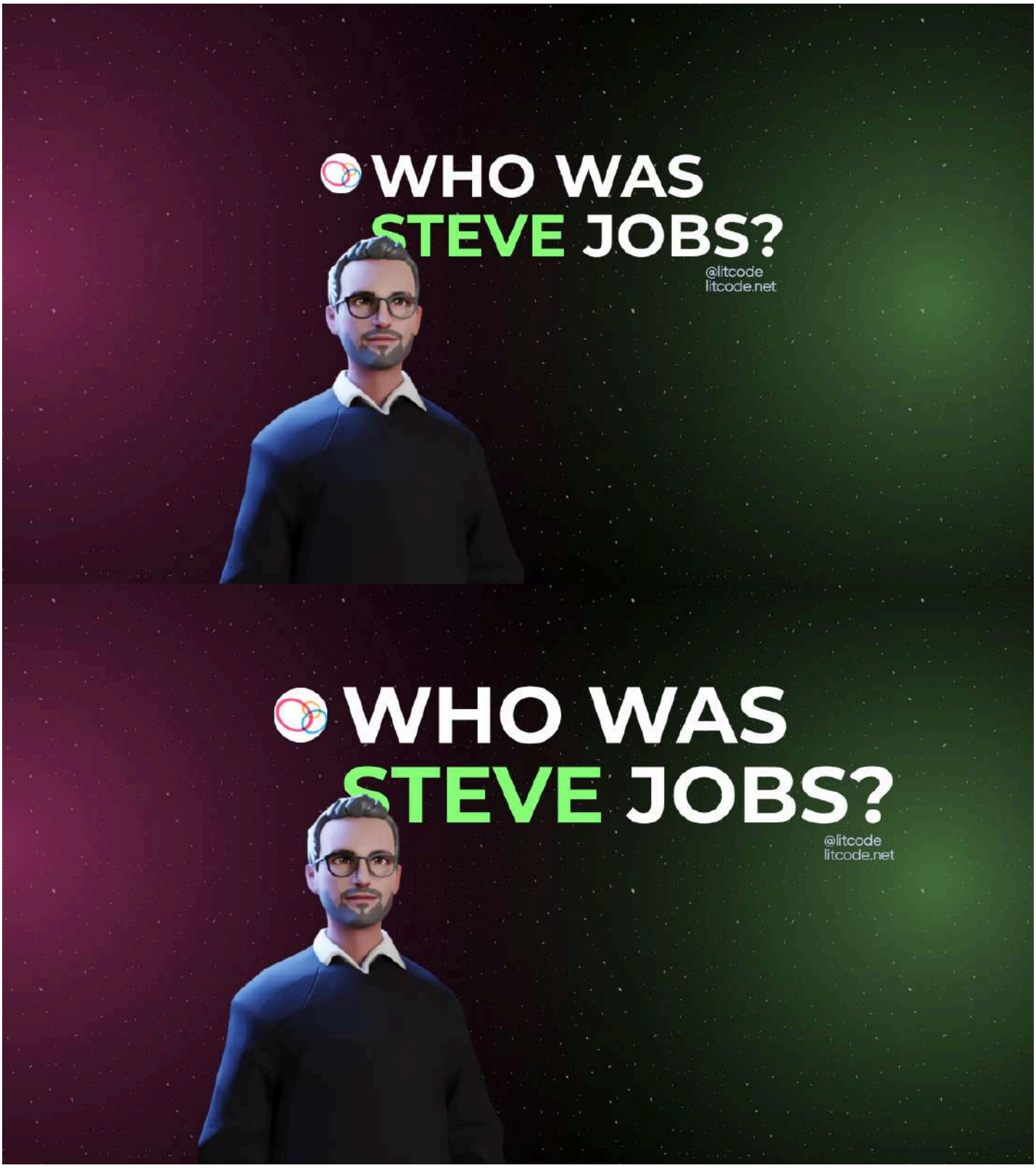
Now let's get into the quiz...

Who was Steve Jobs?

June 5, 2022

Categories: New

Tags: Apple, Computer Science, Life after Death, LitCode, Steve Jobs



Welcome to my new article! After a short break, I am back with regular posting. Today we're talking about Who was Steve Jobs? We're taking a brief look at the biography of this legendary cre and his impact on the tech industry.

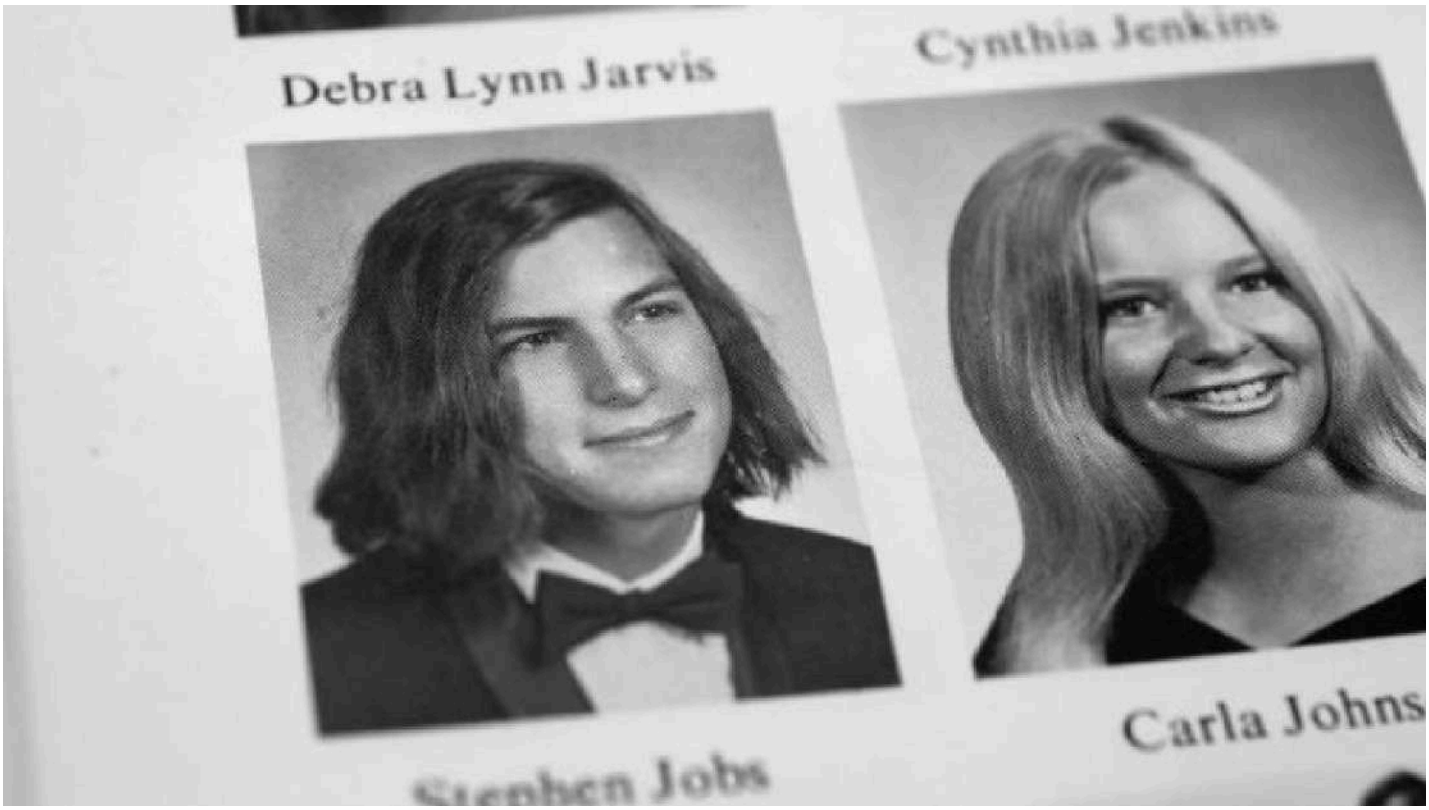
Instagram posts on @thelitcode are back with a new, fresh design! Make sure to check it out.

Now let's jump to the article!

Early days...

Steven Paul Jobs was born in San Francisco, California, on February 24, 1955. Joanne Schieble and Abdulfattah Jandali, his unwed biological parents, placed him for adoption. Paul and Clara Jobs, a lower-middle-class couple, adopted Steve and relocated to Mountain View, California, a few years later.

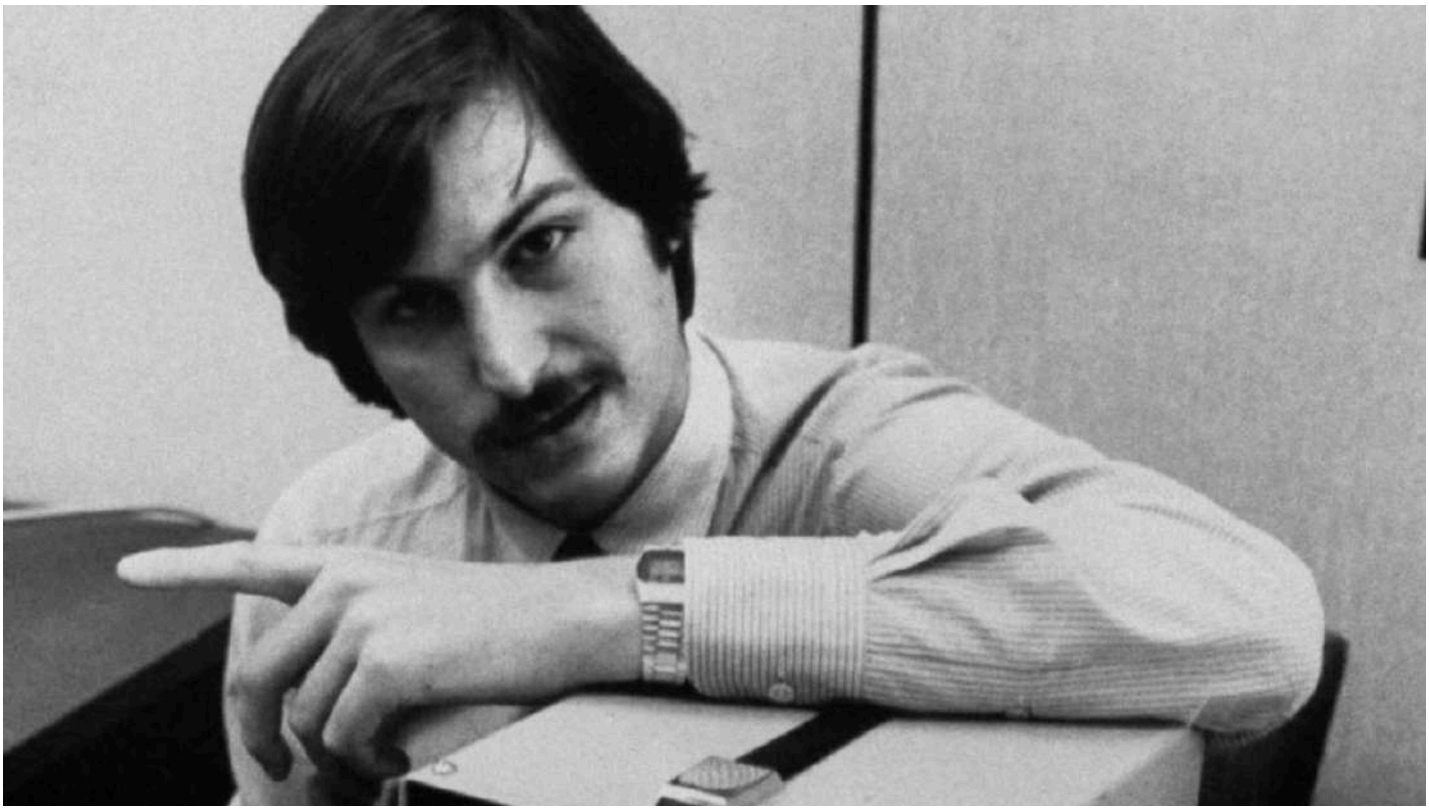
After the emergence of one of the many semi-conductor enterprises in Santa Clara County, south of the Bay Area, the area became known as Silicon Valley in the early 1950s. As a result, a young Steve Jobs grew up in a neighborhood full with engineers who spent their weekends working on electronics and other devices in their garages.



When Steve Jobs was ready for college, he told his parents that he intended to attend Reed College, an expensive liberal arts college in Oregon. **Despite the fact that the tuition costs were exorbitant, the poor couple had promised their son's biological parents that he would receive a college education, so they gave in.**

Steve only stayed at Reed for one semester before dropping out because he was more interested in eastern philosophy, fruitarian diets, and LSD than the courses he was taking. He relocated to a hippy commune in Oregon, where he focused on growing apples.

The Big Bang



Apple Computers, Inc. was founded on April 1, 1976, by college dropouts Steve Jobs and Steve Wozniak, who brought with them a goal of transforming public views of computers to the new firm. Jobs and Wozniak wanted to make computers that could be used in people's homes or offices. Simply put, they desired a user-friendly computer.

Jobs and Wozniak began by hand-building Apple I computers in Jobs' garage and selling them without a monitor, keyboard, or case. With the debut of the first-ever color graphics, the Apple II transformed the computer industry. Apple went public in 1980, and sales increased from $7.8 million in 1978 to $117 million in 1980.

Jobs revolutionized computing by introducing the iBook, which was quickly followed by the iPod, an mp3 player that quickly became the market leader.

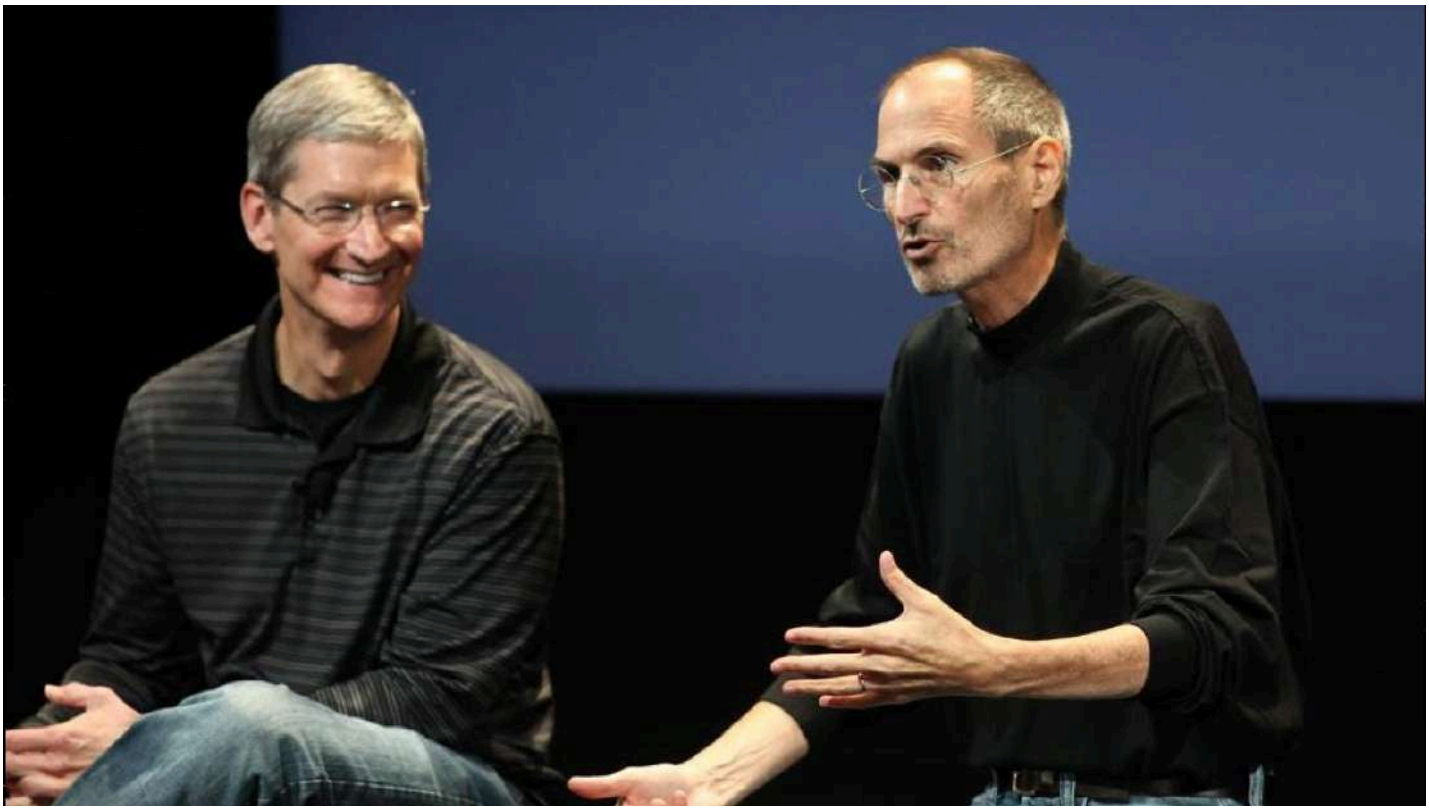
The iPhone, a touch-screen cellular phone, was one of the world's most popular devices when it was launched in 2007, and the business has subsequently developed multiple updated models.

Steve's Work

Steve Jobs and Steve Wozniak co-founded Apple Computers. The corporation pioneered a number of innovative technology under Jobs' leadership, including the iPhone and iPad.

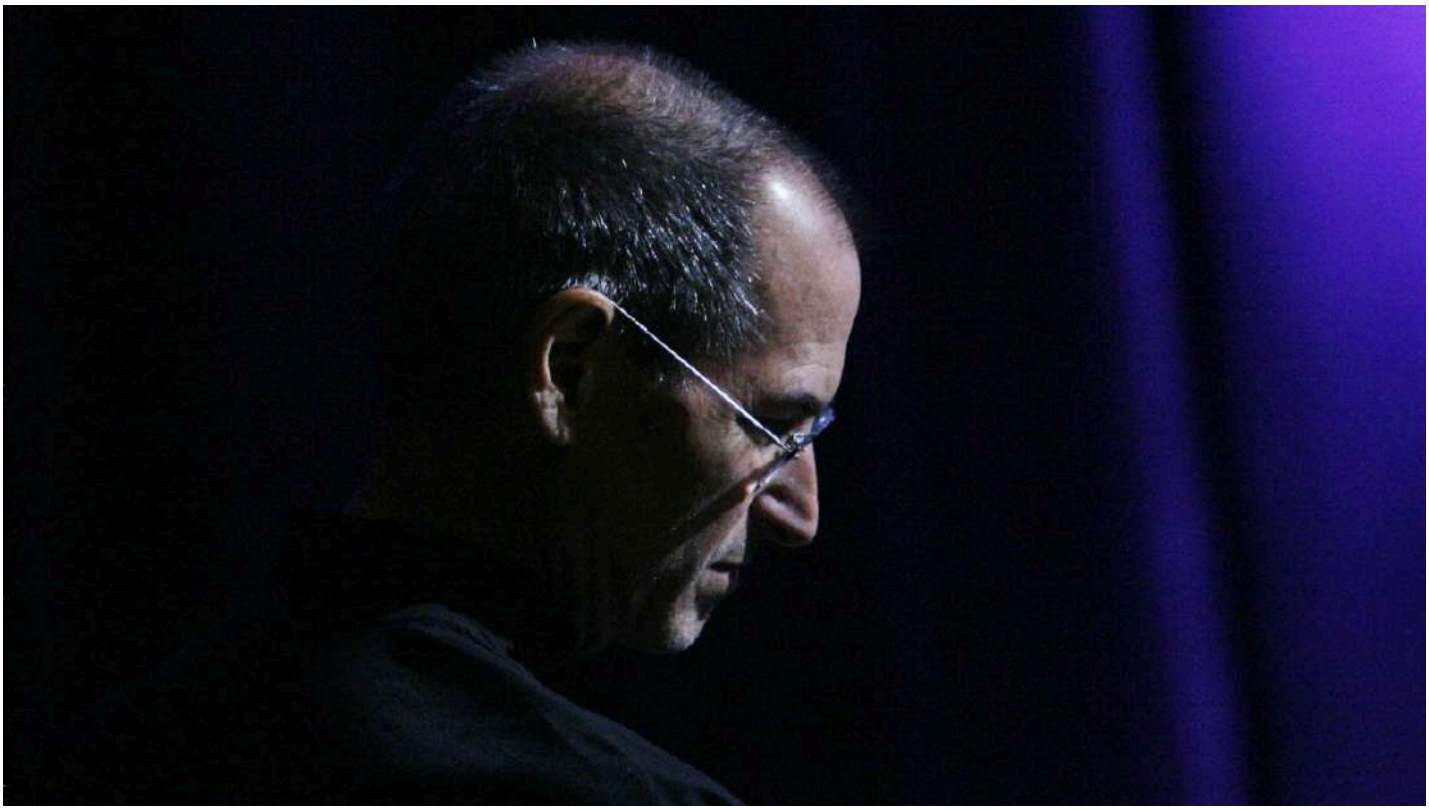
> "Don't let the noise of others' opinions drown out your own inner voice."
>
> Steve Jobs

Steve Jobs took an active role in product design, which was undoubtedly his favorite aspect of his work. He was well-known for frequently visiting the Industrial Design lab to spend time with the designers and offer advice on their prototypes.



Jobs' image as a digital visionary stems not just from Apple's consistent stream of ground-breaking products over the previous decade, but also from a comment from one of his closest associates.

Life after death



Jobs was diagnosed with a rare kind of pancreatic cancer in 2003. He postponed surgery for roughly nine months as he experimented with alternative treatment. Jobs returned to work on June 29, 2009, keeping his promise to return by the end of the month. Jobs, on the other hand, took another medical leave of absence in January 2011. He stepped down as CEO in August, but remained chairman. Two months later, he passed away.

But Steve's vision is seen every day with great products and software Apple launches and has launched. Every part of this legendary designer is seen in Apple's success today.

Resources

You can read more about Steve Jobs in the fantastic books listed below:

1. **Steve Jobs**
2. **I, Steve – Steve Jobs In His Own Words**
3. **Steve Jobs: The man who thought different**
4. **How to Think Like Steve Jobs**
5. **The Innovation Secrets of Steve Jobs**

Summary

Today we took a brief look at the iconic history of Steve Jobs. Even though Steve Jobs wasn't essentially a programmer, he had an enormous impact on the tech industry. Without him, we wouldn't be working with the modern software and products we use today.

Flutter Development – part IV

July 27, 2022

Categories: Let's create

Tags: Dicee, Flutter, Game, LitCode, Mobile App Development, Multiplayer

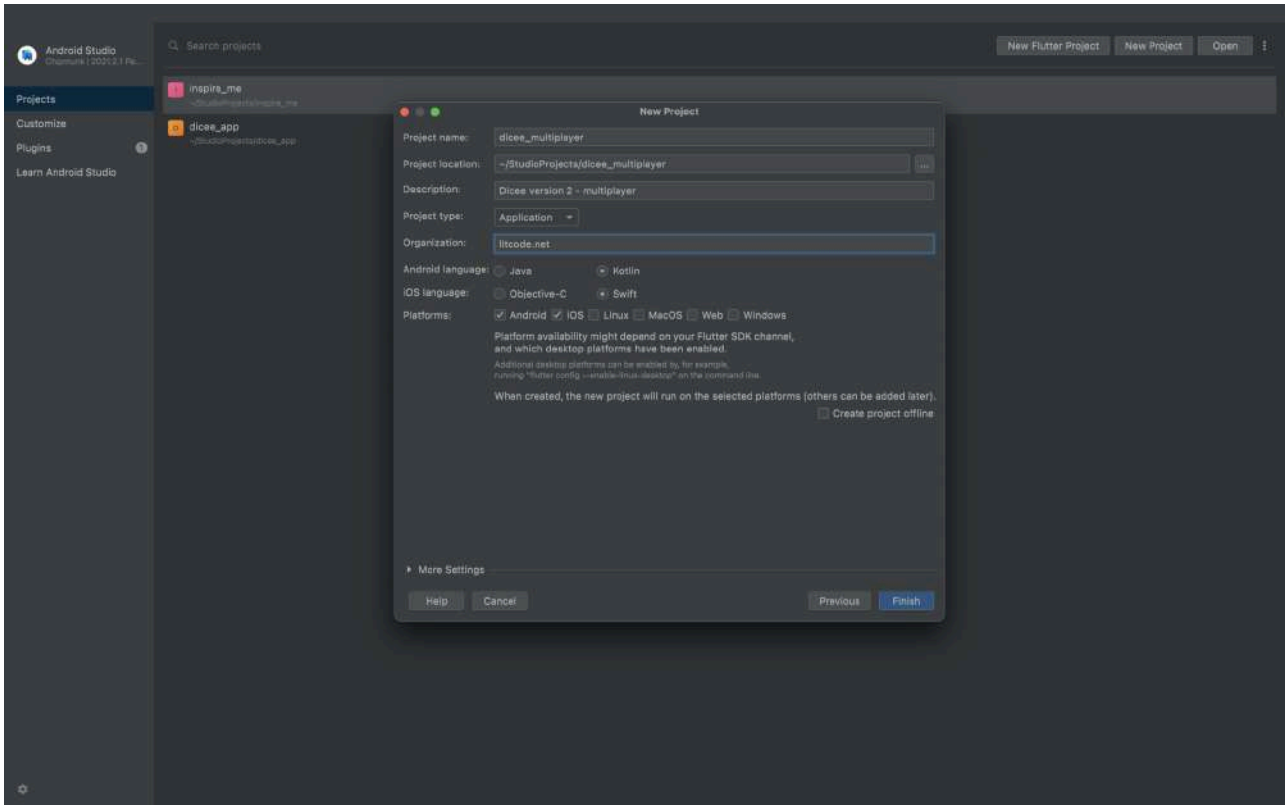
Welcome to Dicee – Developing with Flutter! Today we are working on Part 1 of the dice rolling app in Flutter. Throughout the series, I'll cover the basic logic concepts of Dart, introduce y

new Flutter widgets, cover design and more! We're starting by creating the basic skeleton of the Player 1 interface.

Let's get started!

Creating our Flutter project

We're starting by creating a new Flutter project. **Go to Android Studio and jump right into it.**



For the name of my app, I chose 'dicee_multiplayer' because previously, I created a dicee for a single player. You can add your app description to the README file. For the organization name, type your website link or simply your full name.

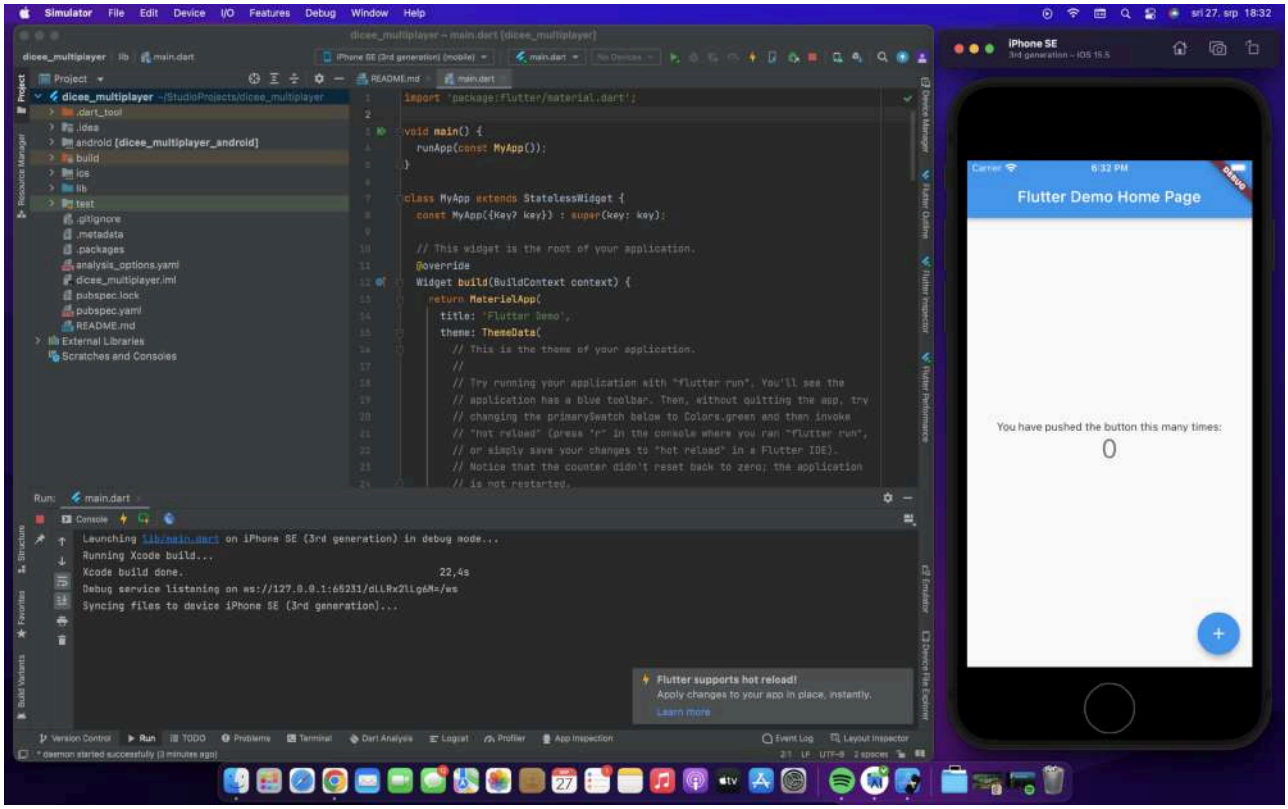
Hit next and you're all set!

Running the App on a Simulator

If you're working on Mac choose the 'Open IOS Simulator' option in the dropdown to run your app on a virtual device. In other cases, open AVD Manager and select your Simulator (process discussed while making the Inspire App series).

After that hit the 'Run' icon in Android Studio to run your app. Every Flutter app comes with previously created counter app. **This is the default app created by the Flutter team.** When you

run your app you will get something like this.

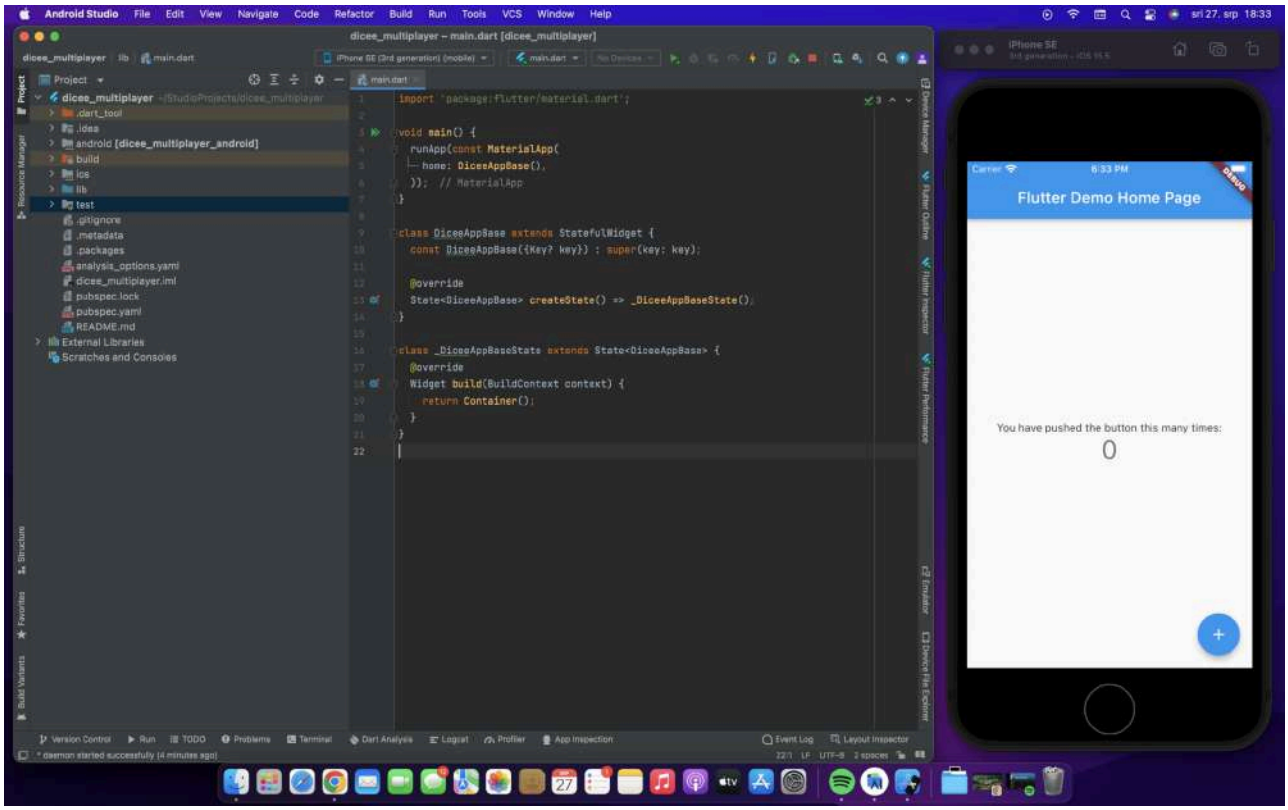


Creating Dicee App widget

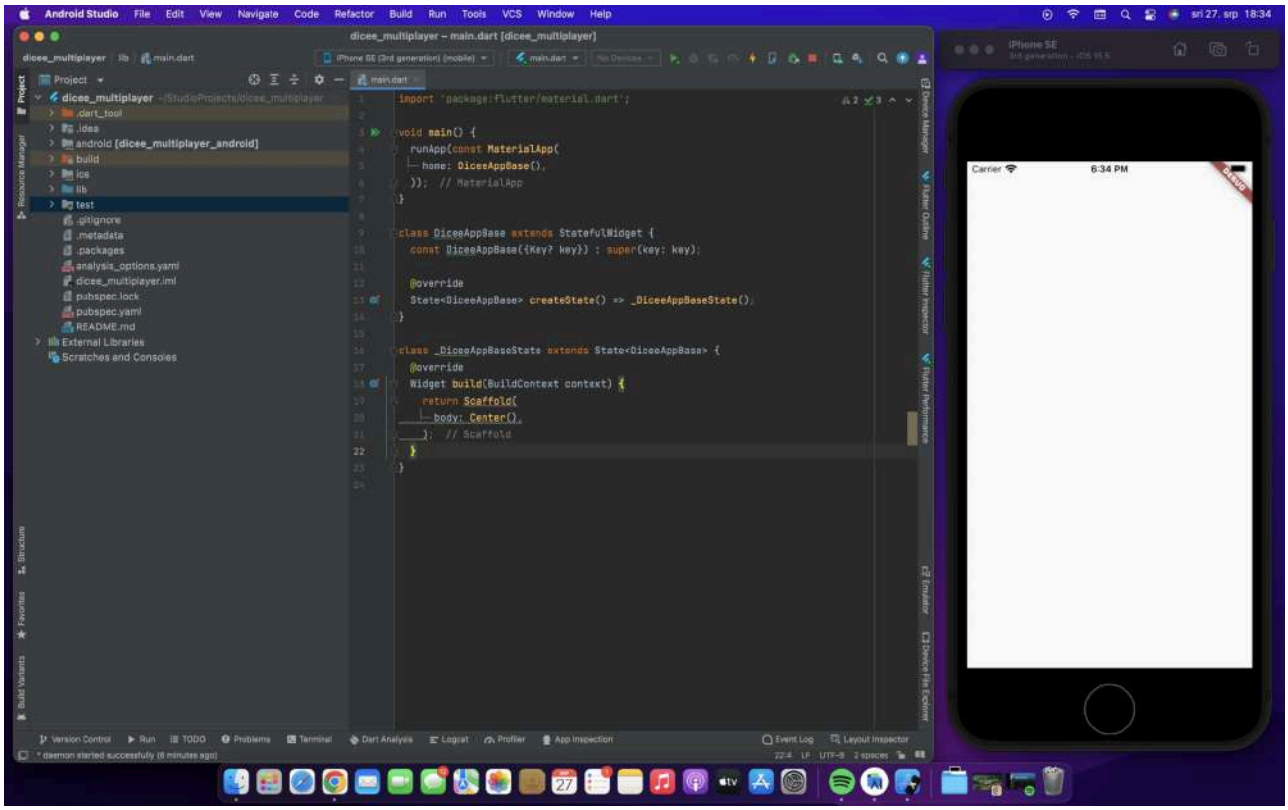
Now, delete the started code and create a new widget. This widget will be a Stateful one for now. I called it 'DiceeAppBase', but you can call it whatever you want.

Make sure to add Material app to the main function above, if you don't you'll get a confusing error. As home set your new widget.

You should get something like this:



I added the Scaffold Widget as the base of our app. We won't need an app bar for this project because it will take too much space on our 'Player Board'. For now add the Center Widget to the Scaffold body.

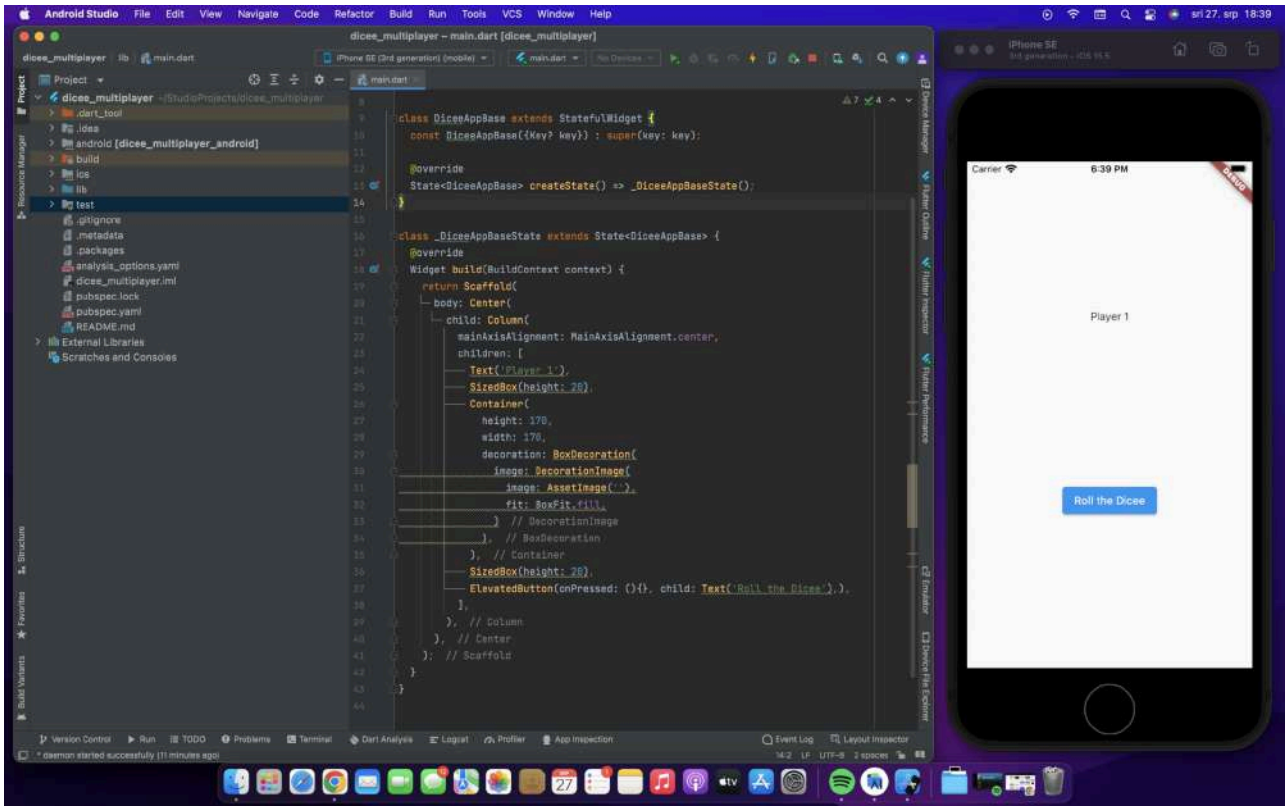


Let's build the skeleton. We add:

1. Column Widget with the main axis alignment centred
2. Text Widget with the 'Player 1' tag
3. Sized Box to divide the Image container and the Text

4. Container which will contain our Image
5. Elevated button

Right now, our Image path is blank. **We will fill it when we add our Image folder to our project.**



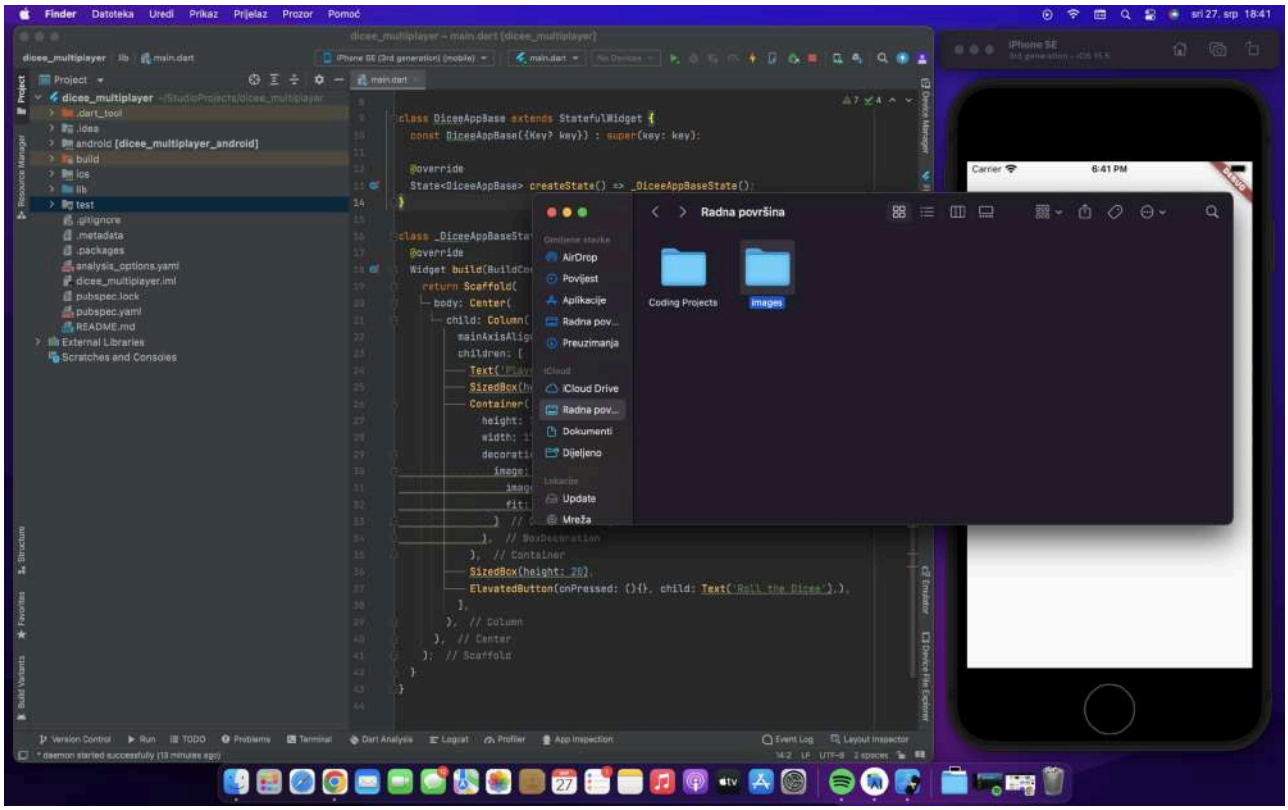
This is the code:

```
Scaffold(
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: ,
        ),
      ),
    );
```

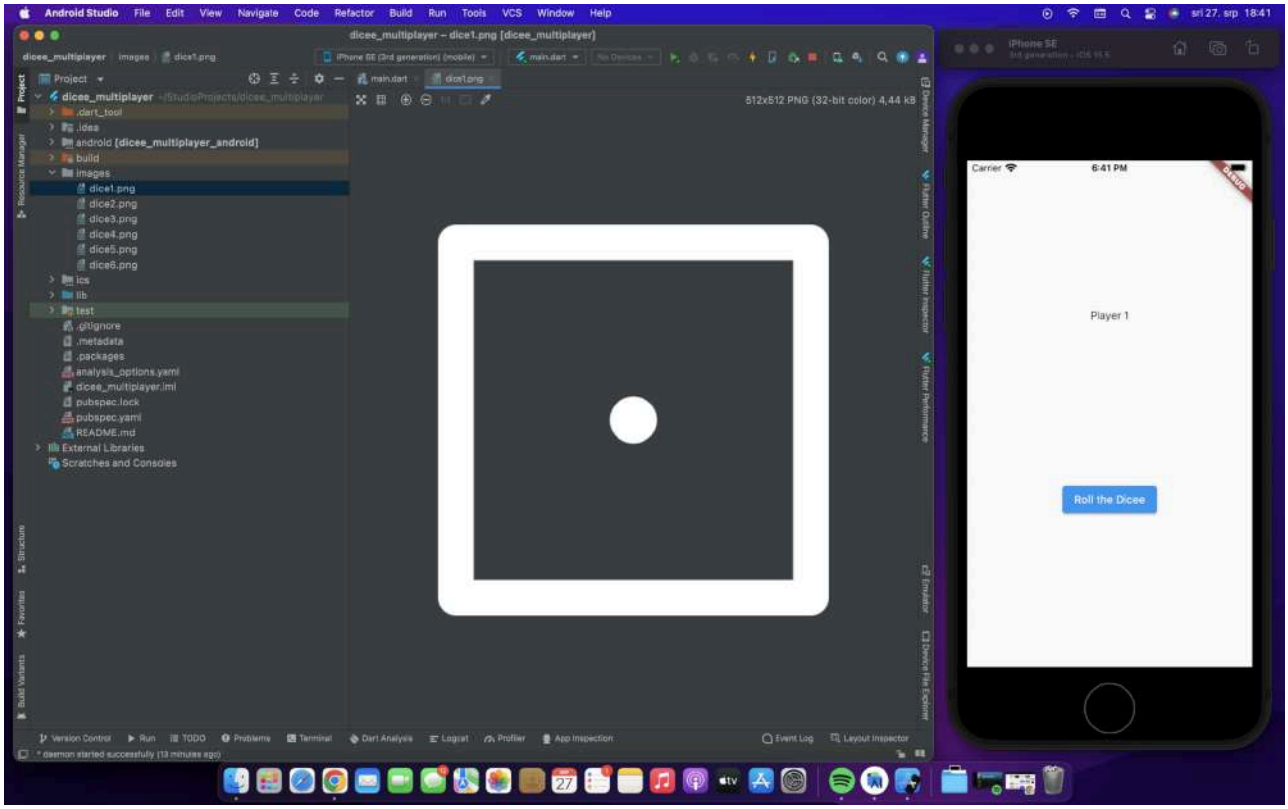
Importing images

The Images folder you can find on my GitHub, the link is:

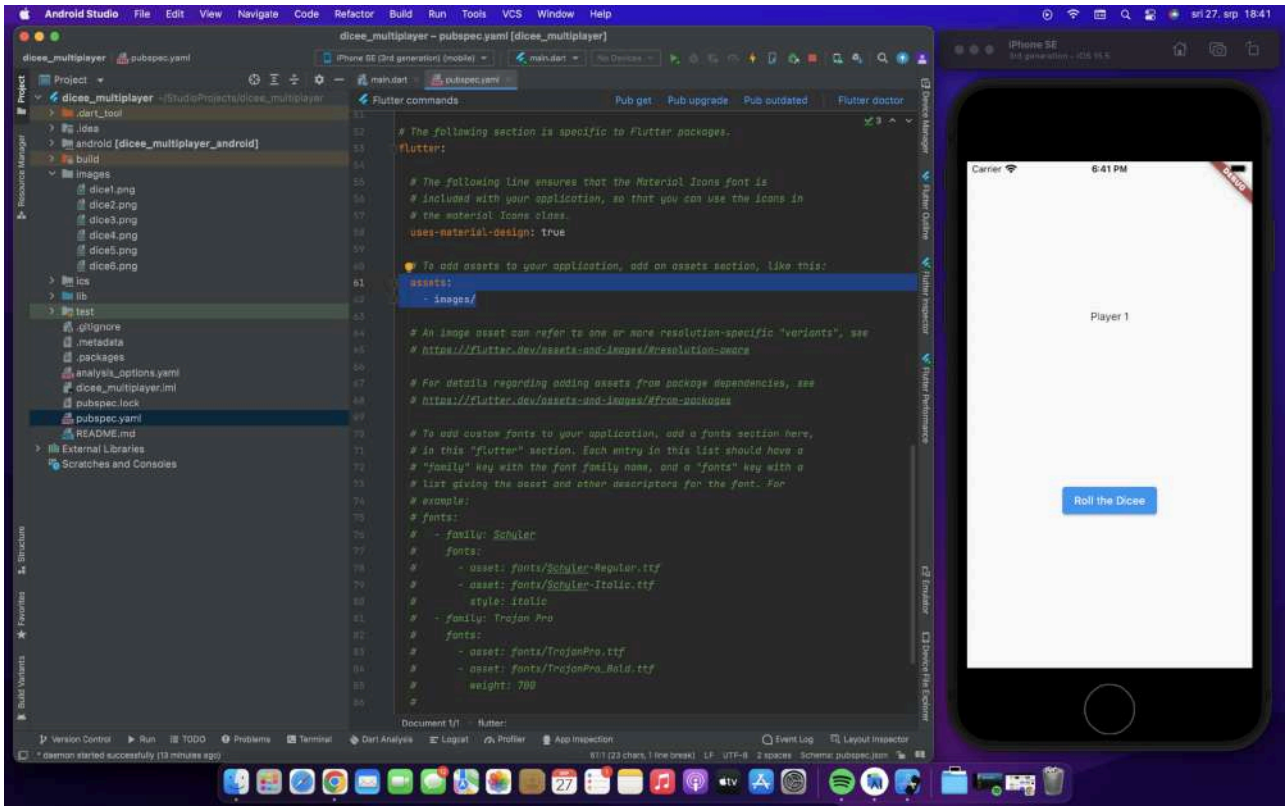
Copy the folder or just download the started code and paste it inside the project level folder.



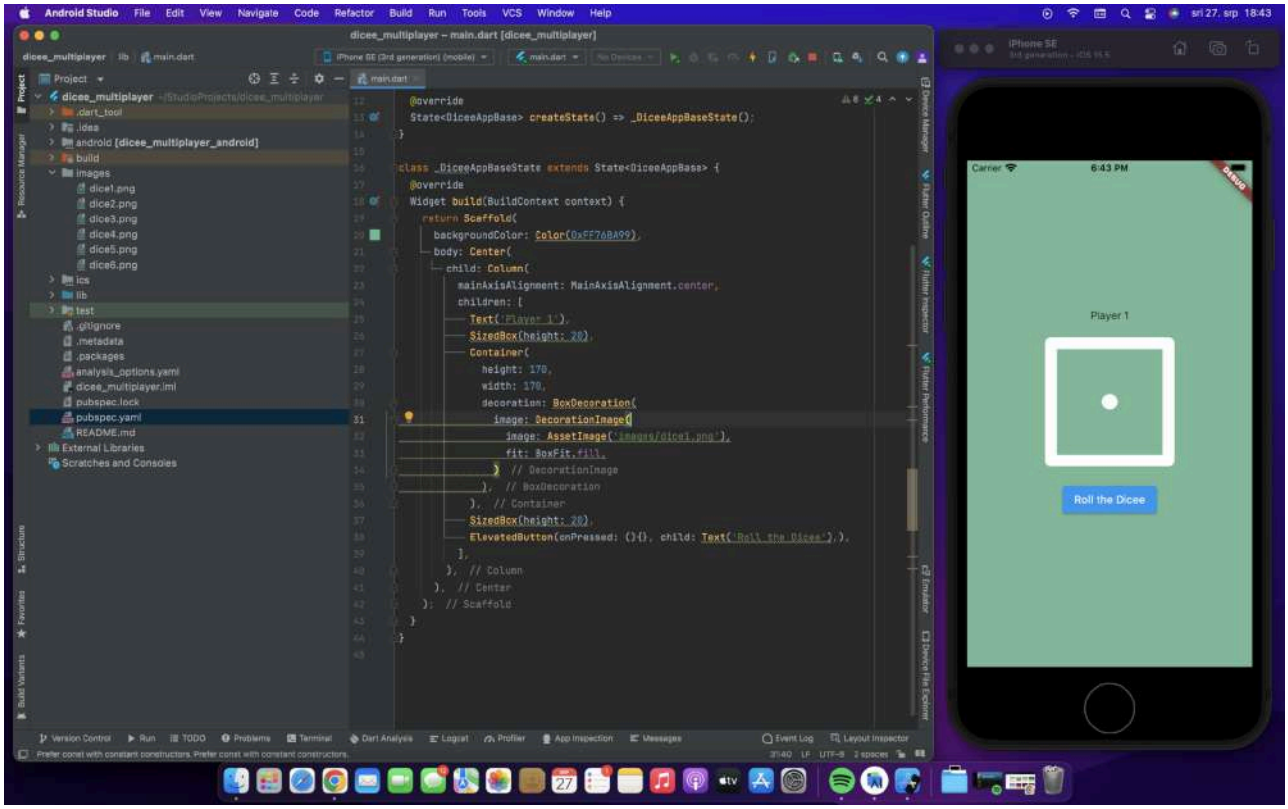
After that, you will see six dice png images. We will need 6 of them to get the randomized order effect in our app.



Before we can use these images in our app, we need to help Flutter recognize and locate them. So we head over to the pubspec.yaml file, uncomment the assets code and add our images folder path. **This is discussed in the Inspire App series.**



Because our dice images are white and png I changed the Scaffold background to a temporary green.



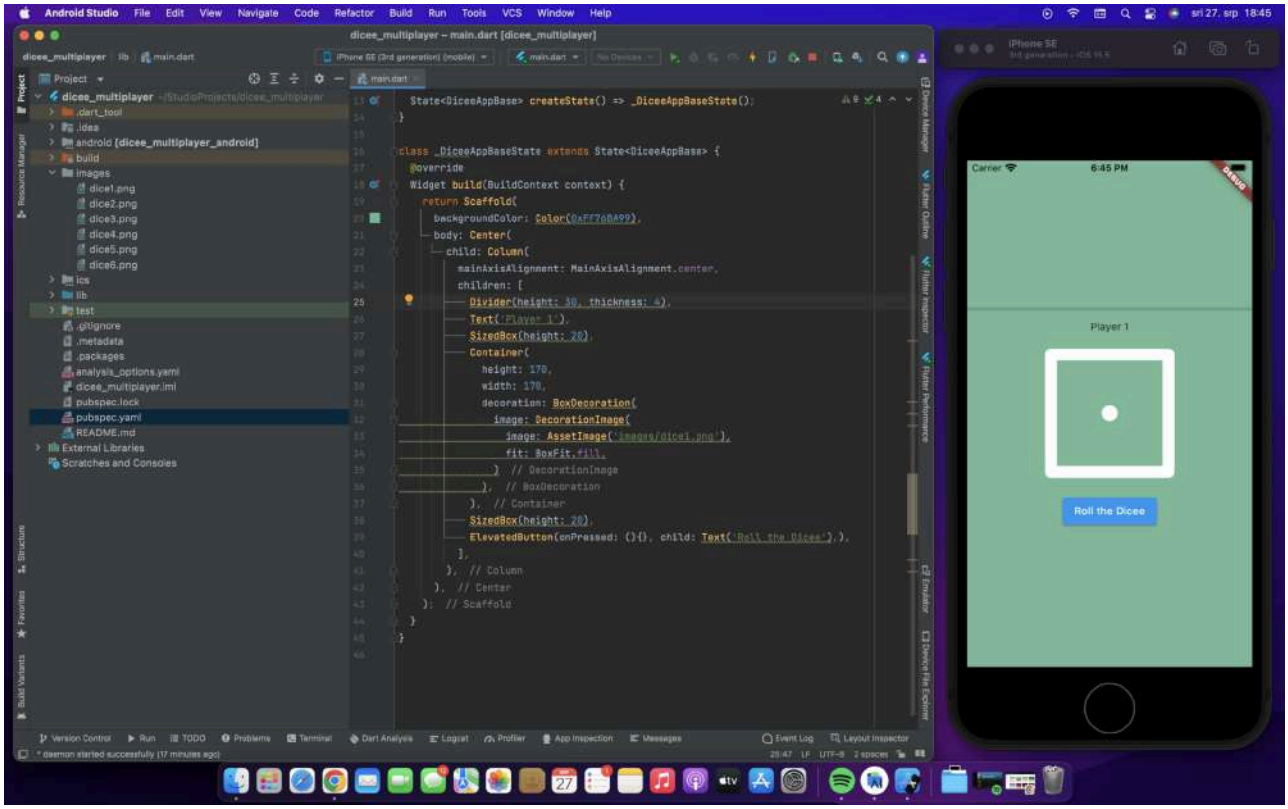
This is the code:

```
Scaffold(
      backgroundColor: Color(0xFF76BA99),
      body: Center(
        child: Column(
```

```
            mainAxisAlignment: MainAxisAlignment.center,
            children: ,
          ),
        ),
      );
```

Adding the Player ‘board’ divider

The final app has divided player boards for Player 1 and Player 2. **We can accomplish that look by adding the Divider Widget.** It creates a horizontal line across the app screen. You can adjust its thickness and height.



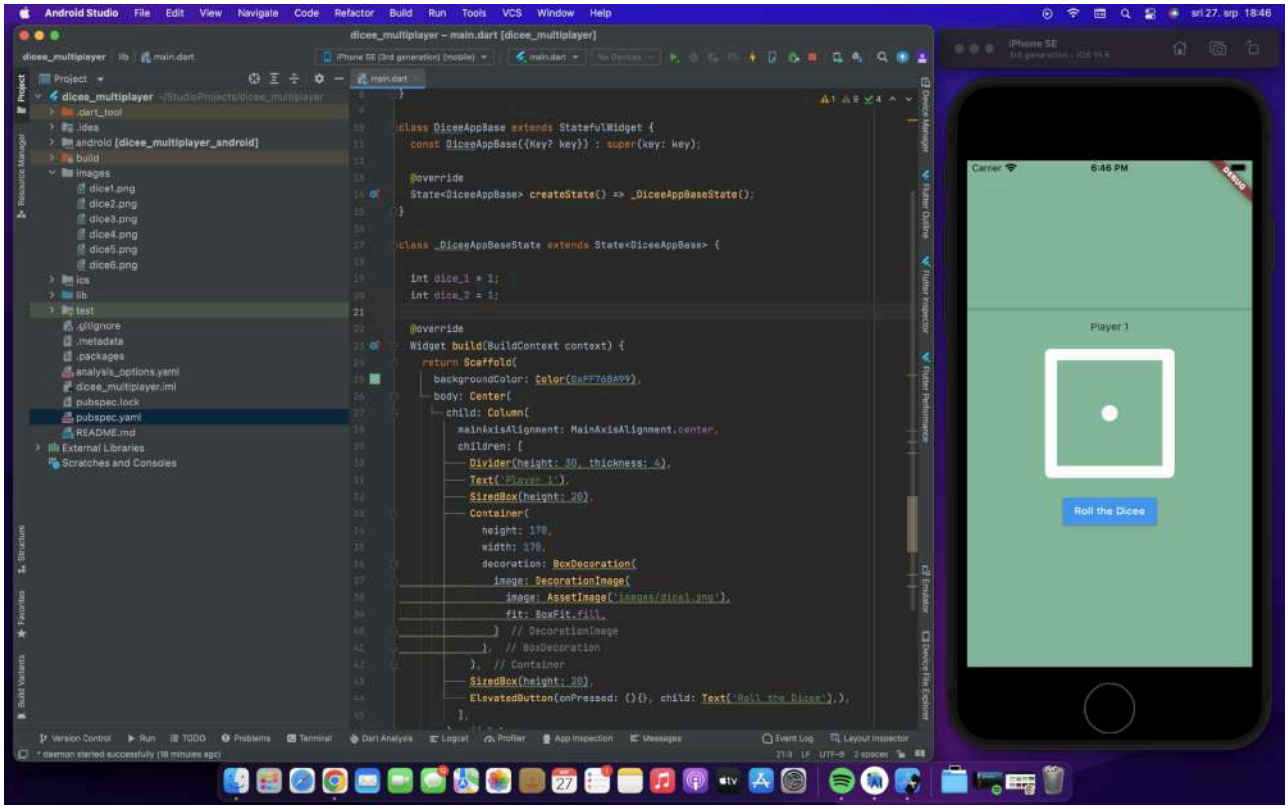
The code:

```
child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Divider(height: 30, thickness: 4),
```

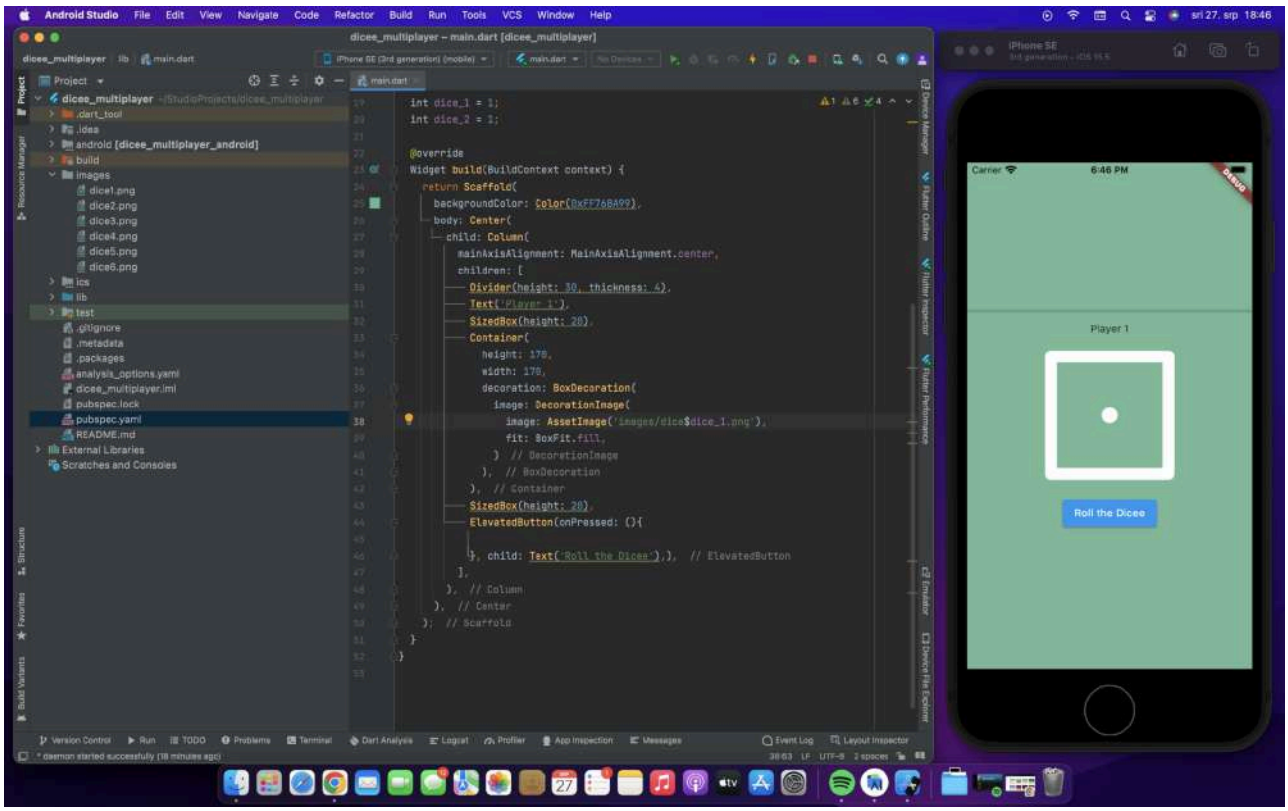
Dicee Rolling logic

The final step in Part 1 is finishing the Player 1 logic of the App.

First, we must import the 'dart:math' library to use the Random class it contains. After that, we create two int variables initially set to 1. I named them 'dice_1' and 'dice_2'. They will be randomized whenever the player hits the button below.



After adding the two variables, we insert the 'dice_1' variable into the image path in our AssetImage Widget. This way, we can dynamically change images because they are named with numbers 1-6.



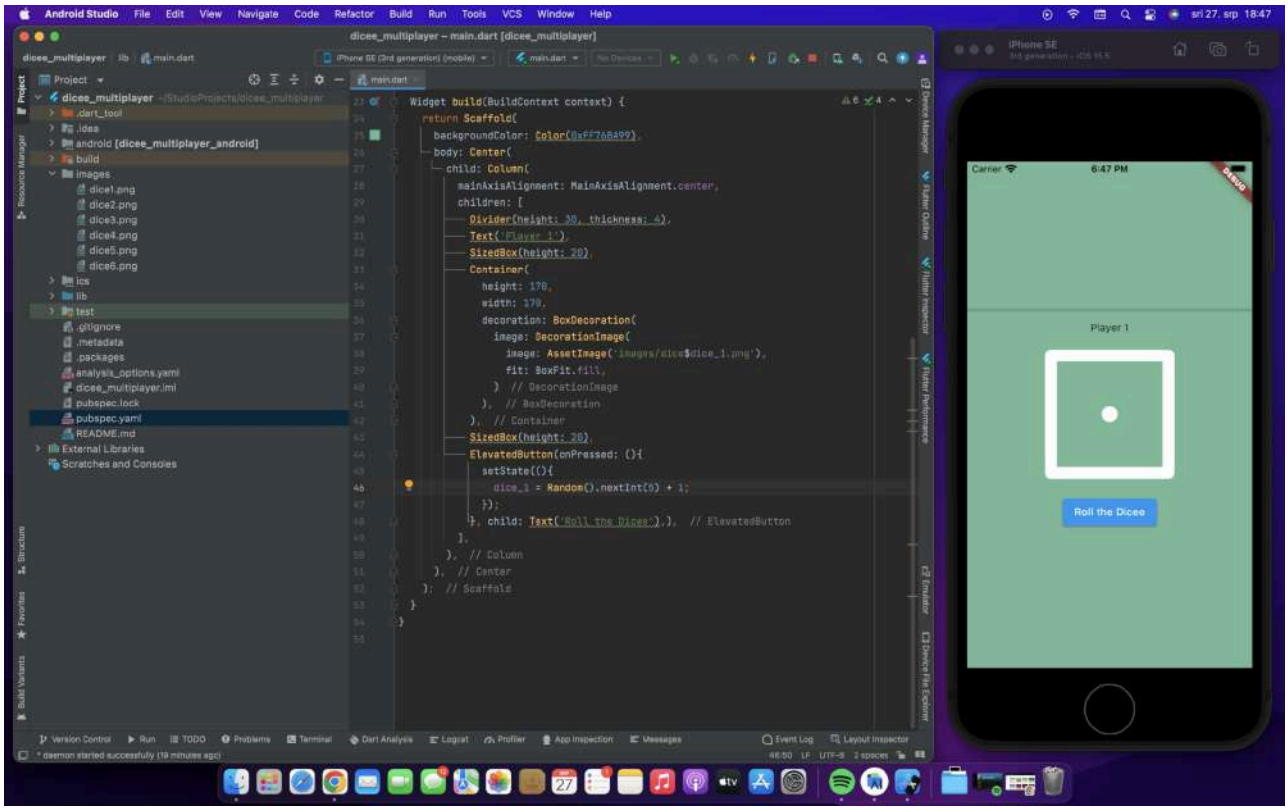
This is how the Asset Widget looks now:

```
Container(
              height: 170,
              width: 170,
              decoration: BoxDecoration(
                image: DecorationImage(
                  image: AssetImage('images/dice$dice_1.png'),
                  fit: BoxFit.fill,
                )
              ),
            ),
```

Now, let's build the random logic.

For this, the state of our app will change. Dicee Images will dynamically change, so we need to alert Flutter of these changes. **We do that by using the setState() method.** Every time a user presses the button the image number (formatted before) should change.

We assign our 'dice_1' variable to a random int value from 1 to 6.



Elevated Button widget looks like this:

```
ElevatedButton(onPressed: (){
              setState(){
                dice_1 = Random().nextInt(5) + 1;
```

```
                  });
                }, child: Text('Roll the Dicee'),),
```

Summary

Today, in part 1, we covered:

1. Creating and running our Flutter App
2. Creating the Dicee Base widget
3. Importing Images
4. Adding the Player Board divider
5. Building Player 1 logic
6. String formatting in Dart

Inspire App series: here.

In part 2, we'll build the second part of our board and work on some code reformatting! I hope you liked this post and learned something new!

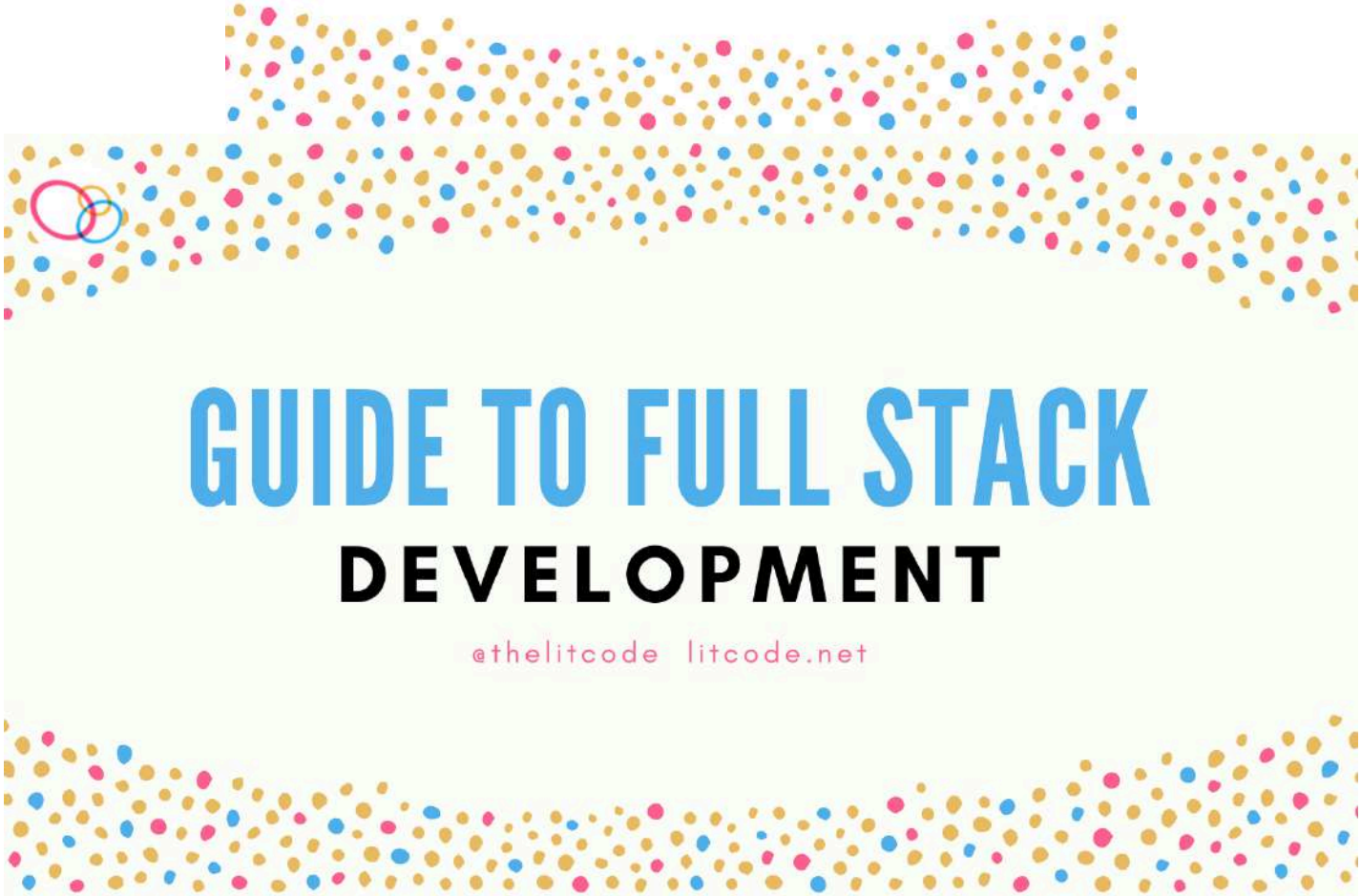
Happy creating! – Irma

Guide to Full Stack Development

September 4, 2022

Categories: Let's learn

Tags: Back End, Coding, Front End, JustCode, LitCode, Web Development



Welcome back to LitCode's new post! **Today we are discovering the world of Full Stack Development.** In today's guide, we'll cover the basic definitions and concepts in this comp

programming area. After reading this post, you'll have a great overview of what it means to be a Full Stack Developer.

I hope you enjoy reading this post! **Now, let's get started...**

What is Full Stack Development?

By the definition, a full stack developer is a programmer who develops both client and server web software. You might know client and server software by their more popular names, front end and back end. The Front End is viewed as the skin of the website. It is what you as a user see. The Front End part of LitCode is exactly this post you're reading now. The Back End is functionality of a website. It is the server and database side of a site.

When Web Development just started it was easy to be a Full Stack Developer. Now this role is much more complicated and the work put into website creation is very complex.

Client Software / Front End

A Front End Developer is a person who creates websites using technologies such as HTML, CSS and JavaScript. **A common misconception is that HTML and CSS are programming languages.** HTML stands for Hyper Text Markup Language and it is not considered a programming language. CSS stands for Cascading Style Sheets and it is used in combination with HTML to improve the website's design. Both of them are often in combination in various frameworks such as Bootstrap and Skeleton. **JavaScript is an actual programming language and it is responsible for some part of the user interaction with the site.**

So in conclusion, front end developers are responsible for the UI parts of the site. Average salary for front end developers in the US and Europe is about $100.000.

Server Software / Back End

A Back End Developer is a person who creates the site's behind the scenes functionality. They're responsible for things such as button navigation, user authentication or any site's actions. **There are so many hidden parts of a website and exactly the back end developers are responsible for them.** The Back End Developer must write high quality and efficient code while thinking about the speed of the site.

Besides the UI, Full Stacks are responsible for programming a browser (**JS, jQuery**), server (**Node, Python**) and database (**SQL, MongoDB**). Average salary for Back End developers is a little less than $100.000.

Summary

In today's post we covered the basic concepts of **Full Stack Development**.

- The definition of a Full Stack Developer
- Developer's responsibilities
- Front End / Client Software
- Back End / Server Software
- Average salaries

I hope you liked the post and make sure to follow LitCode's Instagram! You can find it on @thelitcode. `Happy creating! – Irma`

More posts like this:

- How to start your coding journey?
- How to earn money as a programmer?
- Introduction to Flutter

Introduction to AI

May 6, 2023

Categories: Let's read, New

Tags: AI, Artificial Intelligence, ChatGPT, JustCode, LitCode

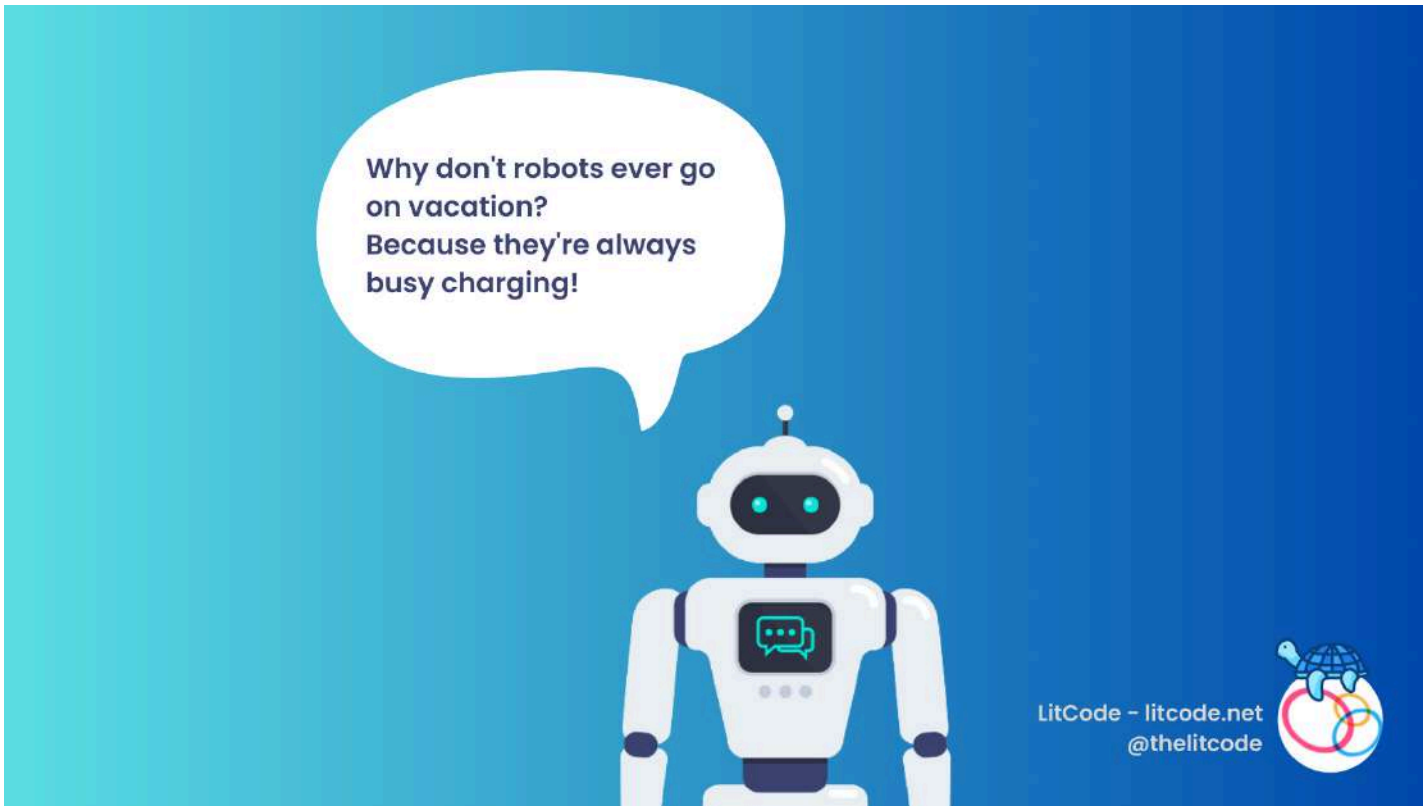
What is AI?

Artificial Intelligence (AI) is a rapidly growing field that involves the development of **algorithms** and **systems** that can perform tasks that would normally require human intelligence. It involves the use of technologies such as **machine learning, deep learning, and natural language processing** to teach machines how to recognize patterns and make predictions. AI has the potential to revolutionize industries such as healthcare, finance, and transportation by enabling faster and more accurate decision-making.

Why is it relevant today?

AI is becoming increasingly relevant today because it has the potential to transform many aspects of our lives, from the way we work to the way we interact with each other. AI is now capable of performing tasks that were once thought to be **only possible by human beings.** This has led to the development of new technologies such as **self-driving cars, virtual personal assistants, and predictive analytics tools.** As these technologies become more widespread, they are expected to have a profound impact on industries ranging from healthcare to finance. However, there are also concerns about the **ethical implications of AI,** particularly with regards to issues such as privacy, bias, and job displacement. As AI continues to evolve and become more integrated into our lives, it will be important to find ways to address these issues and ensure that AI is used in ways that benefit society as a whole.

How to use AI effectively?



Using AI effectively requires **careful planning and execution**. First, it is important to **identify the specific problem** that you want to solve using AI. Once you have identified the problem, you can begin to **gather and analyze the data** that you will need to build your AI algorithms. You will also need to **choose the appropriate AI technology,** such as machine learning and develop a strategy. Finally, it is important to **continually monitor and evaluate the performance** of your AI systems to ensure that they are delivering the desired results.

1. Identify the problem
2. Gather and analyze data needed
3. Choose the AI technology
4. Monitor and evaluate AI performance

Can AI harm us in the long run?

While AI has the potential to revolutionize many aspects of our lives, there are also concerns about the potential risks it poses. There are concerns about the impact of AI on **employment,** as machines become increasingly capable of performing tasks that were once done by humans. Additionally, there is a risk that AI systems could be **biased or discriminatory**, leading to unfair outcomes for certain groups of people. As AI continues to evolve and become more integra nto

our lives, it will be important to address these concerns and ensure that AI is developed and used in ways that are **safe, ethical, and beneficial for society** as a whole.

Reflection on Chat GPT

Chat GPT is a pretty cool artificial intelligence language model that can answer questions and generate text in a conversational style. It uses a technology called **deep learning** to learn from large amounts of text data, which allows it to understand natural language and generate responses that sound like they were written by a human.

USC Experts reflected on this. Professor Swabha Swayamdipta said:

> **"This is definitely a risk that any language model carries – the tendency of 'hallucinating' new information which might seem real but is really not. However, I believe we will get better at spotting ChatGPT generations as opposed to human written language pretty soon, or better we might be able to build that technology. And sure, it can fool peer reviewers, but it cannot appear at conferences or run experiments or do fieldwork. There might be a few cases of it successfully misleading humans before we can learn to spot these fakes though."**

> **Swabha Swayamdipta**

Take a look at this article by USC – ChatGPT: USC Experts Break Down What You Need to Know

Pros and Cons

Let's reflect!

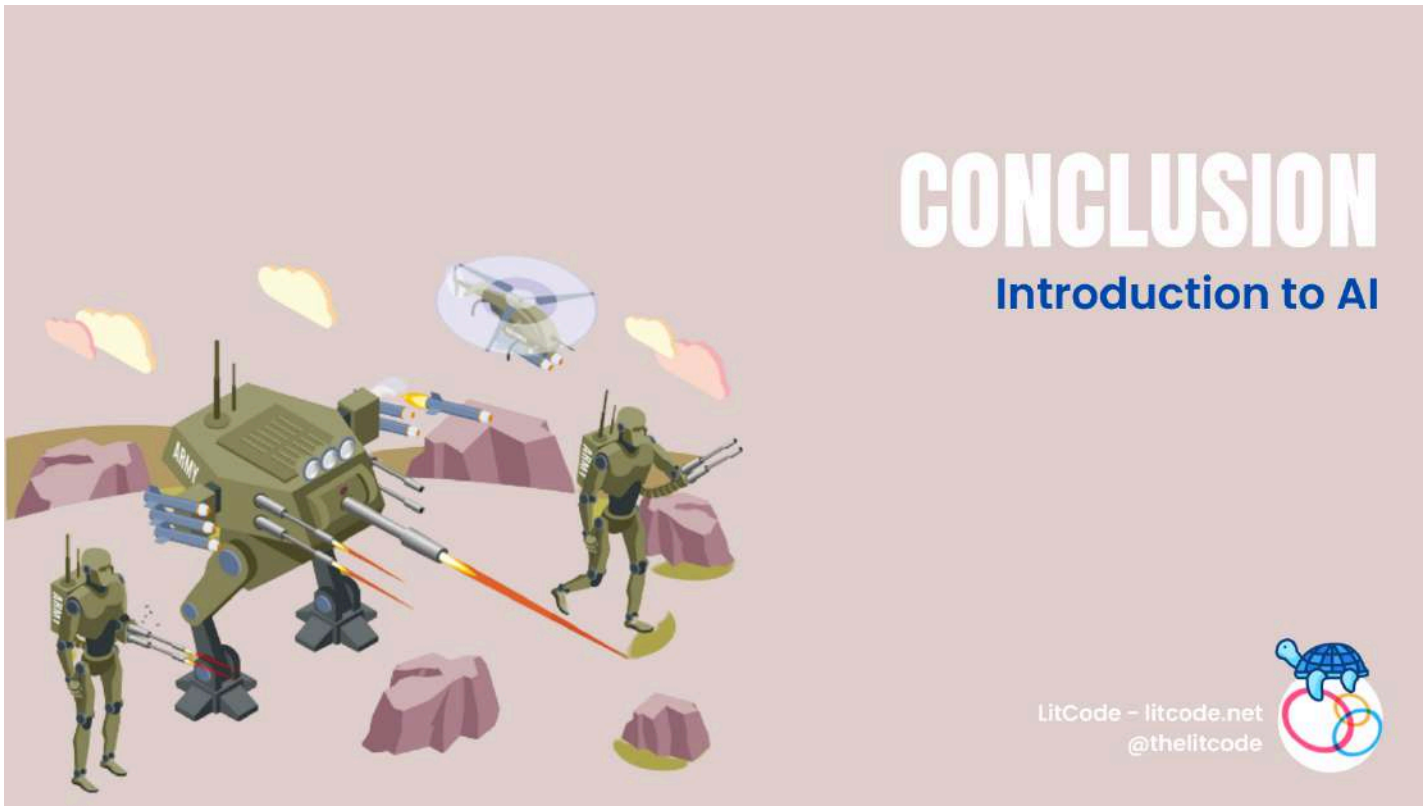
Pros of AI:

1. Increased efficiency: AI can perform certain tasks faster and more accurately than humans, which can increase productivity and efficiency
2. Improved decision-making
3. AI can be used to personalize products and services based on individual preferences
4. Advancements in healthcare
5. Increased safety: AI can be used to monitor and prevent accidents in industries such as transportation

Cons:

1. Job displacement
2. Bias and discrimination: AI systems can be biased or discriminatory, leading to unfair outcomes for certain groups of people
3. Privacy concerns
4. Security risks
5. Lack of transparency: AI systems can be difficult to understand and explain

Conclusion



In conclusion, the use of AI has been widespread and is increasing rapidly. It is important to understand it to be able to effectively use it. Whether you are writing a quick essay or building a security system for your business, before using Chat GPT please talk to the experts 😉

Thank you and I hope you enjoyed this post! Till next time – Irma.

More posts like this:

Guide to Full Stack Development – Who was Steve Jobs? – Think Like a Programmer

The Role of Cryptocurrencies in Tech

June 16, 2023

Categories: Let's learn, New

Tags: Blockchain, Crypto, Cryptocurrencies, LitCode, Programmer, Technology

Introduction

In today's article we'll discuss the impact of **Cryptocurrencies** in tech but also the new **Blockchain technology** uprise. Even though this concept of "Crypto" seems abstract to many it is fairly simple. This article is here to clear some things up.

Let's dive in!

What is a Cryptocurrency?

Cryptocurrency is a form of a digital asset (currency) that **doesn't rely on banks** to verify transactions. This digital payment system is based on a **large distributed, decentralized network of computers. With its decentralized structure it is immune to government control or control of corporations (Google, Facebook, Apple etc.)** Cryptocurrencies of today allow for cheaper and faster money transfers.

"Crypto" refers to algorithms used to guard and secure these assets. Cryptocurrencies are created and run on **Blockchain.** They are created through "mining" using computers to solve mathematical problems and essentially generate these "coins".

Influential cryptocurrencies of today are: Bitcoin, Ethereum, Litecoin, Ripple etc.

What is Blockchain Technology?

Blockchain is a secure database revolutionizing a ***new world without the corporation middlemen.*** It allows for secure information sharing.

All of its data is stored in a database which is cryptographically secure. Transactions and all the work done is stored in an “account book” called **ledger.** Bitcoin blockchain is one of the best examples of blockchain technology.

It allows for secure and fast crypto trading between buyers and sellers. **Not to mention but all of this happens online!** All of it in a decentralized manner without the control and verification of banks.

How is Blockchain changing the future of Tech?

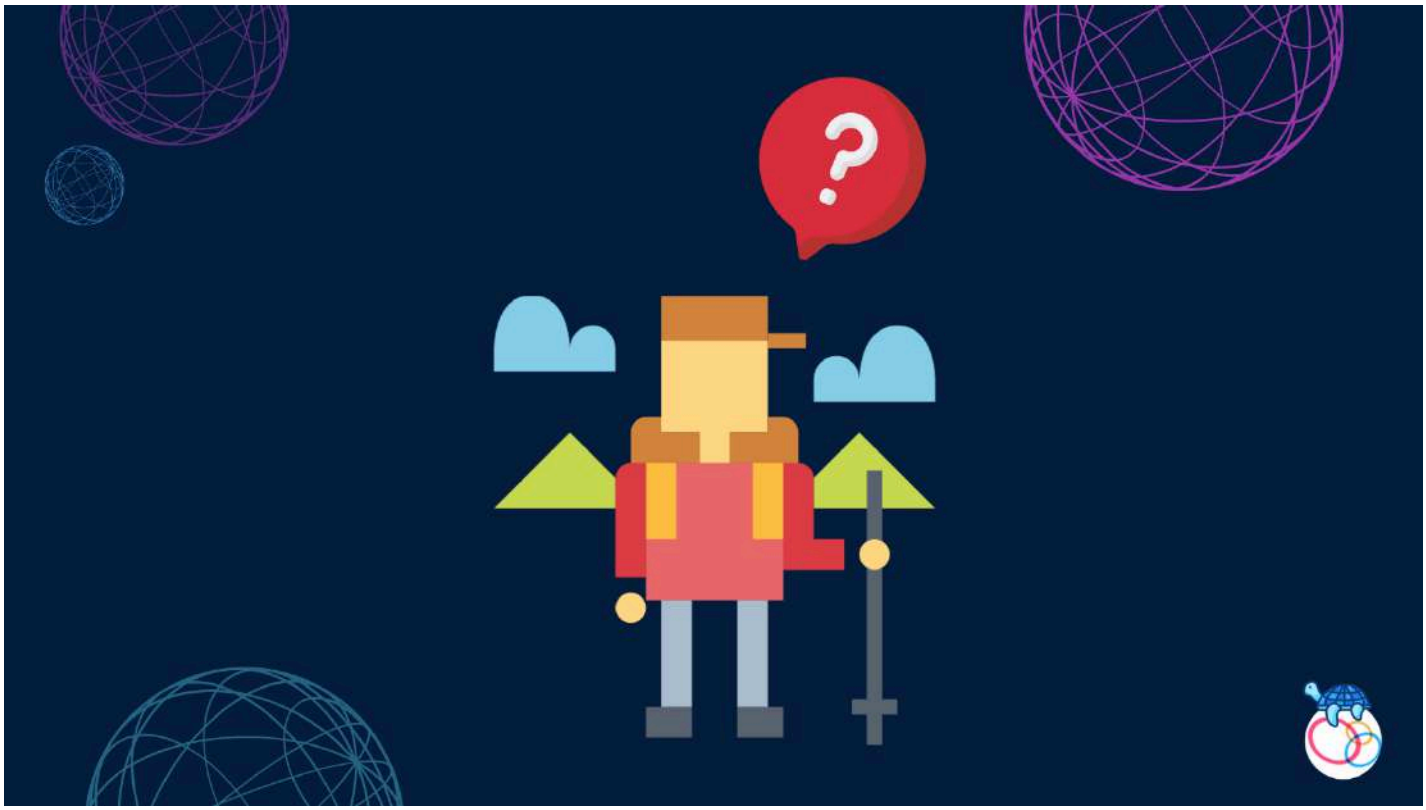
Blockchain technology opened a whole new world of opportunities for programmers. New job openings in this field are in its all time high with an ***average salary of $115,000.***

In the long run, this technology is increasing trust and data security shared across business networks. This is not only influential to World's economies but to developers. **Imagine the opportunities and updates coming after such technology is applied to data storage across our devices!** iCloud, social media, storage can be changed for the good!

Blockchain will ultimately allow access to banking services in places where this couldn't even be imagined.

Some of real life projects you can hop on include: Simple Storage applications, Multi-Send using Ethereum, Wallets, Time-Locks etc.

Your next step...



Today we talked about this world changing network and all of this can be a bit overwhelming. **So now that you know all of this, what is your next step?** I highly encourage you to check out some of Udemy best courses on Blockchain! And hop on building real life projects. If you are just looking to invest in some Crypto I suggest to do your research, talk to the experienced but also trust your gut!

More like this by LitCode:

Introduction to AI by Irma

Who was Steve Jobs? by Irma

How to start your Coding journey? by Irma

Check out these Resources:

Cryptocurrency Explained by Investopedia

What is cryptocurrency and how does it work? by Kaspersky

What is blockchain? by McKinsey & Company

Blockchain Courses on Udemy

Thank you! Till next time, Irma.

Getting started with Python: Language Basics

July 3, 2023

Categories: Let's learn, New

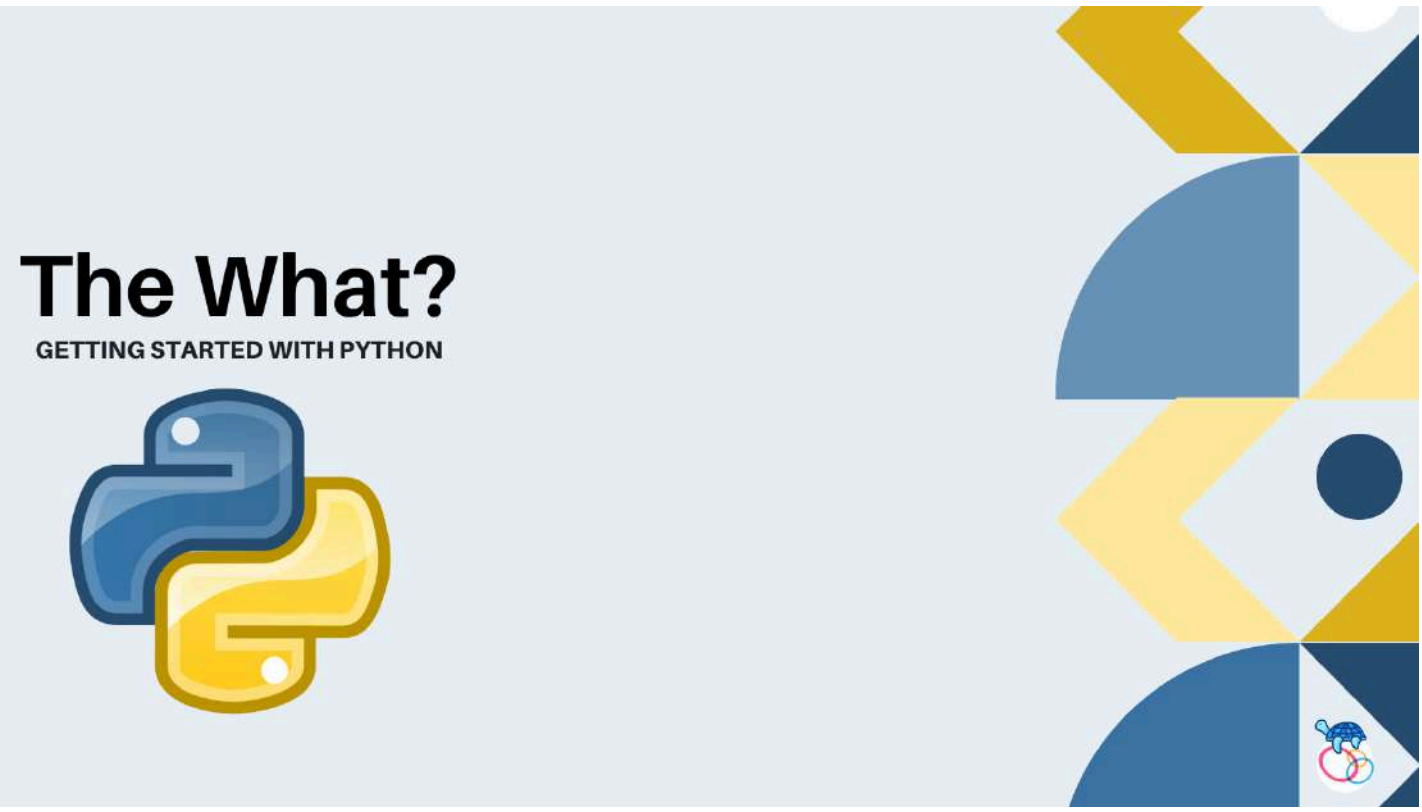
Tags: Challenges, Coding, Course, LitCode, Programmer, Python

Getting Started with Python
LANGUAGE BASICS
WWW.LITCODE.NET
@THELITCODE

In today's article, I will be introducing you to the basics of Python! I believe that Python is the best programming language to get you started on your programming journey. If you are new in this programming world there is no better way to start then with this article.

We will cover the basics in this Part 1 – Getting started with Python. Stay tuned for more in Part 2 of this series!

What is Python?



> **If you decide to design your own language, there are thousands of sort of amateur language designer pitfalls.**
>
> **Guido van Rossum, Python programming language**

Python is one of the most popular programming languages today. **It was created on February 20, 1991 by a Dutch programmer Guido van Rossum.**

He created this programming language just as a hobby hoping to keep him occupied during the holidays. At the time, not even realizing the impact it will make on the technology. Within ***just 3 months*** a prototype version of Python was built.

Guido van Rossum took inspiration from the television program called the ***Monty Python's Flying Circus***. From 2005 to 2012 he spent time developing Python at Google.

The Vision behind Python

There are endless capabilities and functions of this powerful programming language. **Guido van Rossum wanted to create a language with a syntax almost the same as the English language itself.**

Python can be used for many different projects and platforms. Web applications, workflows, connecting database systems, complex calculations, you name it! This programming language is pretty straightforward and relatively easy to learn for beginners. Python can also be used on platforms such as Windows, Linux, Mac etc.

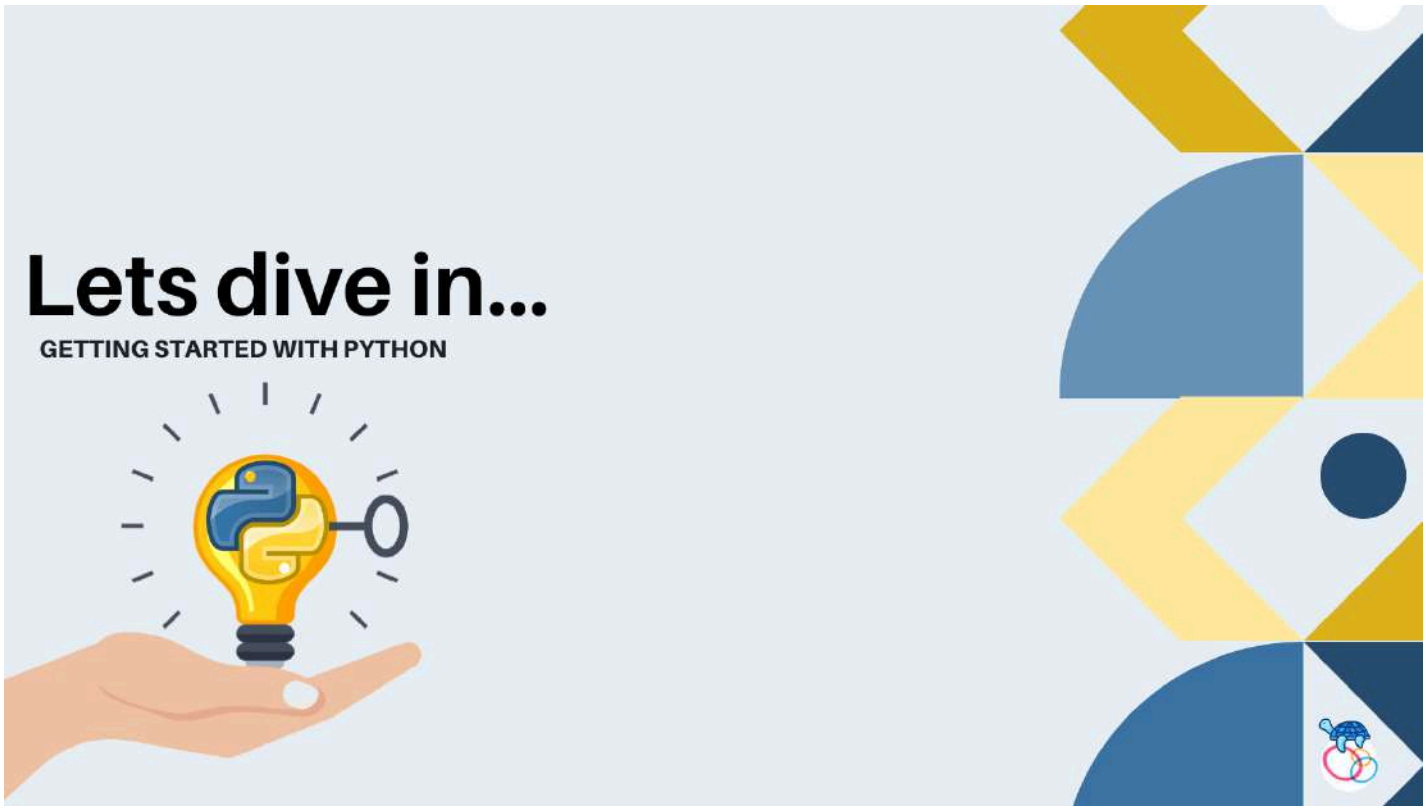
Soon you will see why so many programmers love it!

Python compared to other languages

As any other programming language Python has its specific use cases. It has many pros but also cons. ***Based on the project you're working on you can choose to use Python or maybe some other programming language will fulfill your needs better.***

- Python's syntax is easy to read and straightforward compared to other programming languages.
- It has a gentle learning curve and is one of the best options for beginners.
- With its short compile time Python projects are built faster.
- It has an extensive list of built-in projects and libraries.
- Python is one of the favorites which means a large community of developers ready to help!

Getting started!



In this tutorial we will be using Python 3. You can write your code in a text editor such as VSCode or and IDE such as PyCharm by JetBrains. Also there are many online text editors and compilers easy to use.

Syntax

The rules that define a language's structure are referred to as **syntax.**
The easiest way to get introduced to Python's easy syntax is with the **Hello, World** example.

```
# Program prints Hello, world!

print('Hello, world!')
```

Pretty straightforward am I right? This is the beauty of Python. There is only one line of code written here and it already does what we want. When you press the Start button it prints "Hello, World!" to the console. For reference let me compare it to C++ Hello, World program.

```
#include <iostream>
using namespace std;

int main()
```

```
{
    cout<<"Hello World";

    return 0;
}
```

A lot of stuff going on over there! Our **Python Hello, World** program uses **the print** function and when we put our **desired message in the parenthesis** it prints it out to the console.

Comments

Comment is an explanation or remark in a computer program's source code. Purpose of comments is to make the code more understandable and readable for the one reading it. Comments are ignored by compilers and interpreters.

In Python they are declared with the number sign (#).

```
# this is an example of a comment in Python

# you can write as many comments as you want!
```

Variables

Variables are used to hold data in a computer program that may be accessed and changed.

The syntax for creating a variable in Python is simple. **Variables have a name and a value assigned to it.**

```
nameVariable = value
```

In Python you don't even have to declare the data type of your variable! It recognizes the variable type based on value stored inside the variable.

Based on the value stored we have different types of variables. Here are some:

- **`integers`** – a variable data type that stores whole numbers. They're called integers.
- **`float`** – a variable data type that stores floating-point numbers.
- **`double`** – a variable data type used to store floating-point numbers. **Difference between ble**

and float is that double is more precise and stores larger data.

`bool` – a variable data type used to store true or false values.

`string` – a variable data type that stores text, such as "Hello World". The text is inserted between double quotes (" ") or single quotes (").

```
x = 5 # data type is (integer), name is (x) and value is (5)
b = 5.96 # data type is (float), name is (b) and value is (5.96)
c = 7.88 # data type is (double), name is (c) and value is (7.88)
s = "Hello World" # data type is (string), name is (s) and value is
(Hello World)
t = true # data type is (bool), name is (t) and value is set to (true)
```

Input and Output

print() – to print our output we simply use the PRINT function. For example:

```
print("This is my output.")

# otuput will be "This is my output."
```

input() – to take input from the user we use the INPUT function. Example:

```
input(name)

# ask user for a name
```

Putting it all together:

```
name = "Emma"
print(name) # prints out the current value which is Emma

input(name) # ask a user to input their name
print(name) # now print out the value which will be changed to whatever
user entered
```

Conditional Statements

Conditionals are statements written by programmers that evaluate actions in the program and determine whether they are true or false.

If statement

We use If statement to specify code to be executed if condition is true. The Syntax for an if statement can look like this:

```
if 4<2:
  print("Number 4 is less than 2.")
```

******REMEMBER: indentation in Python is very important!******

Else statement

We use else statement to specify code to be executed if condition is false.

```
if 4<2:
  print('Number 4 is less than 2.')
else:
  print('Number 4 is greater than 2.')
```

Elif (else if)

We use else if statement to specify another condition if our first condition specified is false.

```
if 4<2:
  print('Number 4 is less than 2.')
elif 4 = 2:
  print('4 and 2 are the same.')
else:
  print('Number 4 is greater than 2.')
```

Conclusion and Resources

Today we discussed basic concepts in Python. There is so much more to cover and explore! Stay tuned for more in this series. **Remember practice makes perfect.** In resources below I linked some challenges for you to practice what you've learnt today! Good luck on your journey 🙂

More by LitCode

Getting started with C++ – by Irma P.

Guide to Full Stack Development – by Irma P.

Introduction to Flutter – by Irma P.

Challenges to try!

Simple Conditions – by SoftUni Global

Python Variable Exercises – by W3Schools

Conditional Statements – by W3Resource

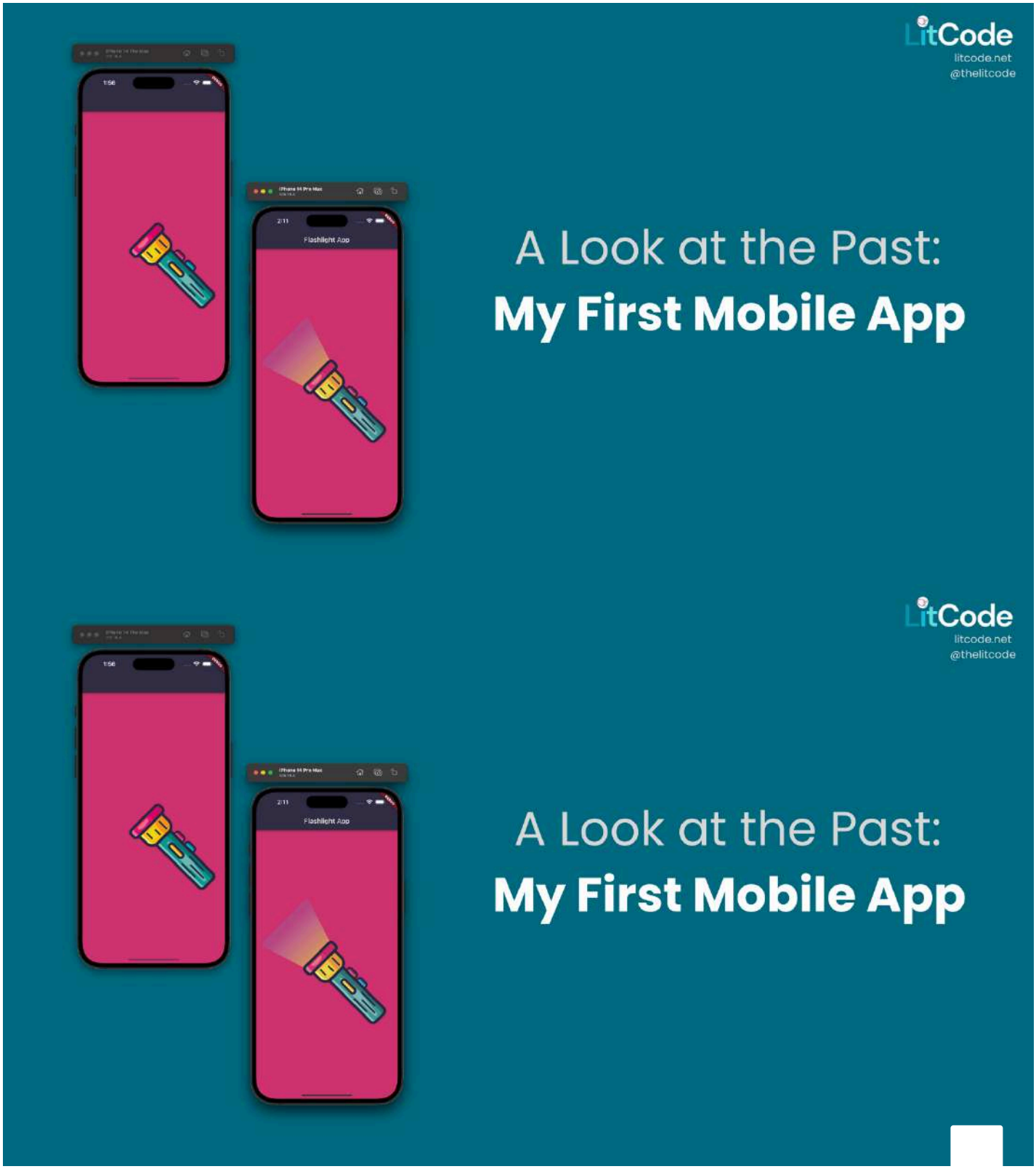
Keep coding – Irma

A Look at the Past: My first Mobile App

August 11, 2023

Categories: Flutter Beginner

Tags: Flutter, LitCode, Mobile App Development, Programmer



Introduction

Welcome to LitCode's new article! In today's post, we'll take a closer look at the development process of *my first EVER Mobile App.*

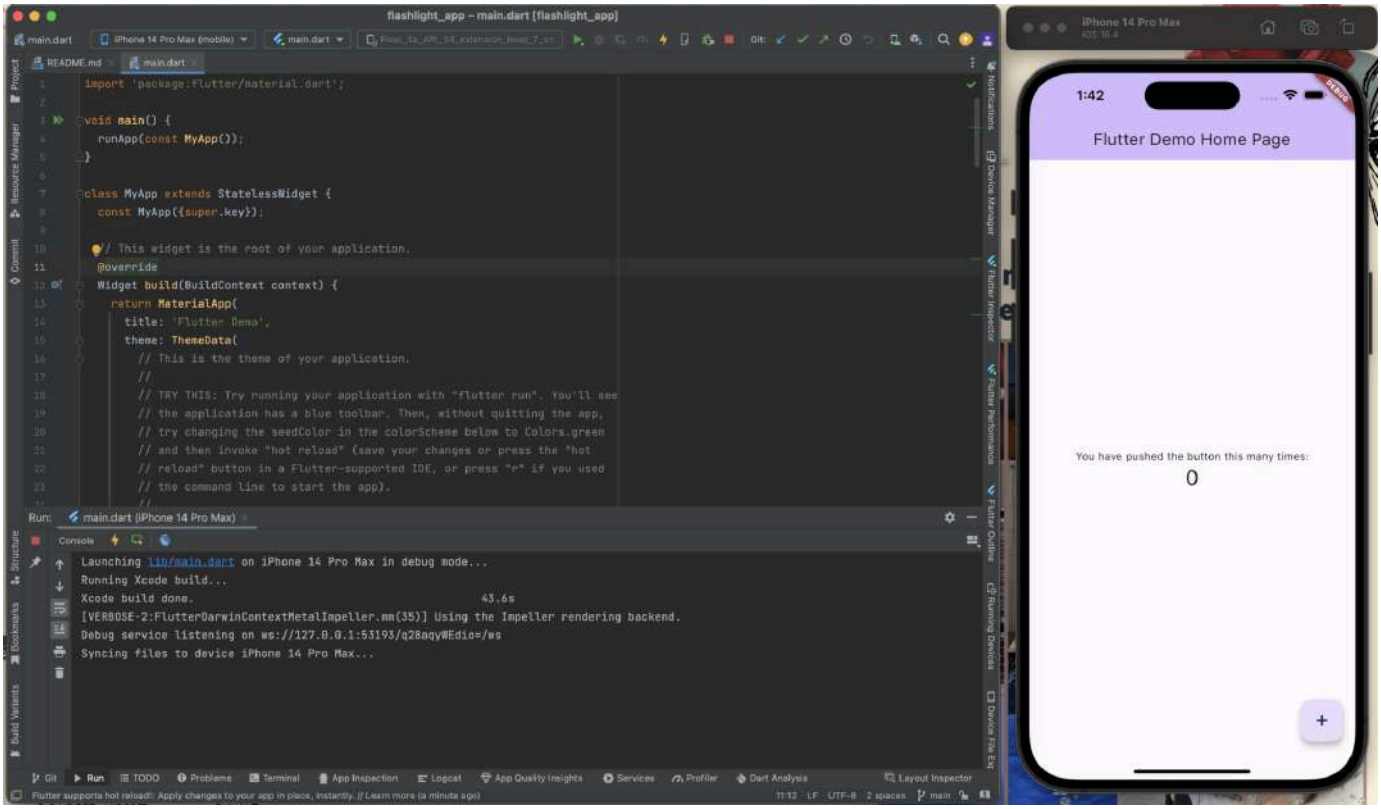
I developed the "Flashlight App," as I like to call it, a couple of years ago when I first started coding in Flutter. These were my beginnings, and I'd like to share them with you.

For anyone just starting out on their coding journey, buckle up; you're off to a great start!

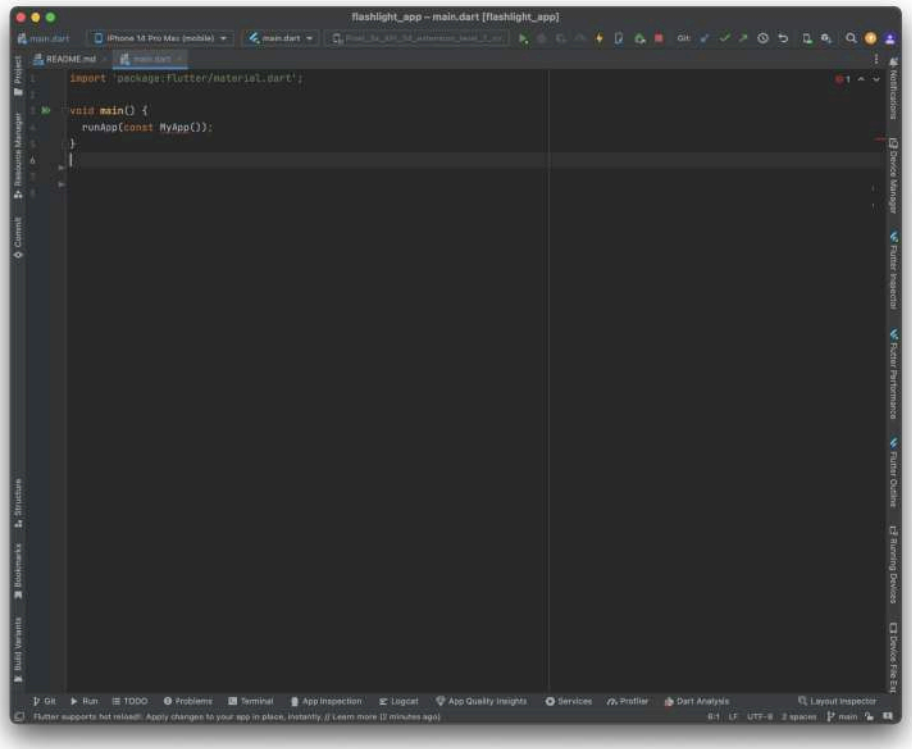
Setting Up

When I first started, I preferred developing in Flutter using Android Studio. I still like it that way but feel free to explore new IDEs and Code Editors! **In this tutorial I will be using Android Studio.**

After you start AS and run your emulator you should see something like this:



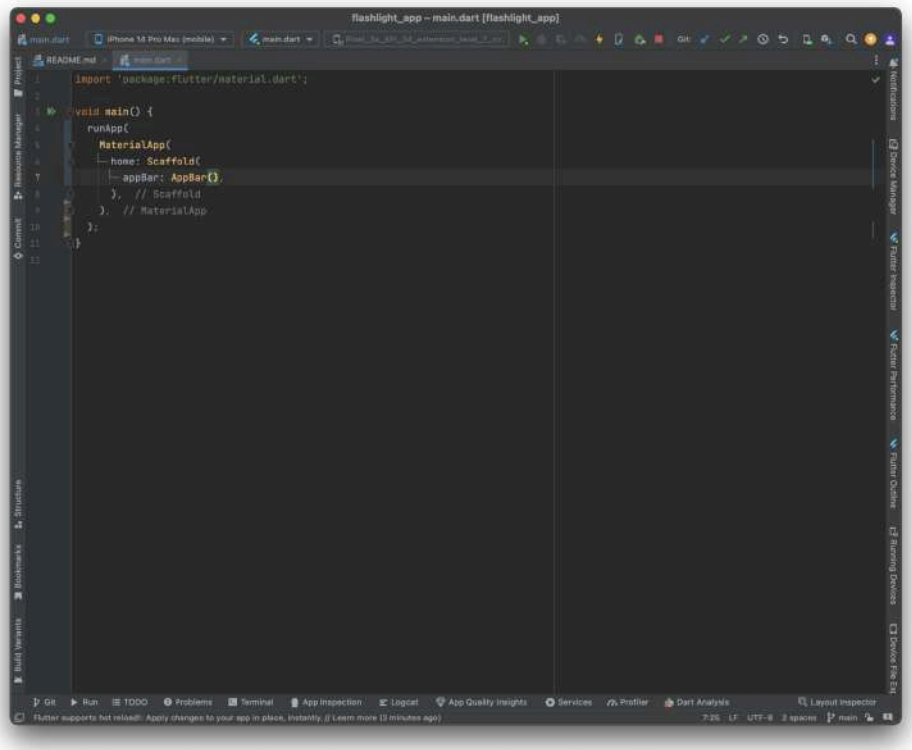
This is our starting point. *When we create a new Flutter project we always see this Flutter Demo App from the Flutter team.* You can play around with it! But for now, **I want you to delete all of it, and when I say all of it I mean everything in the main.dart file.**



Building the Scaffold

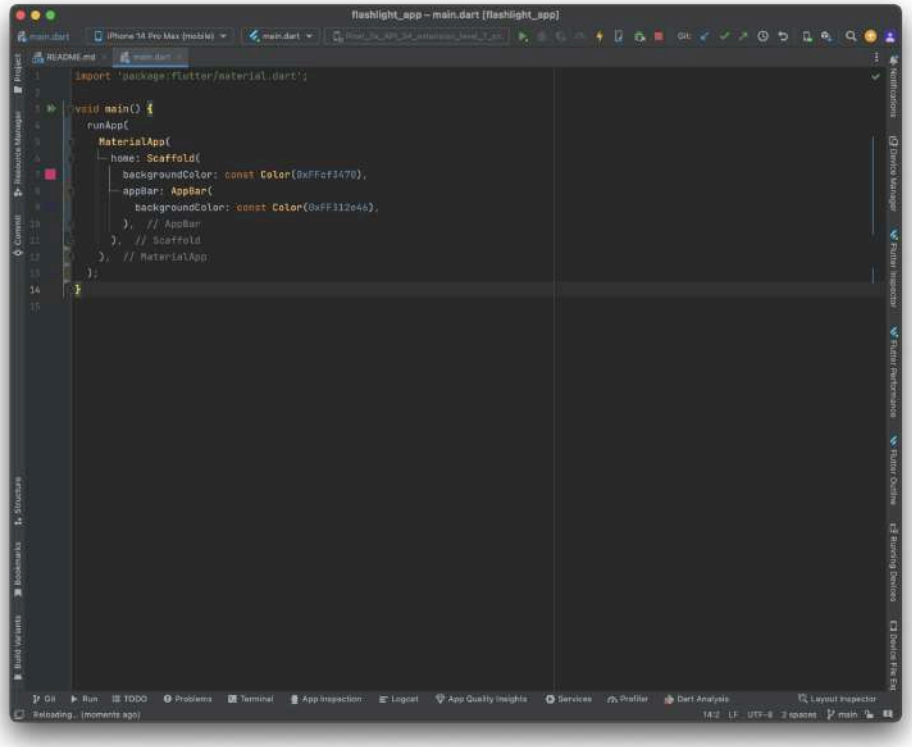
Starting out:

1. Insert a **MaterialApp Widget** as our starting point.
2. As the *home property* we insert the **Scaffold Widget**. This way we can simply add our App Body.
3. Our App is going to need an **AppBar** so let's return one!

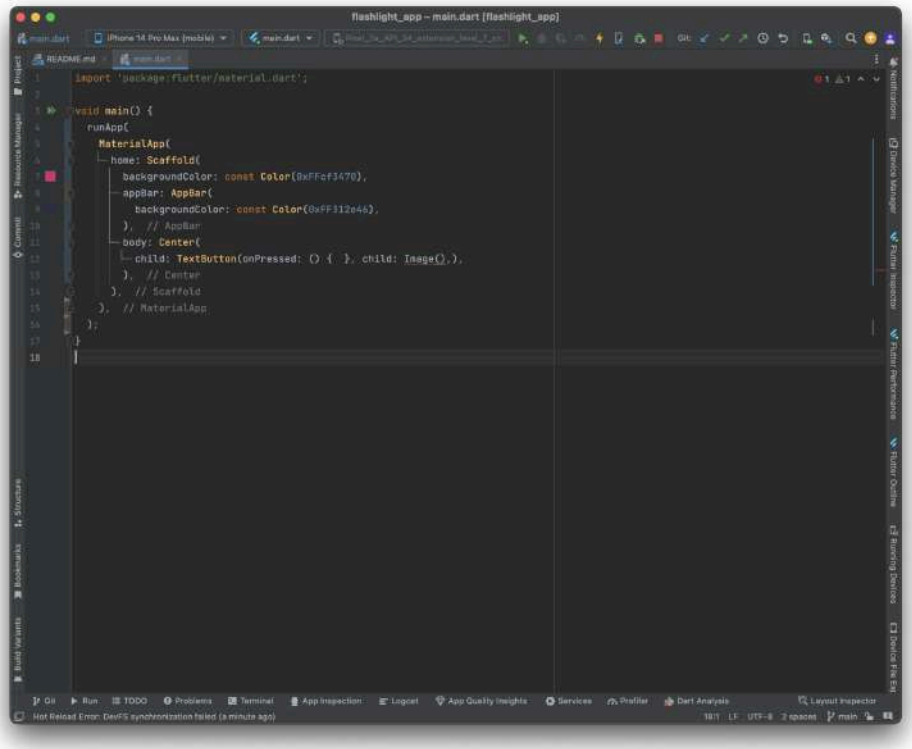


So far Flashlight App looks something like this:

Let's add some color to the ***Background of the Scaffold*** and the ***App Bar***. You can choose any color palette that works for you.



The whole purpose of the "Flashlight App" is that if one image is pressed it will switch to another, creating the "on/off" effect of the real flashlight. For this we will need an Image but also a Button behind it.



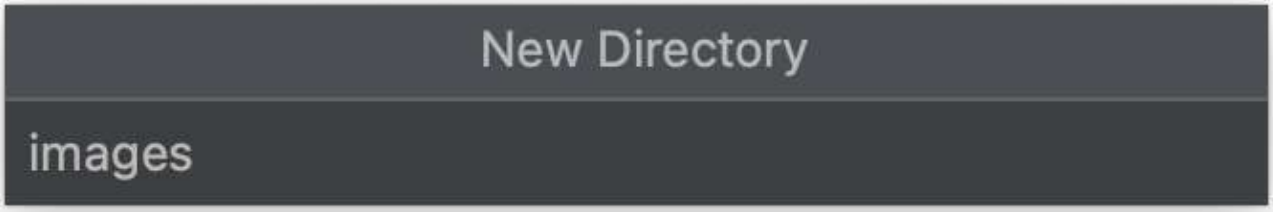
Here in Scaffold body property we added:

1. **Center Widget** – this way our image/button is centered on our device screen.
2. As its child we added the **TextButton Widget.**
3. TextButton can also have a child and this way we can add the **Image Widget**. This Widget also takes in the *onPressed function* which we will deal with later on.

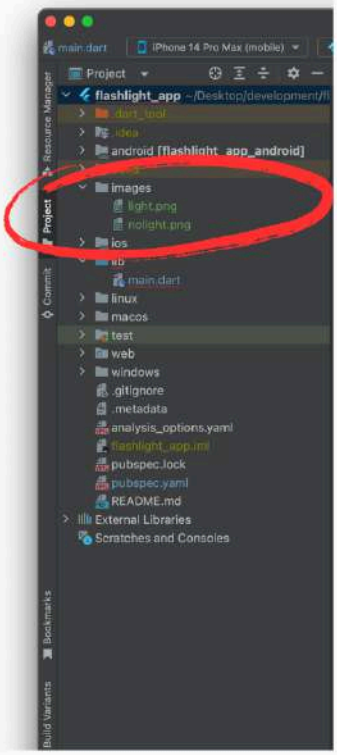
Flashlight Images

In order to use Images in Flutter we first need to setup our pubspec.yaml file located in the Project Folder.

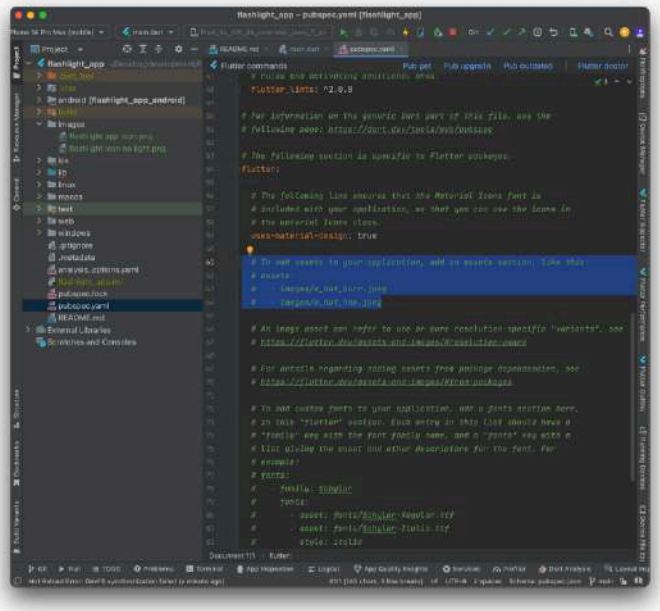
- Create a new “images” directory in the Project Folder.

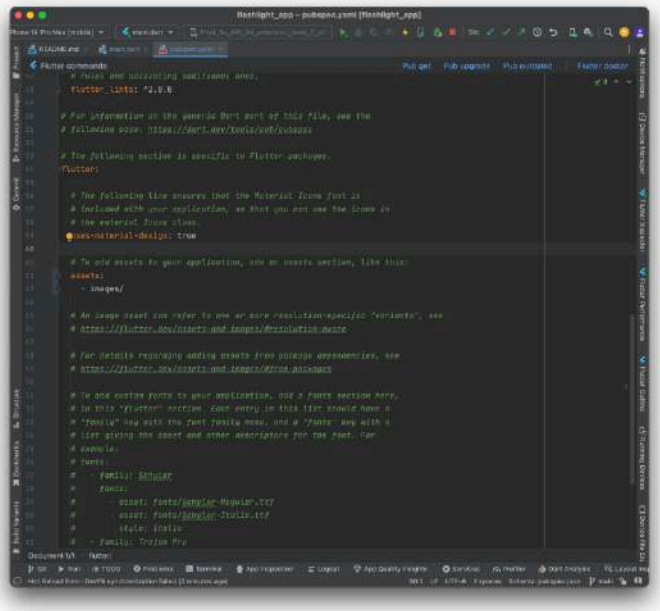


- Add 2 flashlight images; flashlight off and flashlight on. (You can find the images used in this tutorial in the article resources)



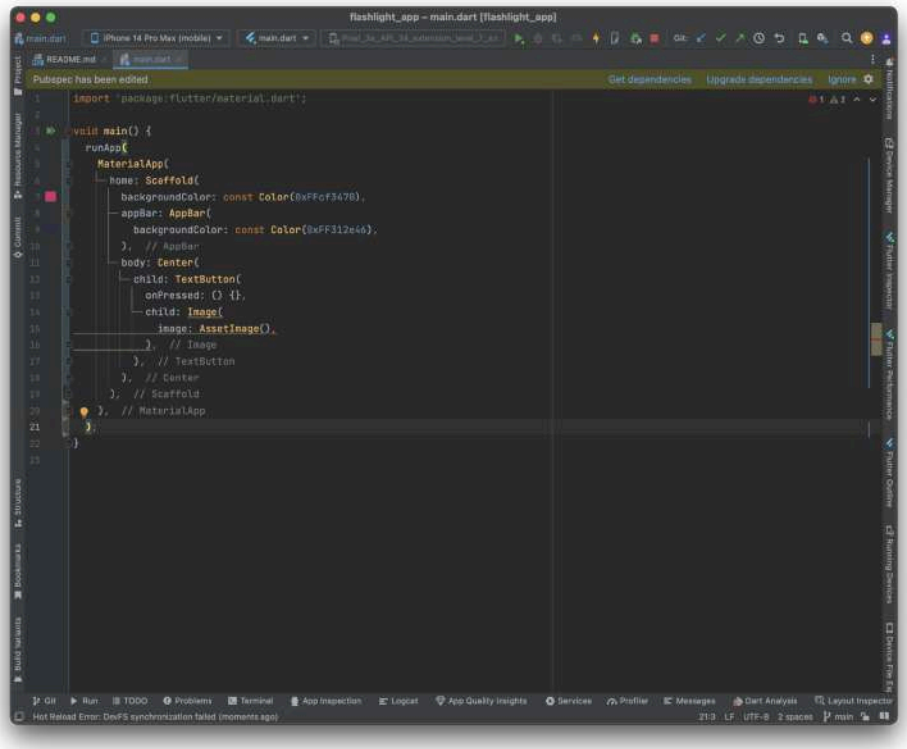
- Open the **pubspec.yaml** file located at the bottom of the app folder. We need to notify Flutter about our new assets. We can do this under the “assets” section by adding the path to our flashlight images.





- **Save changes** and press the **“Get Dependencies”** button in order for our app and files to sync. At this point we can use our images throughout the app.

- As the *image property* of the **Image Widget** return the **AssetImage Widget** and in th h simply enter the folder’s name and the image of a *turned off flashlight*.



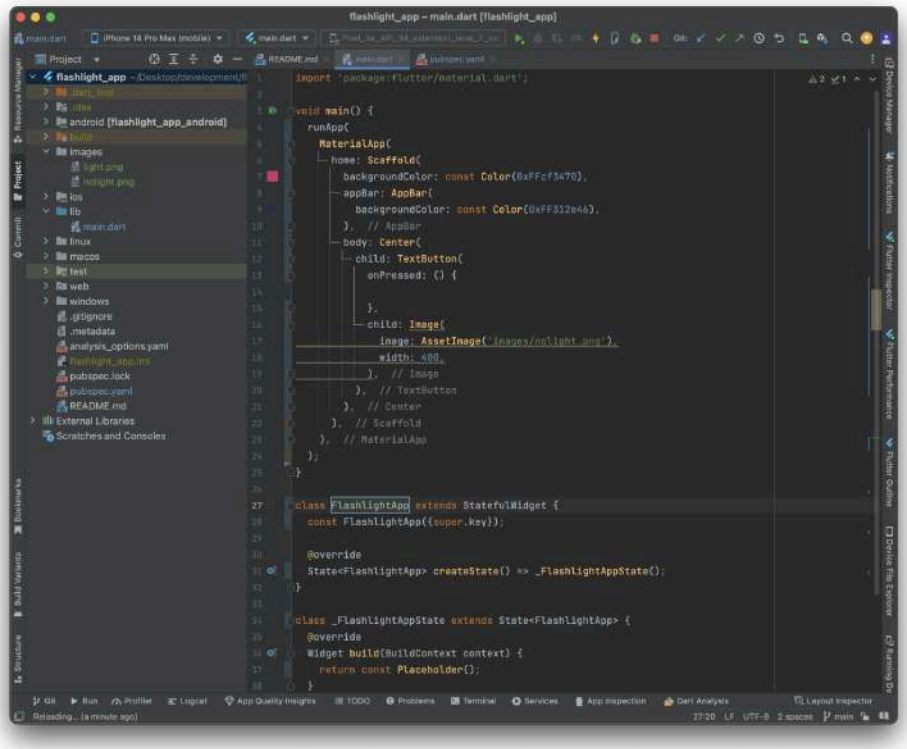
Stateful Widgets

In order for our Images to change we need to add a function that allows us to do so. The TextButton Widget has the **onPressed function** property. This way we can make our images change. **Every time a user presses the image it will create the "on/off" effect of a real flashlight!**

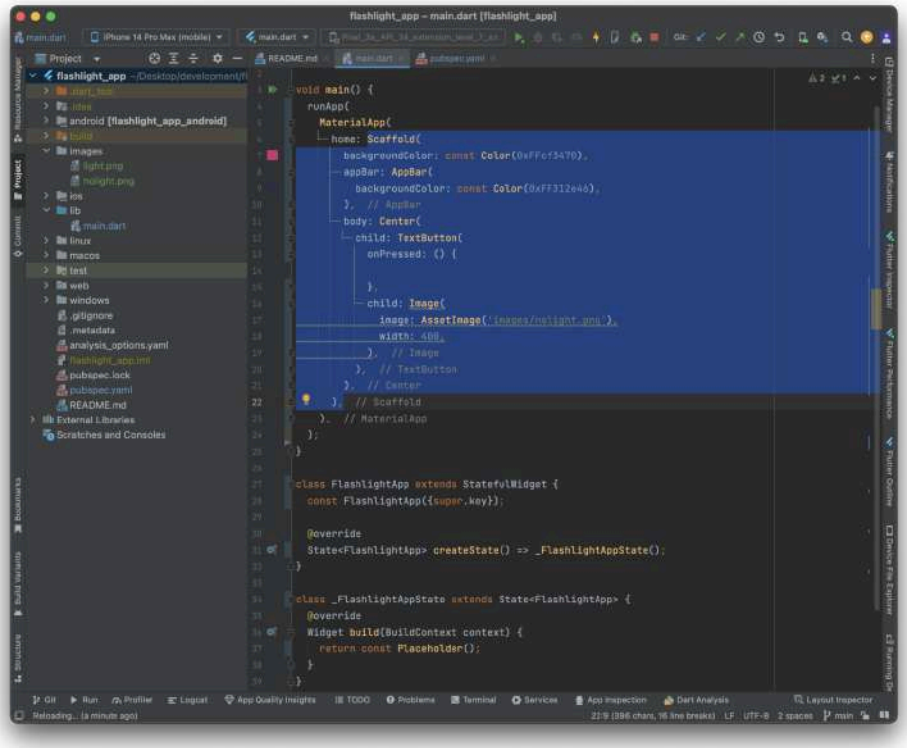
To make this work we will need a **setState function**. This function can only work with **Stateful Widgets** which allow for changes like these (images changing) to happen. Currently, MaterialApp Widget is **Stateless** – the opposite of what we need.

Let's create a seperate Stateful Widget and move our code there! **This Stateful Widget will be returned from the home property of the Material App instead of our Scaffold directly.**

- Under the main function create a Stateful Widget (stful keyword)
- Name it "Flashlight App"



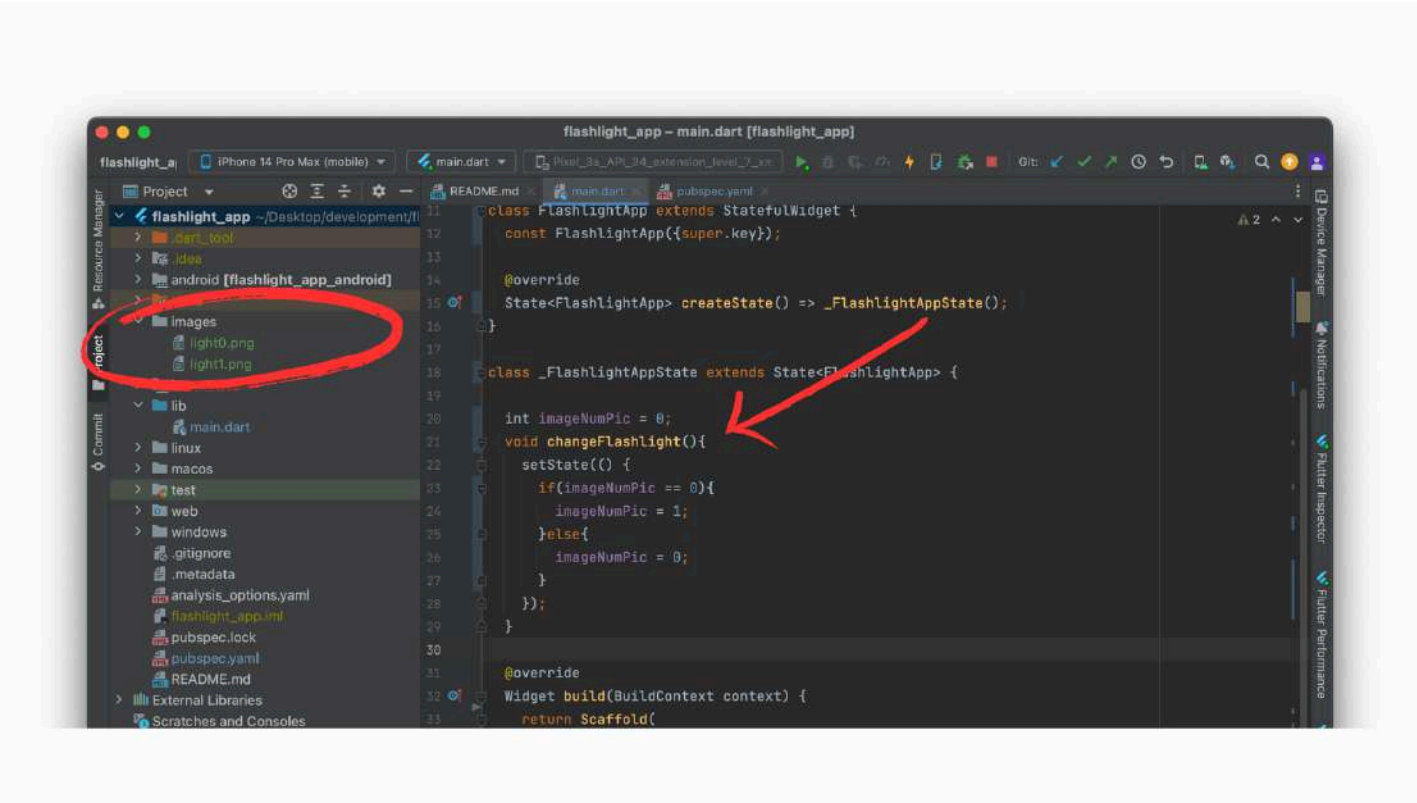
- Return all of the code from the home property in the new Flashlight App Widget



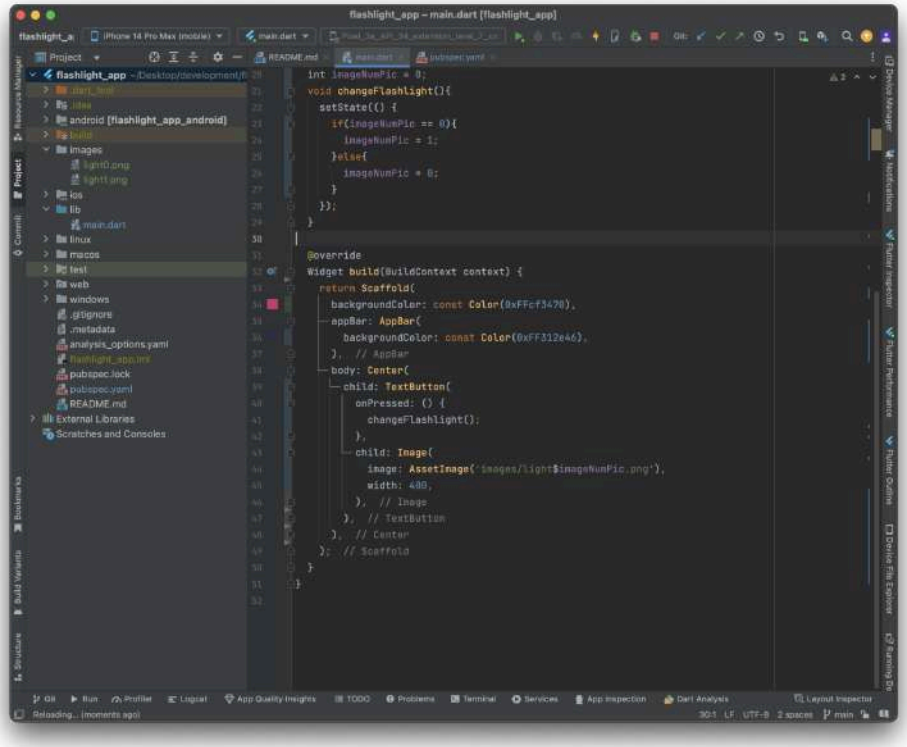
onPressed Function

- Create an **integer variable "imageNum"** and set it to 0
- Create a new function in the **Flashlight App Widget – changeFlashlight()**
- In the body of the function use **setState**
- Rename flashlight images in the flashlight folder to include 0 (no light) and 1 (light)
- If imageNum variable equals 0 set it to 1

- Else set the imageNum value to 0

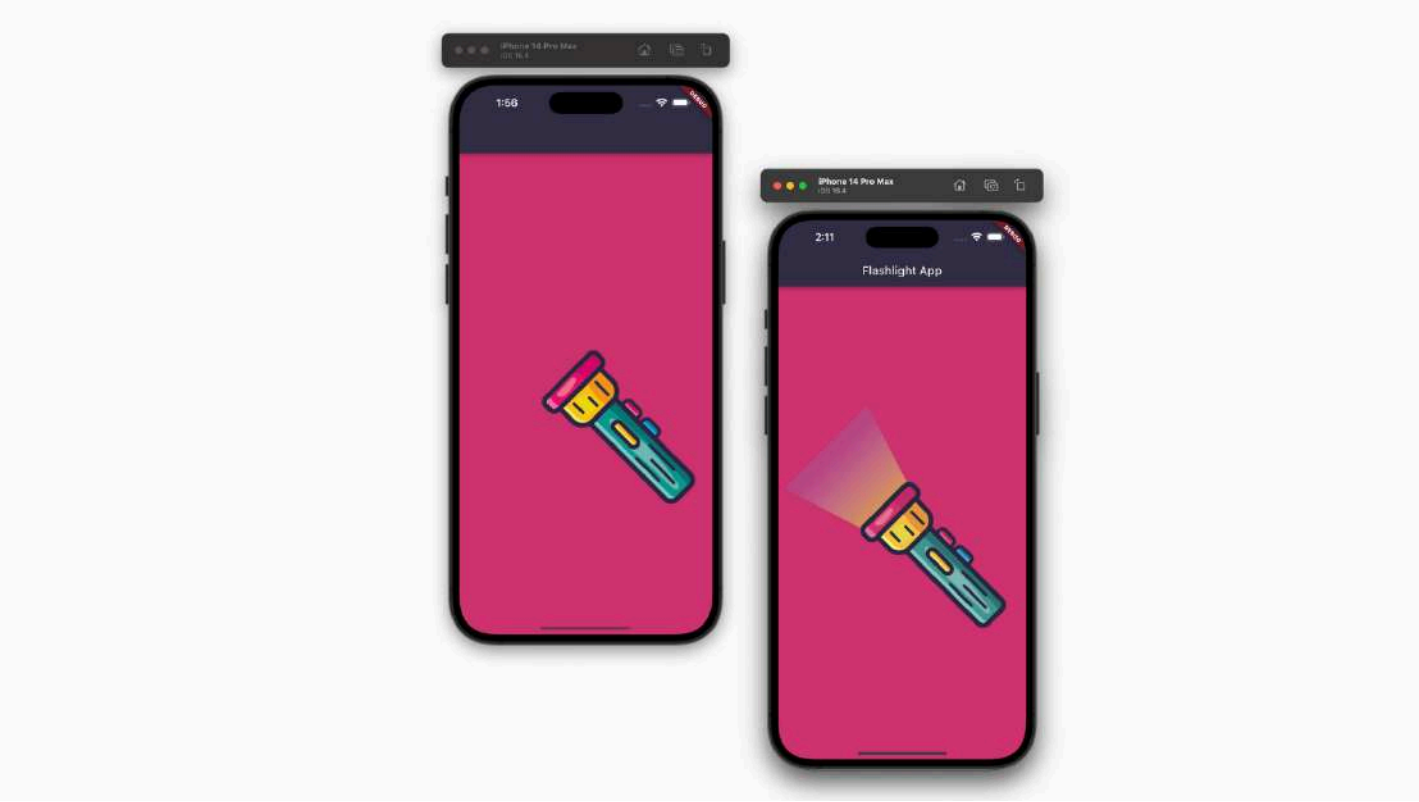


- Return changeFlashlight function from the onPressed property in the FlatButton Widget
- In the AssetImage link add the imageNum variable



Finished Product

And you're set! You should end up with something like this:



There is so much to explore in the **Mobile App Development world!** Stay tuned for more and check out my *Flutter Development* series. **Remember practice makes perfect.** In the resources below you can find assets used for the development of Flashlight App and my Flutter Series. Good luck on your journey! 🙂

Resources

Flashlight App Source Code and Assets – Github

Flutter Series:

Introduction to Flutter – by Irma P.

Developing with Flutter – part II – by Irma P.

Developing with Flutter – part III – by Irma P.

Dicee – Developing with Flutter – by Irma P.

Keep creating! – Irma 🤖

5 Passion Projects to try in 2024

August 18, 2023

Categories: Let's read

Tags: 100DaysOfCode, JustCode, LitCode, Passion, Projects

Introduction

Welcome to LitCode's new article! In today's post, I'll give you an overview of **5 Passion Projects to try in 2024**! There are only a few months left in 2023, and it's time to think about your next big thing!

I worked on projects similar to these and I can guarantee you will learn a ton! I know one thing, **the best way to practice your programming skills is through projects**. Whether that be a project from my list or a mobile app, website, game, give it a try!

For anyone just starting their coding journey, buckle up; you're off to a great start!

What are Passion Projects?

A Passion Project is a creative undertaking born from a deep enthusiasm for a particular topic, skill or idea! That can be a side project where you explore and create something you are excited about.

Sometimes, Passion Projects serve as an escape from regular tasks and obligations. They allow us to dive into something we are truly passionate about. *By doing something we love, we automatically create room for learning, experimenting and showcasing our best skills!*

Imagine someone who loves both coding and music, so they decide to create an app that creates unique, randomized music using just a few key notes. These projects don't have to include strict rules and requirements – **you should enjoy them!**

5 Passion Projects to try in 2024

Now, let's dive in!

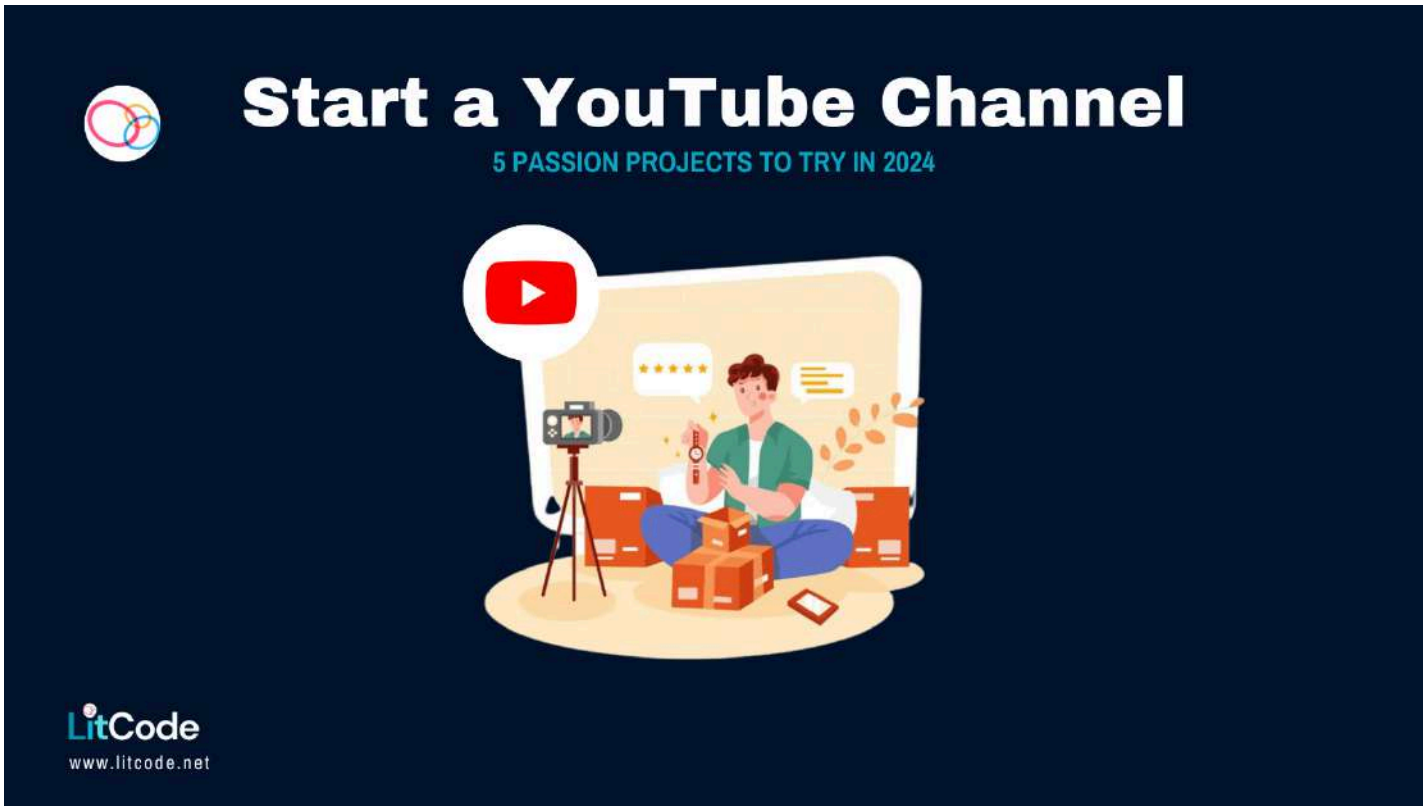
1. Start Blogging

I cannot stress this enough, but **DON'T WASTE YOUR TIME** and start blogging! This way you can both help yourself and others. By writing about topics you're currently working on, you are practicing your skills in that area while also sharing your knowledge with others!

Blogging opens a whole new world of opportunities. It allows you to connect with others interested in the same topics as you! Sharing your journey can inspire and motivate others while also receiving feedback and learning from other people's perspectives.

Through blogging, you can improve your writing and communication skills which are now crucial in tech. Treat blogging as your cheat code for improving your expertise and possibly becoming a respectable voice in the tech world.

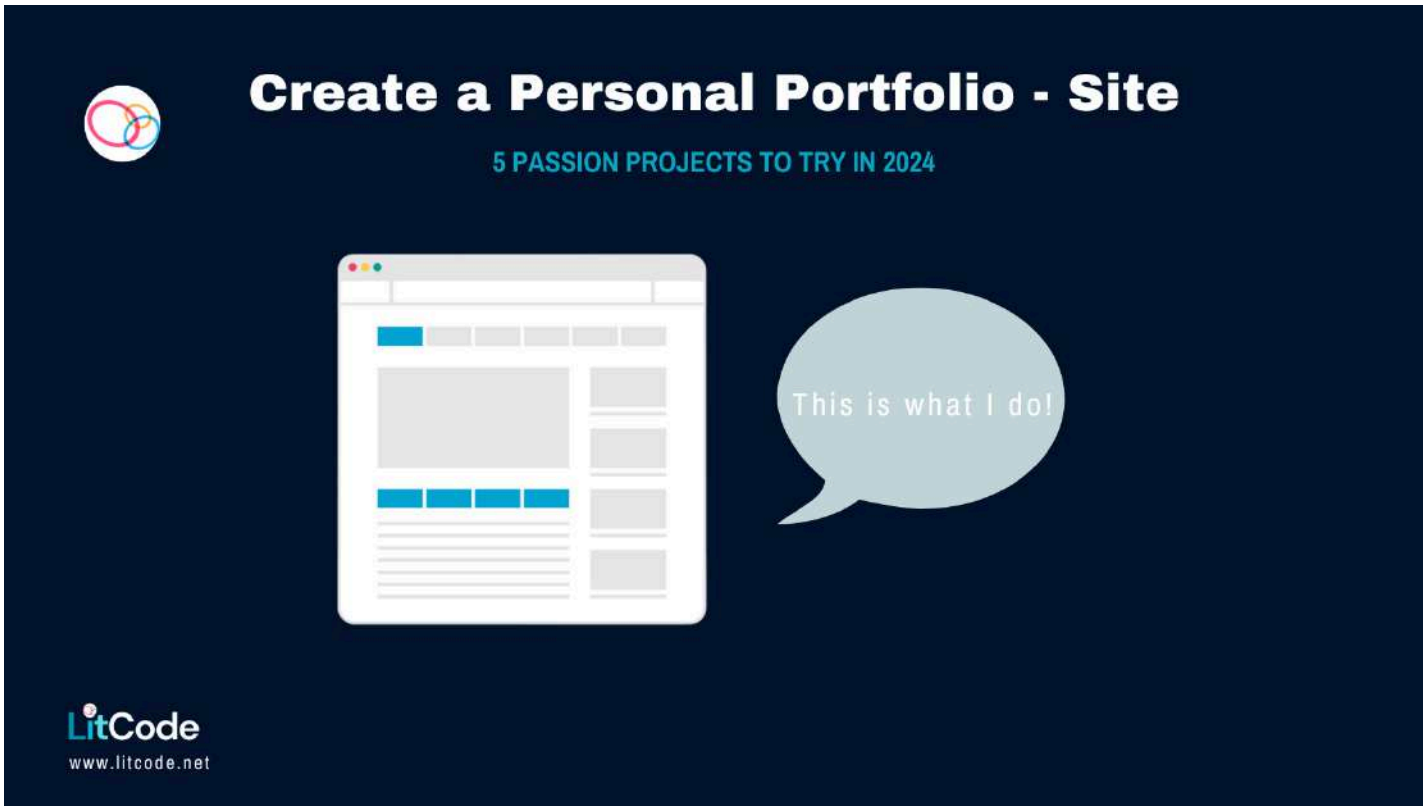
2. Start a YouTube Channel



Starting a Youtube Channel is an amazing opportunity to share your expertise with a global audience! There are so many possibilites. **You can create tutorials, walkthroughs and share your experience on your programming journey.** This way you are developing a solid understanding of the topics you’re interested in while sharing your perspective with others.

You can connect with people who share your interests and possibly collaborate on cool projects! There are so many opportunities for monetization and eventually turning your passion into a job.

3. Create a Personal Portfolio Site



Creating a Personal Portfolio Site is a fantastic way to showcase your skills and work online! This way you can present your projects in an interactive and out of the box way. Websites like this are a great tool for potential employers to see what you're capable of.

This way you can stand out and share your personal style and brand with the world. You can share your growth and progress over time, providing details of your projects and giving your audience a bigger understanding of various topics.

No matter how you decide to create it, a Personal Portfolio serves as a digital record of your projects and accomplishments.

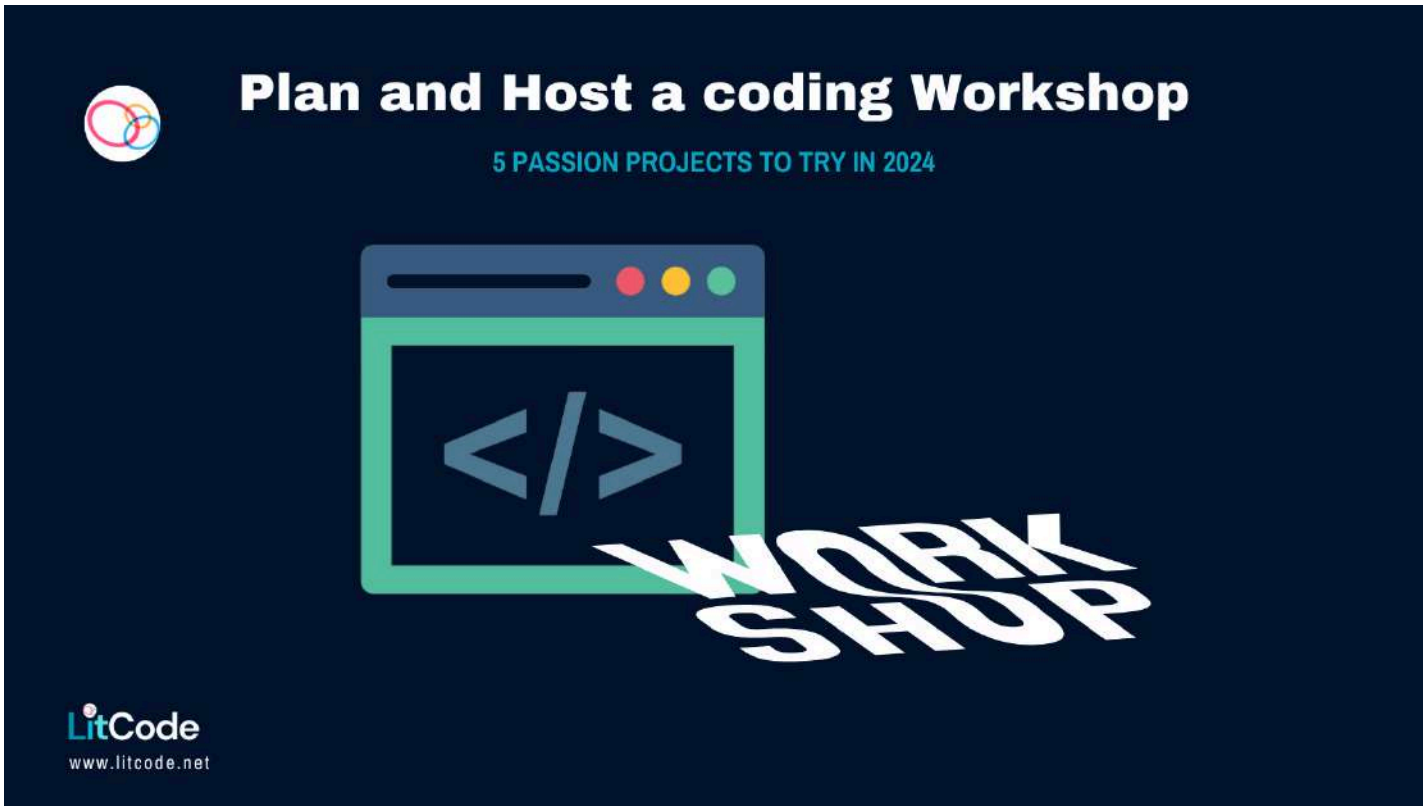
4. Create an Online Course

Online Programming Courses are a way of reaching a global audience of learners interested in topics you are an expert in. They allow students to cover the material at their own pace. This is a flexible way of sharing your knowledge.

Sharing your programming expertise establishes you as an ***"expert"*** in this field (you can't just learn to program, it's a never ending process). Feedback you receive from students can help you improve your content but it also offers a new perspective that you might not even think about otherwise.

Creating an online programming course on platforms like Udemy lets you leave a lasting impact, students from various backgrounds can benefit making something impossible – possible for someone across the world!

5. Plan and host a coding workshop

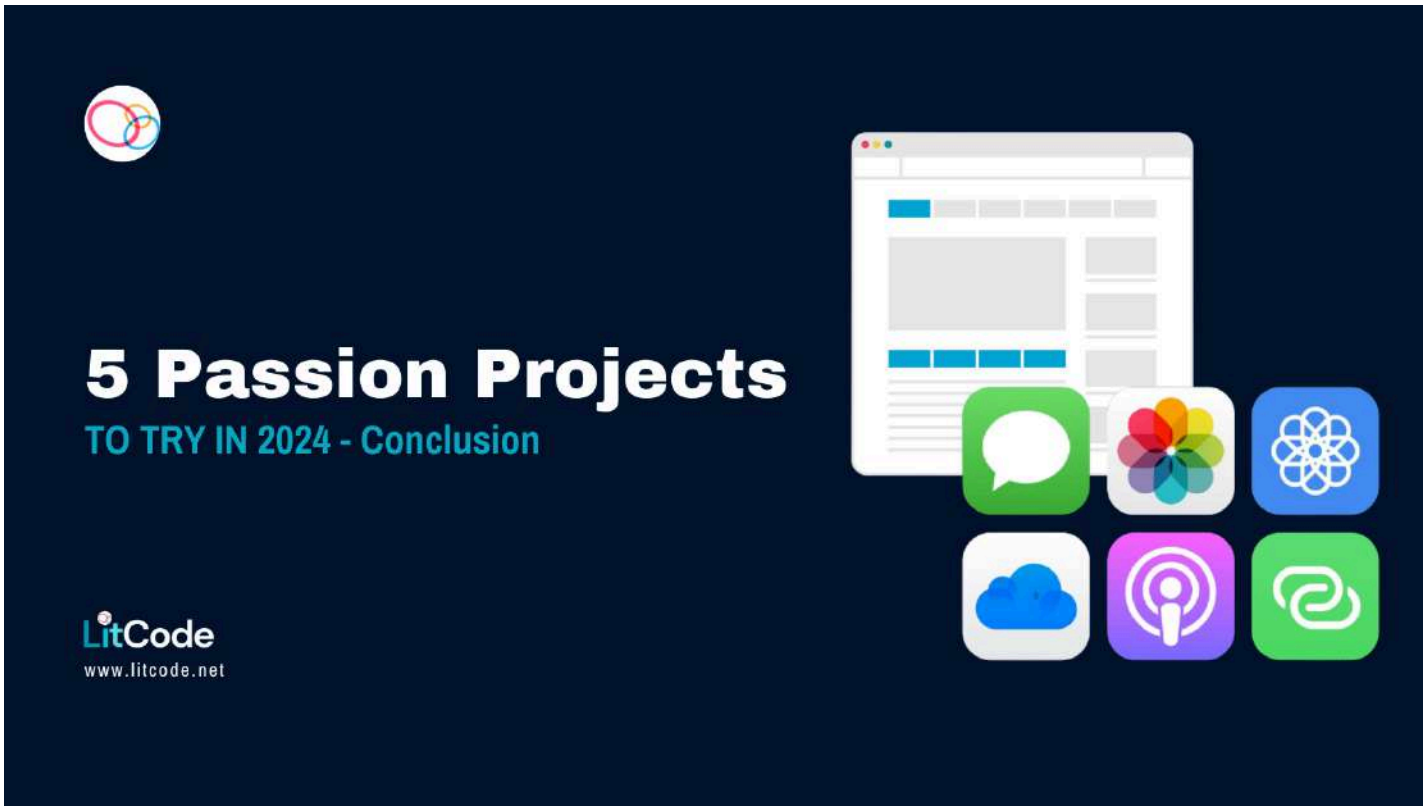


Planning and hosting a coding workshop is a process that involves careful preparation to ensure the event's success. One of your first steps is to decide on a ***topic and target audience.*** This way you can make sure you're organizing a focused and effective workshop.

By organizing this type of event you are fostering a coding community in your area! Participants can benefit from your expertise and by sharing knowledge with each other. Workshops like these offer networking opportunities, allowing you and the attendees an opportunity to connect with each other!

Who knows, by hosting a workshop you just might inspire someone to write their first "Hello, World!".

Conclusion



There are so many more Passion Projects beyond this list, but I hope I cleared up some concepts for you! Stay tuned for more on this topic and check out my other articles! ***Remember practice makes perfect, and passion projects are just that: a way to practice your knowledge.*** Good luck on your journey! 🙂

Resources

Throwback Article: Why should everyone learn to code? – by Irma P.

How to earn money as a programmer? – by Irma P.

Mobile Apps to Develop this Summer – by Irma P.

Think like a programmer – by Irma P.

Keep creating! – Irma 🤖

Create a stunning Portfolio Website 2024

June 3, 2024

Categories: Let's learn

Tags: Goals, LitCode, Programmer, ProgrammersLife, Web Development

Welcome back to LitCode! After a long break (longer than expected) I am finally back 🙂 . Today we are starting fresh and diving into the world of **Web Development**. In today's guide, we'll cover everything you need to know to create your **OWN** portfolio. Get ready to be inspired and create a stunning Portfolio Website for summer 2024!

I hope you enjoy. **Now, let's get to it 🚀 !**

What is a Portfolio Website?

A portfolio website is an individual's own online space where they display their accomplishments, talents, and work. Designers, software developers, photographers, authors, and painters frequently use it to showcase their greatest work to prospective clients, employers, or partners. Portfolio websites, compared to standard resumes, offer a *dynamic* and *visual* way to showcase someone's skills, making it simpler to show off expertise.

Why create one?

A portfolio website plays a **crucial role in career development**, offering endless opportunities that extend beyond traditional resumes and recommendation letters. It serves as **a digital business card**, showcasing your work and abilities.

A portfolio website can be designed to be visually engaging making a strong first impressio

Check out the Designs at the end and achieve the WOW effect 😉

You can **add a variety of media**. It allows you to present examples of real work, projects, designs, articles or code examples.

A portfolio website is **available 24/7 from anywhere in the world**, making it easier for potential employers, clients or collaborators to view your work. This helps you establish and communicate your **personal brand** better and easier!

Key Features to include

Hero Section – the first thing people see after searching you up. This page helps the reader understand who you are, your work and values you offer. This is often the page that redirects users to other sections of your portfolio website.

About Me Section – this section provides a brief introduction to who you are, your background and your work. Here it is best to share your story **(we all have one)**, career journey and showcase your talents!

Contact Information – a portfolio website often includes a contact form and contact details making it easy for potential clients and employers to get in touch with you. These are your **email address, phone number** and links to your **social media** profiles.

My Work Section – here list your skills, services and past projects you've worked on. This helps visitors understand what you offer. This includes technical skills, previous projects, types of services you provide.

Design and Aesthetics – the design of your portfolio should reflect your personal brand. It should be **user-friendly, visually appealing** and **responsive**!

Create your own

Deploying a portfolio website for the first time can be quite a daunting task. This process involves several steps and without prior experience it can be confusing. I am here to help you out. **Let's break down the important steps and make this process smoother for you!**

Pick a Domain Name

Choose a name that is easy to remember and relevant to work and projects you create! Ideally, include keywords related to your personal brand and business.

Decide on the appropriate extension (.com, .org, .net, etc.). Even though .com is the most popular one, other extensions might suite you more!

Register your chosen domain name through a domain registrar like:

- GoDaddy – one of the most popular domain registrars and web hosting providers, offering a wide range of services to help kickstart your portfolio website!
- Google Domains – provided by Google. This is a popular choice, offering a user friendly interface and competitive pricing!
- BlueHost – offers great hosting choices and comes with with built in WordPress platform!
- NameCheap – one of the more popular ones, offers free domain privacy – suitable if you need a domain ASAP.

Get a Hosting Platform

Web hosting is where your website's files are stored and served to visitors. There are several different types of hosting services, such as: **shared hosting, VPS hosting, dedicated hosting, cloud hosting** etc. Often, your domain name registrar offers a hosting plan, making it easier for you to manage everything in one place!

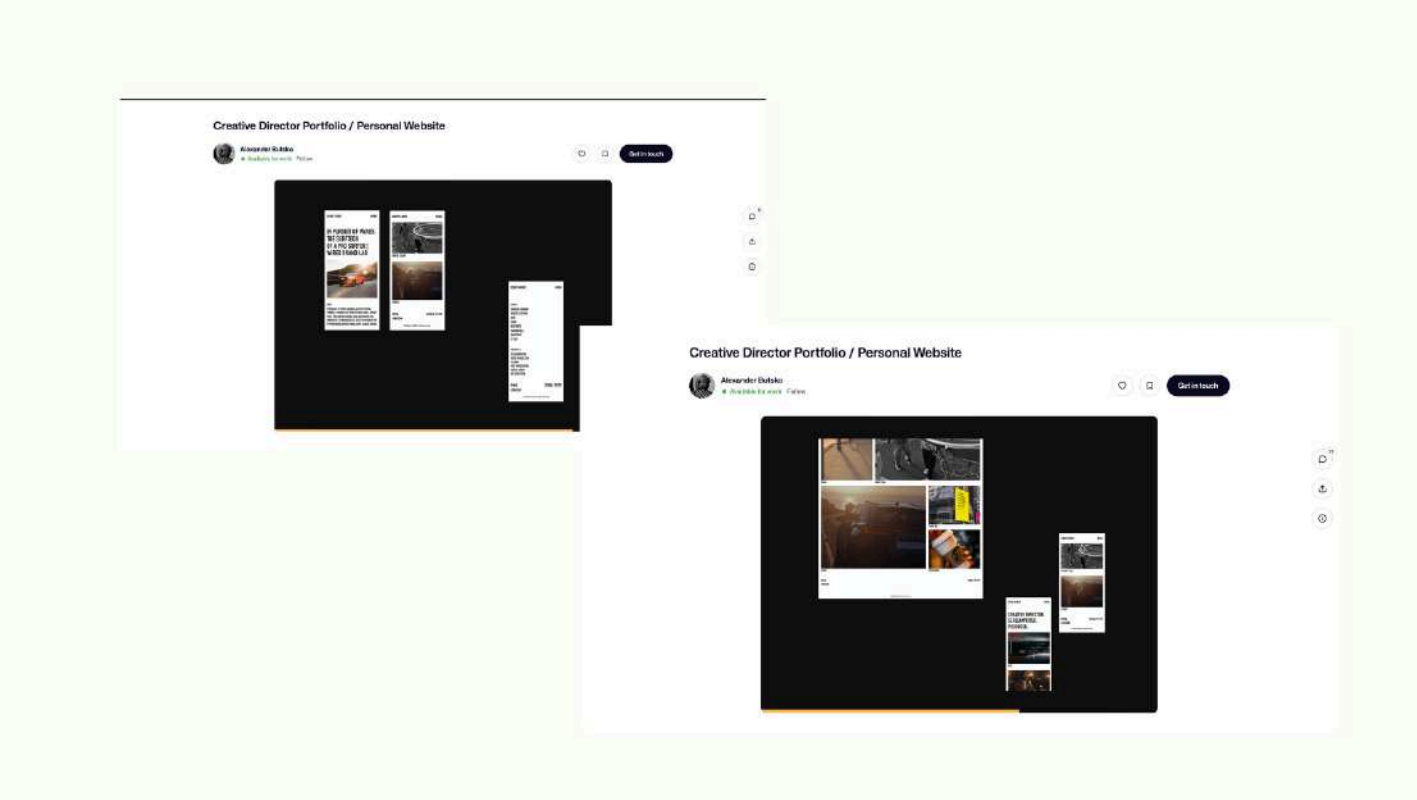
Deploy your site

This process requires several steps regardless of the hosting provider you choose. After securing your domain name and hosting plan, you should ***access your hosting account's control panel*** to set up your server environment. Next ***upload your website files*** using the hosting provider's file manager. Finally, ***update your domain's DNS settings*** to point to your hosting server – this step allows users to access your website by entering your domain name in their browsers!

Portfolio Designs 2024

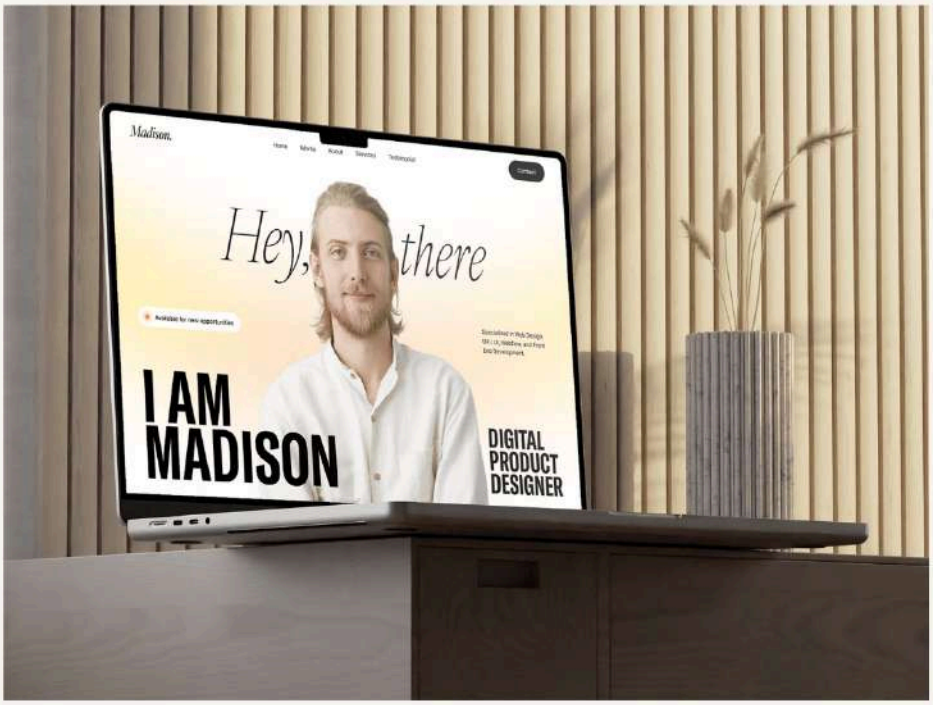
Summer 2024 is just around the corner and here are some stunning designs to inspire your portfolio's aesthetics 🦈 🌞 !

1. Orix Creative – Personal Portfolio Website



2. Devanta Ebison – Portfolio Design Exploration

3. Alexander Butsko – Creative Director Portfolio



Conclusion

In conclusion, creating a portfolio website is an important factor when promoting your personal brand and business. A well-crafted portfolio serves as a powerful tool for showcasing skills and projects. By following the guide provided in today's article, you can navigate this process with confidence, from securing a domain name, hosting provider, deploying your site and designing it professionally!

Good luck on your journey!

Resources

Throwback Article: 5 Passion Projects to try in 2024 – by Irma P.

GoDaddy, Google Domains, NameCheap and BlueHost – domain registrars and hosting providers

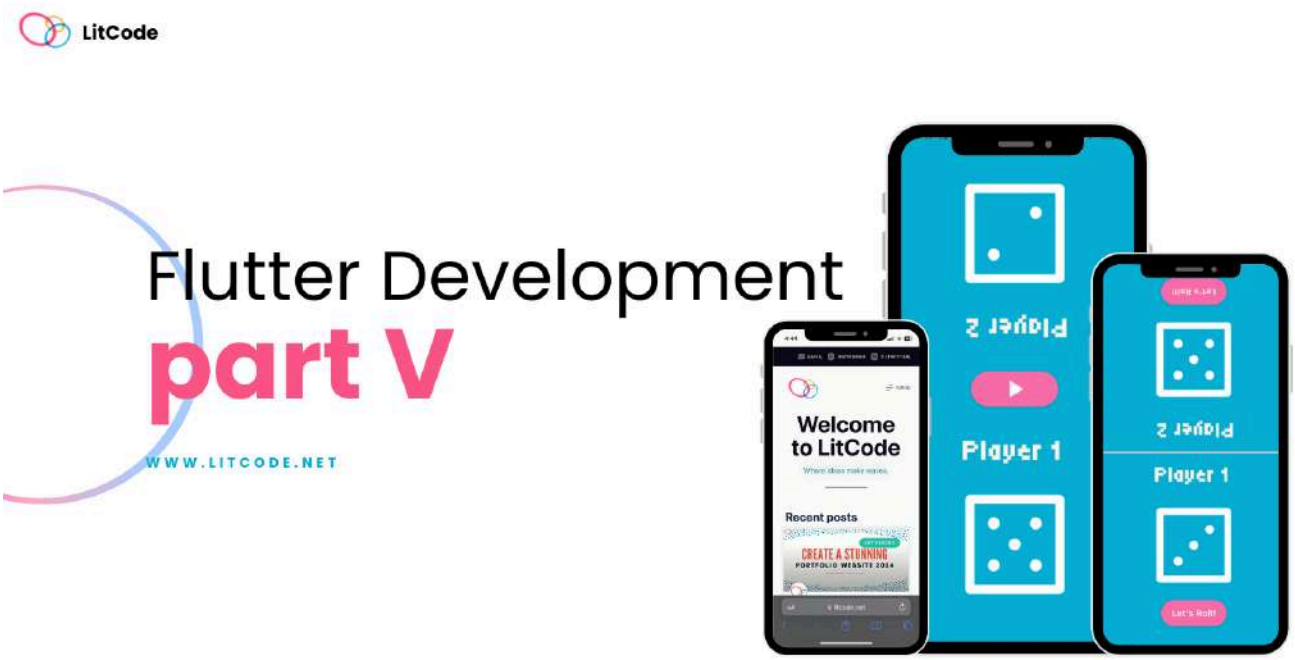
Keep creating! – Irma 😄

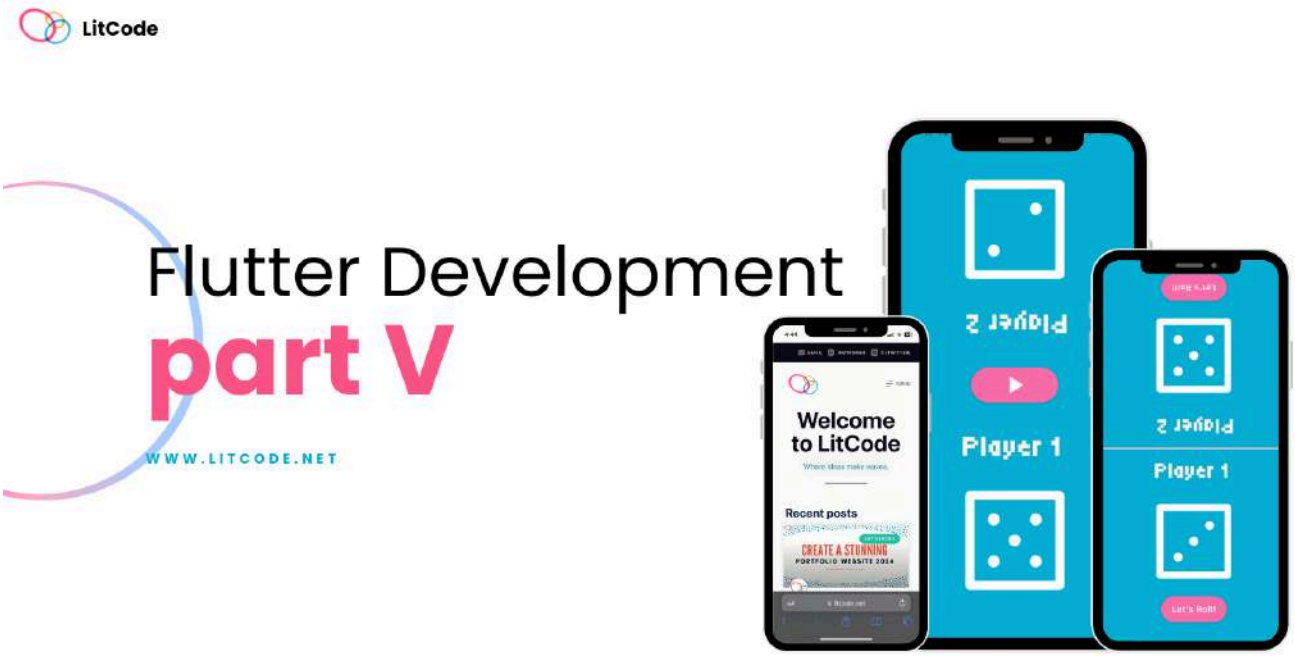
Flutter Development – part V

June 21, 2024

Categories: Let's create, New

Tags: Coding, Flutter, Goals, JustCode, LitCode, Mobile App Development





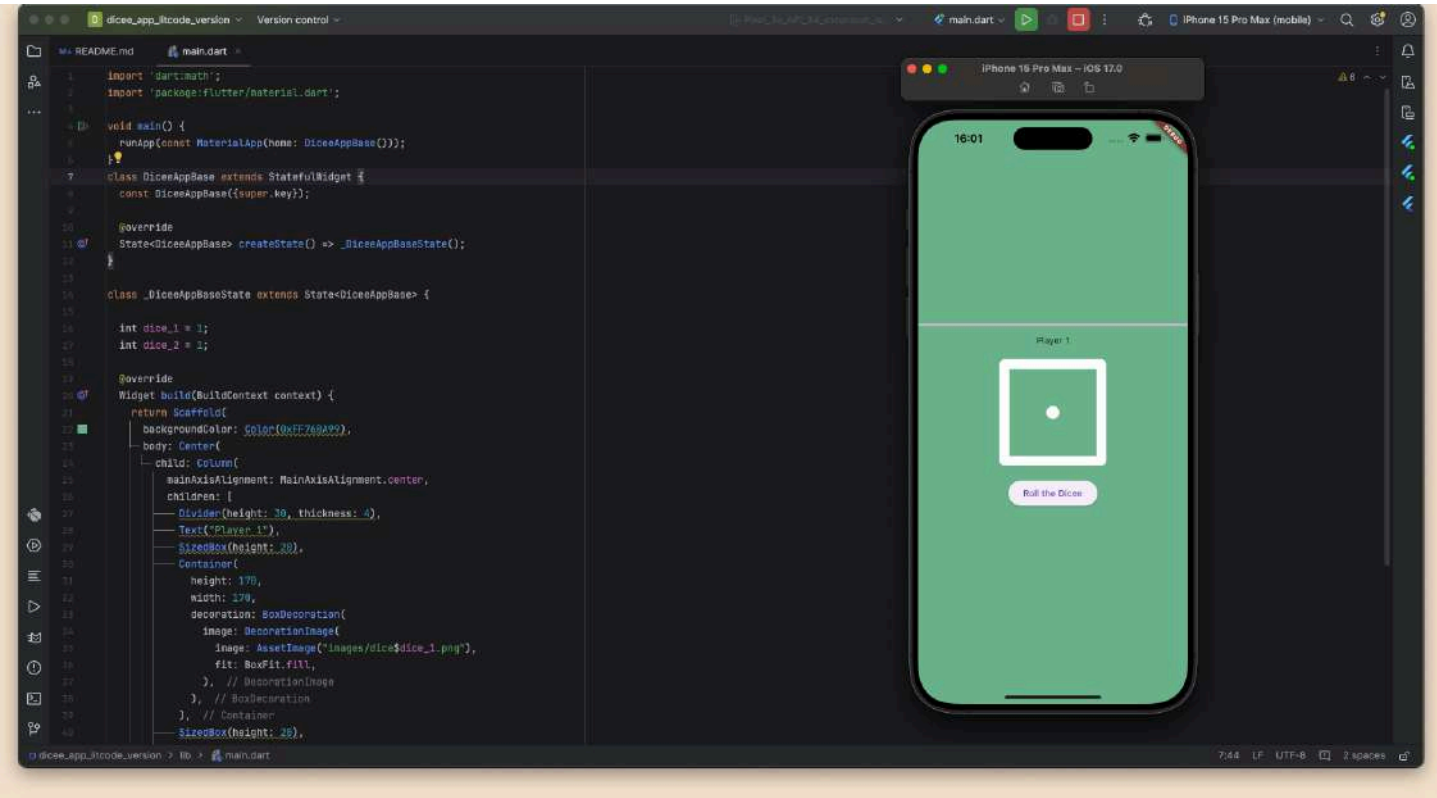
Welcome to Flutter Development – Part V! Today, we will complete the **Dicee Mobile App**. Throughout this series, we've covered the basic syntax of Dart, introduced Flutter Widgets,

customized our app's UI, and much more. I hope you enjoy this tutorial and love the final product 😄 .

Let's code!

What we have built so far

So far the Dicee App looks like this:



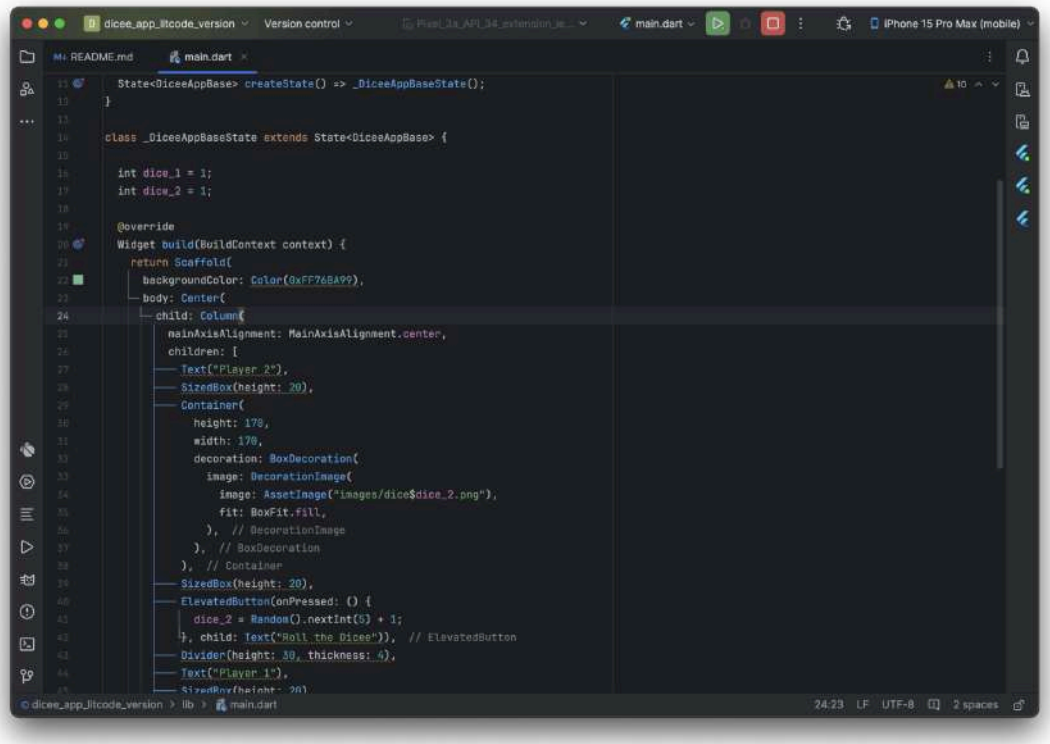
In Part 1 we:

- created and ran our Flutter app for the first time
- created the Dicee Base Widget
- learned how to import Images into our project
- added the Player 1 basic interface

Today we will build the **Player 2 board, work on refactoring our code and polish the UI!**

Adding Player 2 Interface

First we are just going to **copy and paste the code we used to create the Player 1** part of the board. Above portion of the screen belongs to Player 2 so make sure to make Text Widget c ges!



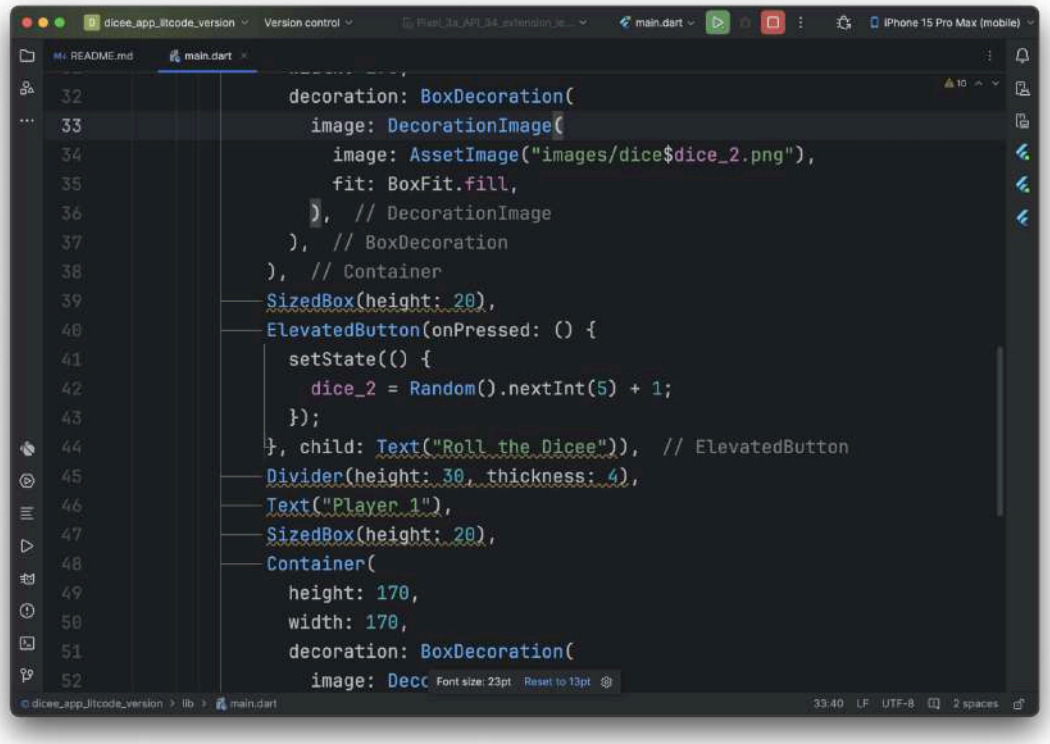
Now we have two identical player screens. As simple as that!

Fixing the Button

If you've tested the app functionality in Part 1 you might have noticed that our "Roll the Dice" button really doesn't do anything. Now is the time to fix that!

Let me introduce you to setState() method. According to the official Flutter documentation: ***setState() is a method provided by the State class in Flutter. It's commonly used for updating the state of a StatefulWidget. Meaning, we use setState() whenever we want to make some dynamic changes to the UI of our app. In our case, the dynamic change is changing the Dice Image every time the user presses the button. setState() method makes this UI change possible.***

So make sure to add setState() inside the onPressed() call in our ElevatedButton Widget.



IMPORTANT NOTE: add setState() to both Player 1 and Player 2 boards!

Flipping second player's perspective

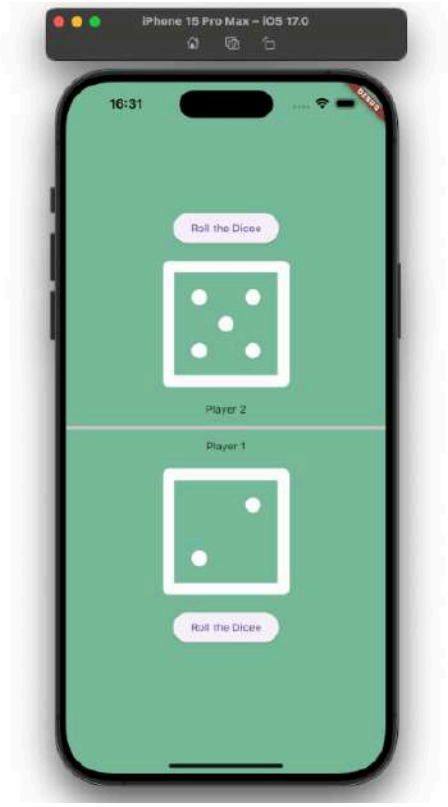
We're not done yet! Even though we added the Player 2 part of the app, the second player's perspective is inverted. The "Player 2" Text and Button Text should be rotated by 180 degrees so the user can easily read it from their POV.

We're going to start by flipping the order of Widgets. Let's reverse it by placing our ElevatedButton on top, below we have our Image Container and finally at the bottom we have our "Player 2" Text.

```
int dice_1 = 1;
int dice_2 = 1;

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Color(0xFF76BA99),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          ElevatedButton(onPressed: () {
            setState(() {
              dice_2 = Random().nextInt(5) + 1;
            });
          }, child: Text("Roll the Dicee")),  // ElevatedButton
          SizedBox(height: 20),
          Container(
            height: 170,
            width: 170,
            decoration: BoxDecoration(
              image: DecorationImage(
                image: AssetImage("images/dice$dice_2.png"),
                fit: BoxFit.fill,
              ),  // DecorationImage
            ),  // BoxDecoration
          ),  // Container
          SizedBox(height: 20),
          Text("Player 2"),
          Divider(height: 30, thickness: 4),
          Text("Player 1"),
          SizedBox(height: 20),
          Container(
            height: 170,
            width: 170,
```

Now we have something like this:



Next we are going to use the Transform Widget to rotate our Text Widgets content by 180 degrees. Wrap the “Player 2” and Button child Widget with Transform and use these settings to achieve the inverted effect.

Player 2 Text Widget:

```
mainAxisAlignment: MainAxisAlignment.center,
children: [
  ElevatedButton(onPressed: () {
    setState(() {
      dice_2 = Random().nextInt(5) + 1;
    });
  }, child: Text("Roll the Dicee")), // ElevatedButton
  SizedBox(height: 20),
  Container(
    height: 170,
    width: 170,
    decoration: BoxDecoration(
      image: DecorationImage(
        image: AssetImage("images/dice$dice_2.png"),
        fit: BoxFit.fill,
      ), // DecorationImage
    ), // BoxDecoration
  ), // Container
  SizedBox(height: 20),
  Transform(alignment: Alignment.center, transform: Matrix4.identity()..rotateZ(3.14159),child: Text("Player 2")),
  Divider(height: 30, thickness: 4),
  Text("Player 1"),
  SizedBox(height: 20),
  Container(
    height: 170,
    width: 170,
    decoration: BoxDecoration(
      image: DecorationImage(
        image: AssetImage("images/dice$dice_1.png"),
        fit: BoxFit.fill,
      ), // DecorationImage
    ), // BoxDecoration
  ), // Container
  SizedBox(height: 20),
  ElevatedButton(onPressed: () {
```

Button child Widget:

```
int dice_1 = 1;
int dice_2 = 1;

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Color(0xFF76BA99),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          ElevatedButton(
            onPressed: () {
              setState(() {
                dice_2 = Random().nextInt(5) + 1;
              });
            },
            child: Transform(
              alignment: Alignment.center,
              transform: Matrix4.identity()..rotateZ(3.14159),
              child: Text("Roll the Dicee"),
            ), // Transform
          ), // ElevatedButton
          SizedBox(height: 20),
          Container(
            height: 170,
            width: 170,
            decoration: BoxDecoration(
              image: DecorationImage(
                image: AssetImage("images/dice$dice_2.png"),
                fit: BoxFit.fill,
              ), // DecorationImage
            ), // BoxDecoration
          ), // Container
```

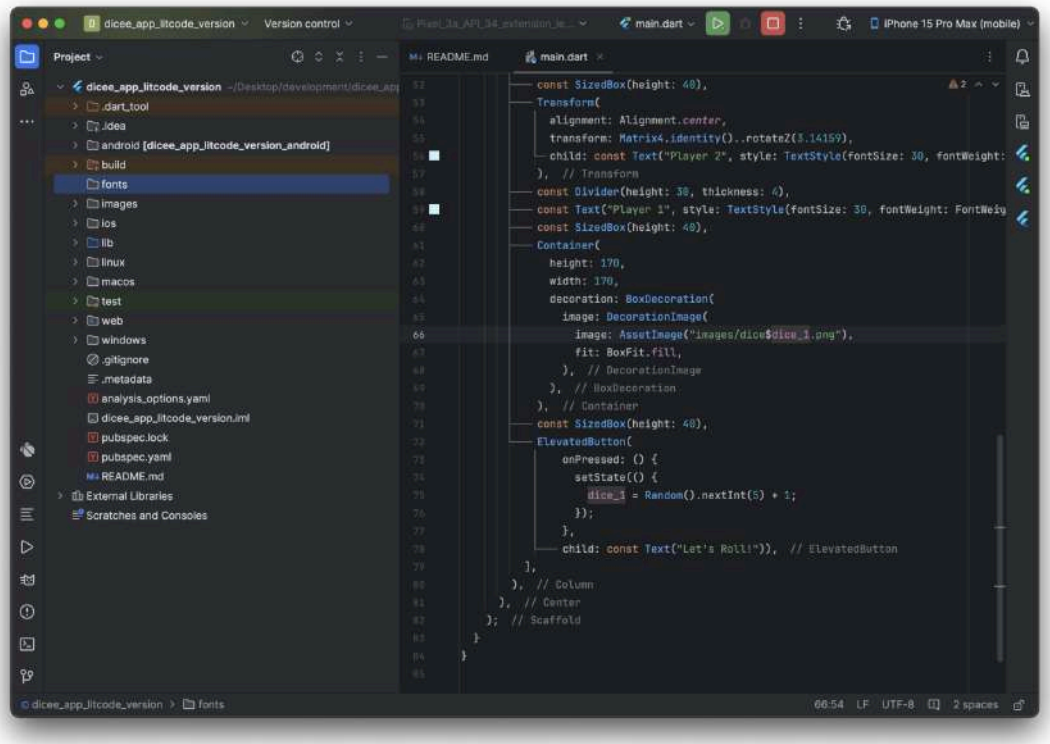
Voilà!

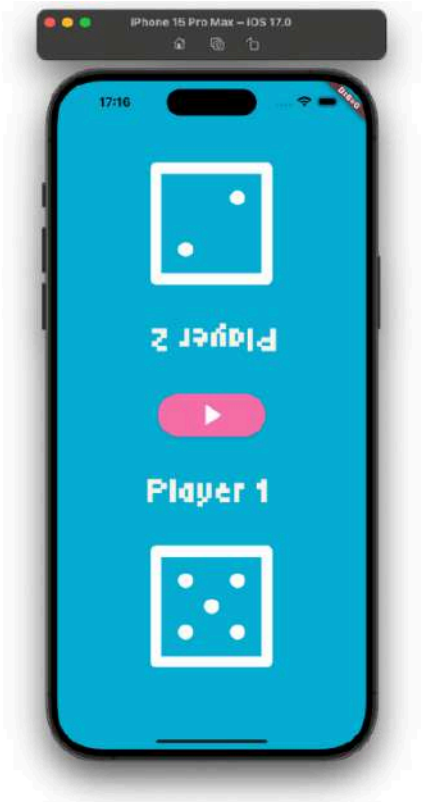
Customizing UI

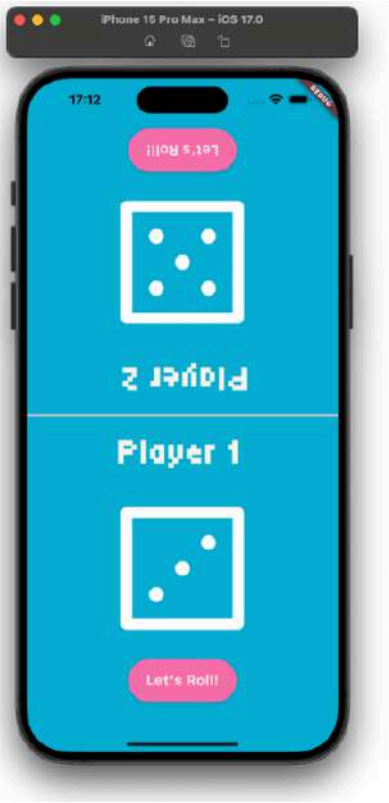
Here is a little challenge for you! Try to figure out what I did to create this UI. Below you can see 2 versions, single and 2 button one! The functionality is pretty much the same, except the single button version changes the images of both player's screens simultaneously!

Some hints:

- imported a new font (Tiny5 get it from Google Fonts)
- change the color scheme (play with it!)
- created the single and 2 button UI







Final Product and Code

You can find the code for the finished product on my GitHub page!

Conclusion

And there you have it – our Dicee Rolling App is complete! Throughout this journey, we've dived into the fundamentals of Dart, explored Flutter Widgets and UI customization. We've built an engaging and interactive app that showcases the power and simplicity of Flutter development.

Dicee App – Part 1 ✅

Inspire App ✅

I hope you enjoyed this tutorial! Stay tuned for more exciting tutorials, projects, and insights into the world of technology.

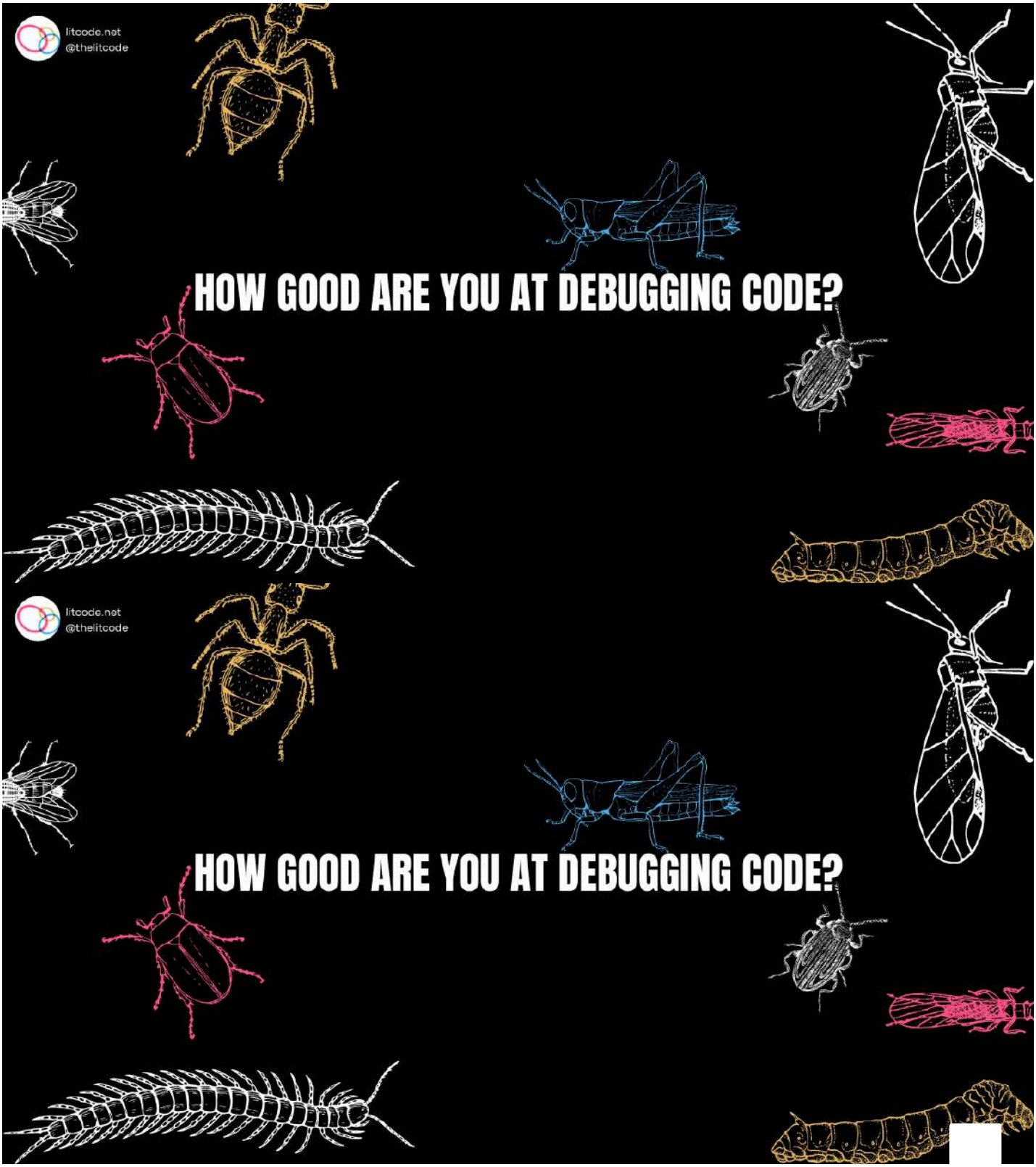
Thank you for following along, and happy coding! – Irma

How good are you at debugging code?

July 30, 2024

Categories: Quiz

Tags: C++, Java, JustCode, LitCode, Python, Quiz



Welcome to How good are you at debugging code – Quiz! Whether you are just a rookie or a seasoned developer this quiz is here to test one of your most essential skills – debugging. Here I will challenge your ability to spot and fix errors in C++, Java and Python. This quiz is here not only to test you, but to teach you how to avoid these mistakes in your code.

> **Don't fix bugs later, fix them now.**
>
> Steve Maguire – Writing solid code

What is debugging?

Debugging is the process of identifying and fixing bugs or errors in your code. It involves an inspection of code to locate the source of problems that cause incorrect or unexpected behavior in your program. Debugging is an important skill for developers, as it makes sure that software runs smoothly and performs its intended functions correctly.

There are some important steps in your debugging process. Even though it can look different for everyone!

- **Identifying the bug** – realizing that our code produced an undesired or unexpected result.
- **Analyzing the code** – read through the code and examine it to figure out the location of the bug.
- **Isolate the bug** – narrow down to the specific lines of code that are causing the issue.
- **Fix the bug** – use your knowledge and resources to modify the code and fix it.
- **Test the fix** – test all possible scenarios which can trigger the lines of code where the bug occurred.

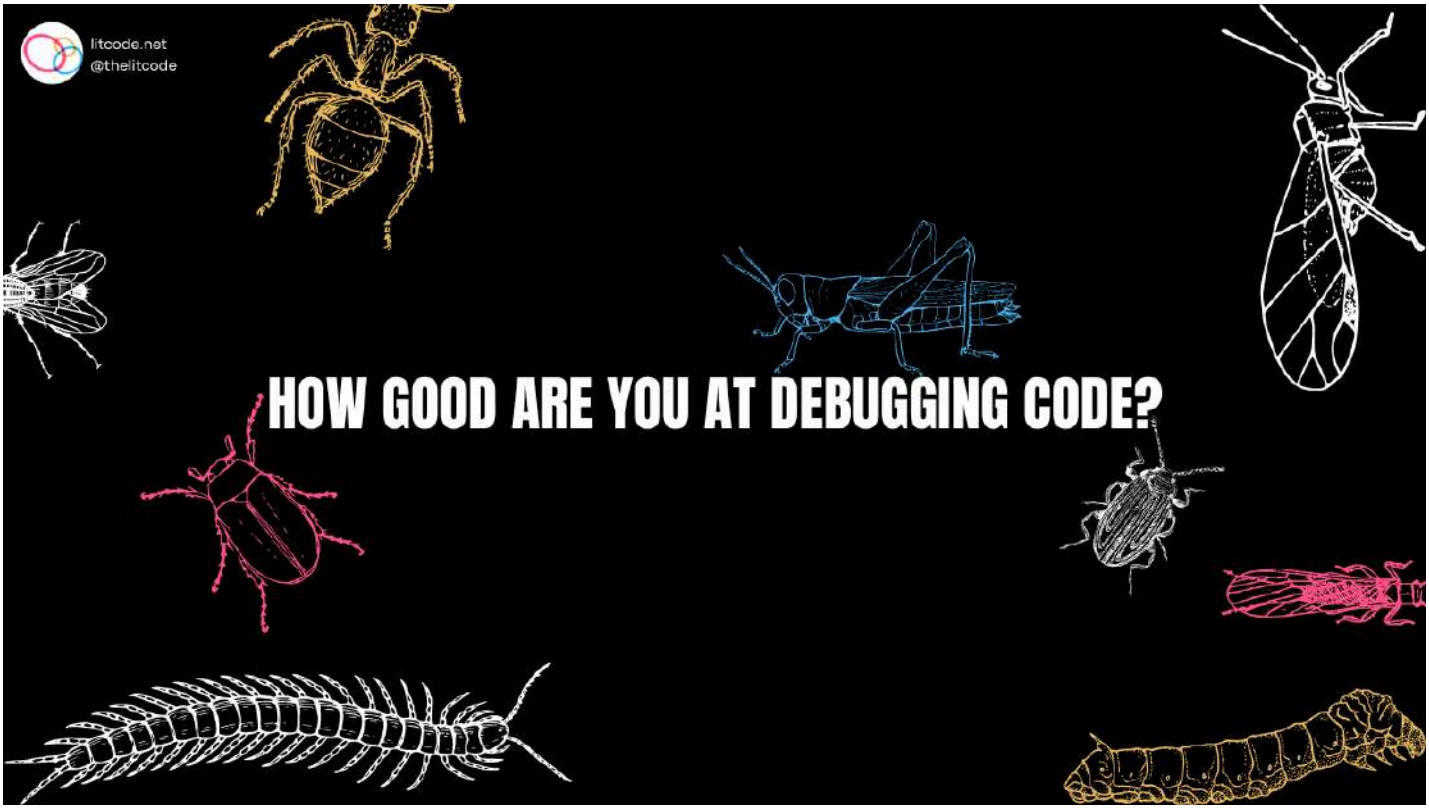
Types of bugs

There are several types of bugs which can occur in code, here are just a few:

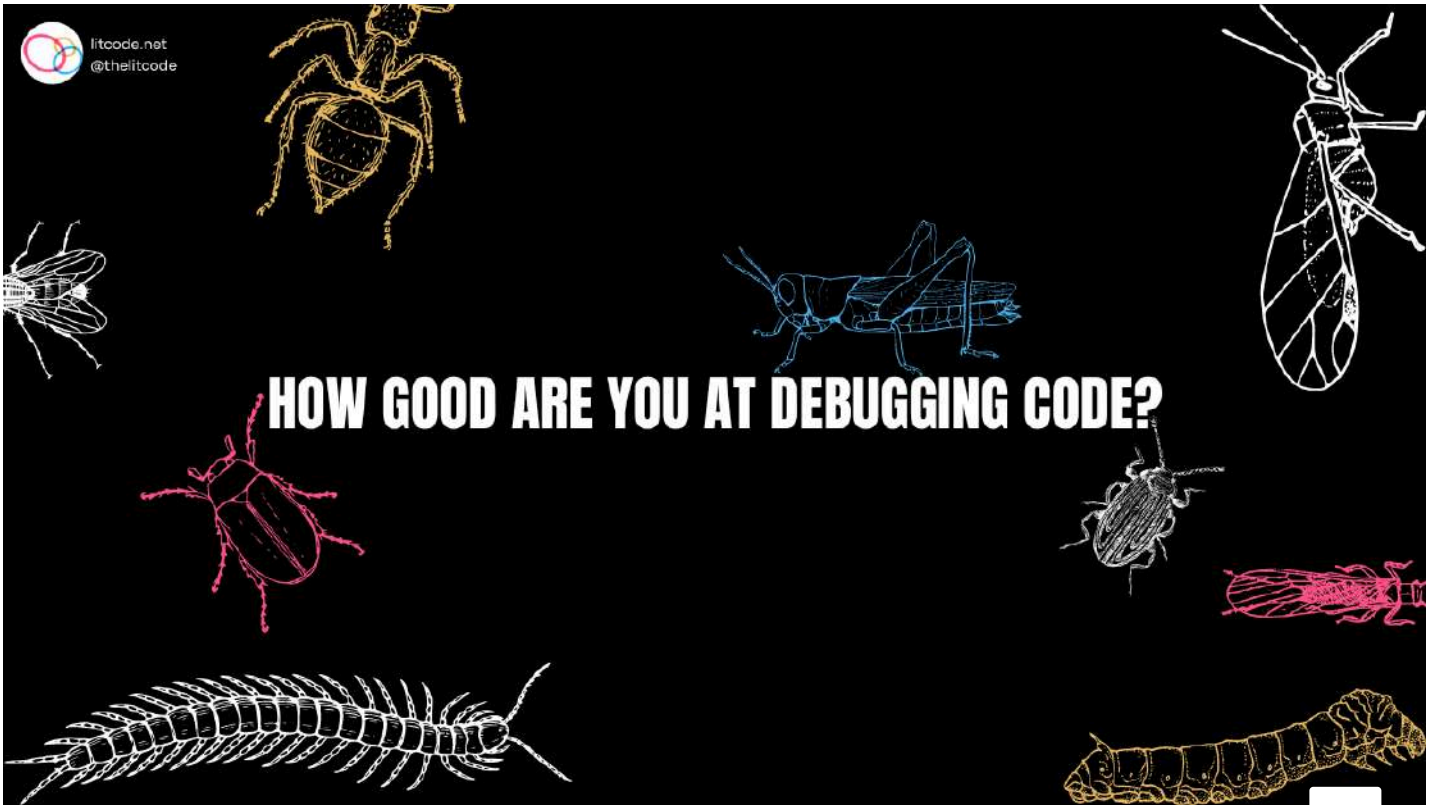
- **Syntax errors** – mistakes in the code that violate the language's syntax rules (forgetting a semicolon)
- **Logical errors** – the code doesn't crash but produces an incorrect result.
- **Runtime errors** – occur during the execution of the program.

- **Memory errors** – related to improper use of memory (memory leaks)

Quiz time!



Conclusion and Resources



Congratulations on completing the "How good are you at debugging code?" quiz! Debugging is an important skill for any developer, and mastering it can greatly improve your efficiency and effectiveness in writing clean, error-free code. Whether you faced challenges or breezed through the questions, remember that debugging is a continuous learning process.

This might help you…

- Python Quiz – by LitCode: Irma
- Flutter Development Quiz – by LitCode: Irma
- C++ Quiz – by LitCode: Irma
- Learn C++ programming – by LitCode: Irma
- Getting started with Python – by LitCode: Irma

I hope this quiz has been both challenging and educational. Keep practicing and stay curious.

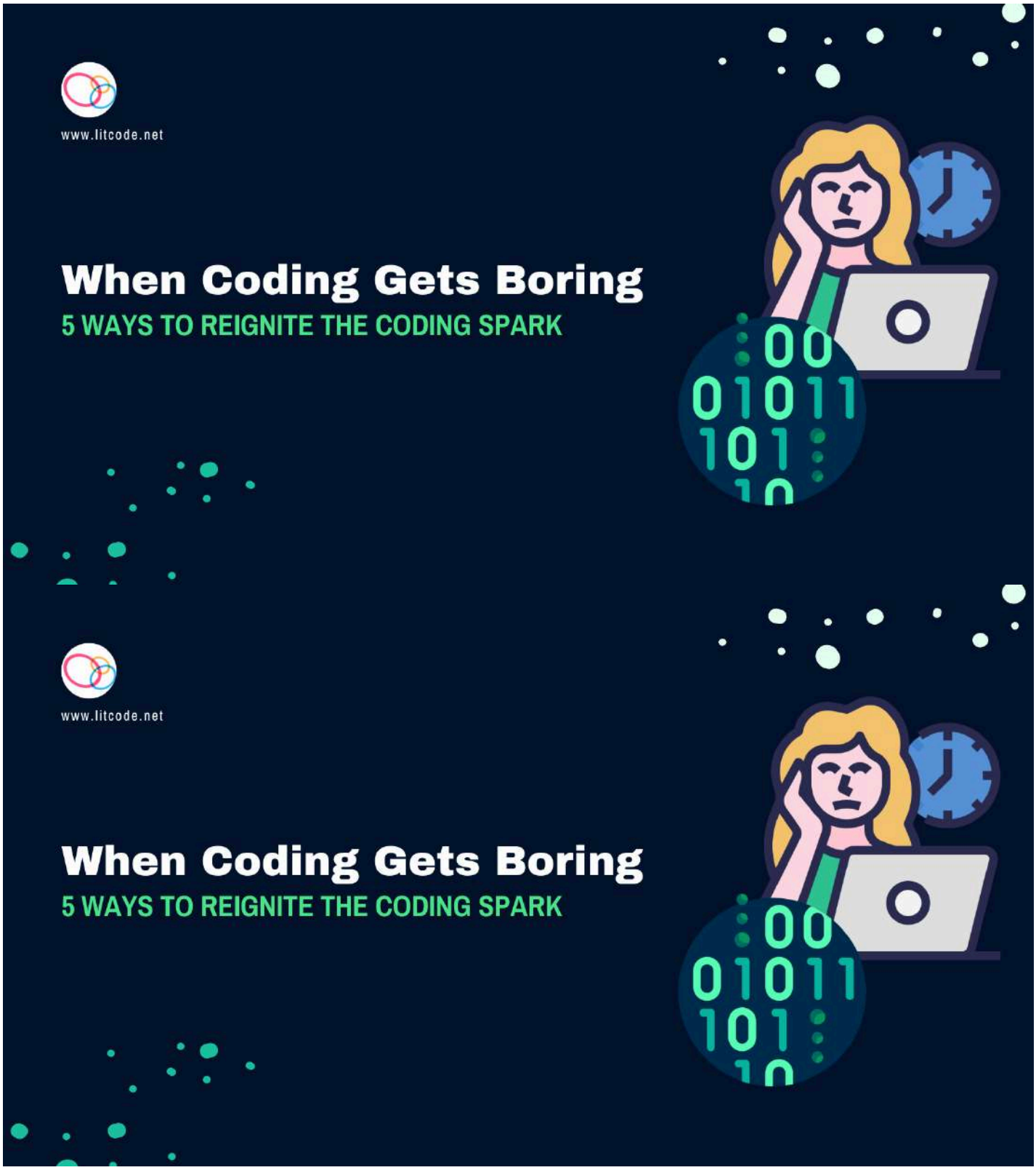
Happy coding! – Irma 😄

When Coding Gets Boring

November 5, 2024

Categories: Let's learn, Let's read

Tags: Coding, Goals, LitCode, ProgrammersLife



Even the experts hit a wall where coding starts to feel...well...**a bit boring**. The excitement might fade, and you might find yourself staring at lines of code feeling uninspired. If this sounds f[illegible]r,

don't worry! ***It is totally normal.*** What matters are the strategies to make your coding journey exciting again.

Here are some that will for sure reignite that spark and keep you moving forward!

> **A ship in a port is safe, but that is not what ships are built for.**
>
> Grace Hopper (Pioneer in Computer Science)

1. Explore a New Programming Language or Framework

If you've been coding in the same language for a while, it might be the time to expand your toolkit. Each language brings its own, unique perspective, which can be refreshing. Here are a few ideas:

- Python for data science or machine learning.
- Swift if you are curious about iOS development.
- Flutter (Dart) for building cross-platform apps.
- Flask or Django for Web Development.

Each language requires a special approach which might reignite that spark!

2. Take on a Creative Project

Sometimes we get bored from coding the same types of projects over and over again. Breaking out of that routine with a creative side project can make a huge difference. Here are a few project ideas:

- Build a Vintage Game – recreate a simple arcade game like Snake or Tetris and test your game development skills.
- Start a Tech Blog (like LitCode!) – share your experiences, challenges and accomplishments.
- Create a Coding Art piece – you can use code to create beautiful visuals and even animated art.

Explore and dive into your creative freedom which allows you to experiment and play around and turn coding into a form of expression.

3. Participate in Coding Competitions and Hackathons

If you are looking for something intense, coding challenges and competitions can give you that adrenaline boost. Here are some popular platforms to play with:

- LeetCode and HackerRank – algorithmic challenges.
- Advent of Code – holiday-themed challenges every December.
- Codewars – to help you learn new syntax and techniques.
- Hackathons – for real time, project-based coding competitions.

These competitions and events can help you improve your problem-solving skills.

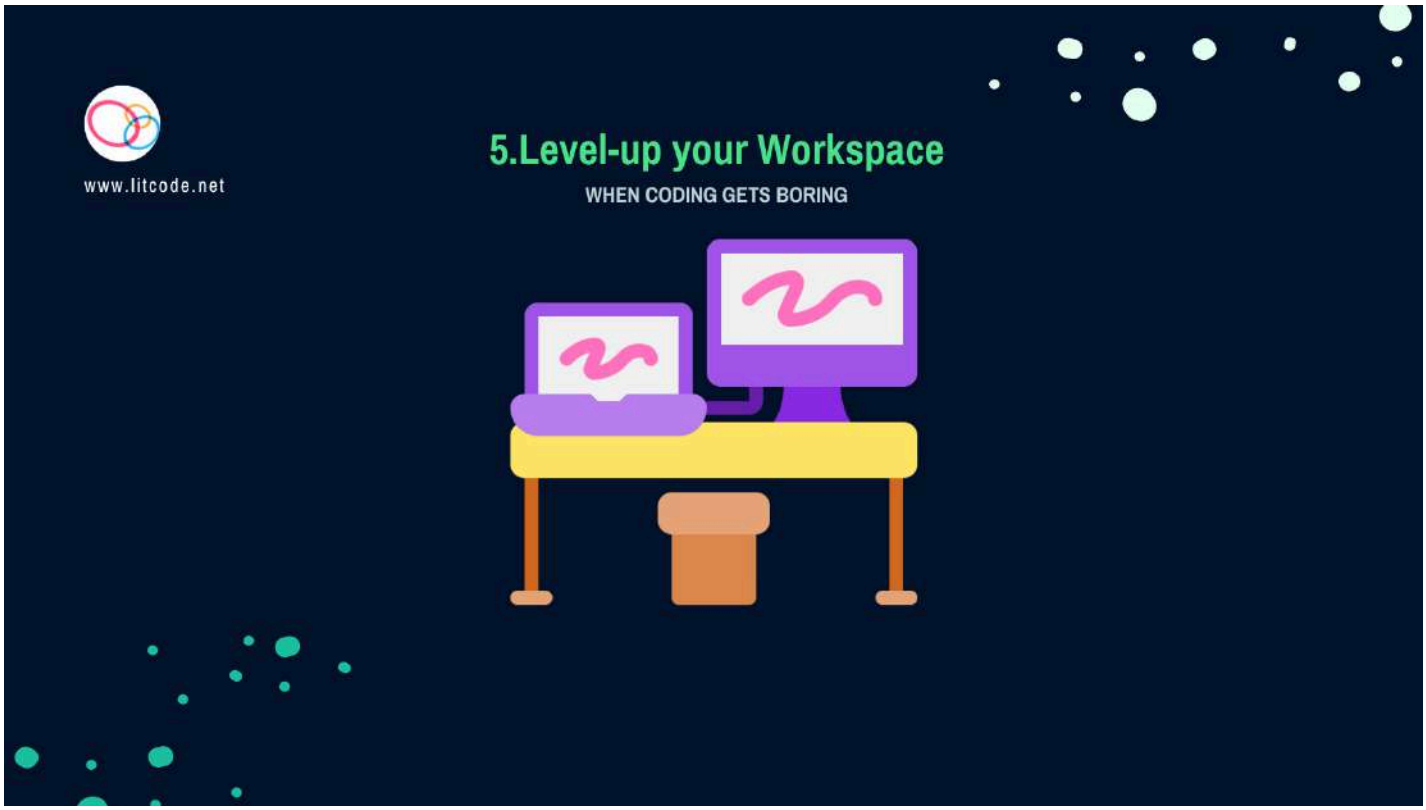
4. Learn New Skills outside Coding

It might seem weird but – stepping away from coding can actually make you a better coder. Huh? If you are feeling stuck, building skills outside of programming can help tremendously. Here are some ideas:

- Design skills (UI/UX) – improve your project's look and feel.
- Public speaking – for presenting that new ground-breaking idea freely.
- Project management – to make your workflow smoother.
- Expanding your book collection – not just coding but something to get your mind cooking!

These skills can add a new layer of creativity and freshness to your projects, bringing new ideas to your life in unique ways.

5. Level-up your Workspace



Sometimes the problem is not code – it's the environment. Refreshing your workspace/office space can have a huge impact on your motivation. You can:

- Optimize your coding setup – try a new editor or experiment with different themes.
- Declutter – your desk, add some plants, new lighting, posters or quotes.
- Try Pomodoro – as a new time management technique to keep coding productive.
- New tech – maybe a new keyboard, monitor or desk can bring a new feel to your coding sessions.

An organized environment can make a significant difference, especially if you are in it for hours at a time!

Remember why you started!

Reflecting on your original motivation for coding can be a powerful reminder of your journey. Maybe you started coding to societal problems, to create something new, or to break into a tech career. **When coding gets boring**, revisiting these motivations can be a great way to refocus and reconnect with your goals.

- Review your early projects and look how far you got!
- Write down your goals and create a vision board to remind yourself of the bigger pictu

- Talk to a newbie, their enthusiasm can remind you of the excitement coding brings.

Conclusion

Remember that it's okay to feel bored with coding sometimes; it happens to everyone. Coding isn't just a skill—it's a journey, full of highs and lows, breakthroughs and setbacks. Embrace the process, and keep pushing yourself to try new things. In the meantime – **Just keep coding!** 🚀

Good luck on your journey!

Keep creating! – Firma 😄

Check out:

Flutter Development Series – by Irma P.

LitCode Quizzes – by Irma P.

5 Passion Projects to try in 2024 – by Irma P.