{POWER.CODERS}

Programming 101

Agenda

Today we will learn about

- What Programming is
- How to think like a programmer
- How to solve problems

What is Programming?

Learning to program means learning how to solve problems using code.

What is Programming?

Learning to program means learning how to solve problems using code.

It consists of 2 parts:

What is Programming?

Learning to program means learning how to solve problems using code.

It consists of 2 parts:

- problem-solving

What is Programming?

Learning to program means learning how to solve problems using code.

It consists of 2 parts:

- problem-solving
- learning programming syntax

PROGRAMMING SYNTAX

PROGRAMMING SYNTAX

Learning a programming language and its **syntax** and semantics gives you the skills to read and **understand code** as well as to write your **own programs**.

Programming Syntax

Learning a programming language and its **syntax** and semantics gives you the skills to read and **understand code** as well as to write your **own programs**.

It is an analytical activity.

Building blocks of Programming

- Data Types
- Variables
- Conditions
- Loops
- Functions

Building blocks of Programming

- Data Types
- Variables
- Conditions
- Loops
- Functions

Each programming language has these building blocks. Only a different syntax.
We will use JavaScript to learn them.

What is JavaScript?

JavaScript is a **high-level** programming language. Nowadays you can (nearly) do anything with Javascript.

What is JavaScript?

JavaScript is a **high-level** programming language. Nowadays you can (nearly) do anything with Javascript.

JavaScript is not Java.

What is JavaScript?

JavaScript is a **high-level** programming language. Nowadays you can (nearly) do anything with Javascript.

JavaScript is not Java.

Java is to **Java**Script as **cat** is to **cat**fish.

Why do we learn JavaScript?

Is JavaScript a good choice as first programming language?

Why do we learn JavaScript?

Is JavaScript a good choice as first programming language?

- It has a **low barrier to entry**

Why do we learn JavaScript?

Is JavaScript a good choice as first programming language?

- It has a **low barrier to entry**
- It gives you **instant feedback**

Why do we learn JavaScript?

Is JavaScript a good choice as first programming language?

- It has a **low barrier to entry**
- It gives you **instant feedback**

JavaScript has also pitfalls and problems - as most programming languages do.

PROBLEM SOLVING

PROBLEM SOLVING

Solving a problem in programming can be defined as writing an original program that performs a particular **set of tasks** and meets all stated **constraints**.

PROBLEM SOLVING

Solving a problem in programming can be defined as writing an original program that performs a particular **set of tasks** and meets all stated **constraints**.

It is an creative activity.

That's what makes it hard.

WHAT IS A CONSTRAINT?

Think about creating a website:

What is a constraint?

Think about creating a website:

- You need to use particular languages: **HTML** and **CSS**

WHAT IS A CONSTRAINT?

Think about creating a website:

- You need to use particular languages: **HTML** and **CSS**
- The website needs to be displayed on different devices: e.g. **Laptop**, **Tablet** or **Mobile phone**

WHAT IS A CONSTRAINT?

Think about creating a website:

- You need to use particular languages: **HTML** and **CSS**
- The website needs to be displayed on different devices: e.g. **Laptop**, **Tablet** or **Mobile phone**
- Its content needs to be searchable and readable by humans and machines: **SEO** and **Accessibility**

What is a constraint?

Think about creating a website:

- You need to use particular languages: **HTML** and **CSS**
- The website needs to be displayed on different devices: e.g. **Laptop**, **Tablet** or **Mobile phone**
- Its content needs to be searchable and readable by humans and machines: **SEO** and **Accessibility**

Constraints are the **limitiations** in how your program can perform the specific set of tasks.

How to think like a programmer?

"The biggest mistake I see new programmers make is focusing on learning syntax instead of learning how to solve problems.

—V. Anton Spraul

5 STEPS TO THINK LIKE A PROGRAMMER

1. Understand
2. Plan
3. Divide & conquer
4. Don't get frustrated
5. Practice

1. UNDERSTAND

1. UNDERSTAND

Read the problem several times until you can explain it to someone else in plain English.

1. Understand

Read the problem several times until you can explain it to someone else in plain English.

> If you can't explain something in simple terms, you don't understand it.
>
> —**Richard Feynman**

1. UNDERSTAND

Read the problem several times until you can explain it to someone else in plain English.

> ”If you can’t explain something in simple terms, you don’t understand it.
>
> —Richard Feynman

Rubber duck debugging: Insights are often found by simply describing the problem aloud.

2. Plan

2. Plan

Take time to analyze the problem and process the information.
Think **very precisely** about how you solve the problem.

2. Plan

Take time to analyze the problem and process the information. Think **very precisely** about how you solve the problem.

Ask yourself: "Given input X, what are the steps necessary to return output Y?"

3. Divide & conquer

3. Divide & conquer

Do not try to solve **one big problem**. Break it down into **steps**. Steps that are so **simple** that a computer can execute them.

3. Divide & conquer

Do not try to solve **one big problem**. Break it down into **steps**. Steps that are so **simple** that a computer can execute them.

Once you solved every step (sub-problem), connect the dots and you will find the **solution to the original problem**. Congratulations!

ANOTHER DEFINITION OF PROGRAMMING

ANOTHER DEFINITION OF PROGRAMMING

Step 3 is so important that you can use it to define programming.

ANOTHER DEFINITION OF PROGRAMMING

Step 3 is so important that you can use it to define programming.

- Programming is the **art** of breaking a problem down into smaller problems.

ANOTHER DEFINITION OF PROGRAMMING

Step 3 is so important that you can use it to define programming.

- Programming is the **art** of breaking a problem down into smaller problems.
- Finding the solutions to these smaller problems.

Another Definition of Programming

Step 3 is so important that you can use it to define programming.

- Programming is the **art** of breaking a problem down into smaller problems.
- Finding the solutions to these smaller problems.
- Putting these pieces back together.

4. Stuck?

4. Stuck?

Don't get frustrated!

4. Stuck?

Don't get frustrated!

- **Debug:** Go step by step through your solution trying to find where you went wrong.

4. Stuck?

Don't get frustrated!

- **Debug:** Go step by step through your solution trying to find where you went wrong.
- **Reasses:** Sometimes we get so lost in the details of a problem that we overlook general principles. Take a step back. Look at the problem from another perspective.

4. Stuck?

Don't get frustrated!

- **Debug:** Go step by step through your solution trying to find where you went wrong.
- **Reasses:** Sometimes we get so lost in the details of a problem that we overlook general principles. Take a step back. Look at the problem from another perspective.
- **Research:** Google is your friend. No matter what problem you have, someone has probably solved it. Find that person/ solution. In fact, do this even if you solved the problem! (You can learn a lot from other people’s solutions).

HOW TO GOOGLE SOLUTIONS

How to google solutions

Don't look for a solution to the big problem, only look for solutions to sub-problems.

How to google solutions

Don't look for a solution to the big problem, only look for solutions to sub-problems.

You will **learn and understand more**. You will need **less research and Google** next time.

HOW TO GOOGLE SOLUTIONS

Don't look for a solution to the big problem, only look for solutions to sub-problems.

You will **learn and understand more**. You will need **less research and Google** next time.

Google is not about Copy & Paste, it is about learning.

BEST PRACTICE

on how to google

- Search in English
- Use more than one word
- Be precise
- Give context
- Use operators
- Check dates and up-to-dateness
- Use more than one source
- Search also images, scholar or books

5. PRACTICE

Practice makes perfect.

5. Practice

Practice makes perfect.

The more problems you solve, the more research you do on other programmers solving the same problem, the more you think like a programmer.

”Demonstrating computational thinking or the ability to break down large, complex problems is just as valuable (if not more so) than the baseline technical skills required for a job.

—Hacker Rank (2018 Developer Skills Report)

How to solve a problem

How to cross the river?

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. The goose cannot be left alone with the corn, or the goose will eat the corn.

How to cross the river?

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. The goose cannot be left alone with the corn, or the goose will eat the corn.

How does the farmer get everything across the river?

Pair Exercise

Find a partner and talk through the puzzle.
You have 15 minutes to try solve the puzzle.
Describe the steps needed.

FIND THE CONSTRAINTS

Find the Constraints

- The farmer can take only one item at a time in the boat

FIND THE CONSTRAINTS

- The farmer can take only one item at a time in the boat
- The fox and goose cannot be left alone on the shore

Find the constraints

- The farmer can take only one item at a time in the boat
- The fox and goose cannot be left alone on the shore
- The goose and corn cannot be left alone on the shore

List the possible actions (operations)

List the possible actions (operations)

- Carry the fox to the far side of the river

List the possible actions (operations)

- Carry the fox to the far side of the river
- Carry the goose to the far side of the river

LIST THE POSSIBLE ACTIONS (OPERATIONS)

- Carry the fox to the far side of the river
- Carry the goose to the far side of the river
- Carry the sack of corn to the far side of the river

By enumerating all the possible operations, we can solve many problems by testing every combination of operations until we find one that works.

By enumerating all the possible operations, we can solve many problems by testing every combination of operations until we find one that works.

The more specific your operations are, the easier it is to miss possible operations.

Generalize the operations as much as possible

GENERALIZE THE OPERATIONS AS MUCH AS POSSIBLE

- Row the boat from one side to the other

GENERALIZE THE OPERATIONS AS MUCH AS POSSIBLE

- Row the boat from one side to the other
- If the boat is empty, load an item from the shore

GENERALIZE THE OPERATIONS AS MUCH AS POSSIBLE

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

GENERALIZE THE OPERATIONS AS MUCH AS POSSIBLE

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

This is also called **abstraction**.

TEST EVERY COMBINATION

TEST EVERY COMBINATION

Now we generate all possible sequences of moves, ending each sequence once it violates one of our contraints or reaches a configuration we've seen before.

TEST EVERY COMBINATION

Now we generate all possible sequences of moves, ending each sequence once it violates one of our contraints or reaches a configuration we've seen before.

Eventually we hit upon the only possible sequence, the solution of our problem.

SOLUTION

1. Transport the goose to the other side
2. Row back alone
3. Transport the fox to the other side
4. Take goose back to original side
5. Transport the sack of corn to the other side (leave goose behind)
6. Row back alone
7. Transport the goose to the other side

How to solve a programming problem?

6 STEPS TO PROGRAM

1. Understand the problem
2. Solve the problem manually
3. Make your manual solution better
4. Write pseudocode
5. Replace pseudocode with real code
6. Simplify and optimize your code

1. Understand the Problem

We had that before.

- Read the problem at least 3 times.
- Ask yourself always following questions: why? what? how?
- Explain the problem in your own words to someone else, remember the rubber duck.

1. Understand the Problem

We had that before.

- Read the problem at least 3 times.
- Ask yourself always following questions: why? what? how?
- Explain the problem in your own words to someone else, remember the rubber duck.

You can use mind mapping tools or whiteboards to visualize your problem, e.g. a notepad or Miro.

2. Solve the problem manually

2. Solve the problem manually

"Nothing can be automated that cannot be done manually!

2. SOLVE THE PROBLEM MANUALLY

”Nothing can be automated that cannot be done manually!

Test your manual process with at least 3 different inputs. Think about what you did to find the solution.

3. Make your manual solution better

3. Make your manual solution better

Simplify and optimize your steps. Look for patterns and see if there's anything you can generalize.

3. Make your manual solution better

Simplify and optimize your steps. Look for patterns and see if there's anything you can generalize.

Can you reduce any steps? Are you repeating steps?

3. Make your manual solution better

Simplify and optimize your steps. Look for patterns and see if there's anything you can generalize.

Can you reduce any steps? Are you repeating steps?

Try for brevity. The lines that you don’t write are the lines where you can be sure that they don’t have bugs.

4. WRITE PSEUDOCODE

4. WRITE PSEUDOCODE

Writing pseudocode line by line, before starting to code, helps you defining the structure of your code.

4. WRITE PSEUDOCODE

Writing pseudocode line by line, before starting to code, helps you defining the structure of your code.

You can do that on a sheet of paper or as comments in your code editor.

4. WRITE PSEUDOCODE

Writing pseudocode line by line, before starting to code, helps you defining the structure of your code.

You can do that on a sheet of paper or as comments in your code editor.

Focus on logic and steps.

5. Replace pseudocode with real code

5. Replace pseudocode with real code

After your pseudocode is ready, translate each line into real code.

5. REPLACE PSEUDOCODE WITH REAL CODE

After your pseudocode is ready, translate each line into real code.

Test each step you translate as early and as thoroughly as possible. Finding a problem in a small and easy piece of code is much simpler than trying to spot it in a large program.

6. Simplify and optimize your code

6. Simplify and optimize your code

This is also called **code refactoring** and we will hear more about that in the coming weeks.

6. Simplify and optimize your code

This is also called **code refactoring** and we will hear more about that in the coming weeks.

”Programs must be written for people to read, and only incidentally for machines to execute.

—Gerald Jay Sussman and Hal Abelson

PRACTICE, PRACTICE, PRACTICE

PRACTICE, PRACTICE, PRACTICE

Again, practice makes perfect.

PRACTICE, PRACTICE, PRACTICE

Again, practice makes perfect.

- Debug every step
- Get feedback through code reviews
- Write useful comments

Books

- Think like a Programmer: An introduction to creative problem solving
- The pragmatic programmer

EXERCISES

Solve the following problems. If you are stuck, ask one of your peers.

- Understand: Visualize the why?how?what?
- Plan: What are the constraints, the inputs, the steps?
- Divide: Break bigger problems into smaller ones.
- Conquer: Write pseudocode.

We will go through them together on THU

Exercise 1

Allow the user to input digits, afterwards sort and print them in numerical order.

Exercise 2

Find the number of the year for the given date. For example, january 1st would be 1, and december 31st is 365.

Exercise 3

Create a random password generator. The password has to have a length of at least 8 characters, at least 1 digit and 1 special char (_ or -).

EXERCISE 4

Validate a telephone number, as if written on an input form. Telephone numbers can be written as ten digits, or with dashes, spaces, or dots between the three segments, or with the area code parenthesized; both the area code and any white space between segments are optional.

STILL HAVE TIME AND ENERGY?

Finish open exercises in HTML and CSS. Work on your website project.