EXERCISES

INTRO TO COMUPTER SCIENCE

SUMMER COURSE

LINNAEUS
UNIVERSITY
SUMMER 2024

WEEK 1 ASSIGNMENTS

ASSIGNMENT 1 - MARKDOWN DOCUMENT

- Write a Markdown document. This document will include questions for self-reflection, as well as some questions from the topics covered in this course. It will also include specific formatting requirements.
- Just 1-2 sentences per question is enough–the goal is to get you thinking, not writing essays! 😊 No need to feel pressured to write more than that.

MARKDOWN DOCUMENT - FORMATTING REQUIREMENTS

- Headings
  - At least two levels of headings (e.g., # for main headings and ## for subheadings) to organize the content.
- Code Blocks
  - Demonstrate how to create a code block to display code snippets. You can write anything you want in this code block.
- Lists
  - Include an ordered list with at least 2 items.
  - Include an unordered list with at least 2 items.
- Links
  - Create a hyperlink to a website of your choice.
- Emphasis and Bold
  - Use italics and bold formatting to highlight specific text within the document.
- Emoji
  - Add an emoji of your choice! 🚀
- Table
  - Create a simple table. 2x2 row and column is enough, but feel free to add more rows/columns if you want! You can write anything you want in the table.

MARKDOWN DOCUMENT - QUESTIONS

- What is your previous experience with computers, computer science and code? For example, have you never really had an interest before? Are you a gamer who has modded games? Have you tried coding a little?
- What made you interested in coding and decide to go to school to learn programming?
- What technology did you grow up with? Did you experience the world before Google and smartphones? If not, how do you think finishing it might have been different from navigating daily life without instant access to services, information, and communication?
- How does technology impact you positively, as well as negatively?
- Have you ever thought about how your personal data is used by companies? How do you feel about the way big data shapes your online experiences?
- Have you tried using ChatGPT or other generative AI tools in your work or hobbies? How did they help or change the way you approach tasks?
- Do you see code as logic or communication? What might be the benefit of each perspective?
- What mental skills listed in “Programming Mentality” do you find challenging? Write one actionable way to improve in that area!
- What did you expect from this course?
- Did you learn what you thought you would during this course? If not, what would you have wanted to be different?
- What did you enjoy most about this course, and what would you have liked to see more of?
- Check Your Disk Type - Use your computer’s disk management tool (e.g., Disk Management on Windows or Disk Utility on macOS) to check your disk type (HDD or SSD) and write it in the document (tip is to use the table for this!)
- Look up how to find your computer’s specifications. What operating system is it running on? What type of processor do you have?
- **Optional:** Write the non-sensitive computer specifications of your machine in your Markdown document. Processor, RAM, OS system version, and System Type are safe to share - they cannot be used to identify your machine or derive sensitive information that could be used maliciously. Write them out on a table

PLEASE DO NOT GIVE OUT ANY SENSITIVE INFORMATION. Here is a breakdown of what could be considered sensitive when sharing device specifications. Always be cautious with sharing details that could uniquely identify or compromise your device’s security.

- **SOFTWARE- Safe to Share**: OS Version, Build Number, System Type (32-bit/64-bit) **Avoid Sharing:** Device Name, Product ID, Product Key.

- **HARDWARE - Safe to Share**: Device Name, Processor, Installed RAM, System Type, Pen and Touch. **Avoid Sharing:** Device ID, Product ID

ASSIGNMENT 2 - BREAK THE ICE

- Ask a question on Slack related to IT / programming that you might be struggling with.
- Ask ChatGPT a question related to IT / programming that you might be struggling with.

ASSIGNMENT 3 - HARDWARE / SOFTWARE

- Task Manager/Activity Monitor- Open Task Manager (Windows) or Activity Monitor (macOS). Locate the CPU usage. Observe which processes are consuming the most CPU resources. Also, check the current RAM usage. Note which applications are using the most memory.

WEEK 2
ASSIGNMENTS

ASSIGNMENT 1 - FOLLOWING TUTORIALS

- **Follow each of the tutorials** “Terminal” and “File Systems” (available on the course site under Week 2).

ASSIGNMENT 2 - DOWNLOAD NODE.JS AND NPM

- **Install Node.js and npm from the official node.js website.**
  - npm should be included in the node.js installation. You’ll see two versions of node.js available: LTS (Long-Term Support) and the current version. Choose the current version - look for: “Want new features sooner?” on the site, and you should find a version that looks something like this: “Node.js v22.6.01 “. That’s the one you want.
  - (For most users, the LTS version is recommended for code that will be used in live applications. However, during school they will recommend the current version so you can have access to new features).

- **Verify the installation:**
  - Open Git Bash and run:
  - ```**node -v**``` to check the Node.js version.
  - ```**npm -v**``` to check the npm version.

ASSIGNMENT 3 - NAVIGATION AND FILE OPERATIONS

- **Open Git Bash and navigate to the desktop folder using Git Bash and text commands.**
- **Create a directory.** Use Git Bash to create a directory - google /ask Chat GPT how to find the command to create a directory using Git Bash.
- **Check that the directory is created.** Google/ask Chat GPT how to find the command to view the contents of the current working directory.
- **Create a file using GUI.** Using the graphical user interface (using your mouse/keyboard and double-clicking) navigate into your newly created directory and create a .txt file. Add some text to the file and save the .txt file.
- **View .txt file contents using Git Bash.**
  - Use Git Bash to navigate into the directory you just created.
  - Check that the .txt file you created is present - use the text command that lets you view the contents of the current working directory to verify that the .txt file is there.

- **Find the absolute file path of your .txt file.** Either google/ask Chat GPT how to do it in the terminal, or use your OS's GUI to look up the file path. The path should be from the root of your filesystem.
  - On Windows, the root directory is represented by the drive letter followed by a backslash. For example, C:\ is the root directory of the C: drive.
  - On macOS and other Unix-like systems, the root directory is represented by a single forward slash /.

- **View the contents of your text file.** Use the command **```cat yourfilename.txt```** in Git Bash.
- **Delete the file** using Git Bash. Use the command **```rm yourfilename.txt```**
- **View the contents of the current working directory again** (the .txt file should be gone).
- **Navigate out.** Move out of the directory in the terminal, to your desktop folder.
- **Delete directory** - remove the directory you created earlier using the command **```rmdir yourdirectoryname```**
- **Use the text command in Git Bash to view the contents of the desktop folder.** The directory you created should be gone.

WEEK 3 ASSIGNMENTS

ASSIGNMENT 1 - HELLO, WORLD!

- Go to Replit, log in, and create a new Repl. Select the programming language you want to use (e.g., JavaScript).
- Use console.log() to write your first "Hello, World!"
- Click the "Run" button to execute your program.
- View the output

ASSIGNMENT 2 - VARIABLES, DATATYPES AND IF-STATEMENTS

1 - Declaring and assigning values

- Go to Replit, log in, and create a new Repl. Select the programming language you want to use (e.g., JavaScript).

- Declare a Variable in the code editor - create a variable in your code using a keyword such as let, const, or var.

- On a separate line of code, assign a value to the variable.

- Print or display the value of the variable using console.log(). For example:

    ```let age = 18
    console.log(age)```

- Reassign the same variable, giving it a new value.

- Use console.log() to print the updated value of the variable, then run the code to print the value again.
    ```

2 - Datatypes and typeof

- Initialize 3 new variables - 1 boolean (true/false), 1 string, 1 number.
  - String is a collection of characters - essentially, any text.
  - **“Hello”** is a string, as is **“ “** and **“Hi, there, coders!!! :)”**
  - If you want to combine a string with a variable in the console log(), this is how:

```javascript
let isRaining = true;
let city = "New York";
let temperature = 75;

console.log("Is it raining? " + isRaining);
// Output: Is it raining? true

console.log("The weather in " + city +
            " is currently " + temperature + " degrees.");
// Output: The weather in New York is currently 75 degrees.

console.log("It's " + temperature + " degrees in " + city +
            ". Is it raining? " + isRaining);
// Output: It's 75 degrees in New York. Is it raining? true
```

- Use **```console.log(variableName)```** to print the value of each of the variables.

- Use the command **```typeof variableName```** on each of the variables inside a console.log() - between the brackets. This will output their data type.

```javascript
let isRaining = true;
let city = "New York";
let temperature = 75;

console.log("Type of isRaining: " + typeof isRaining);
// Output: Type of isRaining: boolean

console.log("Type of city: " + typeof city);
// Output: Type of city: string

console.log("Type of temperature: " + typeof temperature);
// Output: Type of temperature: number
```

3 - If statements

- Write an if statement for each variable.

NOTE!
Using **one = assigns a value** to a variable. Using **two == checks it two different values are equal**.

```
let favoriteColor = "blue";

if (favoriteColor == "blue") {
    console.log("Your favorite color is blue!");
}
```

For easy if-statements, use:

- **Equality Operator ==** (compares two values to determine if they are equal)
- **Relational Operators ><** (compares two values)
  - Greater Than (>): Checks if the value on the left is greater than the value on the right.
  - Less Than (<): Checks if the value on the left is less than the value on the right.

javascript

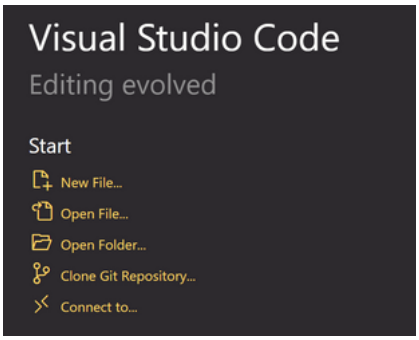
```
let isRaining = true;
let city = "New York";
let temperature = 75;

// Check if it is raining
if (isRaining == true) {
    console.log("It is raining.");
} else {
    console.log("It is not raining.");
}

// Check if the city is "New York"
if (city == "New York") {
    console.log("The city is New York.");
} else {
    console.log("The city is not New York.");
}

// Check if the temperature is greater than 70
if (temperature > 70) {
    console.log("The temperature is above 70 degrees.");
} else {
    console.log("The temperature is 70 degrees or below.");
}

// Check if the temperature is less than 80
if (temperature < 80) {
    console.log("The temperature is below 80 degrees.");
} else {
    console.log("The temperature is 80 degrees or above.");
}
```

ASSIGNMENT 3 - VSCODE & GIT

- Download VSCode.
- Download ANY extension in VSCode. One recommendation is Dracula - it's a theme that changes the colors of the UI. Feel free to choose any theme or extension you like! If you're unsure which one to start with, a theme is a great option.

- Create a directory on your computer. Name it "Local_To_GitHub".

- Create a new project in VSCode. Open VSCode, and open the "Local_To_GitHub" folder when prompted **Open folder...** This will be the root folder of your project.



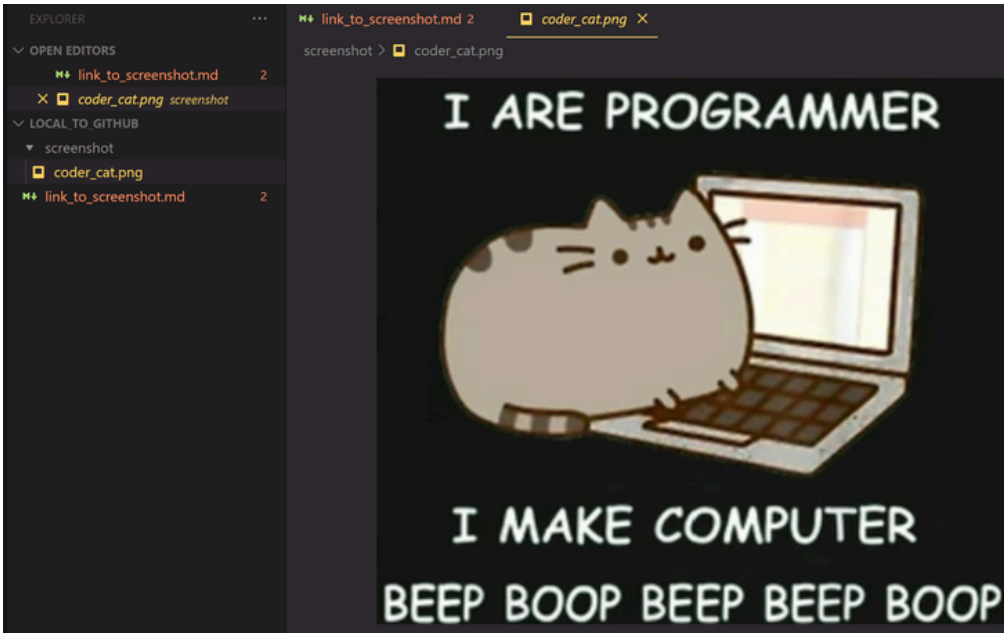
- Create a new folder in your VSCode project "Local_To_GitHub". (Make sure to create it *inside* the "Local_To_GitHub" folder).

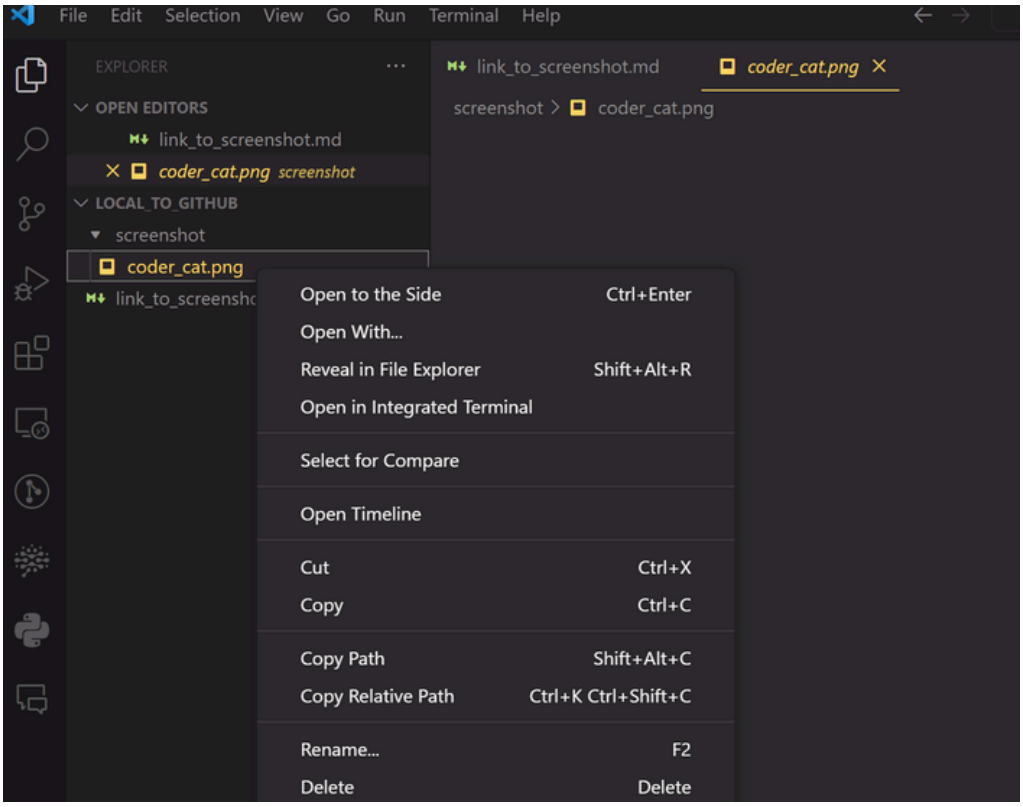
- Create a .md file in the folder "Local_To_GitHub".

- Create a link to your another file in your project(what the file is doesn't matter) and write a link to it *in the Markdown file* you just create in the previous step.

- Practice how to link to files using the relative path!

It should look something like this:





WEEK 4
ASSIGNMENTS

ASSIGNMENT 1 - MINI CALCULATOR

1 - Basic math function

- Go to Replit, log in, and create a new Repl. Select the programming language you want to use (e.g., JavaScript).

- Write a Function:
- Create a function that performs a basic mathematical operation, such as adding two numbers together.

- Make Two Function Calls:
- Call the function twice with different sets of numbers and print the results to the console.
-
- Save and Print the Result:
- Create a variable, and store the result of another function call in the variable.

- Print this variable to the console.

2 - Function with two parameters

- Write a Function with Two Parameters:
- Create a function that performs a mathematical operation using two parameters and division.

- Call the Function with Different Values:
- Call this function once with two different sets of values (such as 20 and 5) as arguments and print the results.

- Swap the values around in another function call to observe the effect and print the result. (This is to notice that switching the parameters affects the result of the operation, as division is order-sensitive!)

3 - Calculate the area of a rectangle

- Write a Function to Calculate Area:
- Create a function that calculates the area of a rectangle given its width and height.

- Print the Result:
- Call the function with specific width and height values and print the resulting area to the console.

BONUS & OPTIONAL (BUT GOOD PRACTICE)

4 - Check if a number is positive or negative

- Define the Function:
- Create a function that accepts a single number as its parameter.

- Check the Number:
- Use an if statement to determine if the number is greater than 0. If it is, the function should return the result "Positive".

- Use an else if statement (google!) to check if the number is less than 0. If it is, the function should return the result "Negative".

- If the number is neither greater than nor less than 0, it must be equal to 0. In this case, return the result "Zero".

- Return the Result:

- Ensure the function returns a string indicating whether the number is positive, negative, or zero based on the checks performed.

HINT

```
function checkNumber(num) {
  if (num > 0) {
    return "Positive";
  } else if (num < 0) {
    return "Negative";
  } else {
    return "Zero";
  }
}
```

5 - Compare two numbers and return the larger one

- Define the Function:
- Create a function that accepts two numbers as parameters.

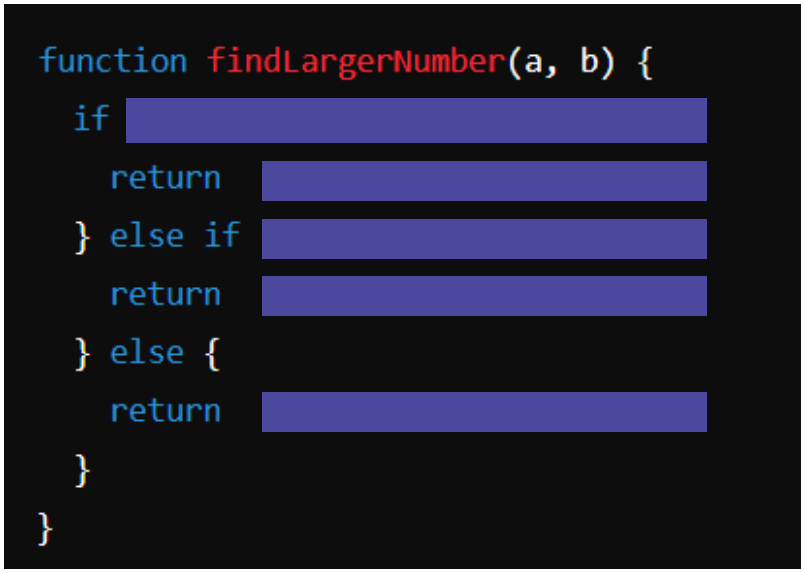
- Compare the Numbers:
- Use an if statement to compare the two numbers. If the first number is greater than the second, the function should return the first number.

- Use an else if statement to check if the second number is greater than the first. If it is, return the second number.

- If neither number is greater (meaning the two numbers are equal), return the result "Both numbers are equal".

- Return the Result:
- Ensure the function returns the larger number or a message indicating that both numbers are equal, based on the comparisons.

HINT



ASSIGNMENT 2 - PINGING AN IP ADDRESS

- **In this assignment, you will learn how to find the IP address of a website (Google in this case) using the command line.**

- **Open GitBash**
- **Find the IP Address of Google:**
  - In the Git Bash terminal, enter the following command

    ```ping google.com```

- **After executing the command, you will see output similar to this:**

  - **```$ ping google.com PING google.com (142.250.190.14): 56 data bytes 64 bytes from 142.250.190.14: icmp_seq=0 ttl=115 time=15.3 ms 64 bytes from 142.250.190.14: icmp_seq=1 ttl=115 time=15.2 ms 64 bytes from 142.250.190.14: icmp_seq=2 ttl=115 time=14.9 ms 64 bytes from 142.250.190.14: icmp_seq=3 ttl=115 time=15.1 ms 64 bytes from 142.250.190.14: icmp_seq=4 ttl=115 time=15.0 ms --- google.com ping statistics --- 5 packets transmitted, 5 packets received, 0% packet loss round-trip min/avg/max/stddev = 14.9/15.1/15.3/0.1 ms```**

The output starts with PING google.com (142.250.190.14), indicating that the **IP address of Google is 142.250.190.14.**