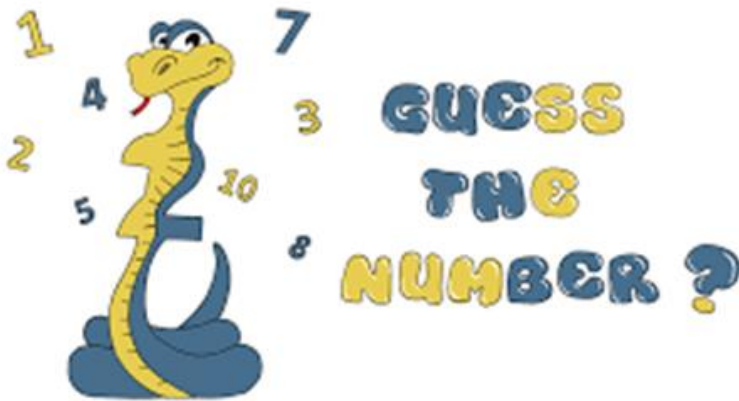


Practical Project: Guess A Number

This is additional practical project and it **is not mandatory and it is not included in the final score**. The main purpose is to use gained knowledge in different type of problems and to improve your portfolio and GitHub skills.



Today we will make the console game "**Guess A Number**". "**Guess A Number**" is a game in which your opponent, "**the computer**" chooses a **random** number between "**1 and 100**" and your task is to **guess** this number. After each number you enter, the computer will give you a **hint** of whether the number is **greater** or **less** than the number you selected until you guess the **correct** number:

```
Guess the number (0-100): 34
Too Low!
Guess the number (0-100): 78
Too High!
```

```
Guess the number (0-100): 78
Too Low!
Guess the number (0-100): 94
You guess it!
```

1. Create GitHub Repository

We already have a **GitHub** account created, so we're moving directly to creating a new **repository**.

Create a **new repository** from: <https://github.com/new>. Choose a **meaningful name**, e. g.

"**GuessANumberByUsername**" add a **short description** and make your repo **public**:

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Owner *

Repository name *



/ GuessANumber ✓

Great repository names are short and memorable. Need inspiration? How about [legendary-garbanzo?](#)

Description (optional)

This is a simple console game "Guess a Number"



Please choose **your original and unique name** for your project!

Your GitHub profile should be **unique**, not the same as your classmates.

You can follow this tutorial, but you can also **make changes** and **implement your project differ** from your classmates.

Also, **add a README.md** file and **.gitignore for Visual Studio**. In Git projects, the **.gitignore file** specifies which files from your repo are not part of the source code and should be ignored (not uploaded in the GitHub repo). Typically in GitHub, we upload in the repo **only the source code**, and we don't upload the compiled binaries and temp files.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☒ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: VisualStudio ▼

Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

License: None ▼

This will set  main as the default branch. Change the default name in your [settings](#).


 You are creating a public repository in your personal account.


Create repository

Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

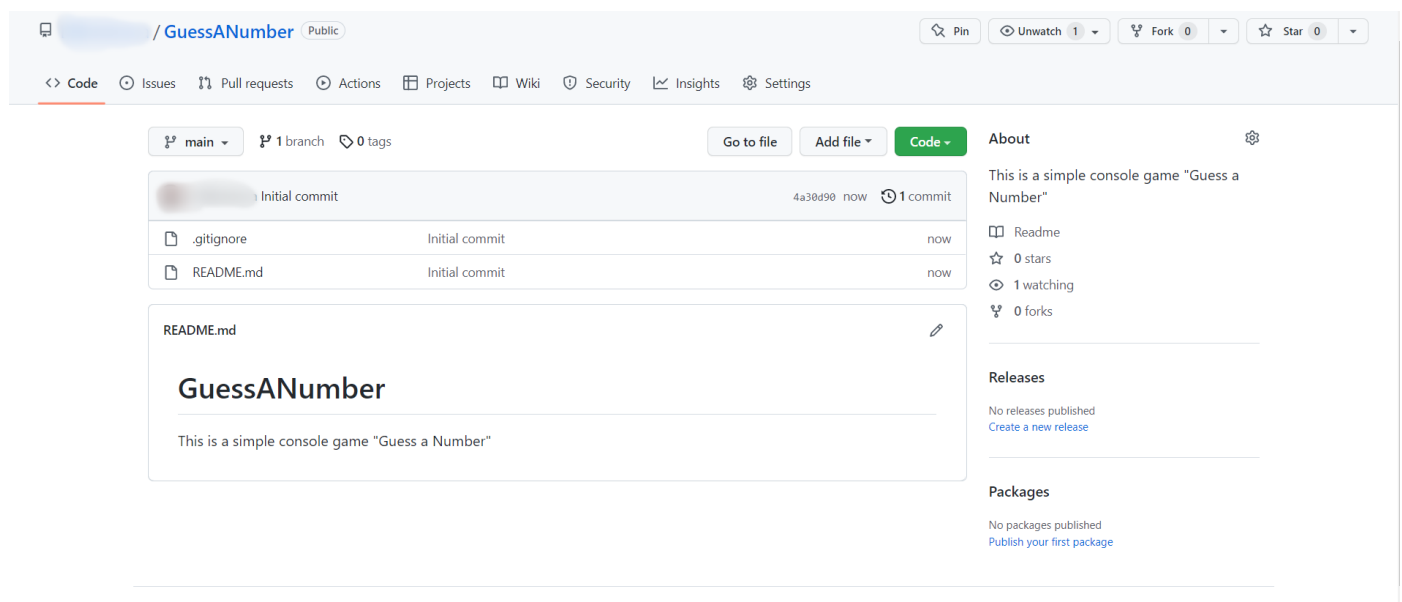
License: MIT License ▼

This will set  main as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

Create repository

Now your **repository is created** and looks like this:



The screenshot shows a GitHub repository named 'GuessANumber' (Public). The repository has 1 branch (main), 0 tags, 0 stars, 1 watching, and 0 forks. The README.md file is visible, containing the title 'GuessANumber' and the description 'This is a simple console game "Guess a Number"'. The repository was created by an initial commit 4a38d98 now.

Now let's see how to **write the code** of our game.

2. Write the Game's Code

Let's create the game and play with it.

Create a Visual Studio Code Project

First, we should **start Visual Studio Code** and **create a new JS file**. Then, **choose an appropriate name** and a **place to save the project**.

Implement the Game Logic

Now let's start working on our project.

Read Player's Move

First, we should create an interface where we can enter the number without stopping the program:

```
const readline = require('readline').createInterface({
  input: process.stdin,
  output: process.stdout
});
```

A little more information about how to create an interface: <https://nodejs.org/api/readline.html>

Create the already known **method random**, which will help us **choose a random number**. We will use this method so the computer can every time choose a number **between "1 and 100"**.

```
let computerGuess = Math.floor(Math.random() * 100);
```

You can learn a little more about it here:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random

Now create a function that stores the question about the number.

```
let recursiveAsyncReadLine = function () {
  readline.question('Guess the number (0-100): ', number =>
  {
  });
}
recursiveAsyncReadLine();
```

More about this **readline.question** you can learn: <https://nodejs.org/api/readline.html>

Now let's run the **app** in the console and check whether our current code **works** properly. Write in the terminal **node {your file's name}.js**:

```
C:\Users\Bobby\Desktop\Guess a Number>node guessANumber.js
Guess the number (0-100):
```

Check the Player's Input

Now check the player's input using **Number(...)**. If it's a number (**what we expect**), the methods will return a **number**, otherwise, **"NaN"**.

Now we have to create our **if-else** statements. First, we should check if the player's input data is valid:

```
if (guess <= 100 && guess >= 0) {
}
```

If data is valid, write a **nested if-else statement** in which we will check all **three** possible cases.

First, if the player's number is **equal** to the computer's number, that means the player **guessed** the computer's number, so you should **write** a message and **stop** the application by using **readline.close()**. Do it like this:

```
if (guess <= 100 && guess >= 0) {  
    if (guess == computerGuess) {  
        console.log('You guess it!');  
        return readline.close();  
    }  
}
```

The other **two** cases are if the player's number is **higher** than the computer's number and the player's number is **less** than the computer's number. Write the rest of the **else if statement** by yourself:

```
if (guess == computerGuess) {  
    console.log('You guess it!');  
    return readline.close();  
} else if (guess < computerGuess) {  
    console.log('Too Low!');  
    recursiveAsyncReadLine()  
} else if (guess > computerGuess) {  
    console.log('Too High!');  
    recursiveAsyncReadLine()  
}
```

To create a **repeating** question after each **entered number**, we have to call the **same function** after every single answer that is **not the guess number**.

```
if (guess == computerGuess) {  
    console.log('You guess it!');  
    return readline.close();  
} else if (guess < computerGuess) {  
    console.log('Too Low!');  
    recursiveAsyncReadLine()  
} else if (guess > computerGuess) {  
    console.log('Too High!');  
    recursiveAsyncReadLine()  
}
```

Recursion is a process of calling itself. A function that calls itself is called a **recursive function**. A recursive function must have a condition to stop calling itself. Otherwise, the function is called indefinitely.

Once the condition is met, the function stops calling itself. This is called a base condition. To prevent infinite recursion, you can use **if...else** statement (or similar approach) where one branch makes the recursive call, and the other doesn't.

More about recursion you can learn: <https://www.javascripttutorial.net/javascript-recursive-function/>

Now let's run the **app** in the console and check whether our current code **works** correctly:

```
Guess the number (0-100): 41
Too High!
Guess the number (0-100): 35
Too Low!
Guess the number (0-100): 38
Too High!
Guess the number (0-100): 36
You guess it!
```

We can see that our application work's appropriately, but we are not finished yet.

Check for Invalid Input

Now what is left is to write **else** statement for the **final** case where the player's input is **invalid**. That's all it takes for the **game to work**.

```
} else {
    console.log('Invalid input! Try again...');
    recursiveAsyncReadLine();
}
```

Your entire code should be similar to the following:

```

function guessANumber() {
  const readline = require('readline').createInterface({
    input: process.stdin,
    output: process.stdout
  });

  let computerGuess = Math.floor(Math.random() * 100);
  let guess;

  let recursiveAsyncReadLine = function () {
    readline.question('Guess the number (0-100): ', number => {
      guess = Number(number);

      if (guess <= 100 && guess >= 0) {
        if (guess == computerGuess) { ...
        } else if (guess < computerGuess) { ...
        } else if (guess > computerGuess) { ...
        }
      } else {
        console.log('Invalid input! Try again...');
        recursiveAsyncReadLine();
      }
    });
  };

  recursiveAsyncReadLine();
}

```

guessANumber()

Test the Application

After you **run it**, the game should look like this:

Guess the number (0-100):

Guess the number (0-100): 12
 Too Low!
 Guess the number (0-100):

```
Guess the number (0-100): 40
Too High!
Guess the number (0-100): █
```

```
Guess the number (0-100): 37
You guess it!
```

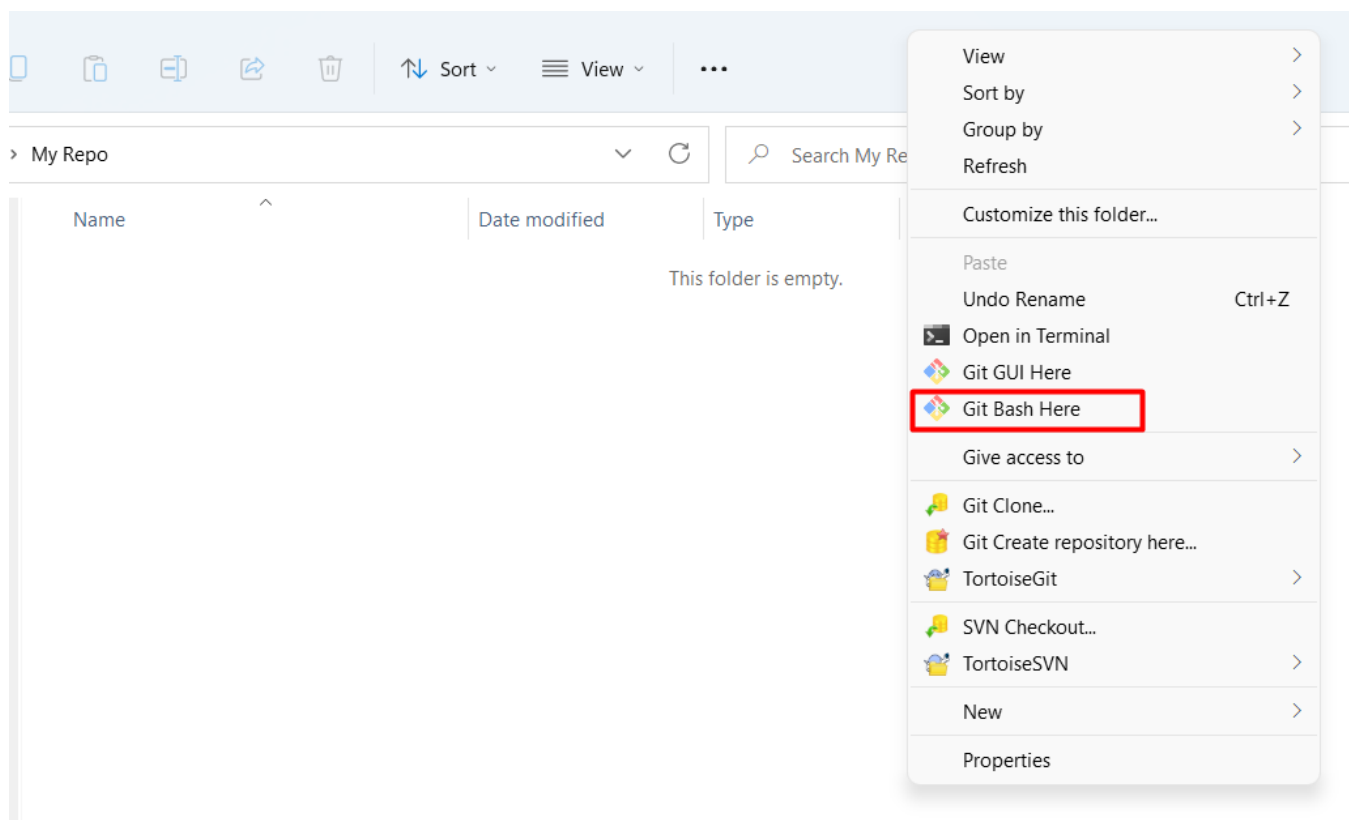
```
Guess the number (0-100): number
Invalid input! Try again...
Guess the number (0-100): █
```

3. Upload Your Project to Github

We already know how to clone our repository using **Git Bash** or **GitHub Desktop**.

Use GitBash (Option 1)

Go to the desired **directory**, right-click on a blank space **anywhere** in the folder, and select **"Git Bash Here"** to open the Git command line console. If the **"Git Bash Here"** menu is missing, you should first install Git.



Type the **"git clone"** command followed by the link to your **repository**:

```
git clone
```



```
Bobby@DESKTOP-DFHSTHV MINGW64 ~/Desktop/GuessANumber
$ git clone https://github.com/softuni/GuessANumber.git
```

The result should be something like this:


```
Bobby@DESKTOP-DFHSTHV MINGW64 ~/Desktop/GuessANumber
$ git clone https://github.com/ /GuessANumber.git
Cloning into 'GuessANumber'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

Your files from your GitHub repo will be downloaded to a **sub-folder** called your project in GitHub, "**GuessANumber**" in our case.

When we open the cloned **repository sub-folder**, it should look like this:

 .gitignore	9.8.2022 г. 12:32
 README.md	9.8.2022 г. 12:32

The next thing to do is to **add** your **project files** to your **cloned repository folder**. It should look like this:

 .gitignore	9.8.2022 г. 12:32
 README.md	9.8.2022 г. 12:32
 guessANumber.js	9.8.2022 г. 12:20

Now we are ready to upload our changes from the "**Git Bash clone**". Go to the desired **folder**, right-click on a blank space anywhere in the folder, select "**Git Bash Here**" and run the following **commands**.

Type the following command:

```
git status
```

The **git status** command displays the state of the working directory and the **staging area**.

```
Bobby@DESKTOP-DFHSTHV MINGW64 ~/Desktop/GuessANumber
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    guessANumber.js

nothing added to commit but untracked files present (use "git add" to track)
```

Now type:

```
git add .
```

This command **adds** all modified files.

Next type:

```
git commit -m "Your message here."
```

This command **commits** your changes. We also should **add** an appropriate **message**.

Second to the last type.

```
git pull
```

This command **updates** your local **repository**.

Now the last thing that we should do is to **push** our changes by using the command:

```
git push
```

This command **pushes** your changes to our local **repository**.

```
Bobby@DESKTOP-DFHSTHV MINGW64 ~/Desktop/Guess a Number/GuessANumber (main)
$ git add .

Bobby@DESKTOP-DFHSTHV MINGW64 ~/Desktop/Guess a Number/GuessANumber (main)
$ git commit -m "Console Game"
[main f827166] Console Game
1 file changed, 37 insertions(+)
create mode 100644 guessANumber.js

Bobby@DESKTOP-DFHSTHV MINGW64 ~/Desktop/Guess a Number/GuessANumber (main)
$ git pull
Already up to date.

Bobby@DESKTOP-DFHSTHV MINGW64 ~/Desktop/Guess a Number/GuessANumber (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 687 bytes | 687.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/BDimitrova/GuessANumber.git
4a30d90..f827166 main -> main
```

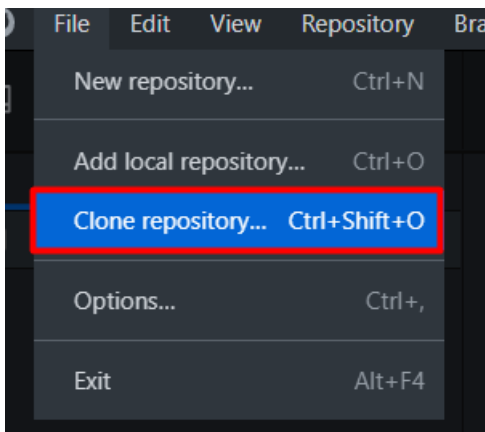
This is all you need to **update** your **repository** with **Git Bash**.

A little more information about it is here: <https://git-scm.com/about>.

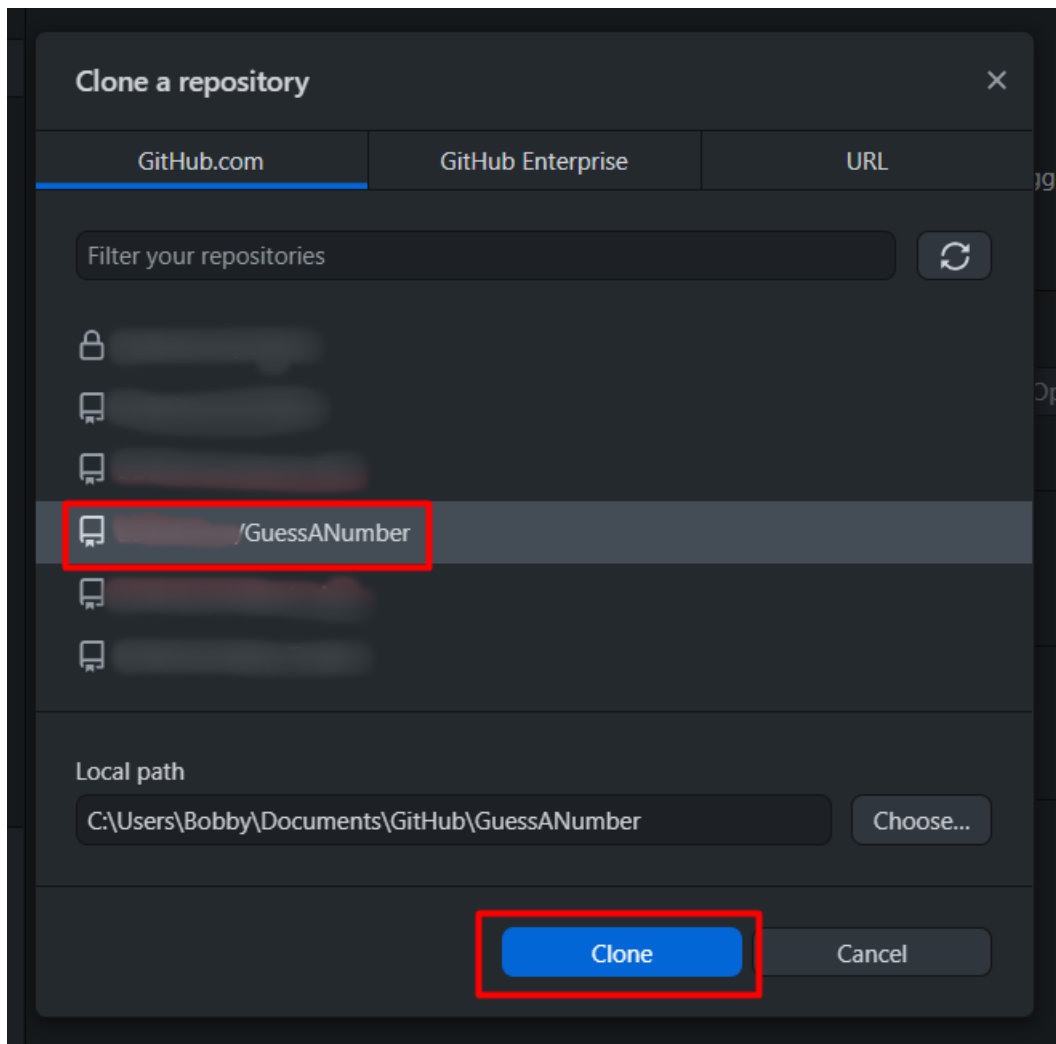
Use GitHub Desktop (Option 2)

If you don't have GitHub Desktop on your computer, download and install it from here: <https://desktop.github.com/>



Go to **"File"** and choose **"Clone repository"**.






Chose the **repository** for the project, in our case, "GuessANumber" and hit the **"Clone"** button.



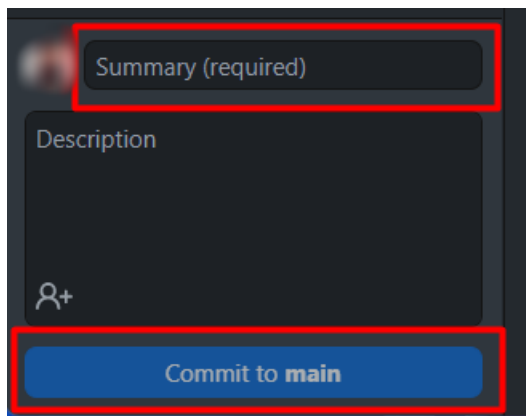
Your files from your GitHub repo will be downloaded to a **sub-folder** called your project in GitHub, "**GuessANumber**" in our case.

Name	Date modified
 .gitignore	9.8.2022 г. 13:02
 README.md	9.8.2022 г. 13:02

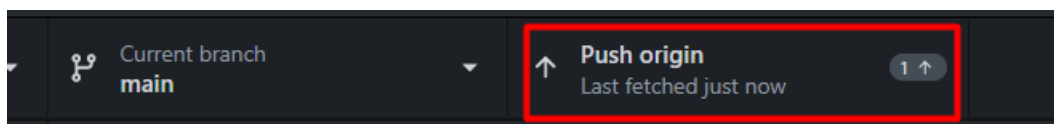
The next thing to do is to **add** your **project files** to your **cloned repository folder**. It should look like this:

 .gitignore	9.8.2022 г. 13:02
 guessANumber.js	9.8.2022 г. 13:02
 README.md	9.8.2022 г. 13:02

Afterward, go to GitHub Desktop and **create a commit**, just like this.



Then **push the commit** to the repository.



This is all you need to **update** your **repository** using **GitHub Desktop**.

4. * Modify the Code, Write Your Features

Now, it's time to **play with the code** and **modify** it.

	<p>This is your project. Be unique. Don't be a copy/paster!</p> <ul style="list-style-type: none">• Implement your features.• Implement the code yourself, using your coding style, code formatting, comments, etc.• Make the project more interesting. Learn by playing with the code and adding your changes.
--	---

Below are a few **ideas** of what you can implement or modify as an addition to your code.

Add Difficulty

You can add logic for difficulty, so the player can have **only a few tries** to guess the number.

Restart the Game

You can automatically **restart the game** after it is finished (or ask the player to play again).

Additional Ideas

- You can **add levels** so whenever the player guesses the number, the range between the minimum and maximum number gets bigger, e. g. **Level 1 (1 - 100)**, **Level 2 (1-200)**, etc.
- Can you add anything else to your code based on your ideas?

Commit to GitHub

Now **commit and push your code changes** to your GitHub repo!



843 contributions in the last year



Learn how we count contributions

Contribution activity

March 2022

Created 36 commits in 1 repository

2022

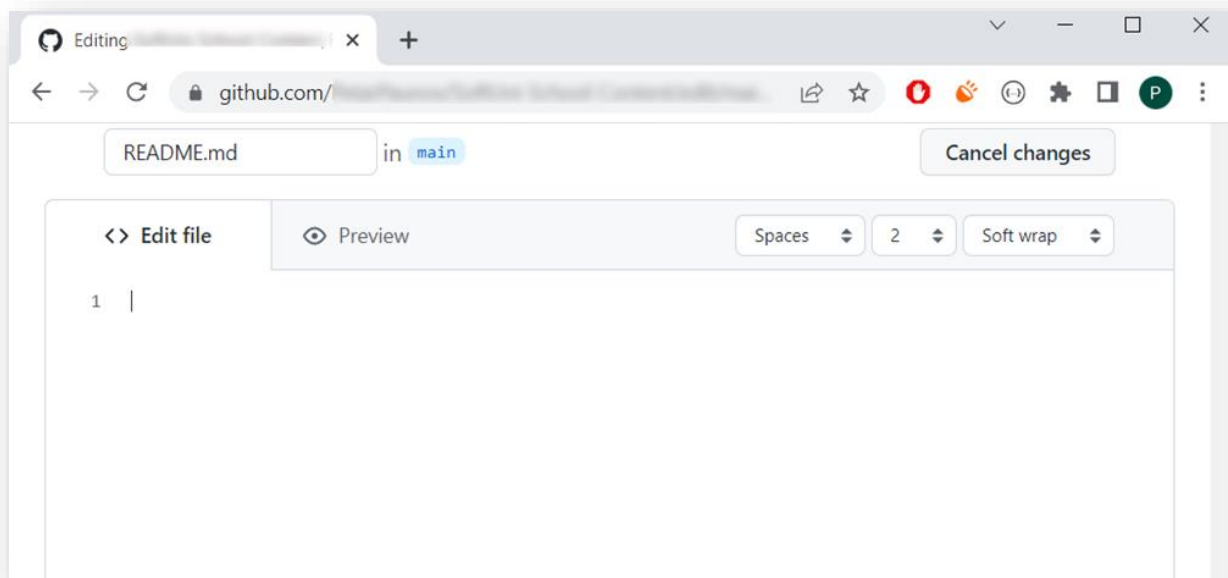
2021

2020

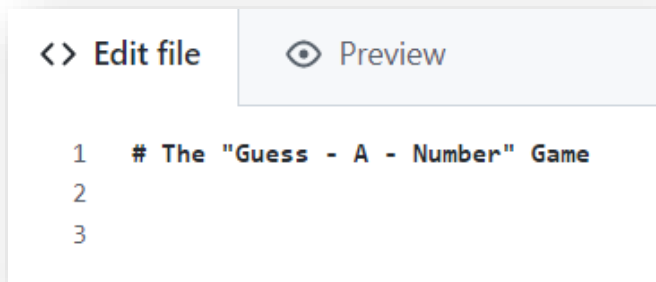
It is very important to **commit your code frequently** to GitHub. This way, you create a **rich commit history** for your project, and your GitHub contribution graph is growing:

5. Create a README.md File

It's highly recommended to provide documentation as part of your project on GitHub to describe what the project is doing. So, let's make one for this **project**. Let's start by editing the **README.md** file from our repo on GitHub:



Add a project name. Use "#" in front of the text to indicate the **title**:



You can **view** the current progress by pressing the **[Preview]** button:

Documentation Sections

Add **information** about your project in your **README.md** file: project goals, technologies used, screenshots, live demo, etc. Typically, you should have the following **sections**:

- **Project title** (should answer the question "What's inside this project")
- **Project goals** (what problem we solve, e. g., we implement a certain game)
- **Solution** (should describe how we solve the problem → algorithms, technologies, libraries, frameworks, tools, etc.)
- **Source code link** (give a direct link to your source code)
- **Screenshots** (add screenshots from your project in different scenarios of its usage)
- **Live demo** (add a one-click live demo of your code)

Use Markdown

Note that the GitHub **README.md** file is written in the **Markdown language**. Markdown combines text and special formatting tags to describe formatted text documents.

You can learn more about **Markdown** here: <https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax>.

Project Goals

Start your documentation by describing your **project goals**. What problem does your project solve?


Sample Documentation

This is an **example** of how you can document your project. Don't copy-paste it!

[Edit file](#) [Preview](#) [Show diff](#)

The "Guess - A - Number" Game

A console-based C# implementation of the "Guess - A - Number" game.



Guess A Number is a simple guessing game where a **user** is supposed to **guess** the number that your **oponent** (the computer) has come up with. The program **randomly** selects a number between **1 and 100**. It will then ask the player to **enter their guess**. Each time you enter a number, the **computer** will tell you whether it is **lower** or **higher** than the **expected** number.

You win the game when your **number** matches your **computer** number



Write the project documentation yourself. Don't copy/paste it!

This is your **unique GitHub profile** and your unique project. **Be different** from others.

Find an **appropriate image** and add it. You can add **images** as follows:

```

```

You can add information about the **inputs** and **outputs** of the project:

Input and Output

Choose **number** between **1** and **100** , then press **Enter** .

The computer selects a **random number**, then returns information whether the number is **less than**, **greater than**, or **equal** to the selected number.

Your Solution

Describe how you **solve the problem**: algorithms, technologies, libraries, frameworks, tools, etc.

Link to the Source Code

[Source Code](guess_a_number.py)

Screenshots

Add **screenshots** of your project:

1. **Take a screenshot** with your favorite tool (e.g., the [Snipping Tool](#) in Windows).
2. **Paste** the screenshot in the GitHub Markdown editor using [**Ctrl+V**]:

Example screenshots for the "Guess a Number" game:

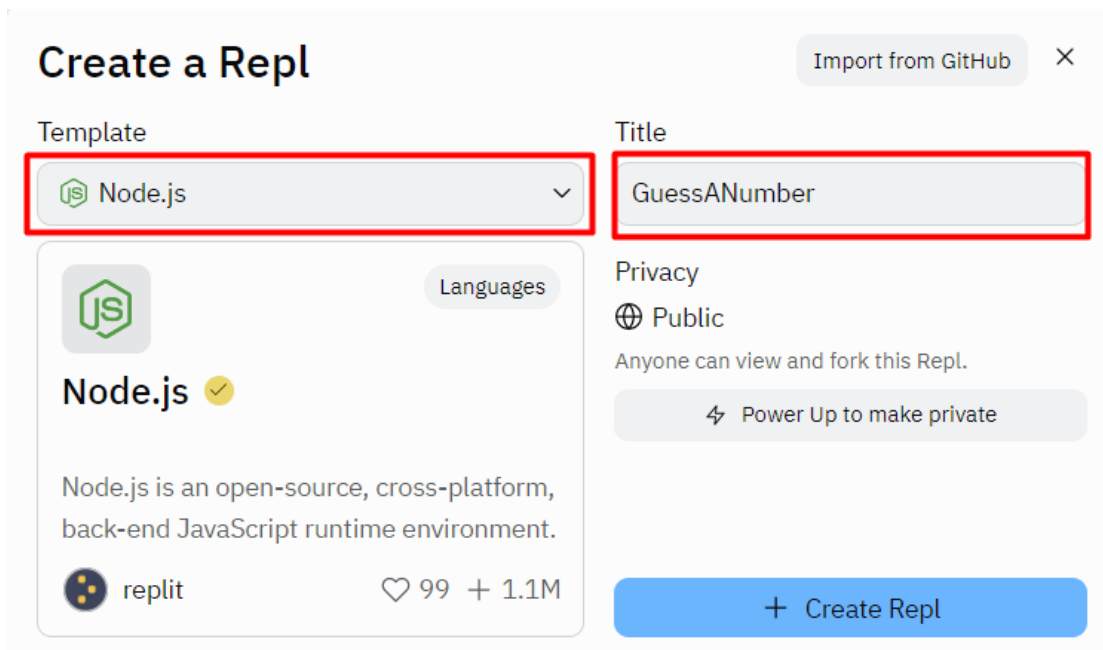
```
Guess the number (1-100): 40
Too Low!
Guess the number (1-100): 45
Too High!
Guess the number (1-100): 43
Too Low!
Guess the number (1-100): 44
You guess it!
```

```
Guess the number (1-100): some text
Invalid input. Try again...
Guess the number (1-100): 1
You guess it!
```

6. Upload Your App to Replit

You already should have a **Replit** profile. Now let's add our **project** there so we can share it with our **friends** and add it to our **GitHub** profile. You already should know how to do that.

Open the **menu** in the upper **left corner**. Click "**Create**", then select the **language** in which your project is **written**, select a name, and **create** the project **Node.js**



Create a Repl Import from GitHub ×

Template: Node.js ▼ Title: GuessANumber

JS Languages

Node.js ✓

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment.

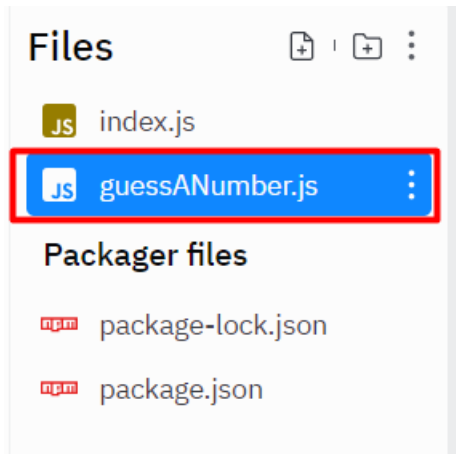
replit ♡ 99 + 1.1M

Privacy: Public
Anyone can view and fork this Repl.

⚡ Power Up to make private

+ Create Repl

Paste your code in the "guessANumber.js" file:



```
~/GuessANumber$ node guessANumber.js
Guess the number (0-100): 40
Too Low!
Guess the number (0-100): 65
Too High!
Guess the number (0-100): 50
Too High!
Guess the number (0-100): 45
Too High!
Guess the number (0-100): 42
Too High!
Guess the number (0-100): 41
You guess it!
~/GuessANumber$
```

You can now **share** your app with your friends.

7. Add Replit Link to Your README.md

Now add a "**one-click live demo**" of your project from your GitHub project documentation. You can do it as follows:

```
## Live Demo

You can play the game directly in your Web browser here:

[]
(https://replit.com/@[username]/Guess-A-Number-Game#main.py)
```

You can take a **screenshot** from Replit.com and **paste it** into the GitHub documentation editor directly with **[Ctrl+V]**.

Now we have completed our **second console game**, and we have our second **project** in our **GitHub** portfolio.