# Practical Project: Rock–Paper–Scissors

Let's make the console game "Rock – Paper – Scissors":

Icon

Description automatically generated

[**Rock-Paper-Scissors**](https://en.wikipedia.org/wiki/Rock_paper_scissors) is a simple **two-player game**, where you and your opponent (the computer) simultaneously choose one of the following three options: "**rock**", "**paper**", or "**scissors**". The rules are as follows:

* **Rock beats scissors** (the scissors get broken by the rock)
* **Scissors beats paper** (the paper gets cut by the scissors)
* **Paper beats rock** (the paper covers the rock)

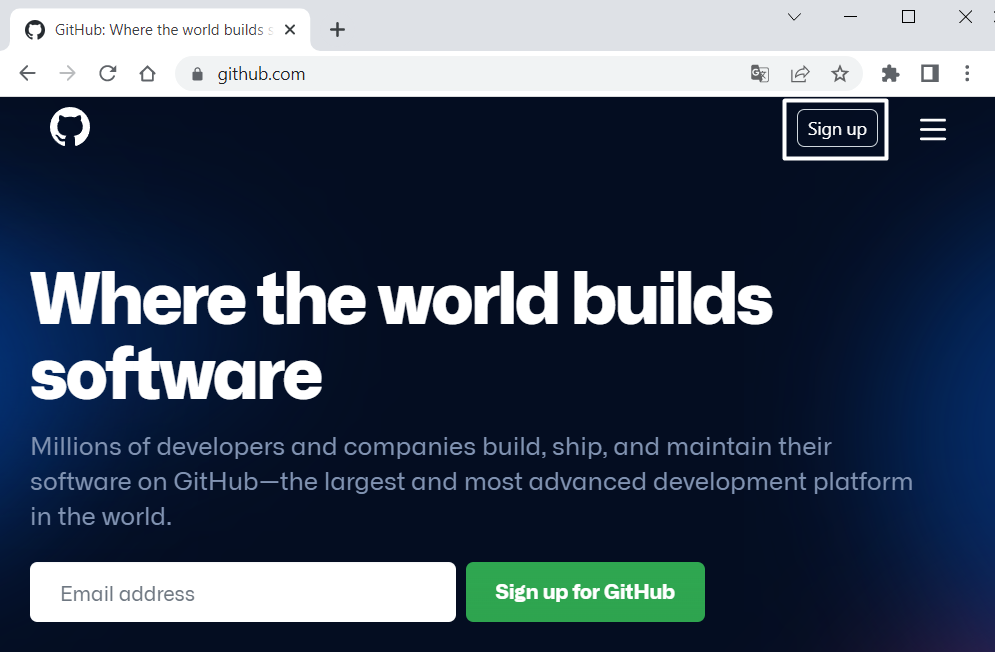
The **winner** is the player whose choice beats the choice of his opponent. If both players choose the same option (e.g., "paper"), the game outcome is "**draw**".

## Create a GitHub Profile and Repo

Everyone should have a GitHub developer profile. First, we should **create our profile on GitHub**.

### Register a GitHub Profile

**Register** for a free **developer account at GitHub** here: <http://github.com> with an email and a username:

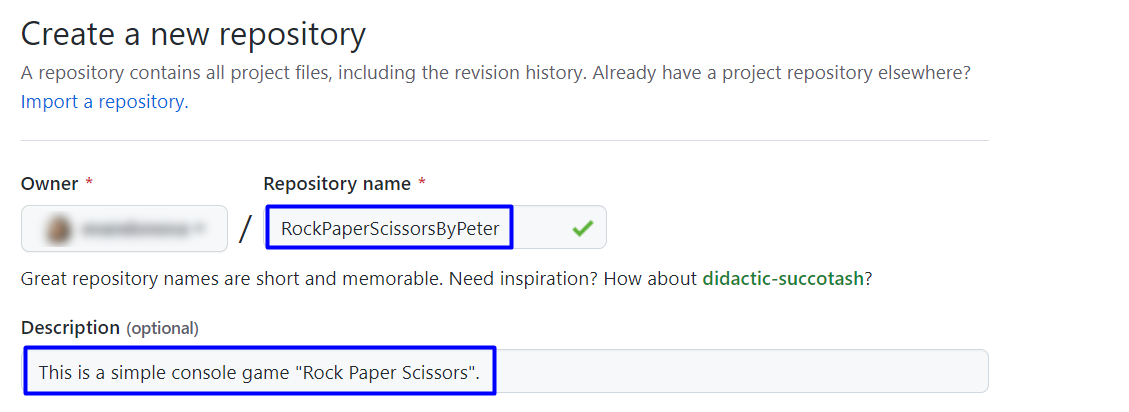
A screenshot of a computer

Description automatically generated with medium confidence

When you are ready, it is time to **create your first repository**. A **repository** contains **all of your project's files** and each file's revision history. You can discuss and manage your project's work within the repository.

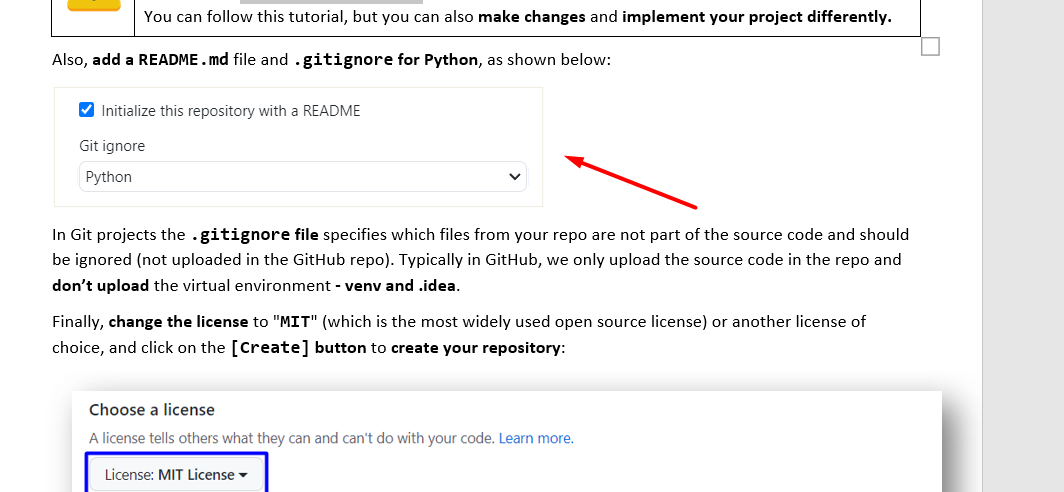
### Create a GitHub Repo

Create a **new repository** from: <https://github.com/new>. Choose a **meaningful name**, e. g. "RockPaperScissorsByUsername". Add a **short description** and make your repo **public**:



|  |  |
| --- | --- |
| Icon  Description automatically generated | Please choose **your own original and unique name** for your project!  Your GitHub profile should be **unique**.  You can follow this tutorial, but you can also **make changes** and **implement your project differently.** |

Also, **add a** README.md file and .gitignore **for Python**, as shown below:



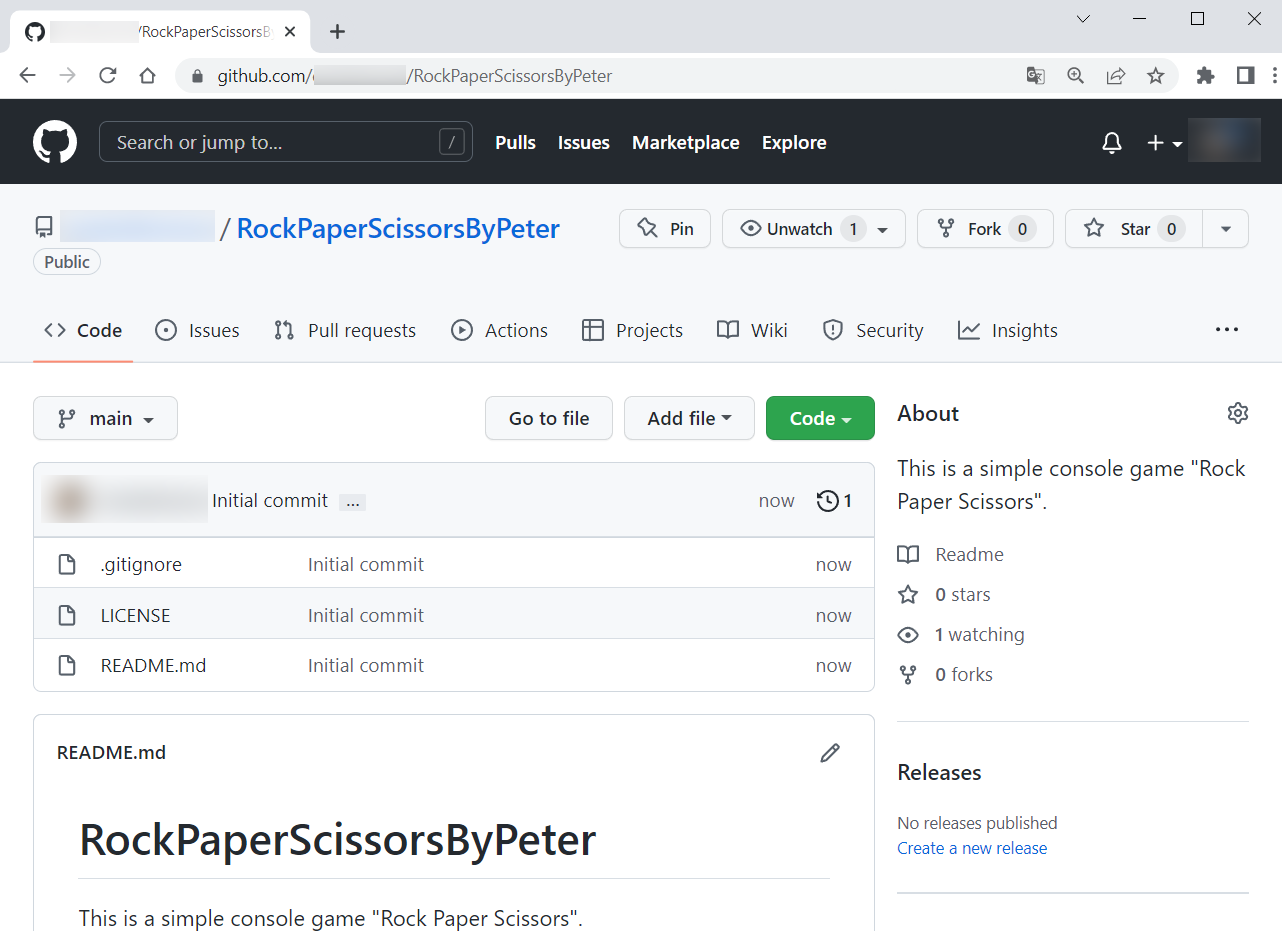
In Git projects, the .gitignore **file** specifies which files from your repo are not part of the source code and should be ignored (not uploaded in the GitHub repo). Typically in GitHub, we only upload the source code in the repo and **don't upload** the virtual environment - **venv and .idea**.

Finally, **change the license** to "MIT" (which is the most widely used open source license) or another license of choice, and click on the [Create] **button** to **create your repository**:

Graphical user interface, text, application, email

Description automatically generated

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



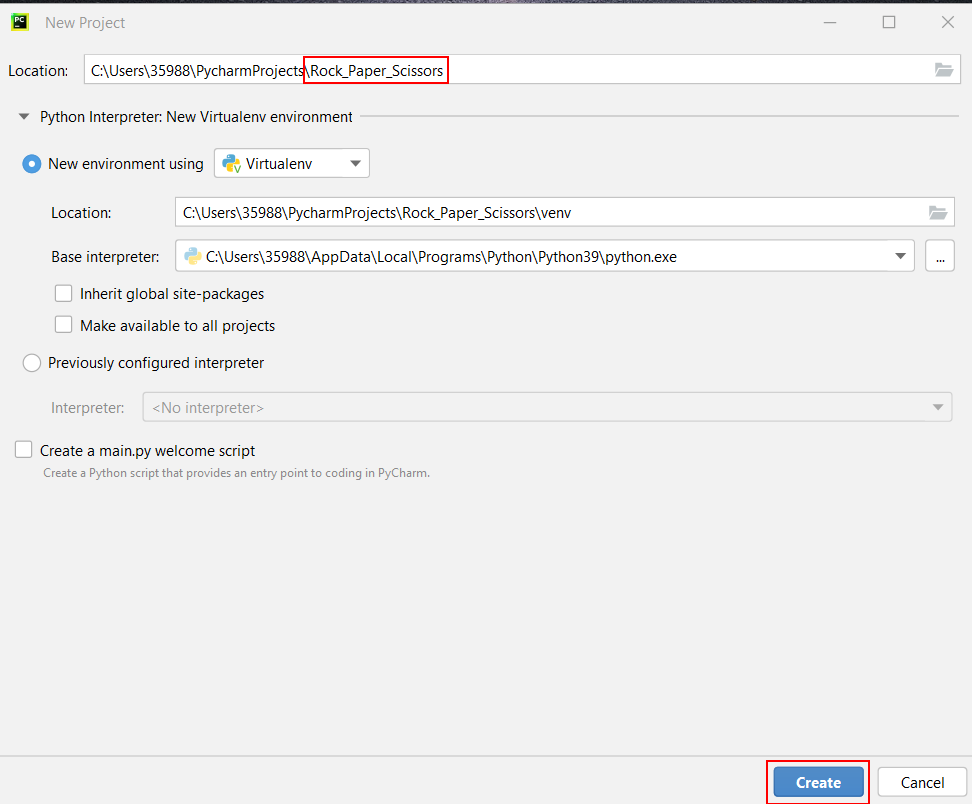
Now let's see how to **write the code** of our game.

## Write the Game's Code

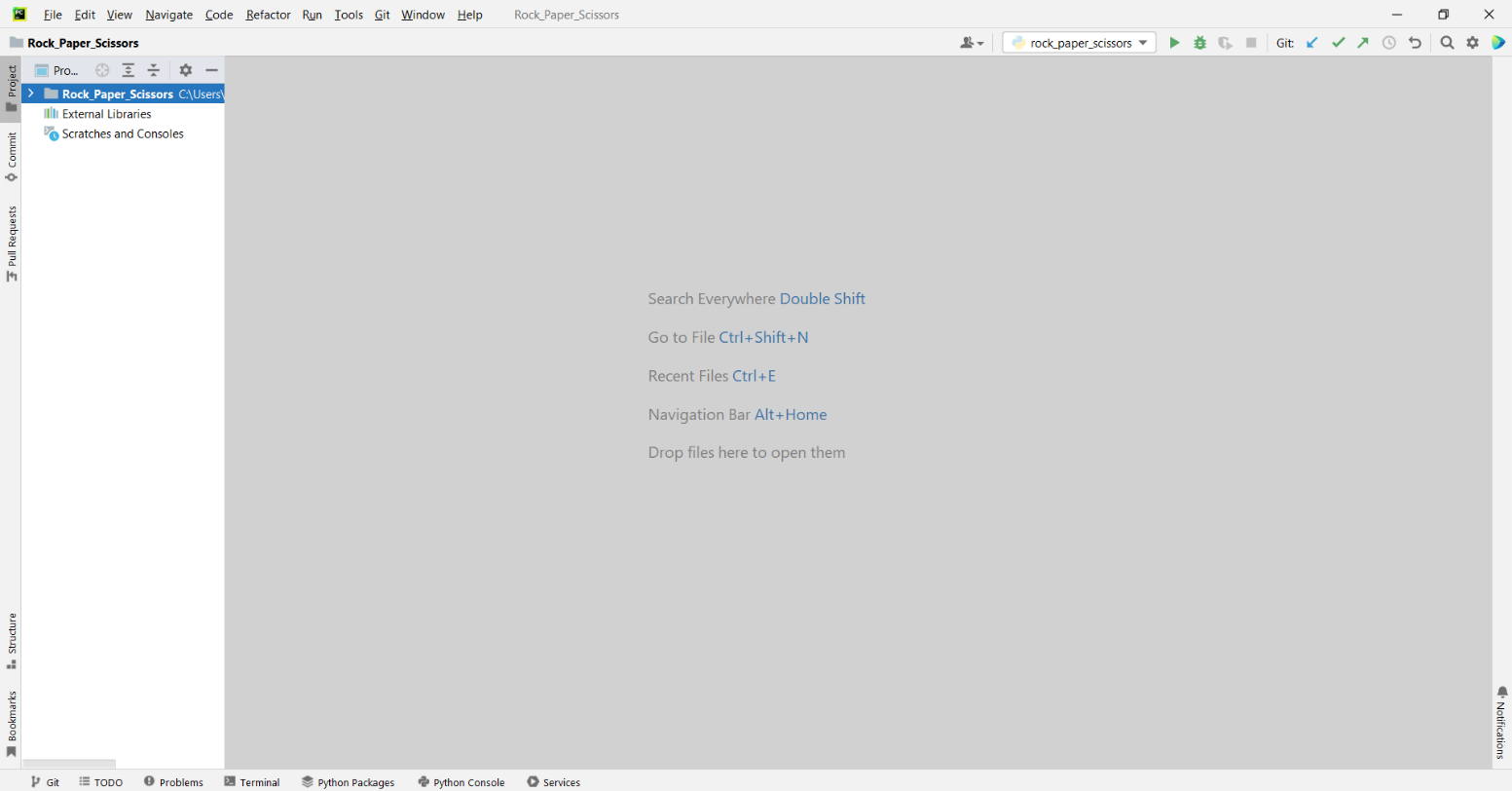
Let's create the game and play with it.

### Create a PyCharm Project

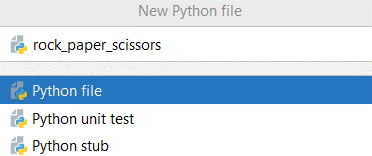
First, we should **start PyCharm** and **create a new Python project**:



Our **project should be created** and should look like this:



We should create a **new Python file** with the name of the game.

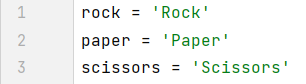


### Implement the Game Logic

#### Read Player's Move

Now let's start working on our code.

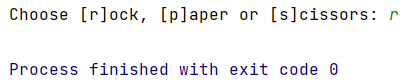
Create **three variables** for our "**Rock**", "**Paper**", and "**Scissors**", which we will use later. They should look like this:



Next, write on the console what options (**"r"** for "rock", **"p"** for "paper",and **"s"** for "scissors") the player can choose from and read his **input** **data**. You already know how to do this:



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

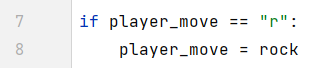


We can see that we have our text **written** on the console, and we can also **write**.

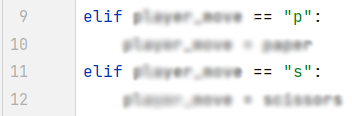
#### Match Player's Move with Possible Options

Now it is time to turn the user input into one of our **player's moves options**. To do this, create an if-else statement with the **possible moves** and change the **player\_move** variable value with the value it represents.

First, if the user has entered **"r"**, they chose **"Rock"**. Write it like this:



And if they have entered **"p"** or **"s"**, then they chose **"Paper"** or **"Scissors"** accordingly. Write the else-if statements by yourself:

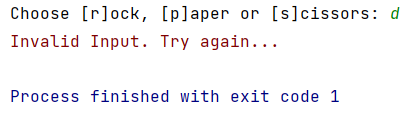


Now we should cover the case, in which the user enters an **invalid value**. To do this, use else and **raise SystemExit** with a message on the console to **stop the program from executing**:



**Note: "raise SystemExit" is an exception, for now, all you need to know is that it exits the whole program.**

Now let's **run** the app in the **console** and check whether our current code **works properly**, at the moment we have **logic** only for the **incorrect input** so the results should be as follow:



#### Choose Computer's Move

**Create a variable of type** Random that will help us **choose a random number** using the randint method. We will use this **number** so that the computer can randomly select from "rock", "paper", or "scissors":



**Note: "randint()" is a method from the build-in library - "random" in Python (like the "math" library that you should know from the "Programming Basics" course). "randint()" accepts two parameters, both inclusive, and returns a random number in this range. For more clarification see these examples** [**here**](https://www.w3schools.com/python/ref_random_randint.asp)**.**

We will need a **variable of type** string to keep our **computer's move:**

****

Choose the computer's **random move**, to make this happen use the **conditional statements** if-else.

****

Think about how you can complete these **conditional statements**.

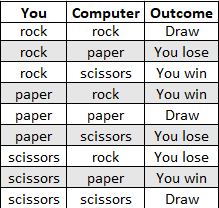
#### Check and Write the Result

Write to the console what is the **random** selection of the computer e.g., "**The computer chose {computer\_move}.**". Now we need to **compare** the choice of the **player** and the **computer**, again using **conditional statements**.

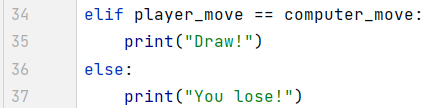
Graphical user interface, text

Description automatically generated with medium confidence

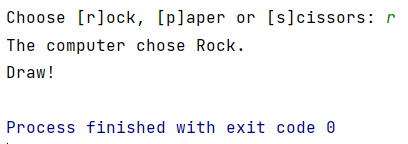
You can use this table for the **possible moves**:

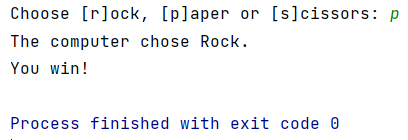


Consider all the cases where the player **loses** or the result between them is **equal** and write down the **conditional** **statements**. That's all it takes for the **game to work**.



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



****

## Upload Your Project to GitHub

Now we want to deploy our project to **GitHub** so the other developers can see it, and if they want to test it, they can clone it and try it themself on their machine. You have **two options**, choose one and follow the steps.

### Use Git Bash (Option 1)

You could use the "**Git Bash**" command line tool to upload your project to your GitHub repo.

First, if you don't have **Git** on your **computer**, you should **install it** from <https://git-scm.com/downloads>.

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.

Graphical user interface, application

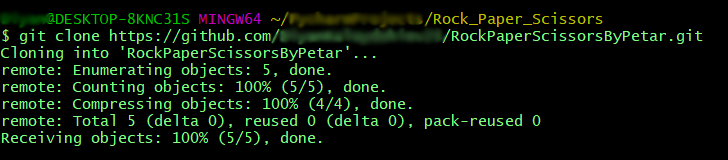
Description automatically generated

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

gitclone



The result should be something like this:

****

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

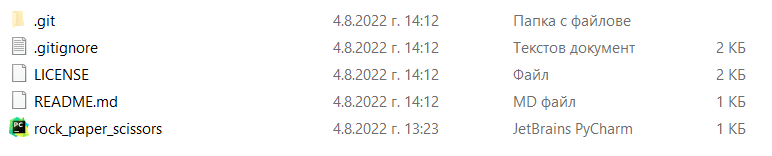
Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

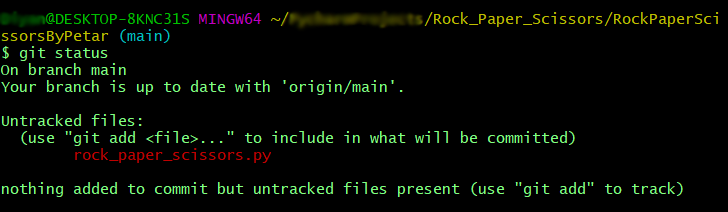
The next thing to do is to **add** your **project files** to your **cloned** **repository folder**. It should look like this:

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 "GitBashHere" 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**.

****

Now type:

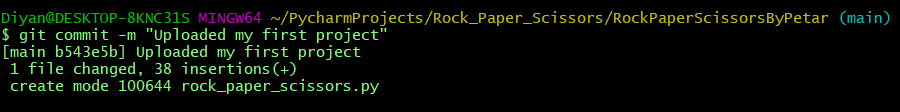
git add .

The above command **adds** all modified files to your local **Git repo**.

Now type:

git commit -m "Uploaded my first project"

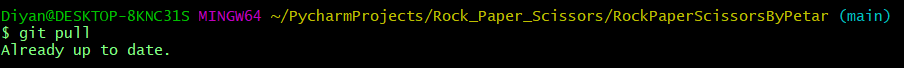
This command**commits** your changes to your local **Git repo**. We also should **add** an appropriate **commit message**.



We have **two** more **commands** left. Second to the last type.

git pull

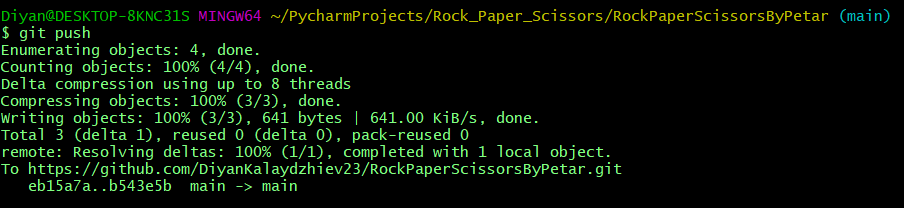
This command **updates** your local **repository** from GitHub. It downloads the latest project version from GitHub and merges it with your local copy.

****

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

git push

This command **pushes your local changes to GitHub**.

****

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

A little more information about Git Bash: <https://git-scm.com/about>.

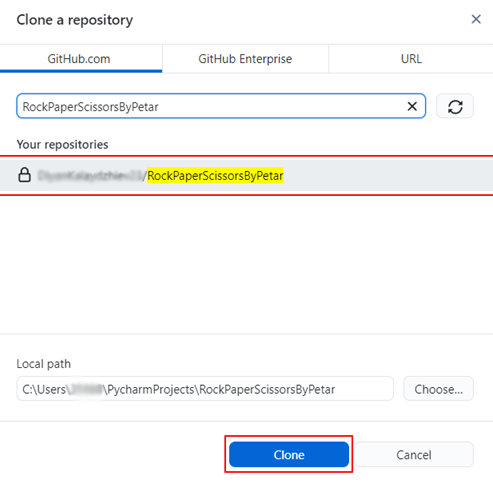
### Use GitHub Desktop (Option 2)

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

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

****

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



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

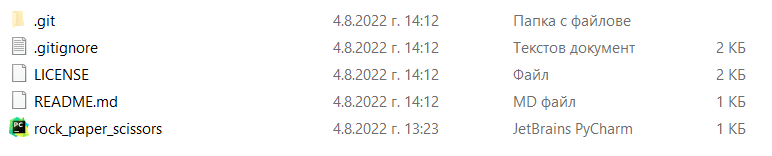
Graphical user interface, text, application

Description automatically generated

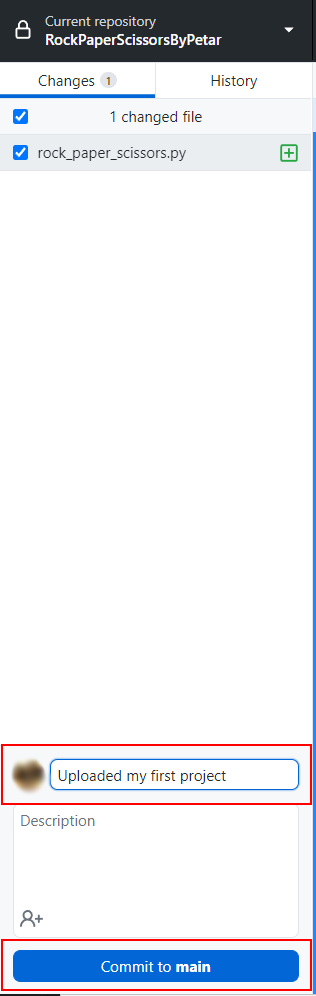
Graphical user interface, text, application

Description automatically generated

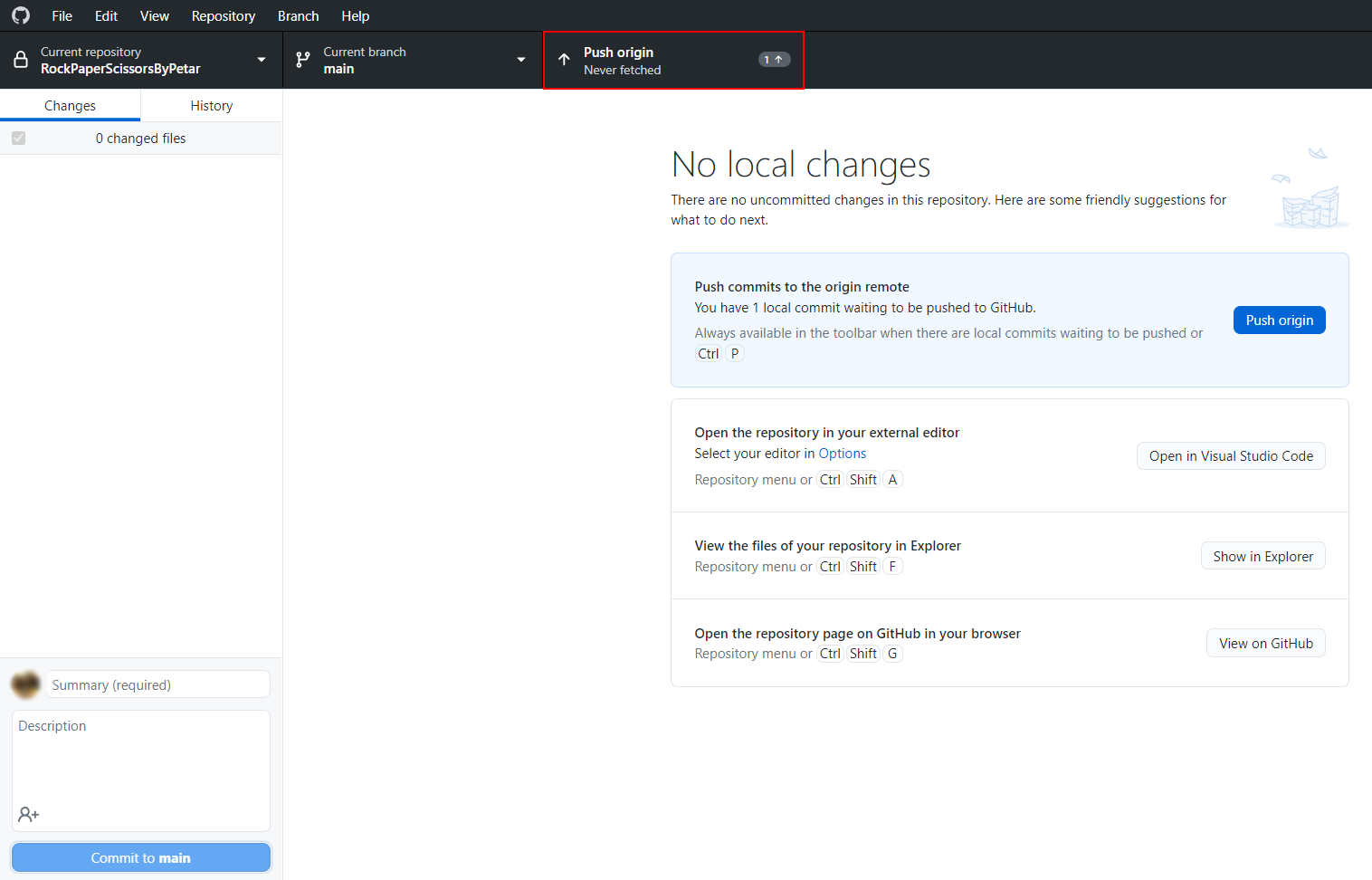
The next thing to do is to **add** your **project files** to your **cloned** **repository folder**. It should look like this:



After that 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** usingGit**Hub Desktop.**

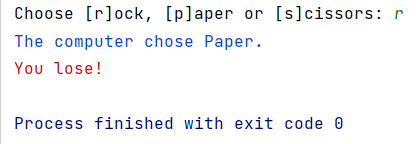
## \*Modify the Code, Write Your Own Features

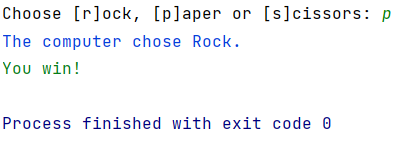
|  |  |
| --- | --- |
| Icon  Description automatically generated | This is your own project. **Be unique**. Don't be a copy/paster!   * Implement your **own features**. * **Implement the code yourself**, using your own coding style, code formatting, comments, etc. * Make the project **more interesting**. Learn by playing with the code and adding your own changes. |

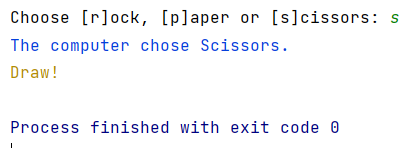
Below are a few **ideas** of what you can implement or modify in addition to your code.

### Add Colors

You can modify the **text color** and **text background** in the console: <https://www.geeksforgeeks.org/print-colors-python-terminal/>

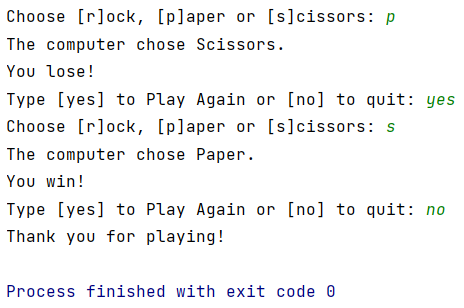






### Restart the Game

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



### Scoring System

You can add a **scoring system** and display the player's and the computer's scores after each game session.

### Additional Ideas

* Can you change your logic, so you can **increase the chances of the player winning**?
* Can you add **anything else** in your code, based on your own ideas?

### Commit to GitHub

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

|  |  |
| --- | --- |
| Icon  Description automatically generated | It is very important to **commit frequently** your code to GitHub. This way you create a **rich commit history** for your project and your **GitHub contribution graph** is growing: |

## 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:

Graphical user interface, text, application, email

Description automatically generated

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

Graphical user interface, application

Description automatically generated

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

Graphical user interface, application

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

### 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 do 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 document.

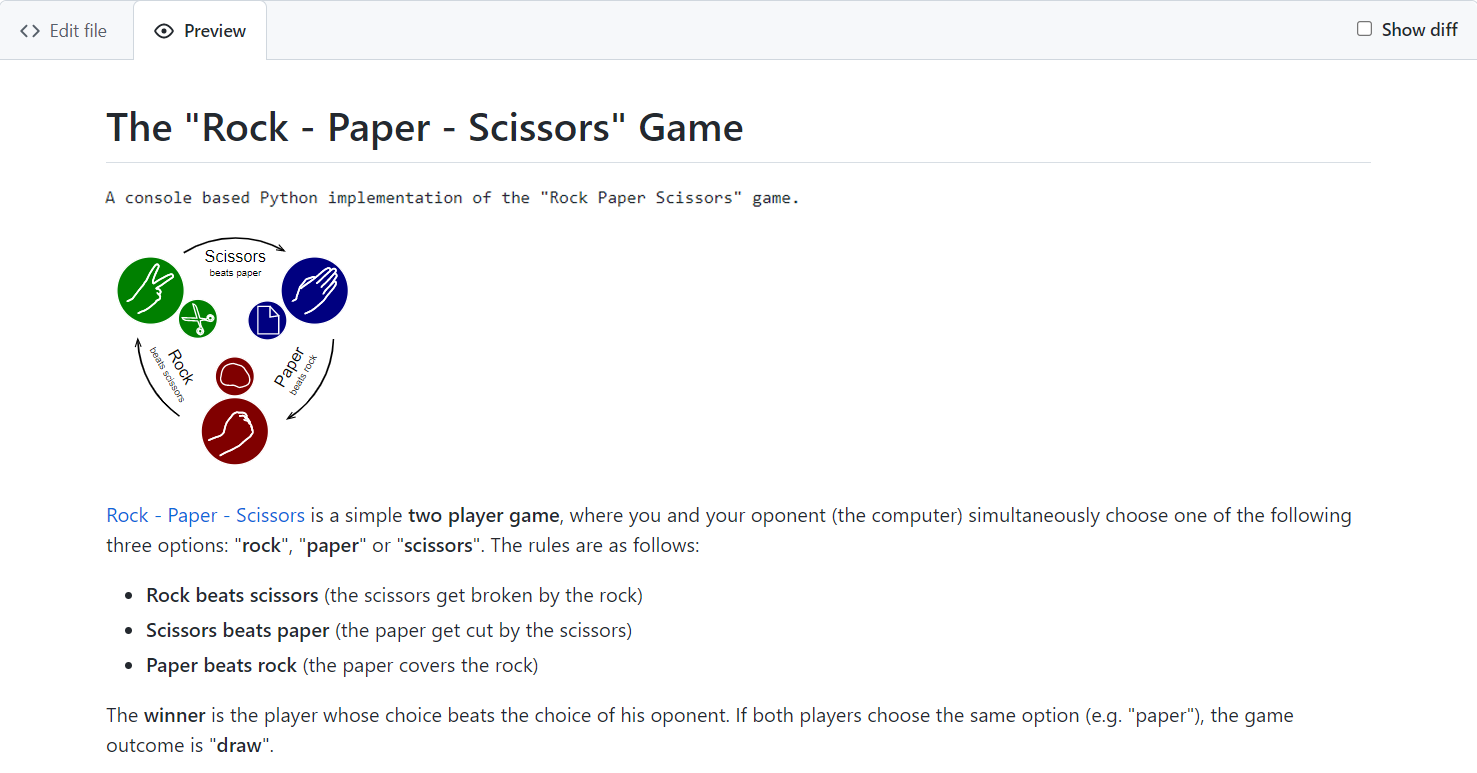
You can learn more about **Markdown** here: <https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax>.

### 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!

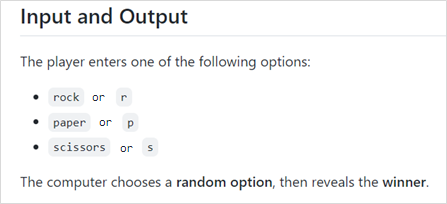


|  |  |
| --- | --- |
| Icon  Description automatically generated | **Write the project documentation yourself**. Don't copy/paste it!  This is your **unique GitHub profile** and your own unique project. **Be different** from others. |

You can add **appropriate** **images** to make your documentation better. You can add an **image** as follows:



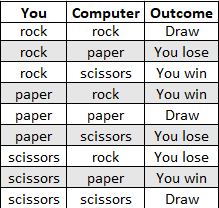
You can add information about the **inputs** and **outputs** of the project:



### Your Solution

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

For example, for our simple game, you may analyze all possible game **situations** in a **table**:



### Link to the Source Code

Add a **link** to your **source code** as follows:

[Source Code](rock\_paper\_scissors.py)

### Screenshots

Add **screenshots** of your project:

1. **Take a screenshot** with your favorite tool (e.g., the [Snipping Tool](https://support.microsoft.com/en-us/windows/open-snipping-tool-and-take-a-screenshot-a35ac9ff-4a58-24c9-3253-f12bac9f9d44) in Windows).

Graphical user interface, text, application, email

Description automatically generated

1. **Paste** the screenshot in the GitHub Markdown editor, using [Ctrl+V]:
2. **Take a screenshot** with your favorite tool (e.g., the [Snipping Tool](https://support.microsoft.com/en-us/windows/open-snipping-tool-and-take-a-screenshot-a35ac9ff-4a58-24c9-3253-f12bac9f9d44) in Windows).

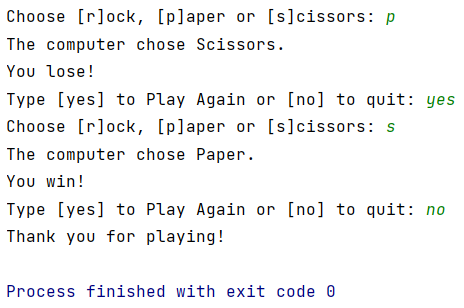
Graphical user interface, text, application, email

Description automatically generated

1. **Paste** the screenshot in the GitHub Markdown editor, using [Ctrl+V]:

Example screenshot

for the "Rock Paper Scissors" game:



## Upload Your App to Replit

Replit is an online coding environment (online IDE) which allows you to **write** software projects, **share** them through a simple link, and **run** your projects directly in the Web browser. We shall upload our project in Replit to allow the users to **run and interact with the project** with just **one click**.

Create your Replit profile so you can show your **projects** to your friends and put "**live demo links**" in your GitHub project documentation. Create a **Replit** account for **free**: [https://replit.com](https://replit.com/).

Graphical user interface, text, website

Description automatically generated

A screenshot of a computer

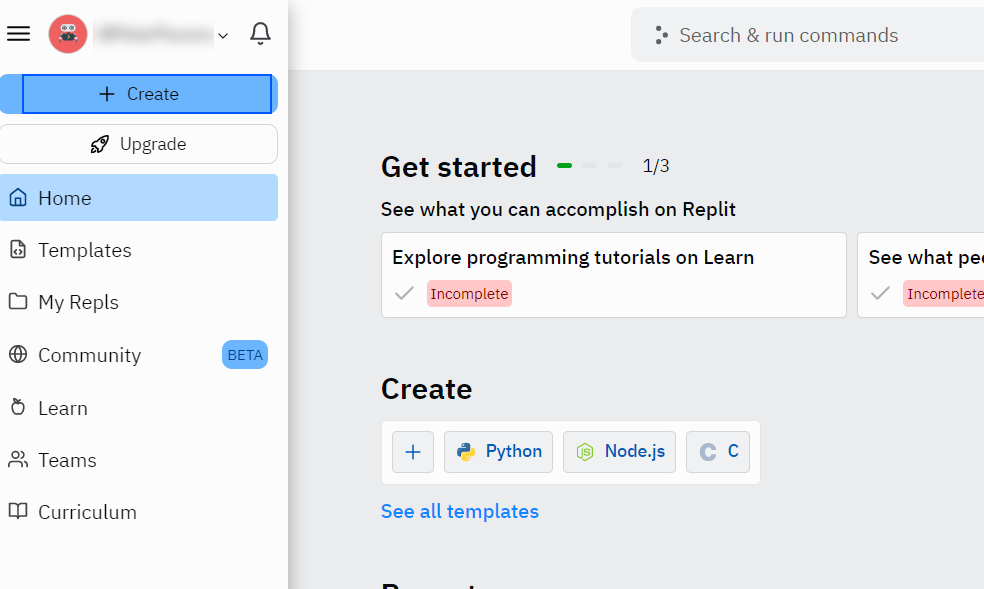
Description automatically generated

Create a **new project** in Replit; open the **menu** in the upper **left corner**.

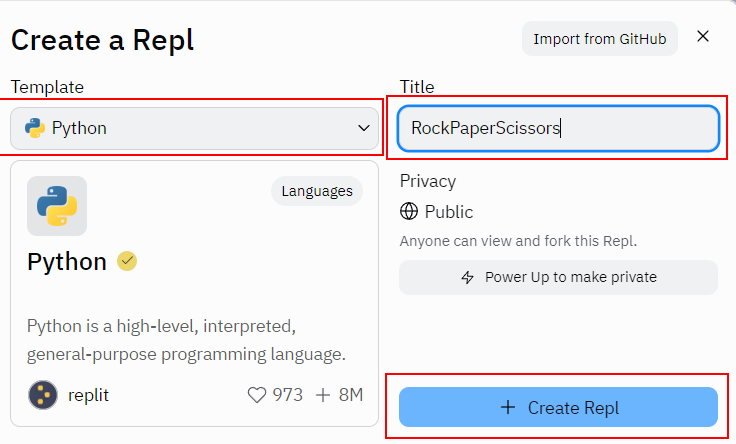
Graphical user interface, text, application, chat or text message, website

Description automatically generated

Click [Create], select the **language** in which your project is **written**, select a name, and **create** the project.



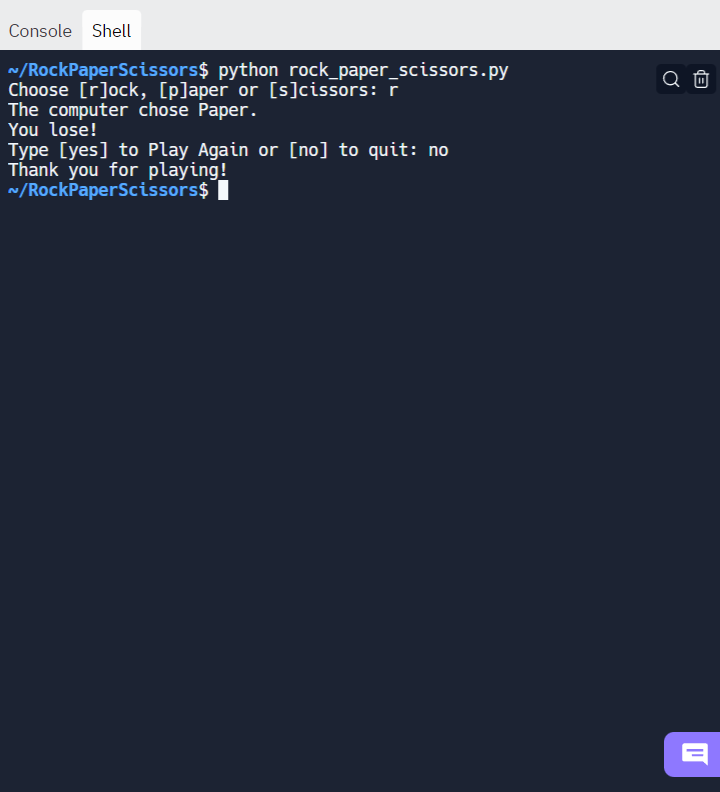
Chose "Python" for your project.



Add a meaningful **name** to your Replit project, e.g., "**RockPaperScissors**".

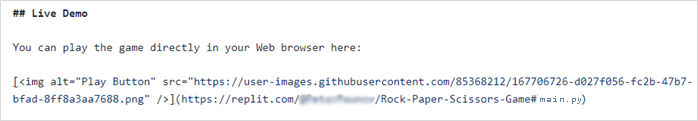
**Paste your code** in the "main.py" file:

**Click** [Run] and enjoy your console application directly in the Web browser:



## Add Replit Link to Your README.md

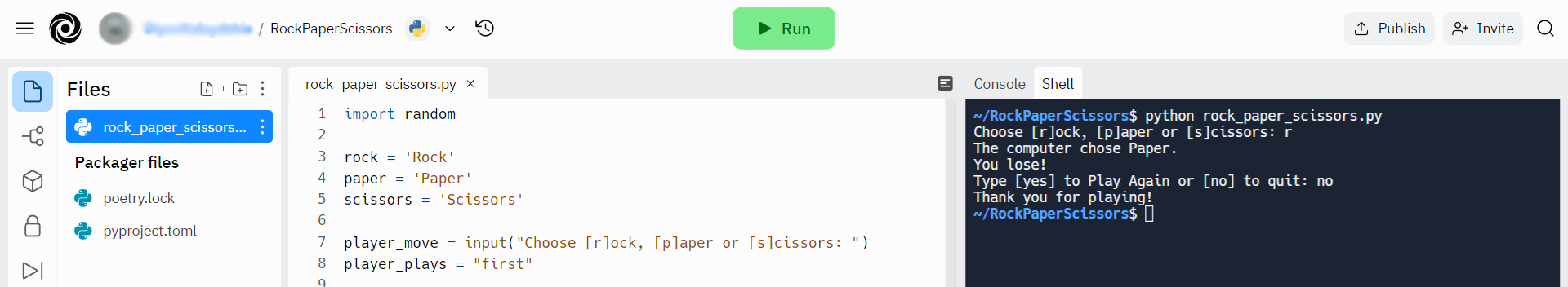
Now add a "**one-click live demo**" of your project from your GitHub project documentation. You can do it as follows:



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

Now when the [**Run**] button is clicked, you will be redirected to your demo in Replit.





We have completed our **first console game and** have our first project in our GitHub portfolio.