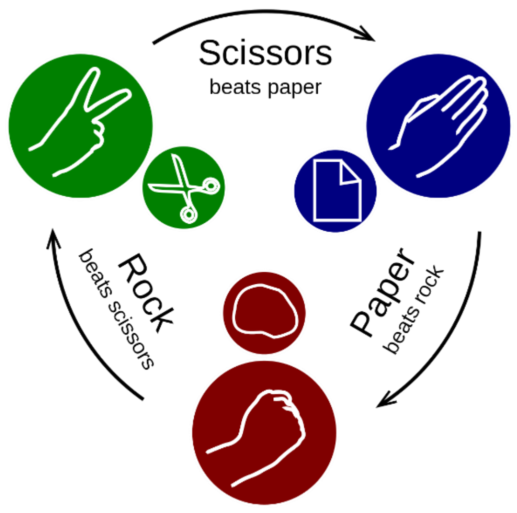
Practical Project: Rock–Paper–Scissors

This is an 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 types of problems and to improve your portfolio and GitHub skills.

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



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)

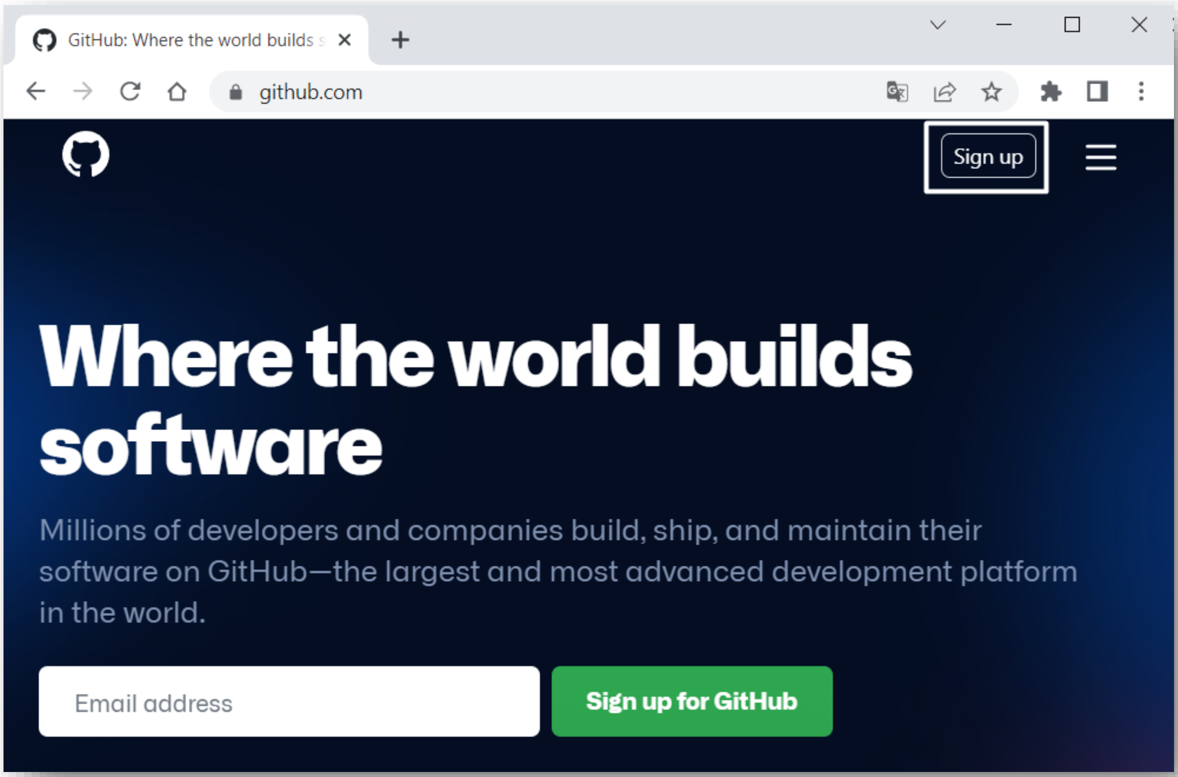
The **winner** is the player whose choice beats the choice of his opponent. If both players choose the same option (e.g., "paper"), the game outcome is "**draw**".

1. Create a GitHub Profile and Repo

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

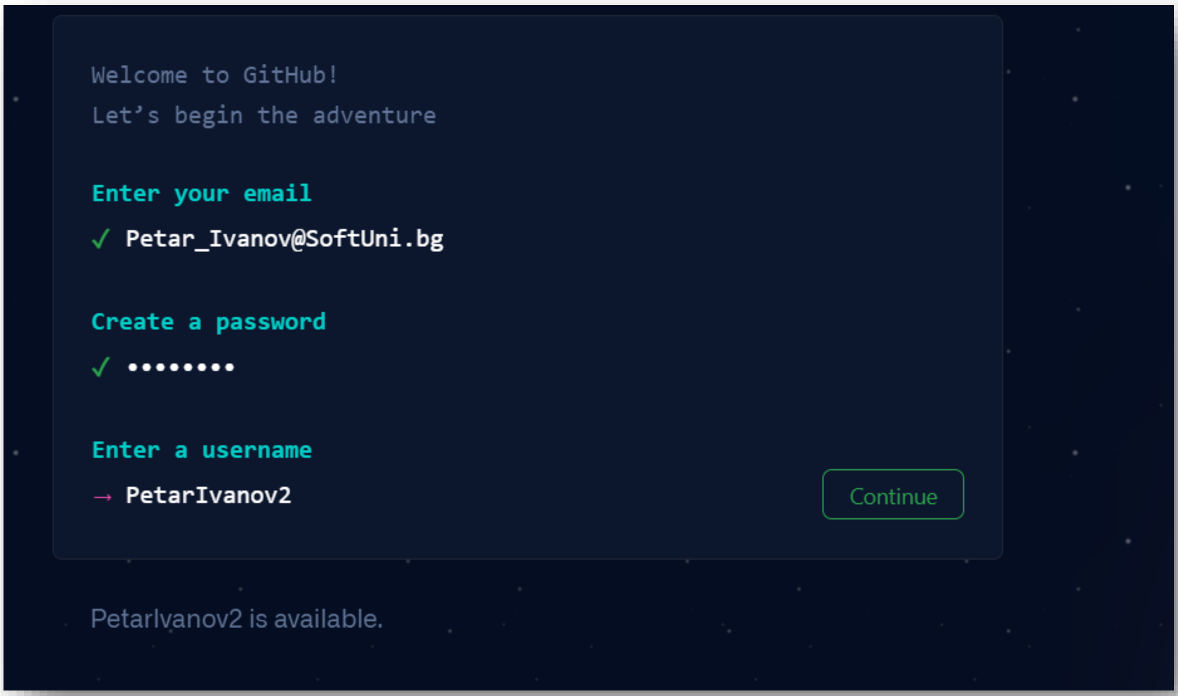
Register a GitHub Profile

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



Where the world builds software

Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.



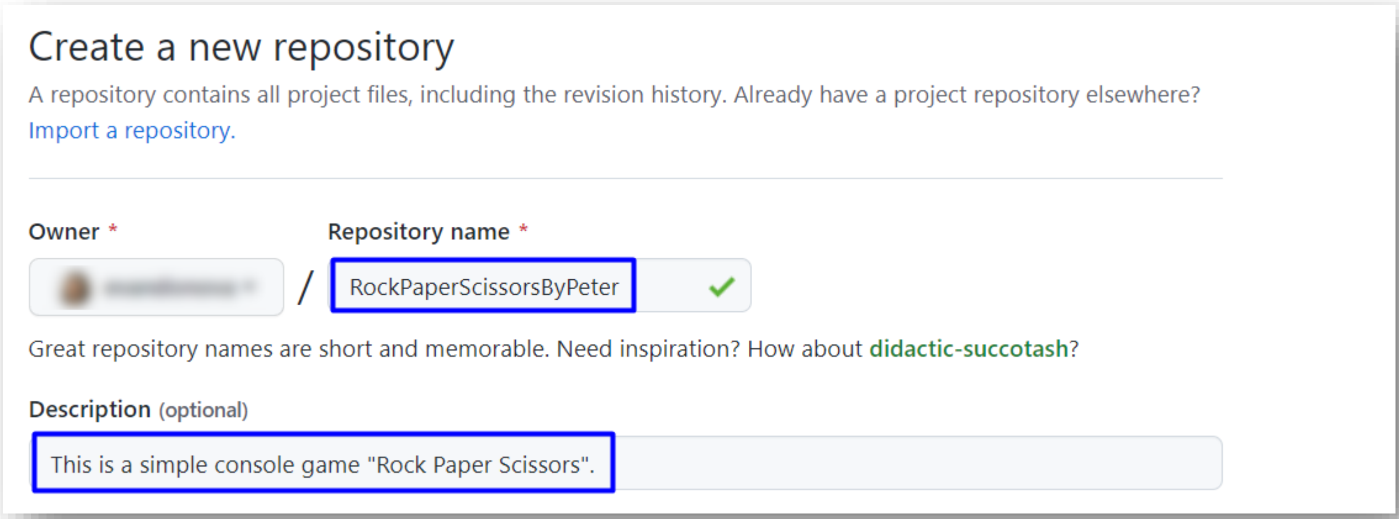
```
Welcome to GitHub!
Let’s begin the adventure

Enter your email
✓ Petar_Ivanov@SoftUni.bg

Create a password
✓ ••••••••

Enter a username
→ PetarIvanov2
```

PetarIvanov2 is available.

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**:


Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * [redacted] / Repository name * RockPaperScissorsByPeter ✓

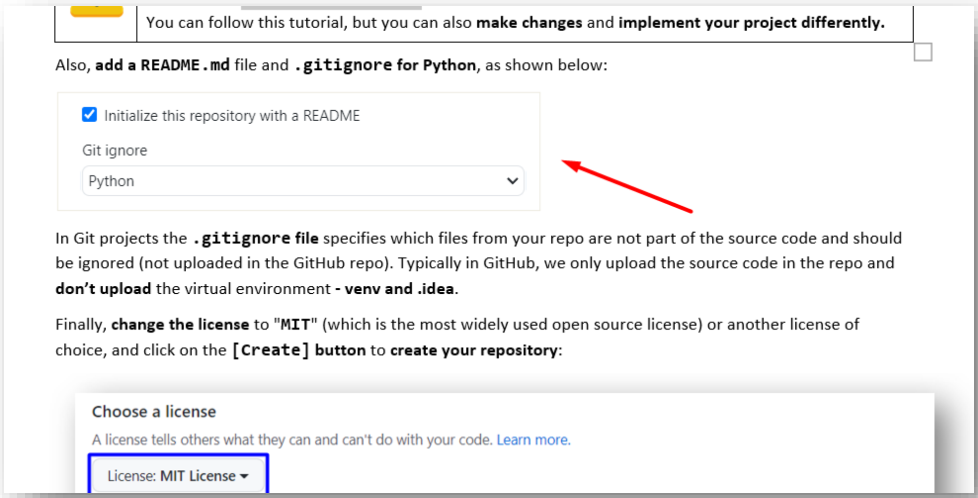
Great repository names are short and memorable. Need inspiration? How about didactic-succotash?

Description (optional)

This is a simple console game "Rock Paper Scissors".

⚠	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:


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:

☑ Initialize this repository with a README

Git ignore: Python

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**:

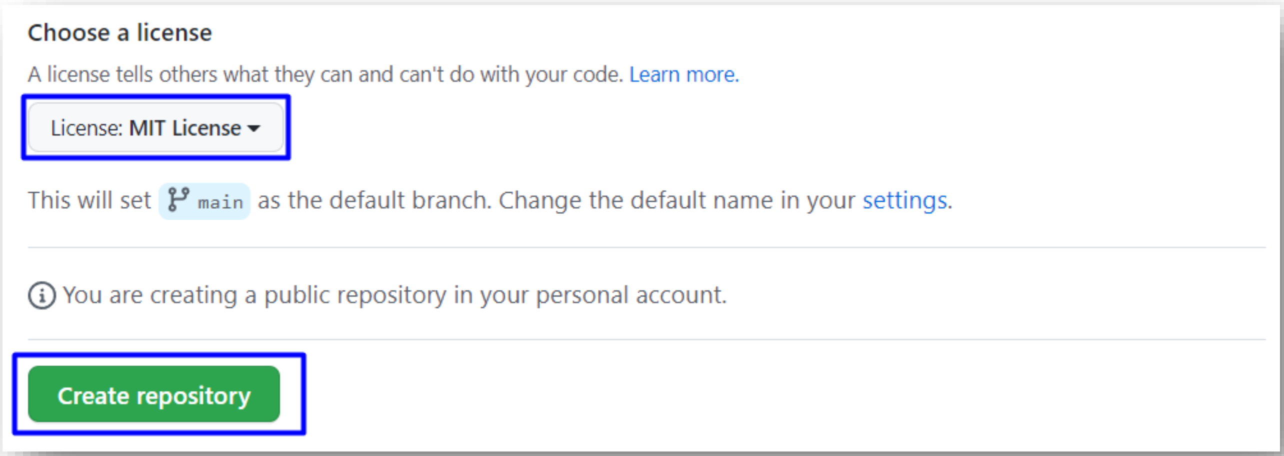
Choose a license

A license tells others what they can and can't do with your code. Learn more.

License: MIT License

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**:


Choose a license

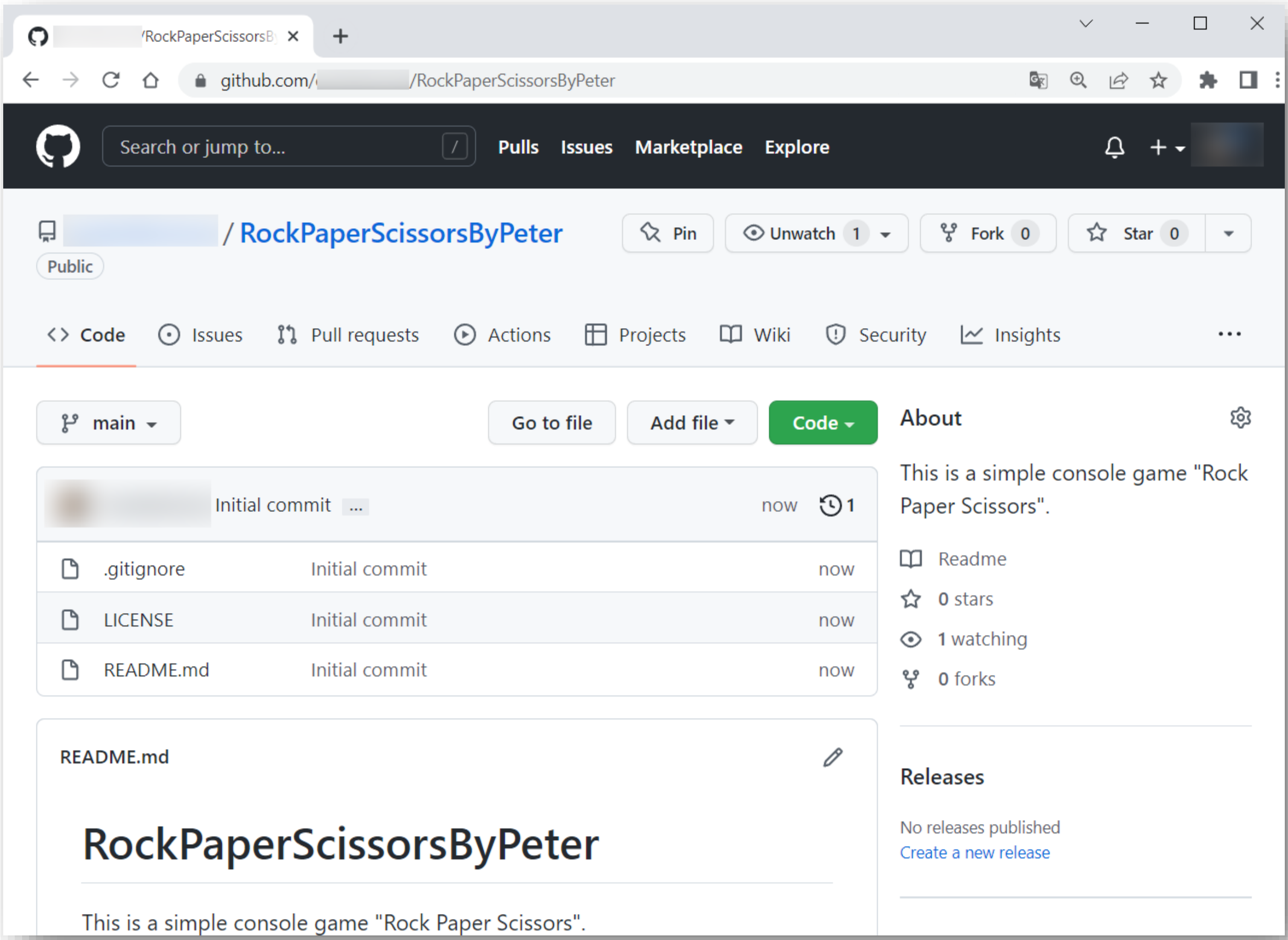
A license tells others what they can and can't do with your code. Learn more.

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.

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



Initial commit ...	now	1
.gitignore	Initial commit	now
LICENSE	Initial commit	now
README.md	Initial commit	now

About

This is a simple console game "Rock Paper Scissors".

Readme
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

README.md

RockPaperScissorsByPeter

This is a simple console game "Rock Paper Scissors".

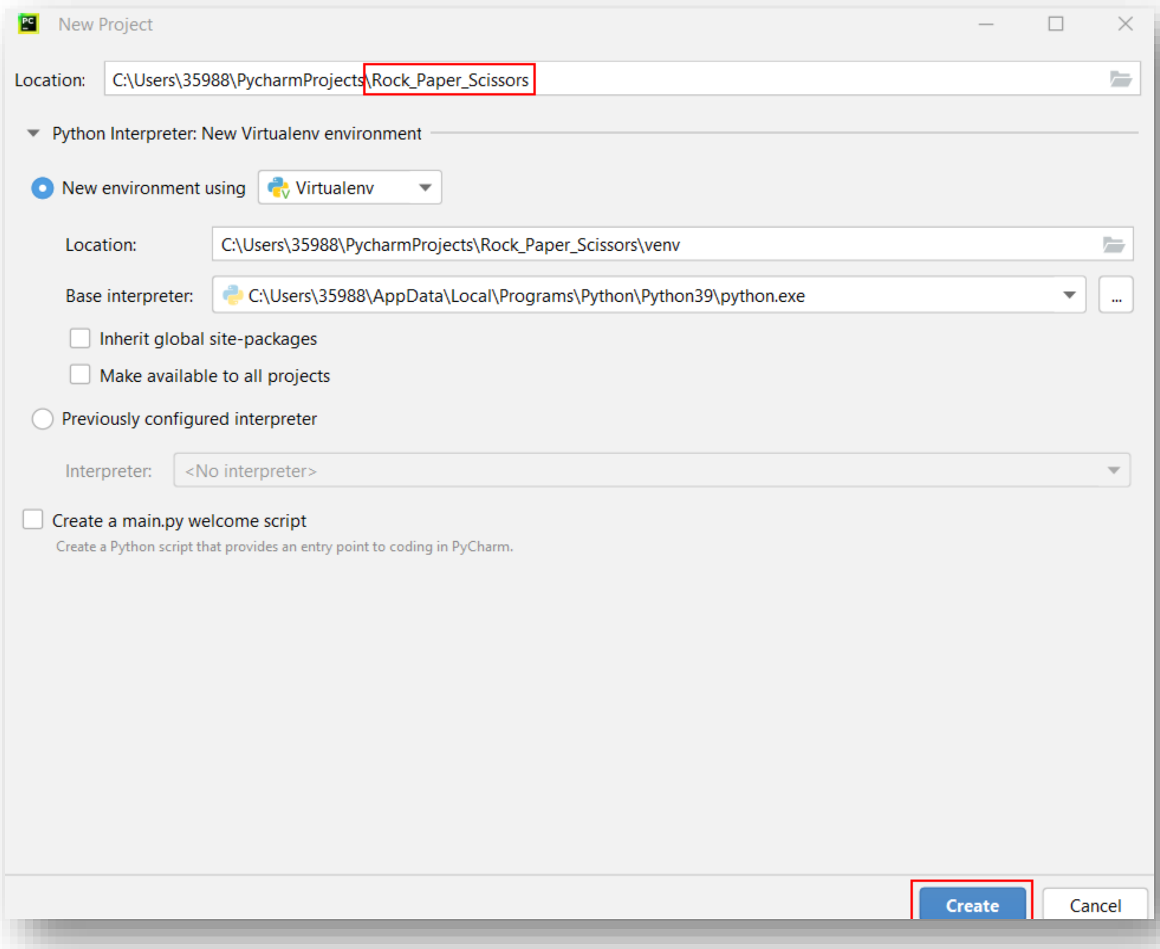
Now let's see how to **write the code** of our game.

2. Write the Game's Code

Let's create the game and play with it.

Create a PyCharm Project

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


New Project

Location: C:\Users\35988\PycharmProjects\Rock_Paper_Scissors

Python Interpreter: New Virtualenv environment

New environment using Virtualenv

Location: C:\Users\35988\PycharmProjects\Rock_Paper_Scissors\venv

Base interpreter: C:\Users\35988\AppData\Local\Programs\Python\Python39\python.exe

Inherit global site-packages

Make available to all projects

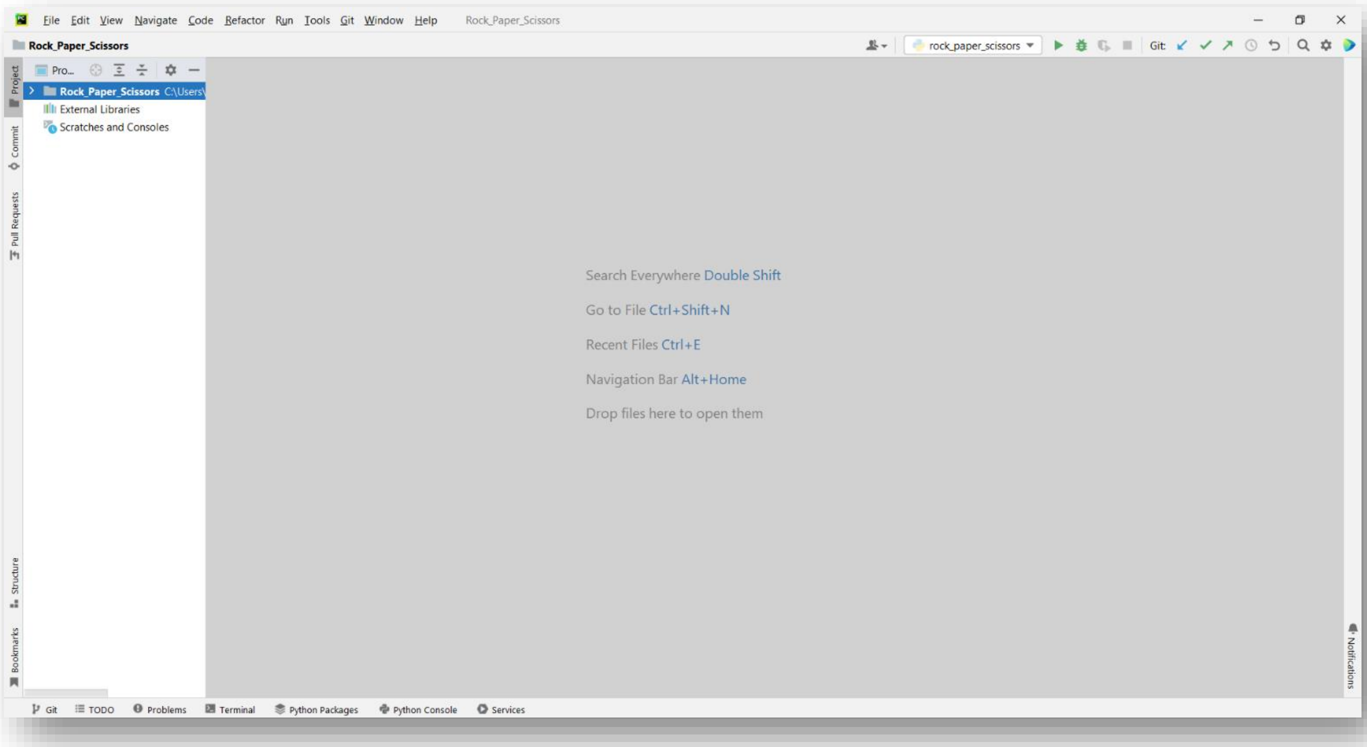
Previously configured interpreter

Interpreter: <No interpreter>

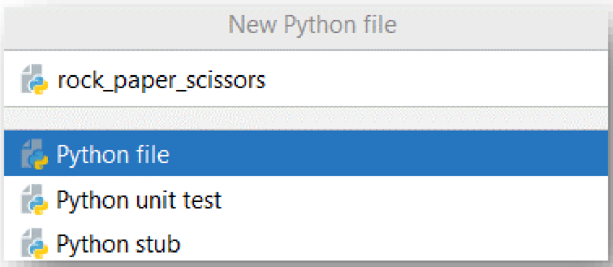
Create a main.py welcome script

Create a Python script that provides an entry point to coding in PyCharm.

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.

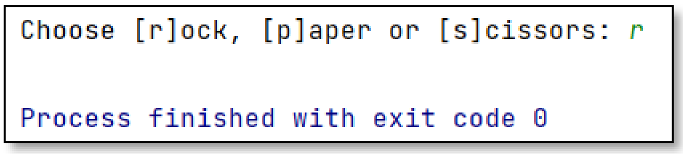
Create **three variables** for our "**Rock**", "**Paper**", and "**Scissors**", which we will use later. They should look like this:

```
rock = 'Rock'
paper = 'Paper'
scissors = 'Scissors'
```

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:

```
player_move = input("Choose [r]ock, [p]aper or [s]cissors: ")
```

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



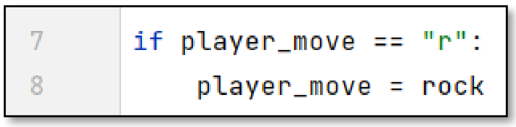
```
Choose [r]ock, [p]aper or [s]cissors: r

Process finished with exit code 0
```

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

Match Player's Move with Possible Options

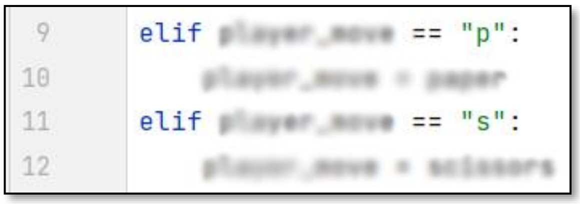
Now it is time to turn the user input into one of our **player's moves options**. To do this, create an **`if-else`** statement with the **possible moves** and change the **`player_move`** variable value with the value it represents.

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



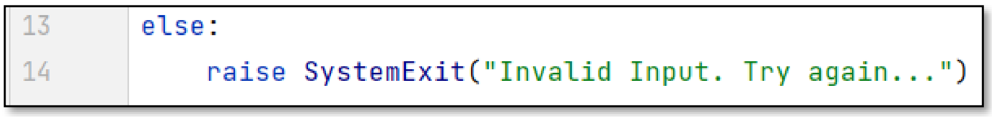
```
if player_move == "r":
    player_move = rock
```

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



```
elif [illegible] == "p":
    [illegible]
elif [illegible] == "s":
    [illegible]
```

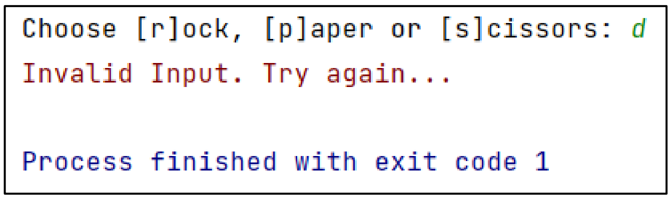
Now we should cover the case, in which the user enters an **invalid value**. To do this, use **`else`** and **`raise SystemExit`** with a message on the console to **stop the program from executing**:



```
else:
    raise SystemExit("Invalid Input. Try again...")
```

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

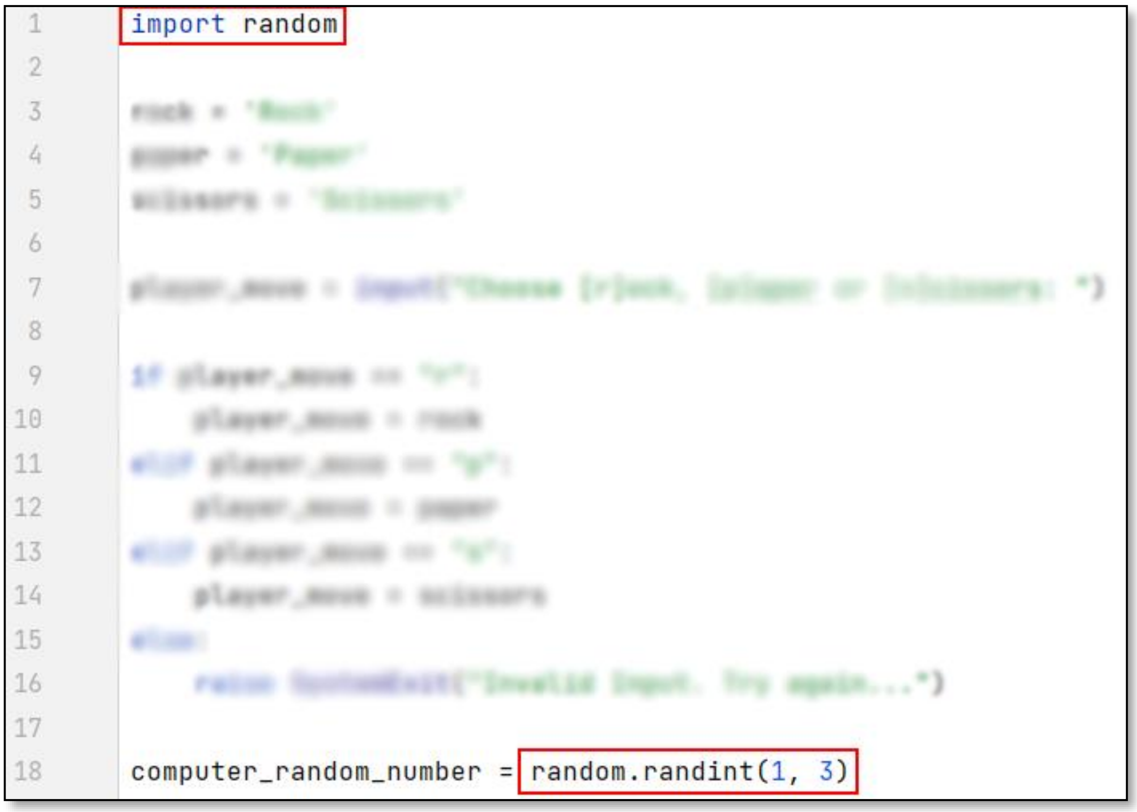
Now let's **run** the app in the **console** and check whether our current code **works properly**, at the moment we have **logic** only for the **incorrect input** so the results should be as follow:



```
Choose [r]ock, [p]aper or [s]cissors: d
Invalid Input. Try again...

Process finished with exit code 1
```

Choose Computer's Move

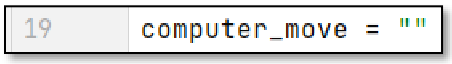
Create a variable of type Random that will help us **choose a random number** using the **`randint`** method. We will use this **number** so that the computer can randomly select from "**`rock`**", "**`paper`**", or "**`scissors`**":



```
import random

[illegible]
[illegible]
[illegible]

[illegible]

[illegible]
    [illegible]
[illegible]
    [illegible]
[illegible]
    [illegible]
[illegible]
    [illegible]

computer_random_number = random.randint(1, 3)
```

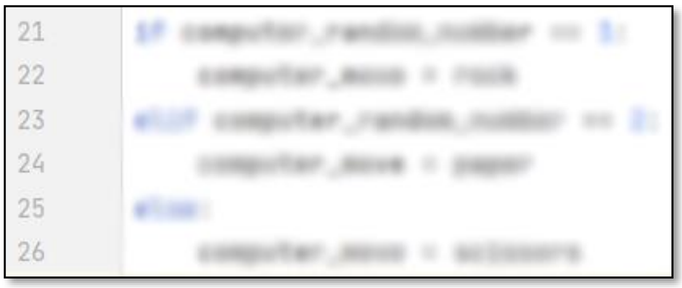
<u>Note: "randint()" is a method from the build-in library - "random" in Python (like the "math" library that you should know from the "Programming Basics" course). "randint()" accepts two parameters, both inclusive, and returns a random number in this range.</u> For more clarification see these examples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

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



```
computer_move = ""
```

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

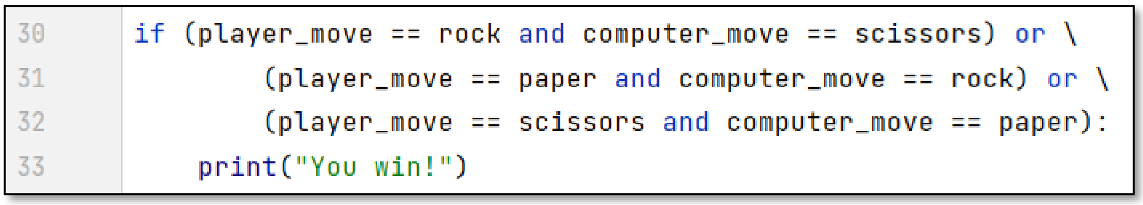


```
[illegible]
    [illegible]
[illegible]
    [illegible]
[illegible]
    [illegible]
```

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**.

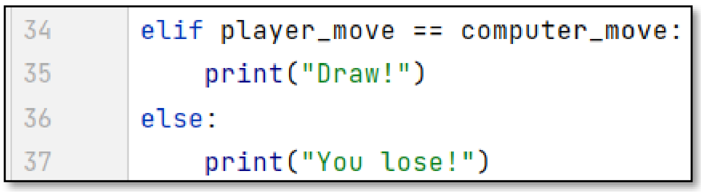


```python
if (player_move == rock and computer_move == scissors) or \
        (player_move == paper and computer_move == rock) or \
        (player_move == scissors and computer_move == paper):
    print("You win!")
```

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

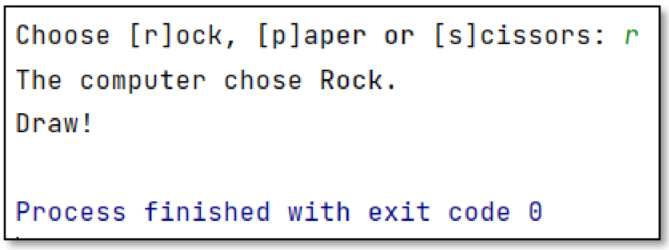
You	Computer	Outcome
rock	rock	Draw
rock	paper	You lose
rock	scissors	You win
paper	rock	You win
paper	paper	Draw
paper	scissors	You lose
scissors	rock	You lose
scissors	paper	You win
scissors	scissors	Draw

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**.

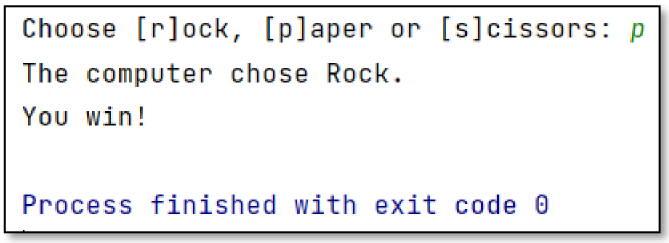


```python
elif player_move == computer_move:
    print("Draw!")
else:
    print("You lose!")
```

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



```
Choose [r]ock, [p]aper or [s]cissors: r
The computer chose Rock.
Draw!

Process finished with exit code 0
```



```
Choose [r]ock, [p]aper or [s]cissors: p
The computer chose Rock.
You win!

Process finished with exit code 0
```

3. Upload Your Project to GitHub

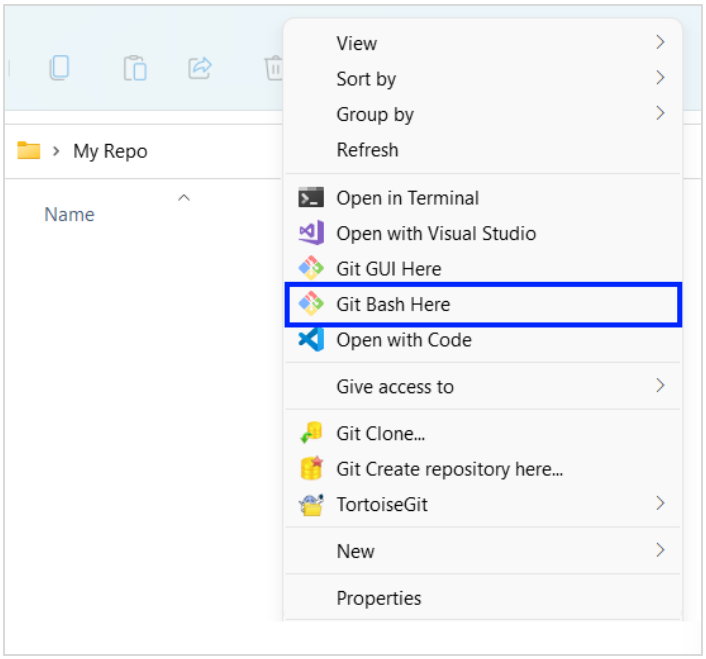
Now we want to deploy our project to **GitHub** so the other developers can see it, and if they want to test it, they can clone it and try it themself on their machine. You have **two options**, choose one and follow the steps.

Use Git Bash (Option 1)

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

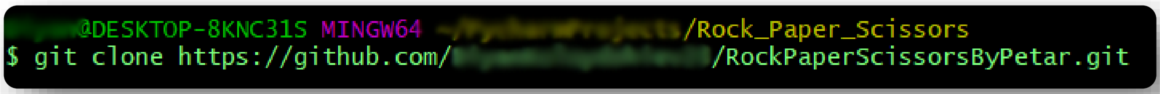
First, if you don't have **Git** on your **computer**, you should **install it** from https://git-scm.com/downloads.

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.



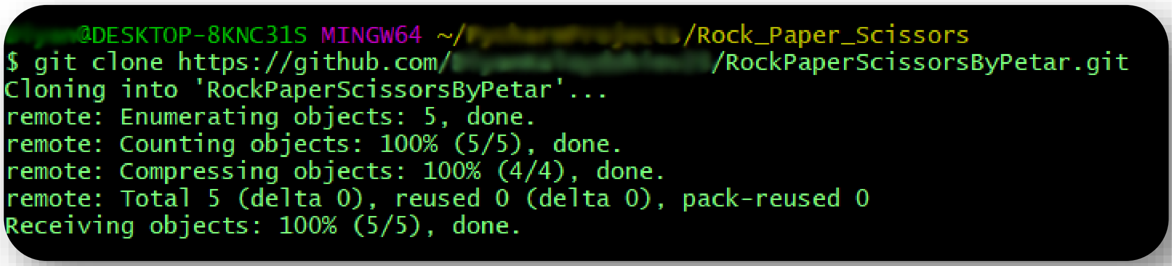
Type the **"git clone"** command followed by the link to your **repository**:

```
git clone
```



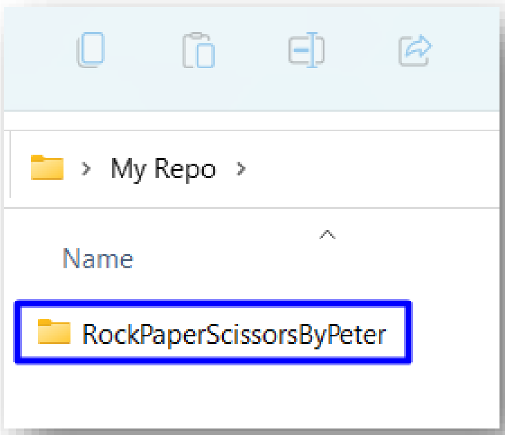
```
[redacted]@DESKTOP-8KNC31S MINGW64 ~/[redacted]/Rock_Paper_Scissors
$ git clone https://github.com/[redacted]/RockPaperScissorsByPetar.git
```

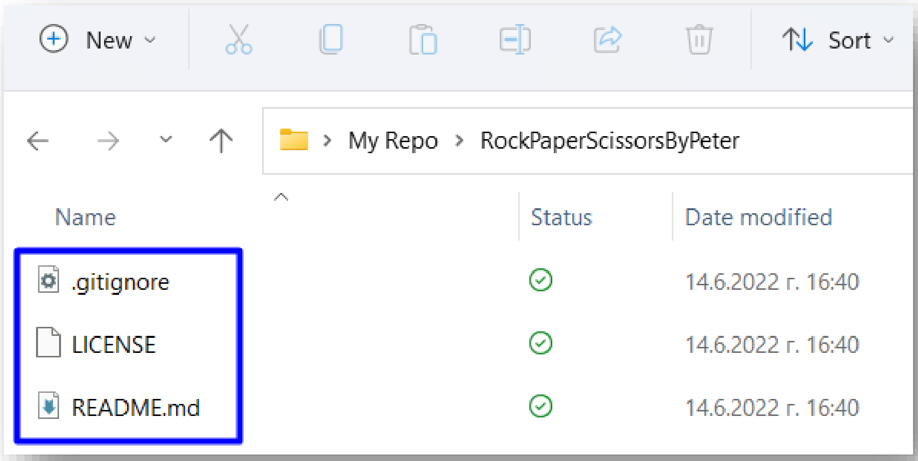
The result should be something like this:



```
[redacted]@DESKTOP-8KNC31S MINGW64 ~/[redacted]/Rock_Paper_Scissors
$ git clone https://github.com/[redacted]/RockPaperScissorsByPetar.git
Cloning into 'RockPaperScissorsByPetar'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

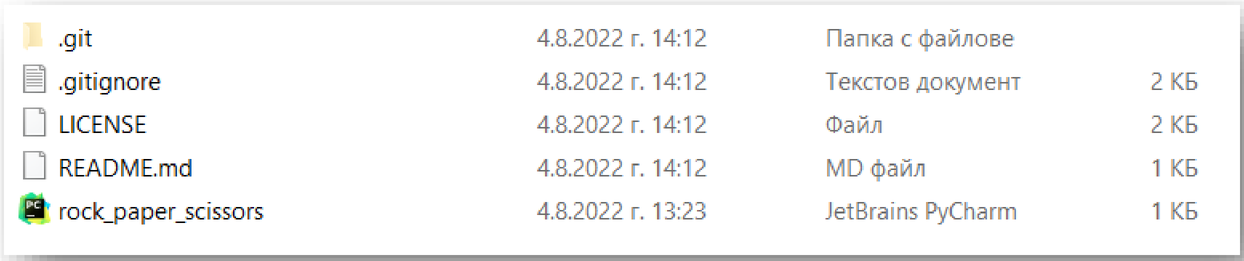
Your files from your GitHub repo will be downloaded to a **sub-folder** called as your project in GitHub, "**RockPaperScissorsByPeter**" in our case.





Name	Status	Date modified
.gitignore	✓	14.6.2022 г. 16:40
LICENSE	✓	14.6.2022 г. 16:40
README.md	✓	14.6.2022 г. 16:40

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



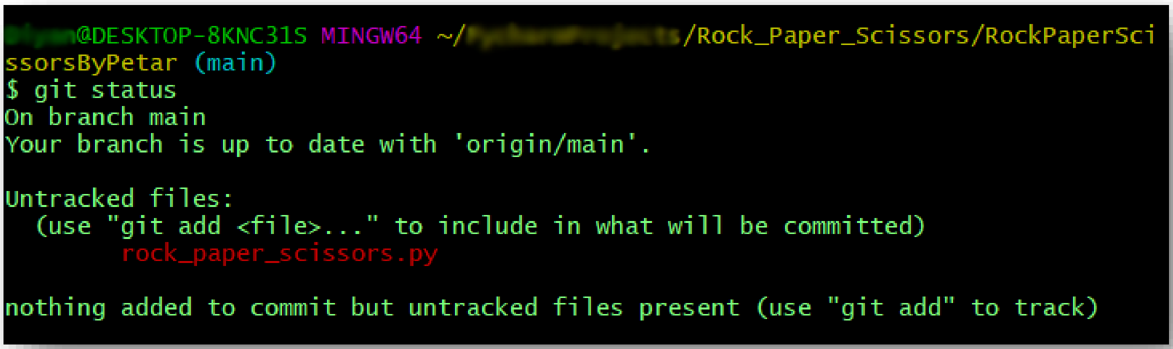
.git	4.8.2022 г. 14:12	Папка с файлове	
.gitignore	4.8.2022 г. 14:12	Текстов документ	2 КБ
LICENSE	4.8.2022 г. 14:12	Файл	2 КБ
README.md	4.8.2022 г. 14:12	MD файл	1 КБ
rock_paper_scissors	4.8.2022 г. 13:23	JetBrains PyCharm	1 КБ

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**.

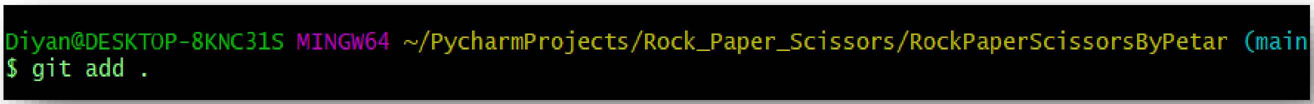


```
[redacted]@DESKTOP-8KNC31S MINGW64 ~/[redacted]/Rock_Paper_Scissors/RockPaperSci
ssorsByPetar (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

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

Now type:

```
git add .
```

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

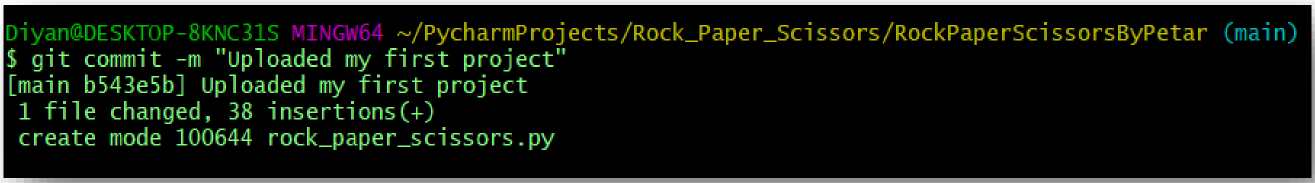


```
Diyan@DESKTOP-8KNC31S MINGW64 ~/PycharmProjects/Rock_Paper_Scissors/RockPaperScissorsByPetar (main
$ git add .
```

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**.

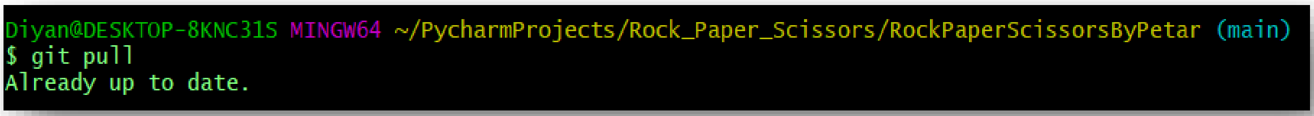


```
Diyan@DESKTOP-8KNC31S MINGW64 ~/PycharmProjects/Rock_Paper_Scissors/RockPaperScissorsByPetar (main)
$ git commit -m "Uploaded my first project"
[main b543e5b] Uploaded my first project
 1 file changed, 38 insertions(+)
 create mode 100644 rock_paper_scissors.py
```

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.

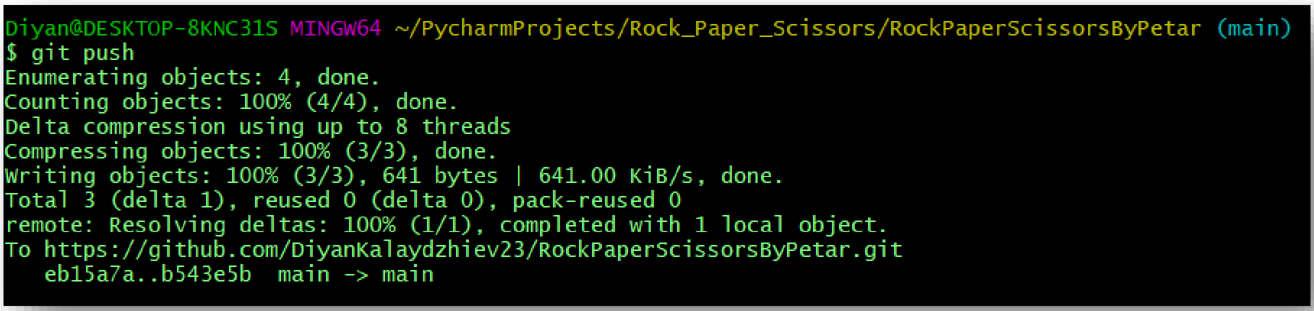


```
Diyan@DESKTOP-8KNC31S MINGW64 ~/PycharmProjects/Rock_Paper_Scissors/RockPaperScissorsByPetar (main)
$ git pull
Already up to date.
```

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**.



```
Diyan@DESKTOP-8KNC31S MINGW64 ~/PycharmProjects/Rock_Paper_Scissors/RockPaperScissorsByPetar (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), 641 bytes | 641.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/DiyanKalaydzhiev23/RockPaperScissorsByPetar.git
   eb15a7a..b543e5b  main -> main
```

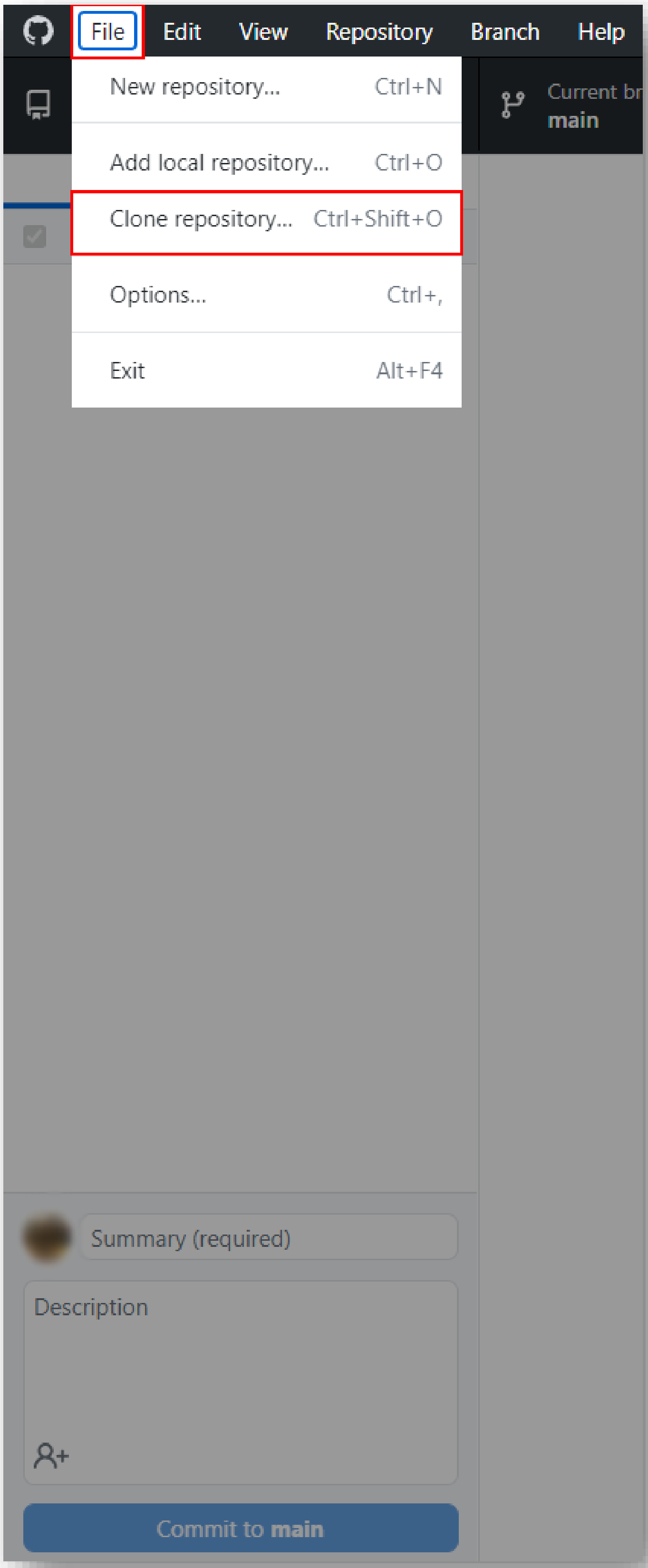
This is all you need to **update** your **repository** using **`Git Bash`**.

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

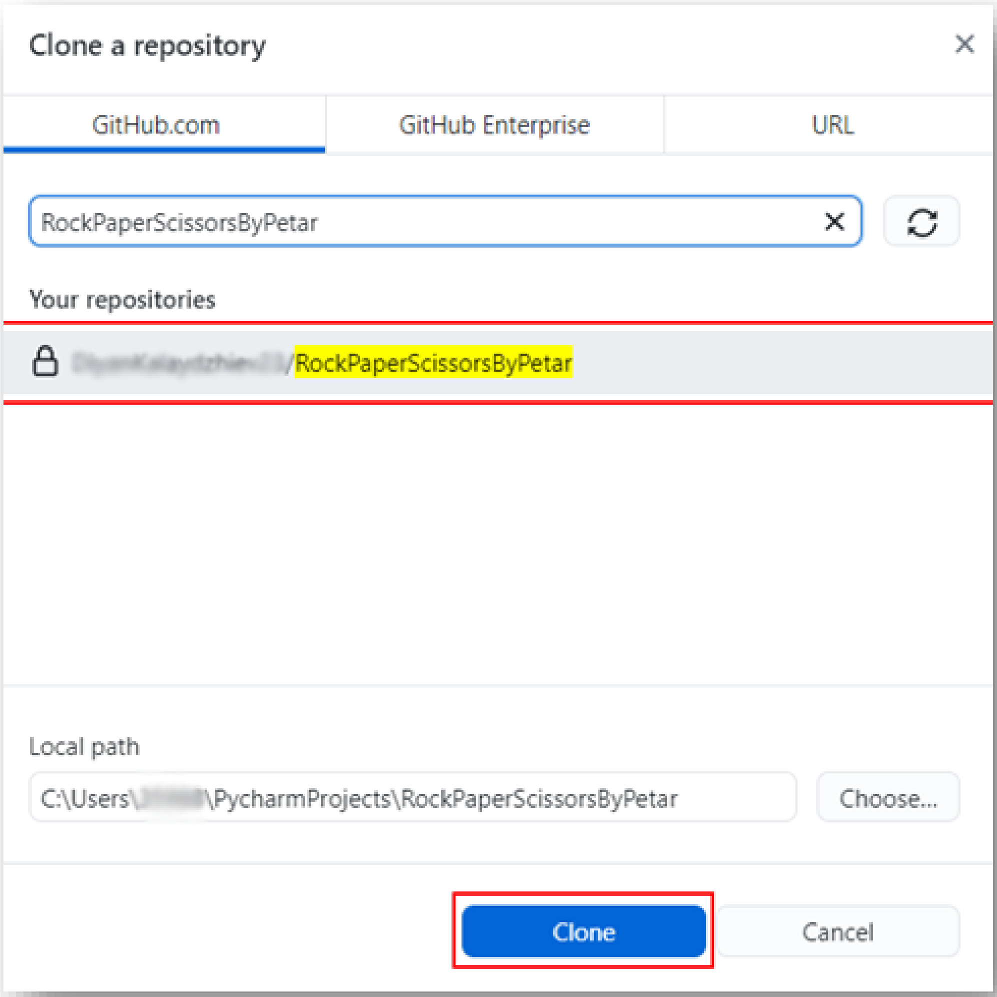
Use GitHub Desktop (Option 2)

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

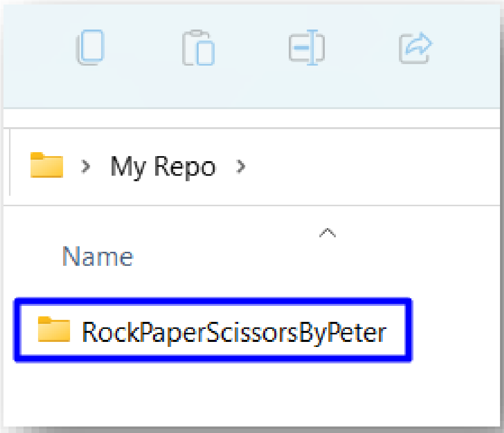
Go to **"File"** and chose **"Clone repository".**

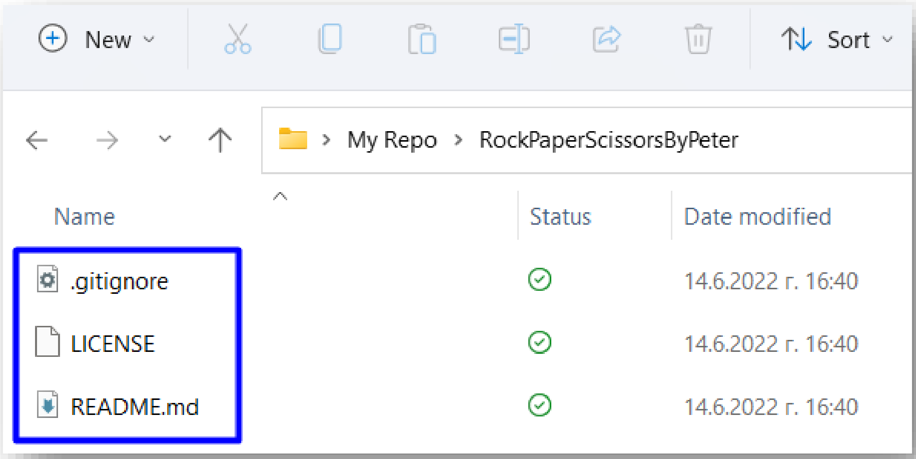


Chose the repository for the project, in our case "RockPaperScissorsByPetar" and hit the **"Clone"** button**.**



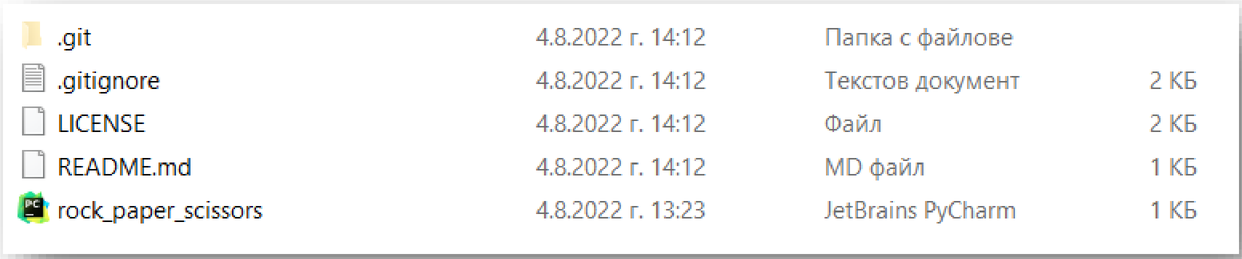
Your files from your GitHub repo will be downloaded to a **sub-folder** called as your project in GitHub, "**RockPaperScissorsByPeter**" in our case:





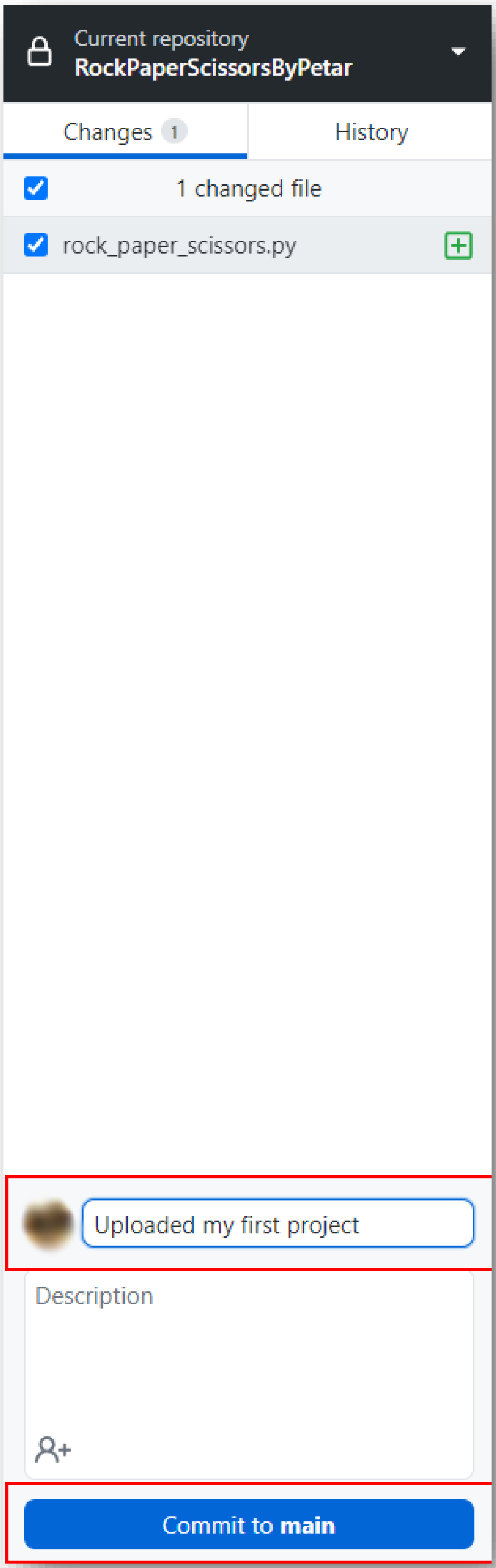
Name	Status	Date modified
.gitignore	✓	14.6.2022 г. 16:40
LICENSE	✓	14.6.2022 г. 16:40
README.md	✓	14.6.2022 г. 16:40

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



.git	4.8.2022 г. 14:12	Папка с файлове	
.gitignore	4.8.2022 г. 14:12	Текстов документ	2 КБ
LICENSE	4.8.2022 г. 14:12	Файл	2 КБ
README.md	4.8.2022 г. 14:12	MD файл	1 КБ
rock_paper_scissors	4.8.2022 г. 13:23	JetBrains PyCharm	1 КБ

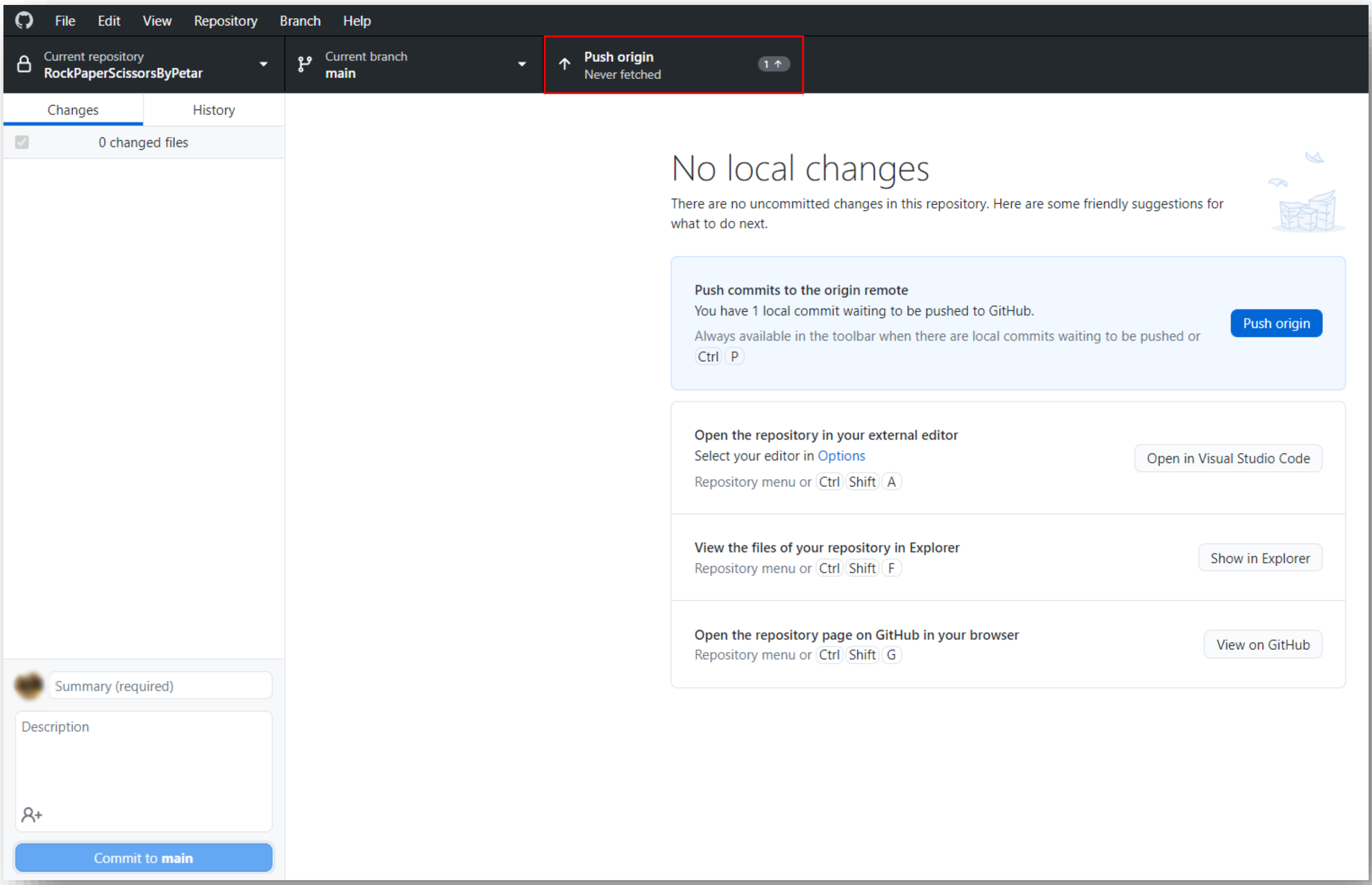
After that go to GitHub Desktop and **create a commit**, just like this.


Current repository
RockPaperScissorsByPetar

1 changed file
rock_paper_scissors.py

Uploaded my first project

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



No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Push commits to the origin remote
You have 1 local commit waiting to be pushed to GitHub.
Always available in the toolbar when there are local commits waiting to be pushed or Ctrl P

Open the repository in your external editor
Select your editor in Options
Repository menu or Ctrl Shift A

View the files of your repository in Explorer
Repository menu or Ctrl Shift F

Open the repository page on GitHub in your browser
Repository menu or Ctrl Shift G

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

4. *Modify the Code, Write Your Own Features

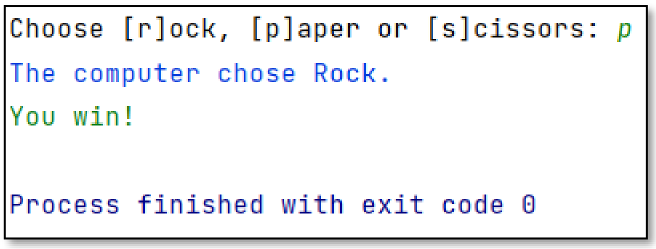
⚠	This is your own project. **Be unique**. Don't be a copy/paster! • Implement your **own features**. • **Implement the code yourself**, using your own coding style, code formatting, comments, etc. • Make the project **more interesting**. Learn by playing with the code and adding your own changes.

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

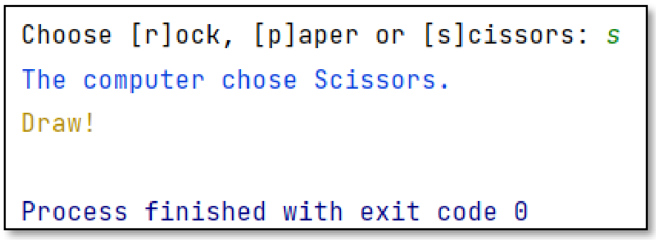
Add Colors

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

```
Choose [r]ock, [p]aper or [s]cissors: r
The computer chose Paper.
You lose!

Process finished with exit code 0
```



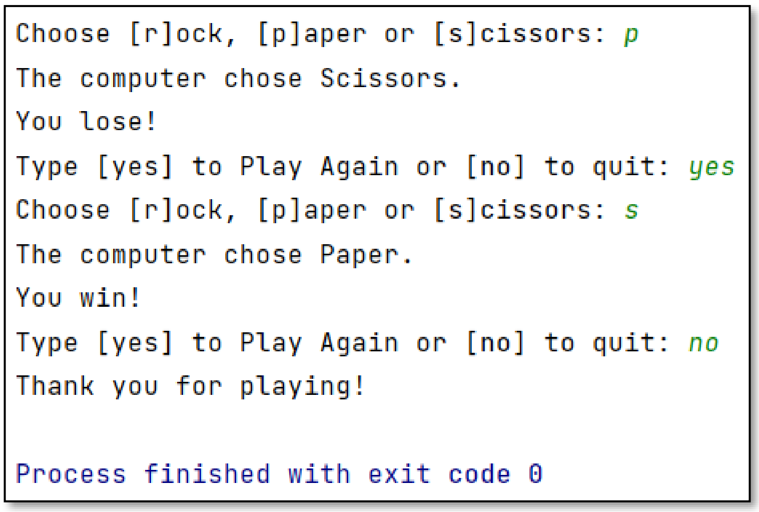
```
Choose [r]ock, [p]aper or [s]cissors: p
The computer chose Rock.
You win!

Process finished with exit code 0
```



```
Choose [r]ock, [p]aper or [s]cissors: s
The computer chose Scissors.
Draw!

Process finished with exit code 0
```

Restart the Game

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



```
Choose [r]ock, [p]aper or [s]cissors: p
The computer chose Scissors.
You lose!
Type [yes] to Play Again or [no] to quit: yes
Choose [r]ock, [p]aper or [s]cissors: s
The computer chose Paper.
You win!
Type [yes] to Play Again or [no] to quit: no
Thank you for playing!

Process finished with exit code 0
```

Scoring System

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

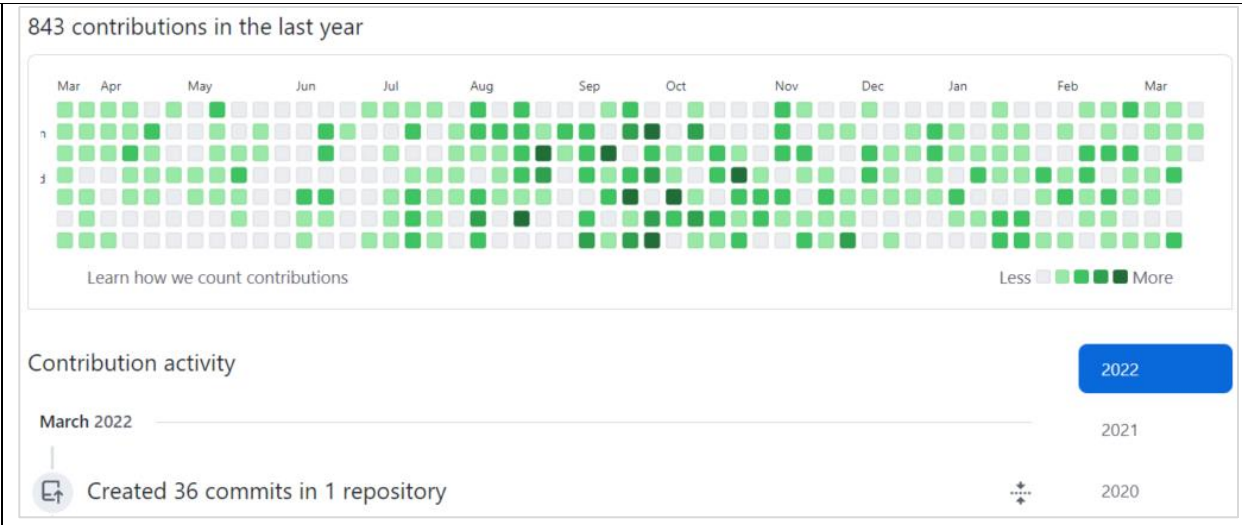
Additional Ideas

- 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!

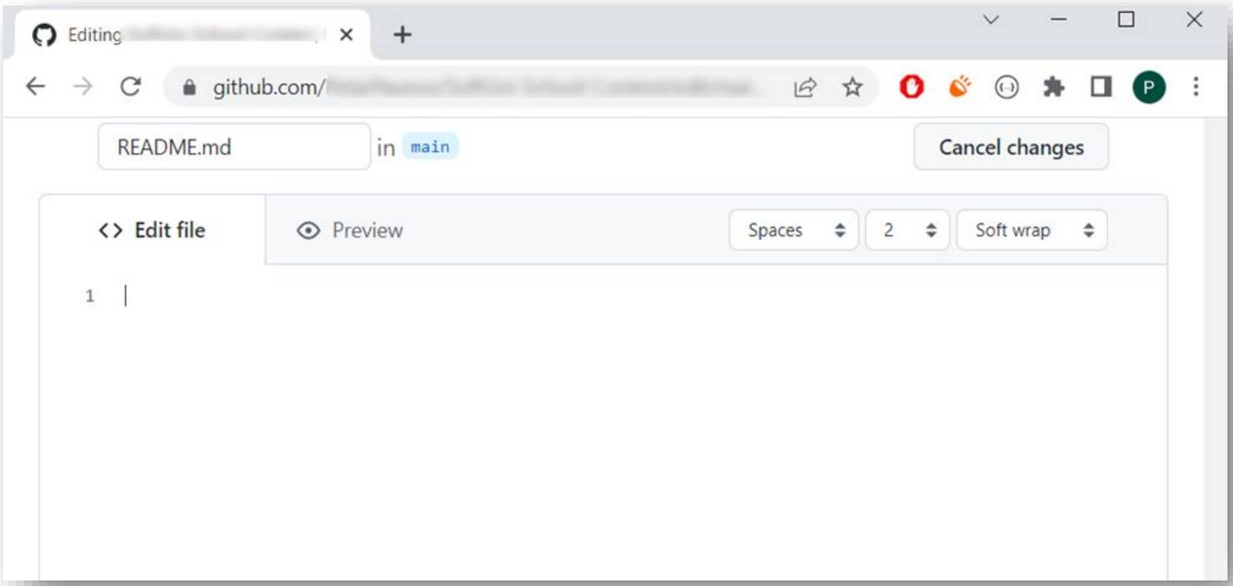
!	It is very important to **commit frequently** your code to GitHub. This way you create a **rich commit history** for your project and your **GitHub contribution graph** is growing:


843 contributions in the last year

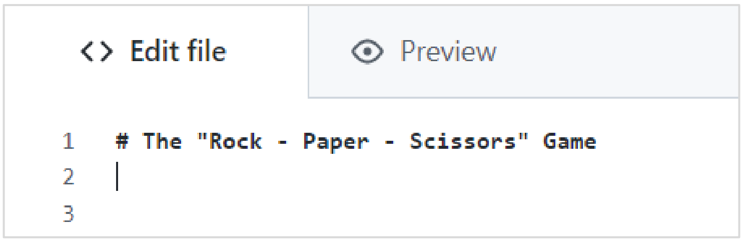
Created 36 commits in 1 repository

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:

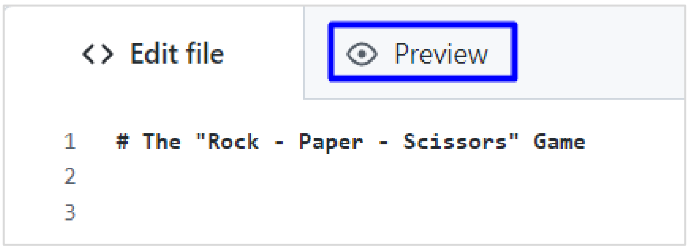


Add a project name. Use **"#"** in front of the text to indicate the **title**:

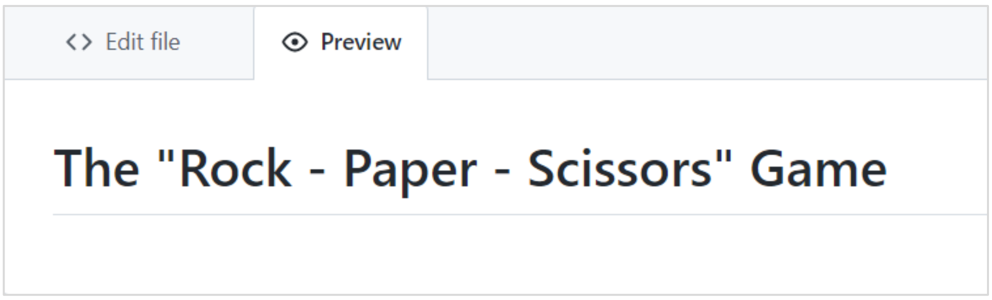


```
# The "Rock - Paper - Scissors" Game
```

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



```
# The "Rock - Paper - Scissors" Game
```



The "Rock - Paper - Scissors" Game

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.

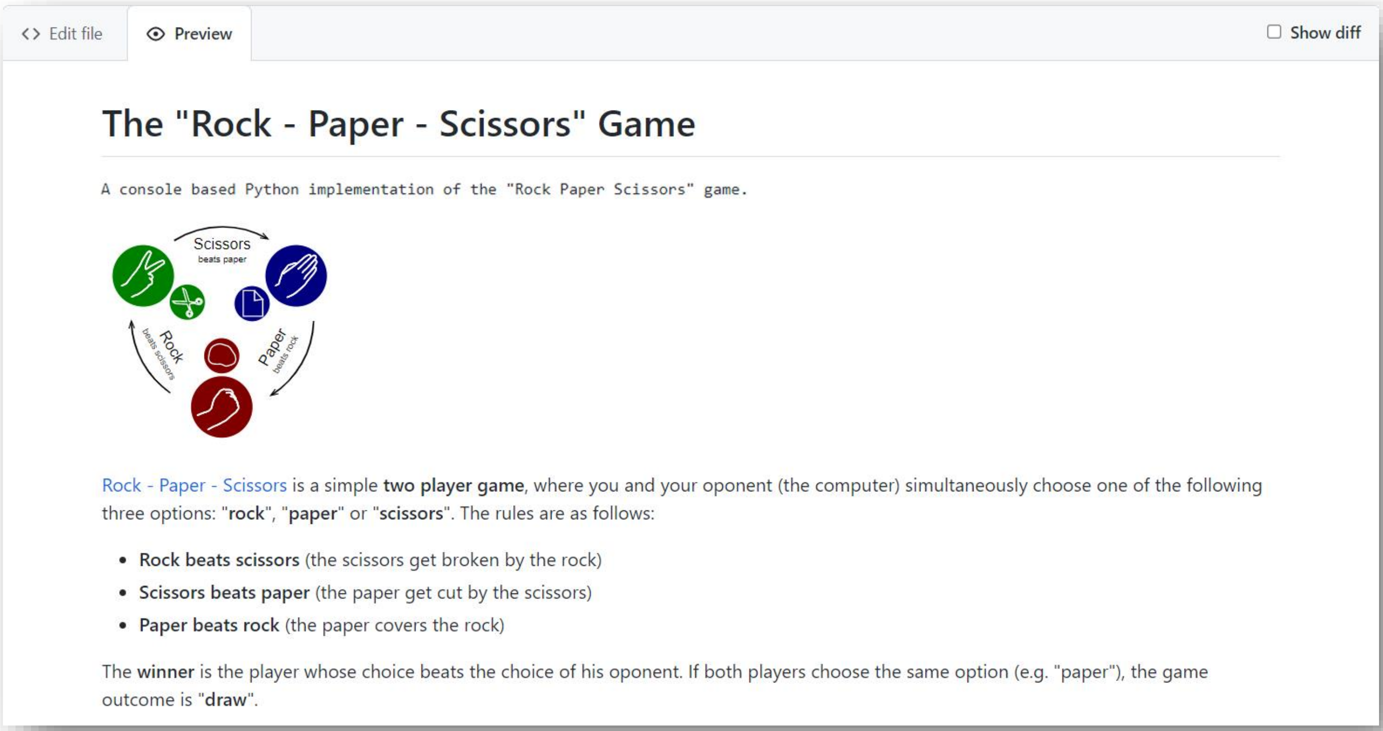
You can learn more about **Markdown** here: https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax.

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!



The "Rock - Paper - Scissors" Game

`A console based Python implementation of the "Rock Paper Scissors" game.`

Rock - Paper - Scissors is a simple **two player game**, where you and your oponent (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 get cut by the scissors)
- **Paper beats rock** (the paper covers the rock)

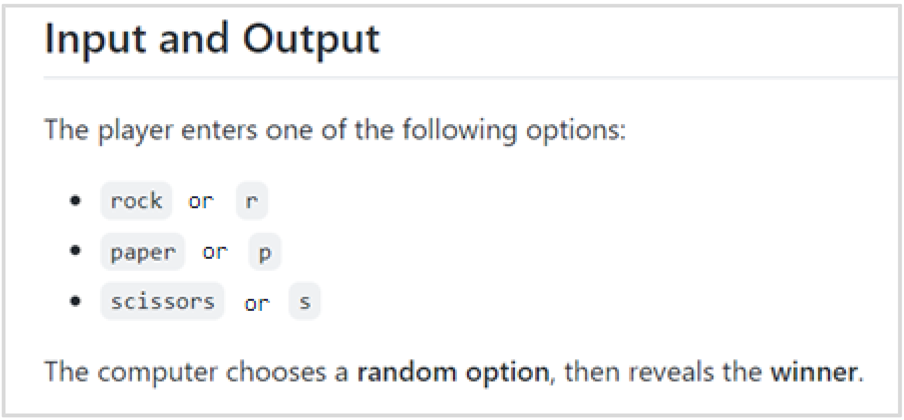
The **winner** is the player whose choice beats the choice of his oponent. If both players choose the same option (e.g. "paper"), the game outcome is "**draw**".

!	**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:

```
<img alt="Image" width="200px" src="[redacted]" />
```

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



Input and Output

The player enters one of the following options:

- `rock` or `r`
- `paper` or `p`
- `scissors` or `s`

The computer chooses a **random option**, then reveals the **winner**.

Your Solution

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

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

You	Computer	Outcome
rock	rock	Draw
rock	paper	You lose
rock	scissors	You win
paper	rock	You win
paper	paper	Draw
paper	scissors	You lose
scissors	rock	You lose
scissors	paper	You win
scissors	scissors	Draw

Link to the Source Code

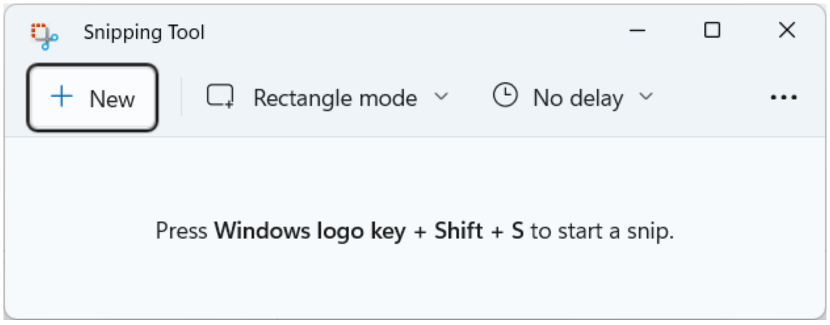
Add a **link** to your **source code** as follows:

```
[Source Code](rock_paper_scissors.py)
```

Screenshots

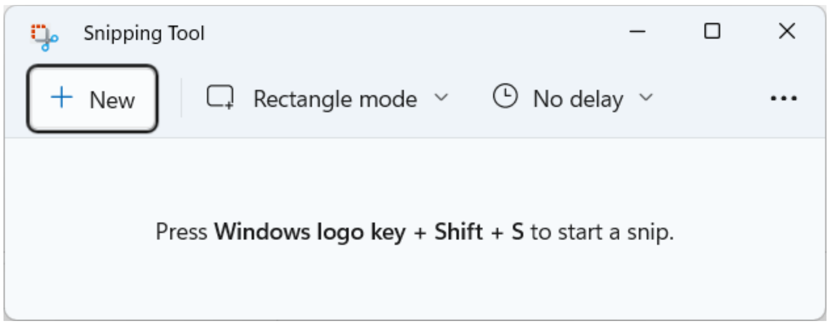
Add **screenshots** of your project:

1. **Take a screenshot** with your favorite tool (e.g., the Snipping Tool in Windows).



Press **Windows logo key** + **Shift** + **S** to start a snip.

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



Press **Windows logo key** + **Shift** + **S** to start a snip.

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

Example screenshot

for the "**`Rock Paper Scissors`**" game:

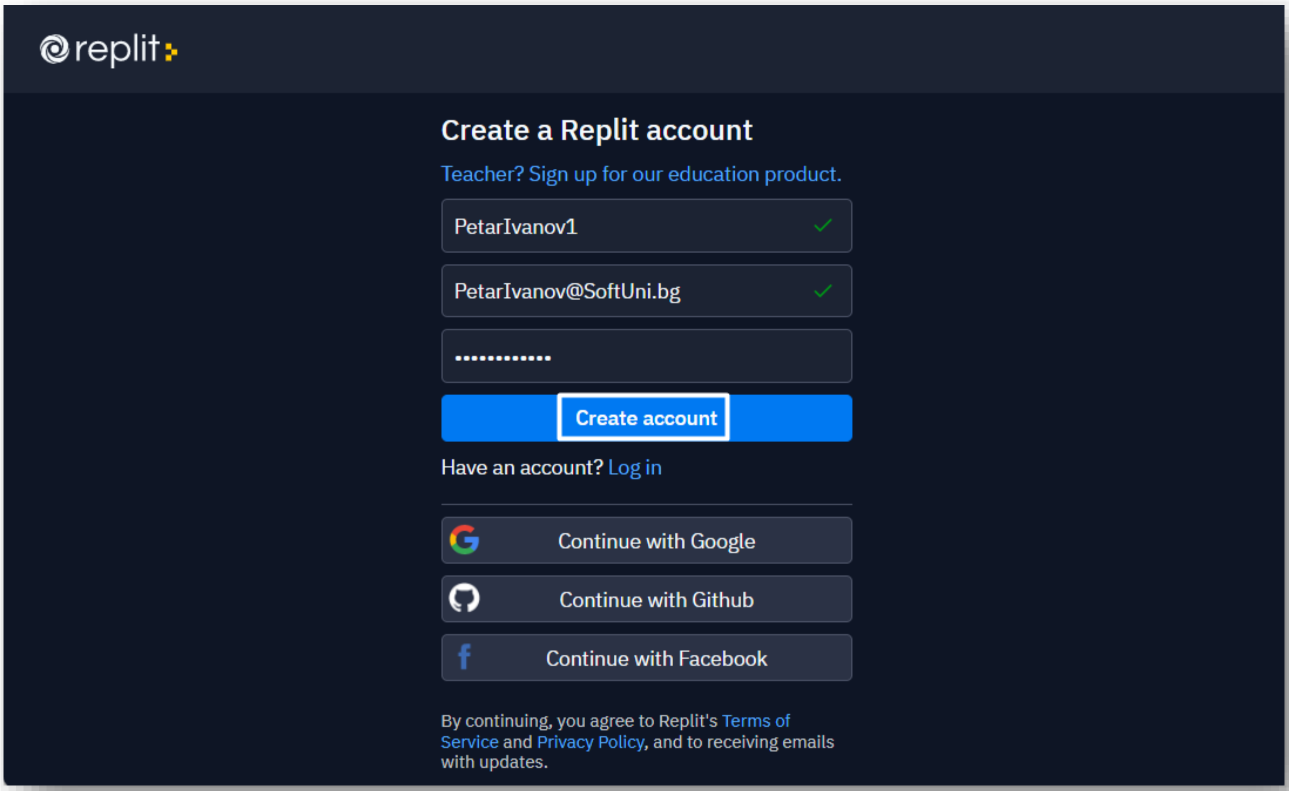
```
Choose [r]ock, [p]aper or [s]cissors: p
The computer chose Scissors.
You lose!
Type [yes] to Play Again or [no] to quit: yes
Choose [r]ock, [p]aper or [s]cissors: s
The computer chose Paper.
You win!
Type [yes] to Play Again or [no] to quit: no
Thank you for playing!

Process finished with exit code 0
```

6. Upload Your App to Replit

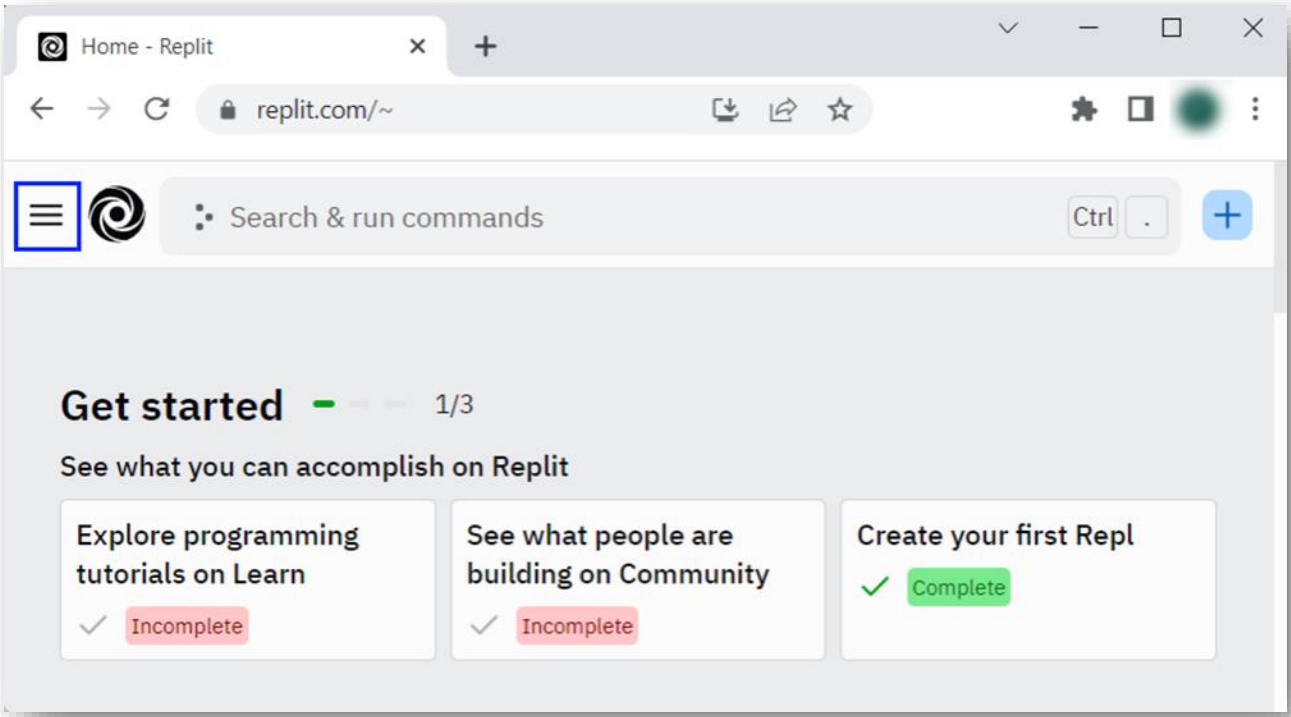
`Replit` is an online coding environment (online IDE) which allows you to **write** software projects, **share** them through a simple link, and **run** your projects directly in the Web browser. We shall upload our project in **`Replit`** to allow the users to **run and interact with the project** with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

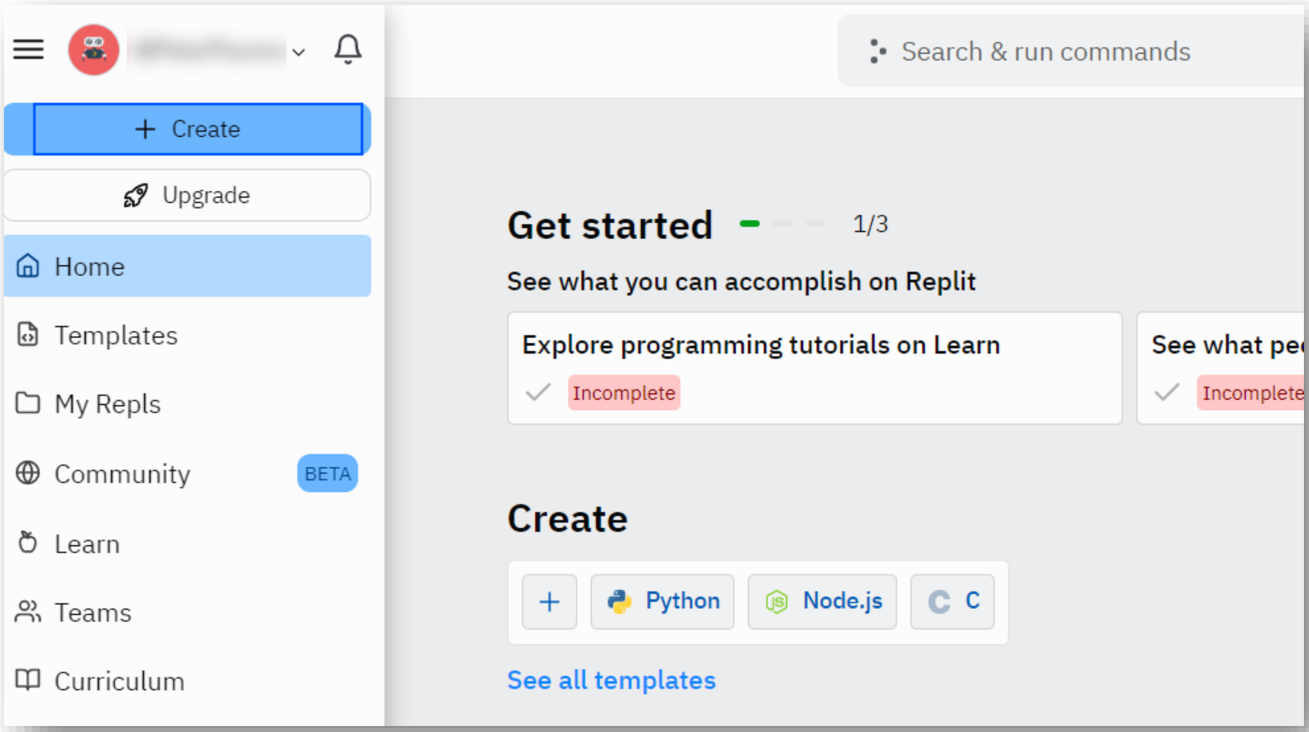
replit
Features
Blog
Pricing
Jam
Teams Pro
Teams for Education
Careers
Log in
Sign up
Code, create, and learn together
Use our free, collaborative, in-browser IDE to code in 50+ languages — without spending a second on setup.
<> Start coding



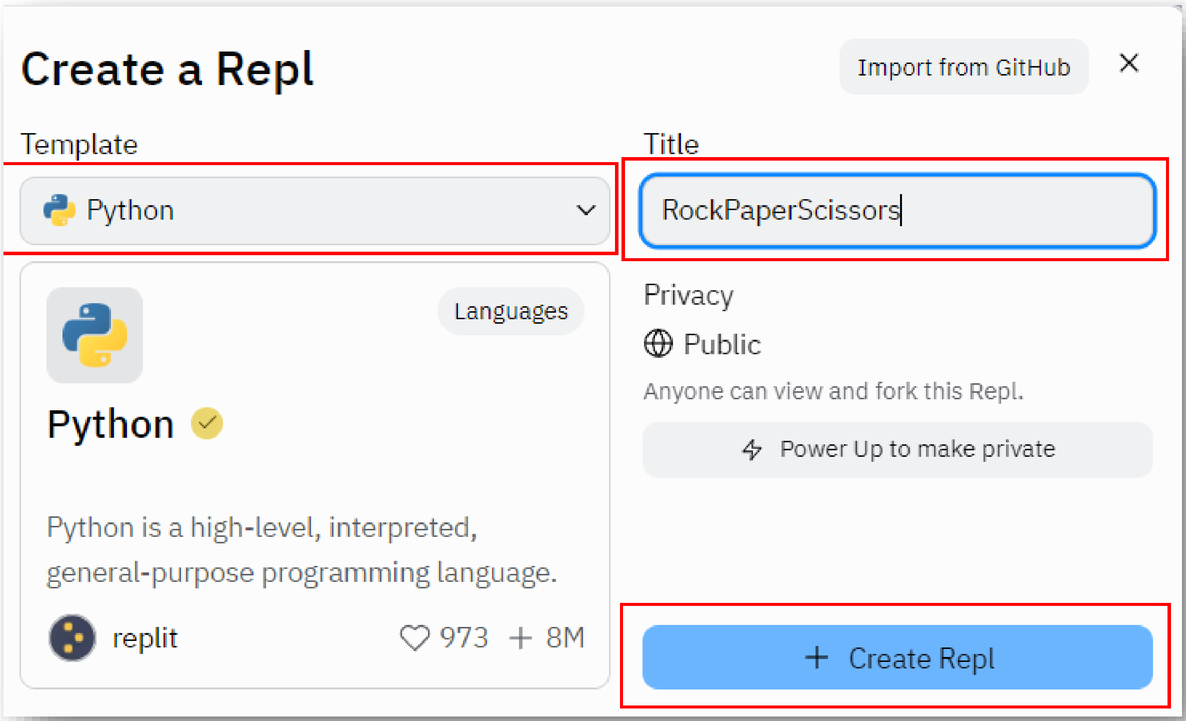
Create a **new project** in **`Replit`**; open the **menu** in the upper **left corner**.



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

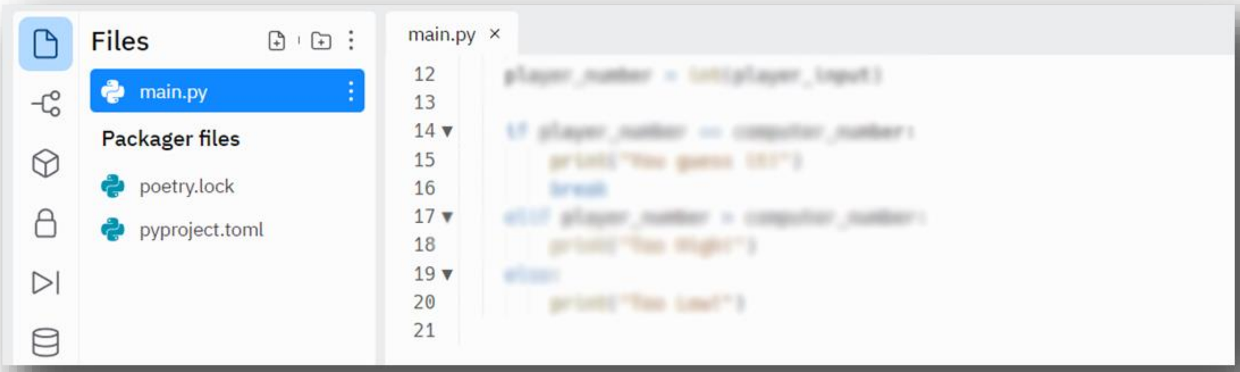


Chose "**Python**" for your project.

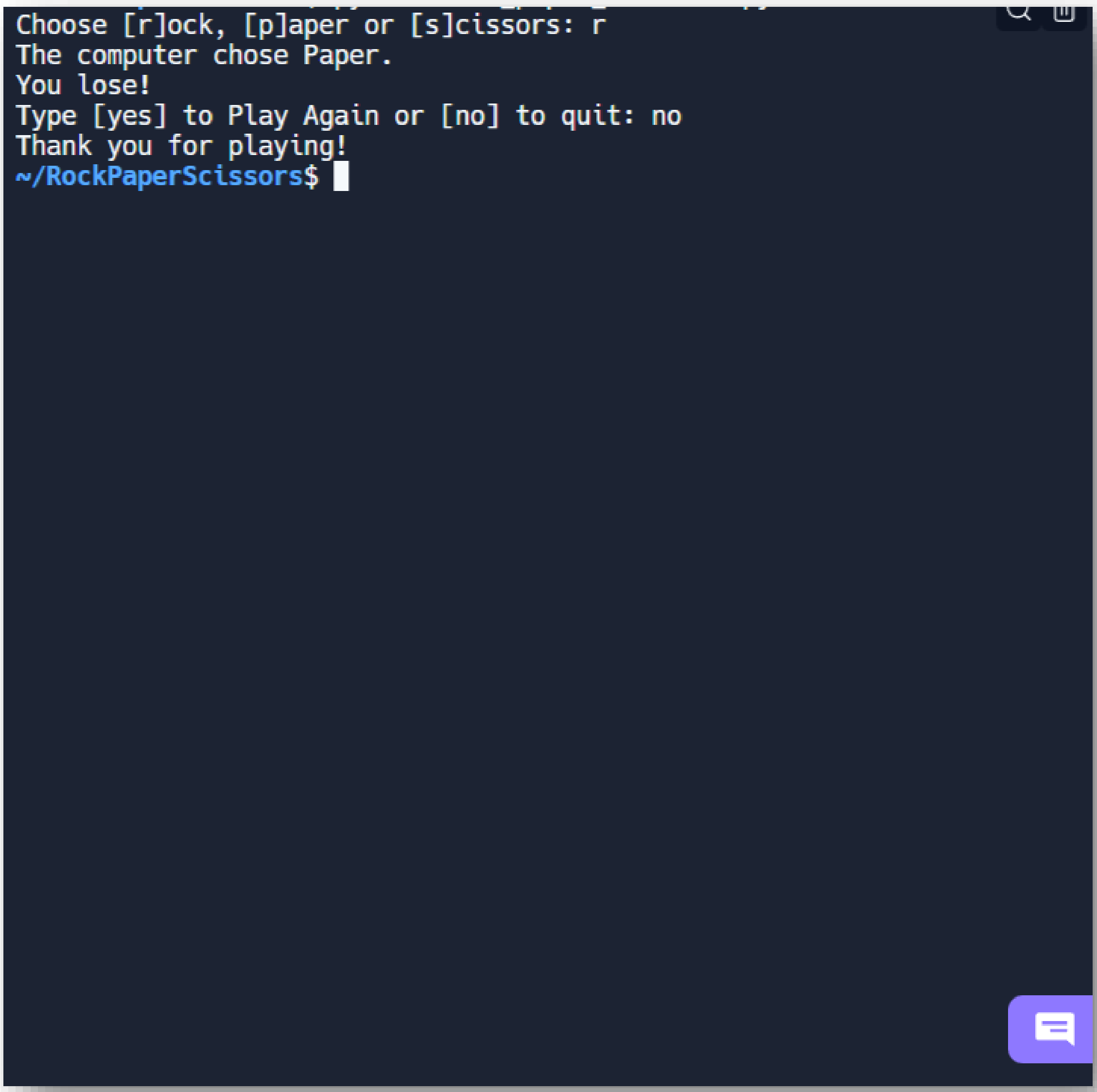


Add a meaningful **name** to your **`Replit`** project, e.g., "**RockPaperScissors**".

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



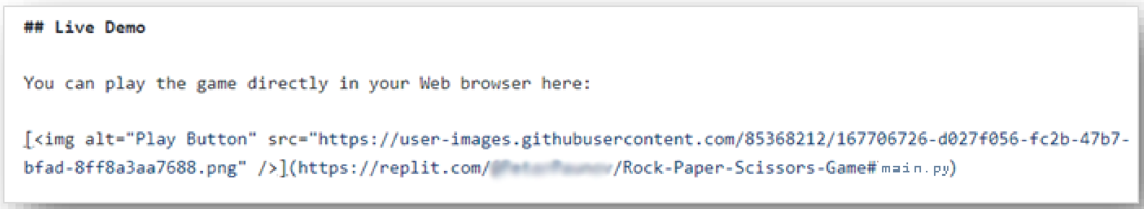
Click `[Run]` and enjoy your console application directly in the Web browser:



```
Choose [r]ock, [p]aper or [s]cissors: r
The computer chose Paper.
You lose!
Type [yes] to Play Again or [no] to quit: no
Thank you for playing!
~/RockPaperScissors$
```

7. Add Replit Link to Your README.md

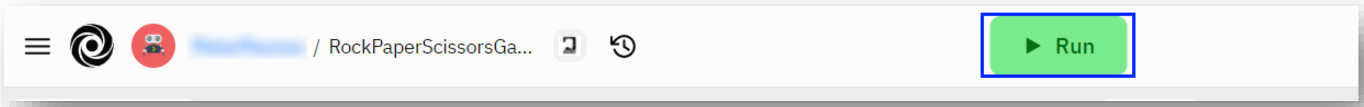
Now add a "**one-click live demo**" of your project from your GitHub project documentation. You can do it as follows:

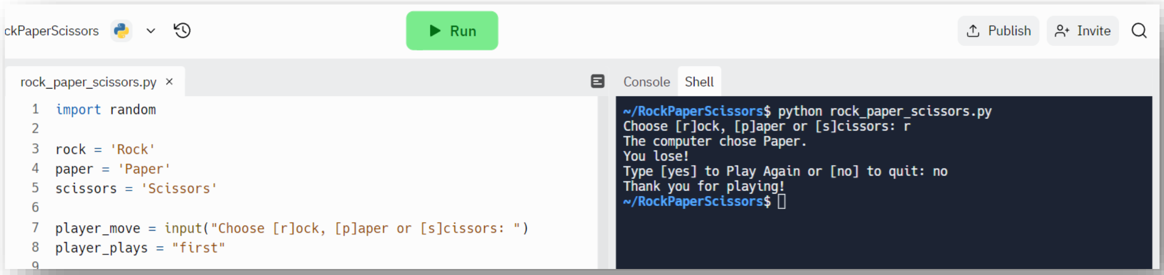


```
## Live Demo

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

[<img alt="Play Button" src="https://user-images.githubusercontent.com/85368212/167706726-d027f056-fc2b-47b7-bfad-8ff8a3aa7688.png" />](https://replit.com/[redacted]/Rock-Paper-Scissors-Game#main.py)
```

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`**.





```
import random

rock = 'Rock'
paper = 'Paper'
scissors = 'Scissors'

player_move = input("Choose [r]ock, [p]aper or [s]cissors: ")
player_plays = "first"
```

```
~/RockPaperScissors$ python rock_paper_scissors.py
Choose [r]ock, [p]aper or [s]cissors: r
The computer chose Paper.
You lose!
Type [yes] to Play Again or [no] to quit: no
Thank you for playing!
~/RockPaperScissors$
```

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