main.py

CODE WITH REPL.IT

FROM BEGINNER TO EXPERT
THROUGH GUIDED TUTORIALS

repl.it

CONTENT BY RITZA

Programming walkthroughs: Coding with Python and Repl.it

Ritza

Extract Tutorial 1
Setting Up an Account and First Steps

Contents

Welcome to Code with Repl.it

In a series of tutorials, you'll go from beginner to expert in coding with Repl.it. While these lessons are designed to be taken in order, they each make sense on their own too, so feel free to jump in wherever looks the most interesting to you.

Part 1 covers the basics of using Repl.it: how to create projects, work with files, use third party dependencies, do plotting and graphing, and use multiplayer to code as part of a team.

Part 2 covers more advanced Repl.it use. You'll see how to pull projects from GitHub and collaborate on open source software, build a game, and keep your code secure. You'll build a full web application using Test Driven Development (TDD), and find out how to be an elite hacker by using the shortcuts offered by Repl.it

Once you've completed Part 1 and Part 2, you'll be able to build nearly any project that you want, and deploy it for the world to use. If you're stuck for ideas, you can go through the examples given in Part 3, which consists of practical tutorials to build everything from web scrapers to chat bots.

Note that this set of lessons does not focus on teaching you to code, though we will explain some key concepts along the way. If you don't already know how to code, it's best to take this course in conjunction with a more traditional course. If you're not sure what to do next, jump right in and see if you can keep up. We're beginner friendly.

Part 1: The basics of Repl.it

In this section of the course, you'll learn the basic of Repl.it. But that doesn't mean you won't build some fun stuff along the way.

Tutorial 1: Introduction to Repl.it and using the IDE

Learn the basics of the Repl.it IDE. Why use an online IDE and what are all those different panes? Build a simple program to solve your maths homework.

Tutorial 2: Working with files using Repl.it

Computers were initially created to read and write files, and although we've come a long way files remain central to everything we do. Learn how to create them, read from them, write to them, and import and export them in bulk.

Understanding the Repl.it IDE: a practical guide to building your first project with Repl.it

Software developers can get pretty attached to their Integrated Development Environments (IDEs) and if you look for advice on which one to use, you'll find no end of people advocating strongly for one over another: VS Code, Sublime Text, IntelliJ, Atom, Vim, Emacs, and no shortage of others.

In the end, an IDE is just a glorified text editor. It lets you type text into files and save those files, functionality that has been present in nearly all computers since those controlled by punch cards.

In this lesson, you'll learn how to use the Repl.it IDE. It has some features you won't find in many other IDEs, namely:

- It's fully online. You can use it from any computer that can connect to the internet and run a web browser, including a phone or tablet.
- It'll fully manage your environment for building and running code: you won't need to mess around with making sure you have the right version of Python or the correct NodeJS libraries.
- You can deploy any code you build to the public in one click: no messing around with servers, or copying code around.

In the first part of this guide, we'll cover the basics and also show you how multiplayer works so that you can code alone or with friends.

Introduction: creating an account and starting a project

Although you don't need an account to use Repl.it (you can just navigate to repl.it[3] and press the "start coding" button), let's set one up in order to have access to all of the features.

Visit https://repl.it/signup[4] and follow the prompts to create a user account, either by entering a username and password or by logging in with Google, GitHub, or Facebook.

Once you're done, hit the `+ new repl` button in the top right. In the example below, we choose to create a new Python project. Repl.it will automatically choose a random name for your project, or

[3] https://repl.it/
[4] https://repl.it/signup

you can pick one yourself. Note that by default your repl will be public to anyone on the internet; this is great for sharing and collaboration, but we'll have to be careful to not include passwords or other sensitive information in any of our projects.

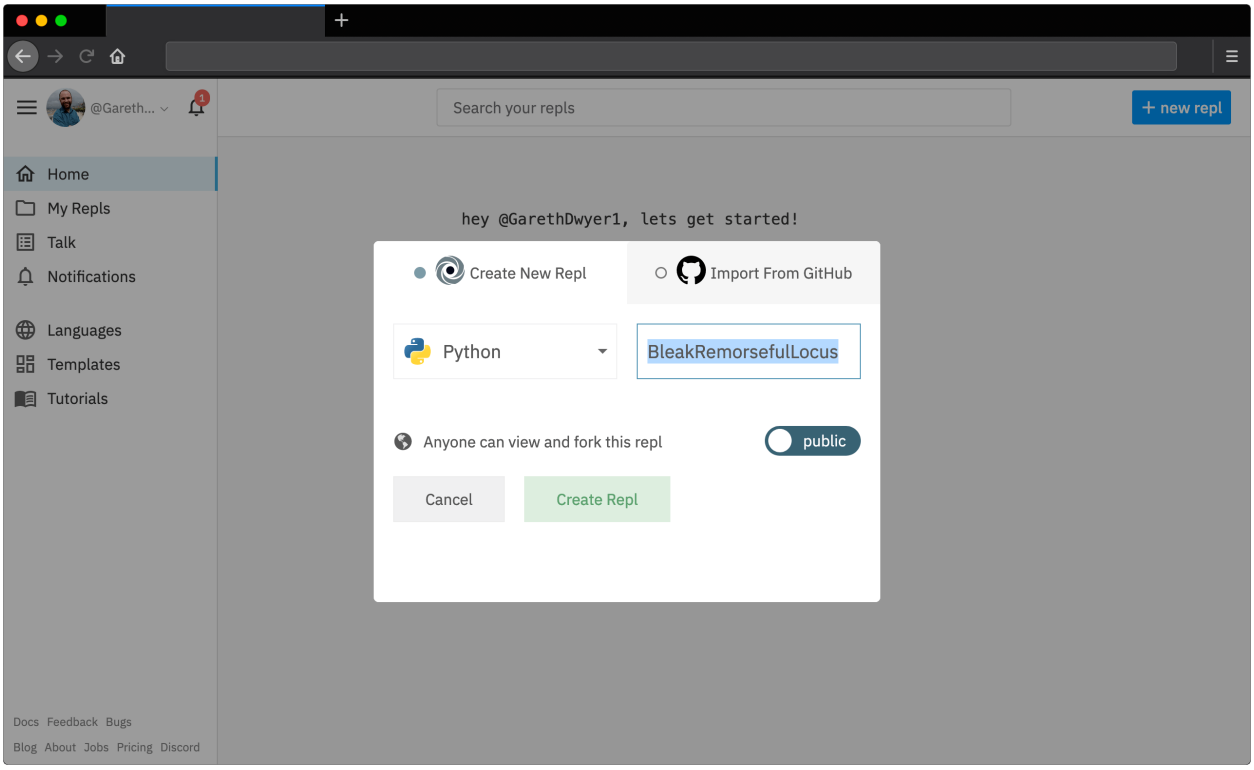

Image 1: ***Creating a new Python project***

You'll also notice an "Import from GitHub" option. Repl.it allows you to import existing software projects directly from GitHub, but we'll create our own for now. Once your project is created, you'll be taken to a new view with several panes. Let's take a look at what these are.

Understanding the Repl.it panes

You'll soon see how configurable Repl.it is and how most things can be moved around to suit your fancy. However, by default, you'll get the following layout.

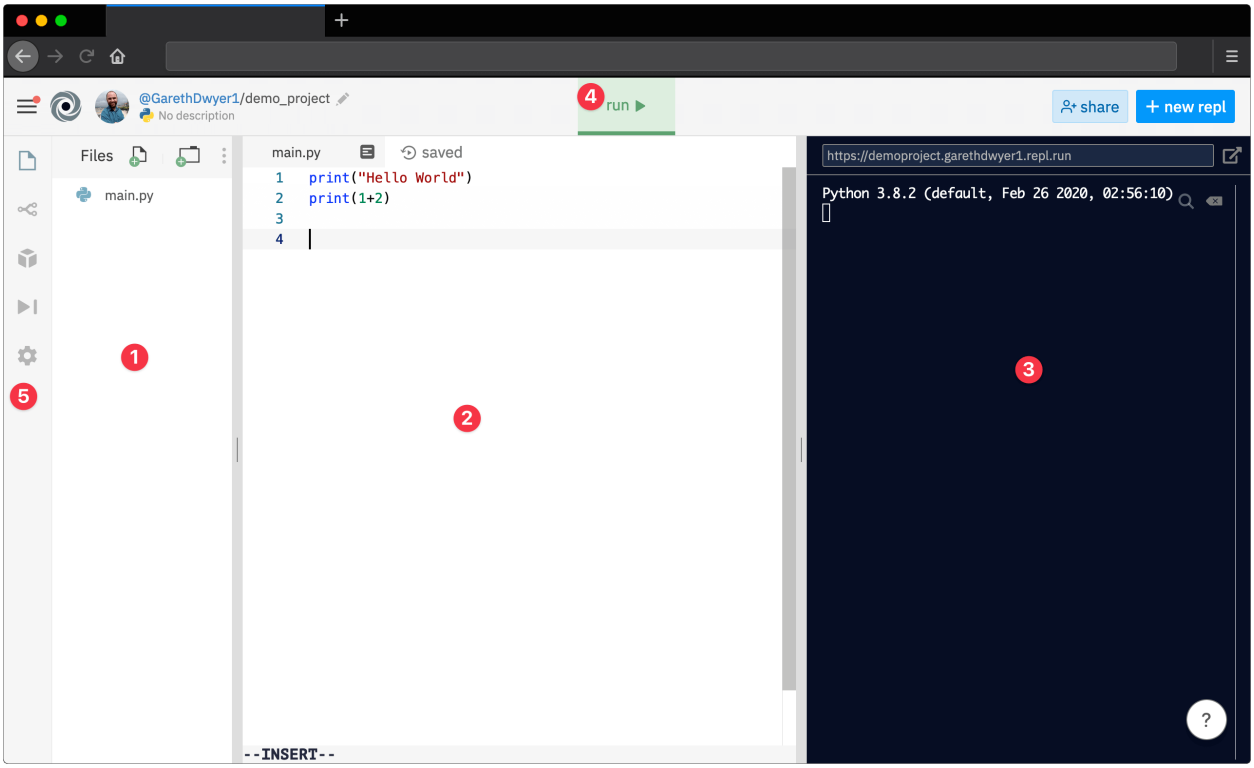

Image 2: *The Repl.it panes*

1. **Left pane: files and configuration**. This, by default, shows all the files that make up your project. Because we chose a Python project, Repl.it has gone ahead and created a `main.py` file.
2. **Middle pane: code editor**. You'll probably spend most of your time using this pane. It's a text editor where you can write code. In the screenshot, we've added two lines of Python code, which we'll run in a bit.
3. **Right pane: output sandbox**. This is where you'll see your code in action. All output that your program produces will appear in this pane, and it also acts as a quick sandbox to run small pieces of code, which we'll look at more later.
4. **Run button**. If you click the big green `run` button, your code will be executed and the output will appear on the right.
5. **Menu bar**. This lets you control what you see in the main left pane (pane 1). By default, you'll see the files that make up your project but you can use this bar to view other things here too by clicking on the various icons. We'll take a look at these options later.

Don't worry too much about all of the functionality offered right away. For now, we have a simple goal: write some code and run it.

Running code from a file

Usually, you'll enter your code as text in a file, and run it from there. Let's do this now. Enter the following code in the middle pane (pane 2), and hit the run button.

```
1 print("Hello World")
2 print(1+2)
```

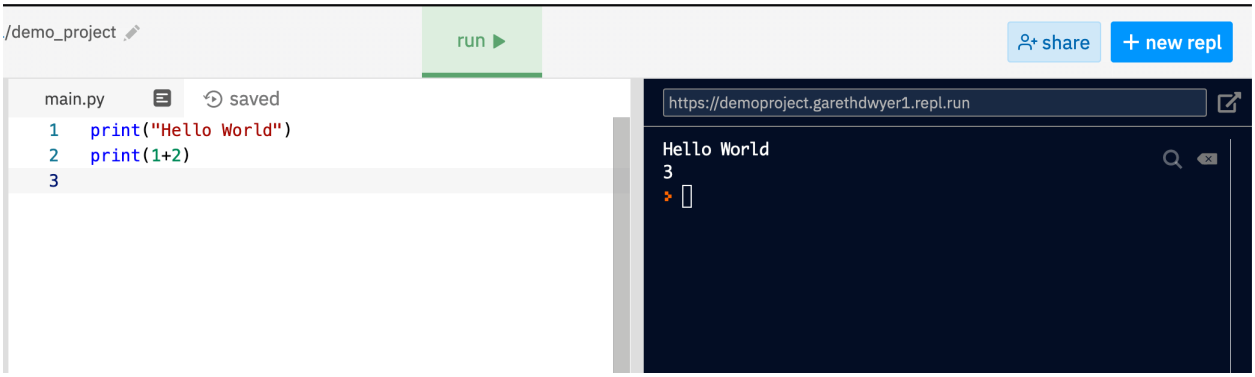

Image 3: *Your first program*

Your script will run and the output it generates will appear on the right pane (pane 3). Our code output the phrase "Hello World" (it's a long-standing tradition that when you learn something new the first thing you do is build a 'hello world' project), and then output the answer to the sum `1 + 2`.

You probably won't be able to turn this script into the next startup unicorn quite yet, but let's keep going.

Running code from Repl.it's REPL

In computer programming, a REPL is a read-eval-print loop[5], and a REPL interface is often the simplest way to run short computer programs (and where Repl.it got its name).

While in the previous example we saved our code to a file and then executed the file, it's sometimes quicker to execute code directly.

You can type code in the right-hand pane (pane 3) and press the "Enter" key to run it. Take a look at the example below where we print "Hello World" again and do a different sum, without changing our code file.

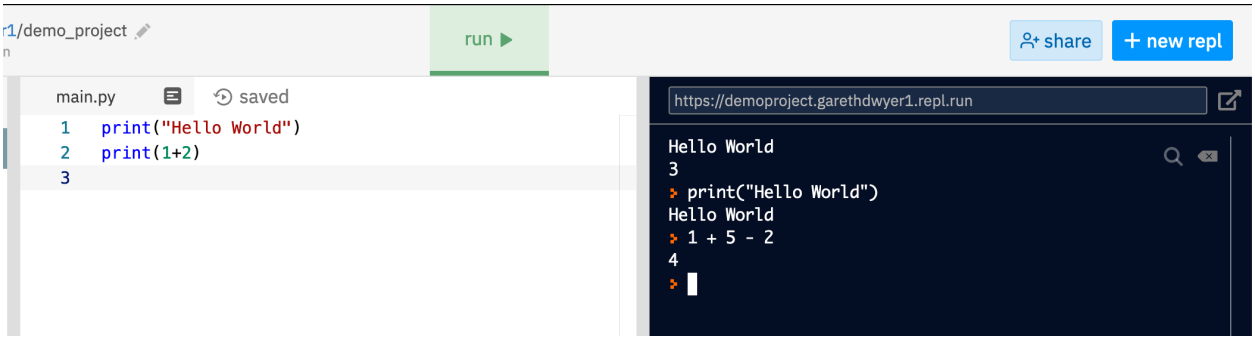

Image 4: *Running code from the REPL*

[5] https://en.wikipedia.org/wiki/Read%E2%80%93eval%E2%80%93print_loop

This is useful for prototyping or checking how things work, but for any serious program you write you'll want to be able to save it, and that means writing the code in a file like in our earlier example.

Adding more files to your software project

If you build larger projects, you'll want to use more than a single file to stay organised, grouping related code in different files.

So far, we've been using the 'main.py' file that was automatically created for us when we started the project, but we're not limited to this file. Let's add a new file, write some code in that, and import it into the main file for use.

As an example, we'll write code to solve quadratic equations[6]. If you've done this before, you'll know it can be tedious without a computer to go through all of the steps.

[6] https://www.mathsisfun.com/algebra/quadratic-equation.html

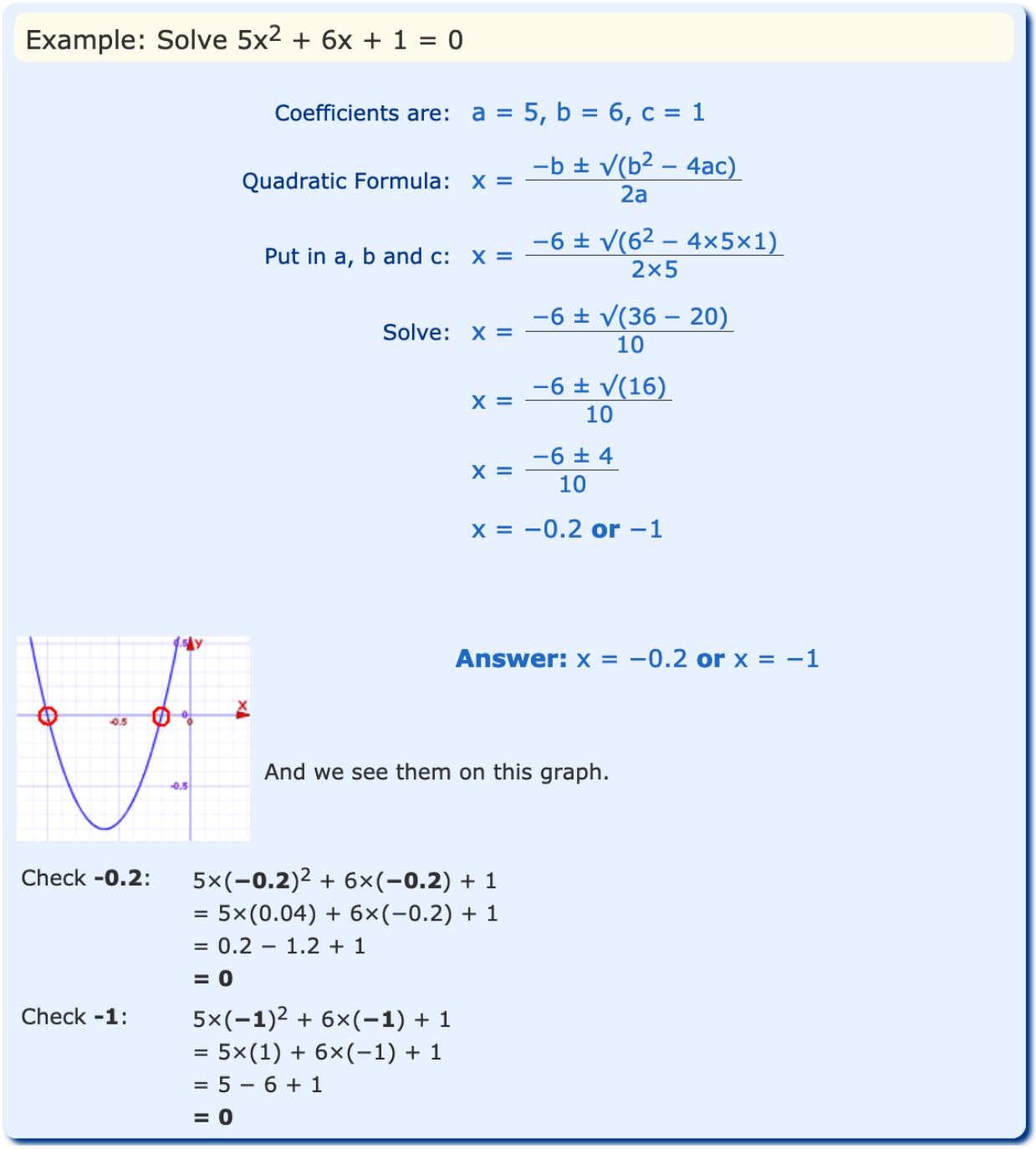

Image 5: *A quadratic equation example*

Let's make Python do the repetitive steps for us by creating a program called "solver". This could eventually have a lot of different solvers, but for now we'll just write one: `solve_quadratic`.

Add a new file to your project by clicking on the `new file` button, as shown below. Call the file `solver.py`. You now have two files in your project: `main.py` and `solver.py`. You can switch between your files by clicking on them.

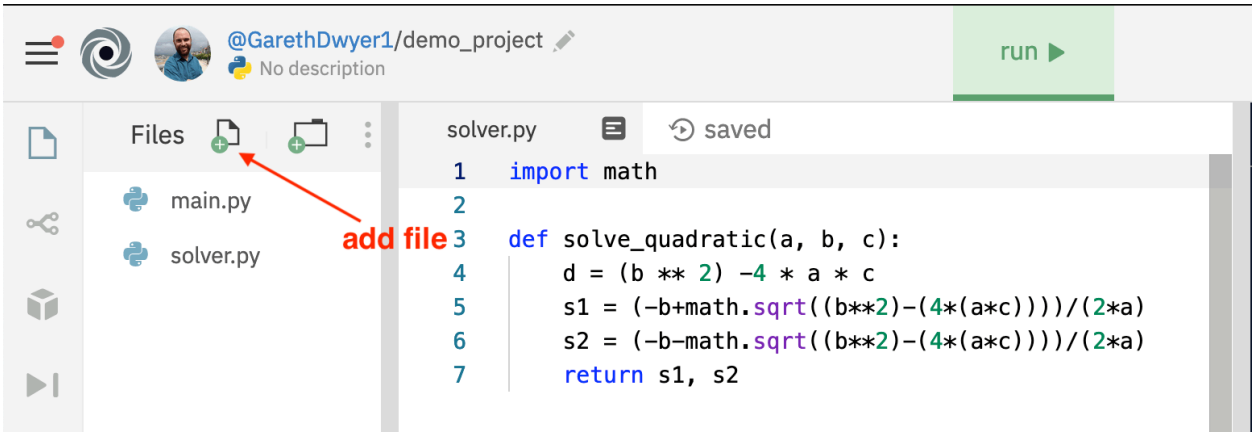

Image 6: *Adding a new file*

Open the `solver.py` file and add the following code to it.

```
import math

def solve_quadratic(a, b, c):
    d = (b ** 2) - 4 * a * c
    s1 = (-b + math.sqrt(d)) / (2 * a)
    s2 = (-b - math.sqrt(d)) / (2 * a)
    return s1, s2
```

Note that this won't solve all quadratic equations as it doesn't handle cases where `d`, the discriminant, is 0 or negative. However, it'll do for now.

Navigate back to the `main.py` file. Delete all the code we had before and add the following code instead.

```
from solver import solve_quadratic

answer = solve_quadratic(5, 6, 1)
print(answer)
```

Note how we use Python's import functionality to import the code from our new solver script into the `main.py` file so that we can run it. Python looks for `.py` (Python) files automatically, so you omit the `.py` suffix when importing code. Here we import the `solve_quadratic` function (which we just defined) from the `solver.py` file.

Run the code and you should see the solution to the equation, as shown below.

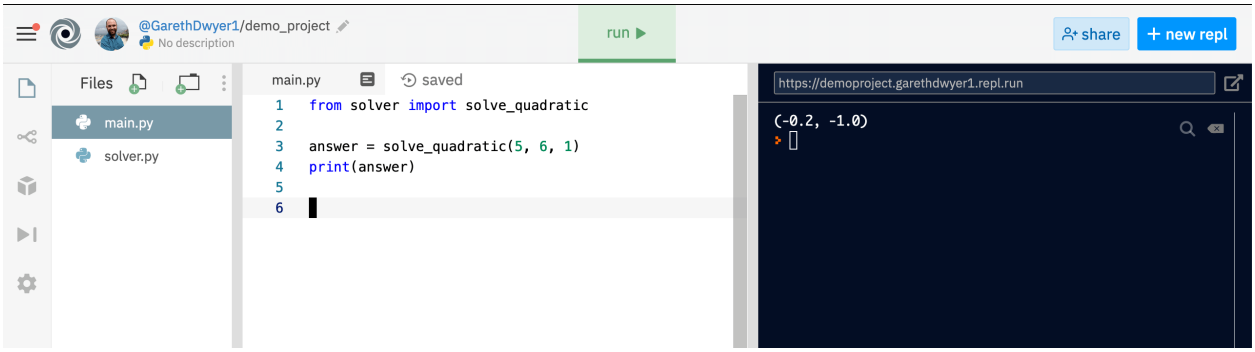

Image 7: *The solution to the quadratic equation*

Congratulations! You've written your first useful program.

Sharing your application with others

Coding is more fun with friends or as part of a team. If you want to share your code with others, it's as easy as copying the URL and sending it. In this case, the URL is `https://repl.it/@GarethDwyer1/demoproject`, but yours will be different based on your Repl.it username and the project name you chose.

You can copy the link and open it in an incognito tab (or a different web browser) to see how others would experience your project if you were to share it. By default, they'll be able to see all of your files and code and run your code, but not make any changes. If someone else tries to make changes to your repl, it'll automatically get copied ("forked") to their account, or an anonymous account if they haven't signed up for Repl.it. Any changes your friends make will only happen in their copies, and won't affect your code at all.

To understand this, compare the three versions of the same repl below.

- As you see it, with all of the controls
- As your friend would see it, a read-only version
- As your friend would see it after forking it, on an anonymous account

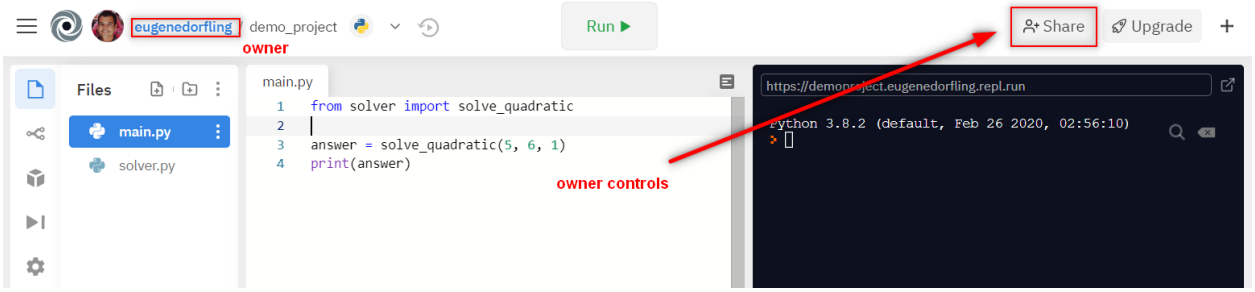

Image 8: *The owner's view of a repl*

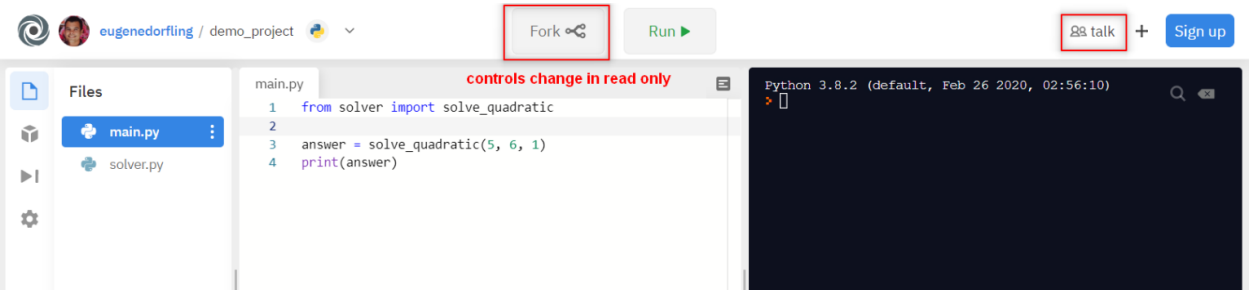

Image 9: *A guest's 'read-only' view of a repl*

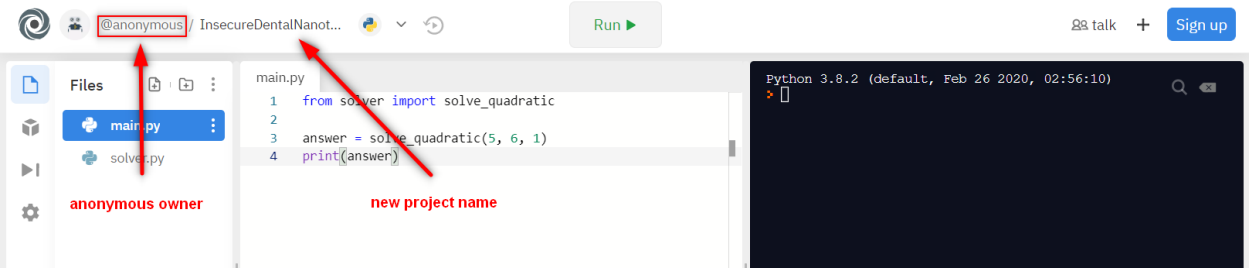

Image 10: *An anonymous owner's view of a copy of a repl*

What does this mean? Because no one else can edit your repl, you can share it far and wide. But because anyone can read your repl, you should be careful that you don't share anything private or secret in it.

Sharing write-access: Multiplayer

Of course, sometimes you might want others to have write access to your repl so that they can contribute, or help you out with a problem. In these cases, you can use Repl.it's "multiplayer" functionality.

If you invite someone to your repl, it's different from sharing the URL with them. You can invite someone by clicking `Share` in the top right and sending them the secret link that starts with "https://repl.it/join". This link will give people edit access to your repl.

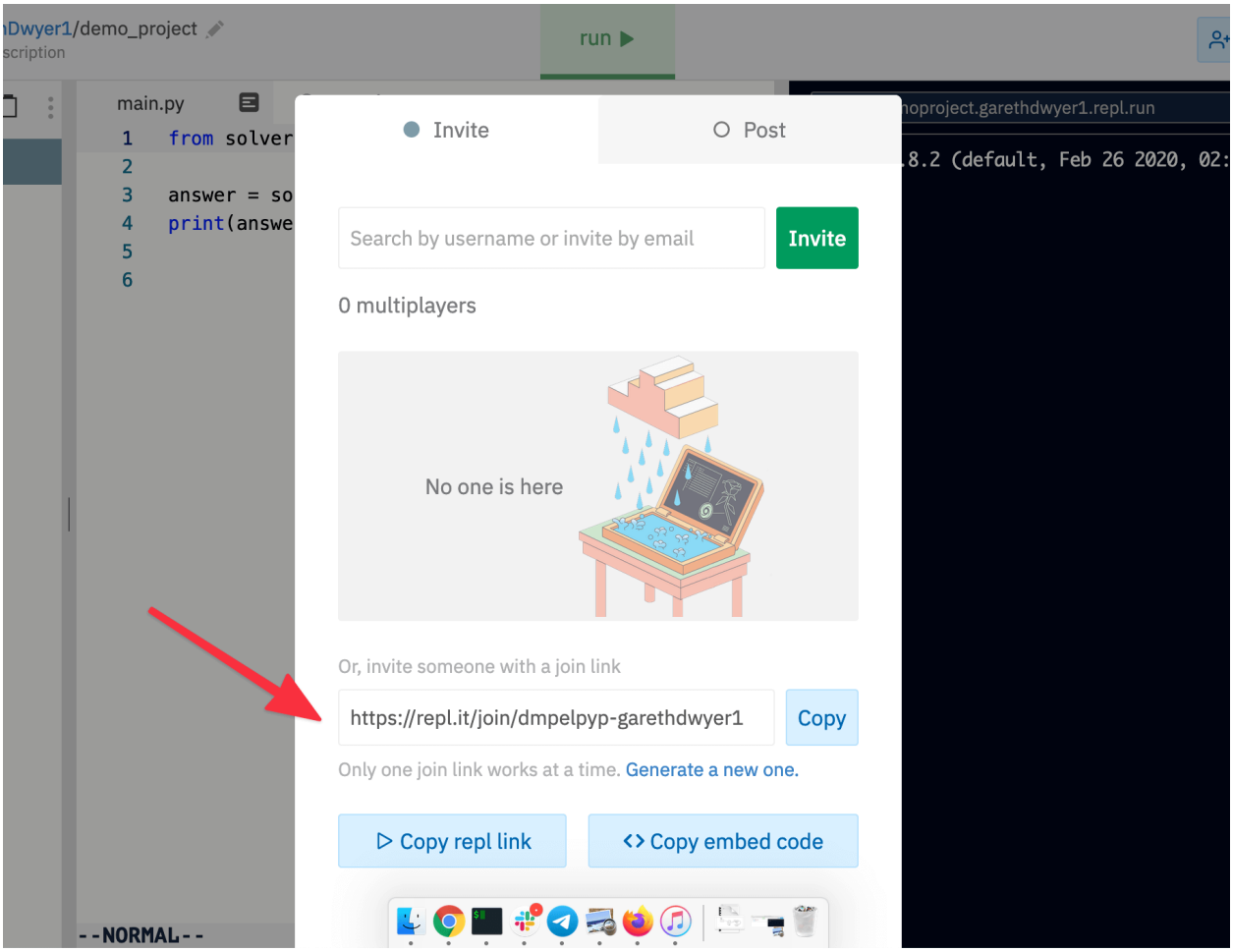

Image 11: ***Inviting someone to your repl***

If you have a friend handy, send it to them to try it out. If not, you can try out multiplayer anyway using a separate incognito window again. Below is our main Repl.it account on the left and a second account which opened the multiplayer invite link on the right. As you can see, all keystrokes can be seen by all parties in real time.

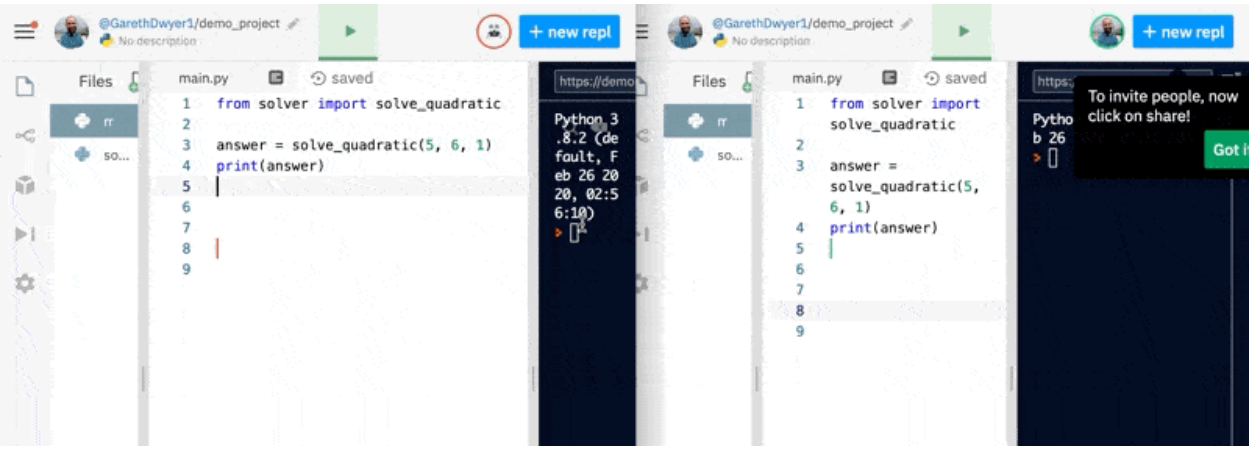

Image 12: ***Using multiplayer***

Make it your own

If you followed along, you'll already have your own version of the repl to extend. If not, start from ours. Fork it from https://repl.it/@GarethDwyer1/cwr-01-quadratic-equations[7].

Where next?

You can now create basic programs on your own or with friends, and you are familiar with the most important Repl.it features. There's a lot more to learn though. In the next nine lessons, you'll work through a series of projects that will teach you more about Repl.it features and programming concepts along the way.

If you get stuck, you can get help from the Repl.it community[8] or on the Repl.it Discord server[9].

[7] https://repl.it/@GarethDwyer1/cwr-01-quadratic-equations
[8] https://repl.it/talk/all
[9] https://repl.it/discord