



Girls Who Code At Home

Password Generator

Activity Overview

We use passwords for almost everything we do digitally! You may have a password for your email, shopping websites, Instagram, food delivery apps, etc. Did you know that in order to secure your personal information, such as your name, birthday, or location, you should create a different password for all of your accounts? In 2019 about 1 in 15 people became victims of [identity fraud](#) where one of the most common reasons for attack was due to weak passwords.

Cyber Security Specialists are responsible for implementing security measures in any computer and recognizing potential threats. [Cyber Security](#) is a fast growing industry that makes up about 32-45% of all U.S tech jobs with an average base salary of **\$83,000**. There are many things that you can do at home to secure your systems to prevent a cyber attack! In this activity you will be randomly generating 10-character long passwords that are strong enough to prevent cyber attacks on your personal information.

If you want to learn more about cyber security, check out our activity: [Cyber Detective](#).

Learning Goals

By the end of this activity you will be able to...

- understand the importance of creating strong passwords to protect personal information.
- create and use [Lists](#) in Python to store information.
- use the [random](#) library to randomize numbers within a specified range.

Materials

- [Repl.it Editor](#)
- [Password Generator Sample Project](#)
- [Password Generator Reference Guide](#)

Prior Knowledge

Before embarking on this project, we recommend that you:

- can explain what a [variable](#) is in your own words and describe how they can be used in a program.
- can explain what a [conditional statement](#) is in your own words and describe how they can be used in a program.

If you want a quick refresher on Python we highly suggest you check out our activity: [Can I Help You?](#)

Women in Tech Spotlight: Violet Blue



Image Source:
[Columbia Journalism Review](#)

Violet Blue is a journalist, author, advisor, and educator on hacking and cyber crime. Violet found that women and the [LGBTQIA](#) community experience heightened risks of cyber crimes. Hackers may use or sell personal information leaving you vulnerable for strangers to [troll](#), impersonate, or physically stalk you. Violet continues to use her online platform to raise awareness of cyber crimes, educate others how to protect their information, and advise companies and individuals on cyber security measures.

In 2015 Violet wrote a book, [The Smart Girl's Guide to Privacy](#). This book educated its readers on how to:

- Delete personal content from websites
- Use website and browser privacy controls effectively
- Recover from and prevent identity theft
- Figure out where the law protects you—and where it doesn't
- Set up safe online profiles
- Remove yourself from people-finder websites

Watch this [video on Violet's view of hacking culture](#) (up to timestamp 3:20). Want to learn more about Violet? Read this [article on suggestions to fellow female journalists](#) on protecting against cyber crimes. [Learn more about using password managers](#) or [learn how to use VPN](#), Virtual Private Network, to keep your data private when browsing the internet here.

Reflect

Being a computer scientist is more than just being great at coding. Take some time to reflect on how Violet and their work relates to the strengths that great computer scientists focus on building - bravery, resilience, creativity, and purpose.



CREATIVITY

Violet continues to write articles and educate others about cybersecurity through her online platforms. If you had your own platform, what issues would you educate others about?

Share your responses with a family member or friend. Encourage others to read more about Violet to join in the discussion!

Step 1: What is Cyber Security? (2-5 mins)



Cyber security is the practice of defending computers (any electronic device that can store and process data), servers, network, and data in general from cyber attacks. With almost all types of electronic devices connected to the internet through [WiFi](#) or [5G](#), protecting our information from cyber attacks is more important than ever. You may already be familiar with some famous attacks stemming from privacy leaks of celebrities to the release of sensitive information from governments to crime shows on TV or you may personally know someone who has faced a cyber attack.

In 2019 about 1 in 15 people became victims of [identity fraud](#). One of the most common attacks are due to the lack of strong and secure passwords for any online account. In this activity we will discuss the qualities and importance of a strong password, then you will learn how to code a password generator in Python!

Strong/Secure Passwords (2 mins)

If a hacker was able to access your account, most likely this is due to a **weak/vulnerable password**. When creating a password, there may be some simple requirements before you are able to create a valid account. Some restrictions may include:

- 6-8 minimum characters
- A variety of uppercase and lowercase letters
- At least one number used
- At least one special character

Take a look at the chart to the right and compare how long it takes for hackers to crack your password.

TIME IT TAKES FOR A HACKER TO CRACK YOUR PASSWORD					
Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	1 sec	5 secs
7	Instantly	Instantly	25 secs	1 min	6 mins
8	Instantly	5 secs	22 mins	1 hour	8 hours
9	Instantly	2 mins	19 hours	3 days	3 weeks
10	Instantly	58 mins	1 month	7 months	5 years
11	2 secs	1 day	5 years	41 years	400 years
12	25 secs	3 weeks	300 years	2k years	34k years
13	4 mins	1 year	16k years	100k years	2m years
14	41 mins	51 years	800k years	9m years	200m years
15	6 hours	1k years	43m years	600m years	15 bn years
16	2 days	34k years	2bn years	37bn years	1tn years
17	4 weeks	800k years	100bn years	2tn years	93tn years
18	9 months	23m years	6tn years	100 tn years	7qd years

 **HIVE SYSTEMS** Cybersecurity that's approachable. Find out more at hivesystems.io

Image Source: [Reddit](#)

Oftentimes people will use the same password for all of their online accounts, from their Facebook, Apple, to bank accounts. While this makes it easier for people to remember their passwords it also leaves them even more vulnerable to cyber attacks.

In order to secure your accounts, it is best to create unique passwords with a variety of letters, numbers, and special characters. But as you can imagine, this is incredibly difficult to remember. Here come the password managers! These secure software helps people create and remember strong passwords for all accounts, all you need to do is remember one password.

Step 2: Plan Your Password (15-20 mins)

Explore It! (10-15 mins)



For this project we will be using the [Repl.it](https://repl.it) web editor. Repl.it is a free, collaborative, browser-based editor that supports multiple programming languages. This powerful tool can even allow you to code and talk with a group of friends all at the same time!

- ☐ **Rank the following passwords from 1-5, where 1 represents the most secure and 5 represents the least secure password.** In order to help you rank the passwords, we included a few columns: number of characters and character variation.
 - ☐ **Fill out the # of characters column and character variation for each password.**
 - ☐ **Rank the passwords from 1-5.**
 - ☐ **Check your answers by using [this website](#) and type in each option.** Fill out the last column, time for hacker to crack to compare your rankings.

Rank <i>1: most secure to 5: least secure</i>	Password	# of characters <i>How many characters are in the password?</i>	Character Variation <i>Does the password include upper and/or lower case letters? Numbers? Symbols?</i>	Time for Hacker to Crack <i>Based on this website, how long would it take for a hacker to crack the password?</i>
<i>Example Password</i>	ku8@}:'\$	8	Lowercase letters, symbols, numbers	4 hours
	hcVESx			
	vWESp3Tt			
	Sg3Jpezyhv			
	password1			
	jG/8ab{s			



Don't forget to check your ideas with the Reference Guide on pg 2.

Step 2: Plan Your Password (cont.)

- ❑ Take **2 minutes** to come up with your own secure passwords and write them down in the space below. Your passwords should take *at least 1 year* for hackers to crack.

1.

2.

3.

- ❑ Navigate to [this website](#) we used in the previous step to check how secure your passwords are.

Create the Pattern of Your Password Generator (2-3 mins)

In this activity you will be creating a project that randomly generates a password that is 10 characters long. You will decide the *order of characters* in your password.

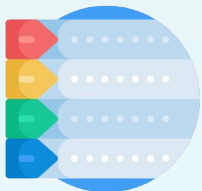
Example

For example in this sample project, all password generated will have the following pattern:

- **0st-5th characters:** A random letter either upper or lower case
- **6th character:** A random number
- **7th character:** A random special character
- **8th-9th character:** A random letter either upper or lowercase

```
q1Qe1B7-En  
ihhniK2&kR  
iGVRNz3^Hb  
oYJTJC2+s0  
bTedeQ7&Rm
```

These are 5 examples of a randomly generated password that uses the same pattern as above!



You may be thinking, why did we start with the 0? In fact, in order to create our password we will need to use an object called [Lists](#) in Python. A **List** is an ordered data set that can be used to store different types of information. **Lists** uses indexes, or a number, to reference each element inside. This makes it easy to retrieve, add, delete, and modify information within the **List**. Indexes start at 0 and increase by 1, which is why we chose to start at 0 in our character pattern for the password in the example above! We will learn more about how **Lists** store data later in this activity.

Step 2: Plan Your Password (cont.)

- ❑ **Take 2 minutes to plan out the pattern of your password in the table below.** For simplicity, we suggest you combine both upper and lowercase letters together. If you want to make your pattern more complex you can specify locations that will be either upper or lowercase letters.

Character #	Type of Character <i>Letter (upper or lowercase), number, or special character</i>
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

Note: Your password pattern might be different than what is depicted in the example throughout this activity. Please keep this in mind as you compare your code with our examples in the Reference Guide.

Tip: Print out this page so that you can easily reference this later!

Step 3: Get Started with Repl.it (15-20 mins)



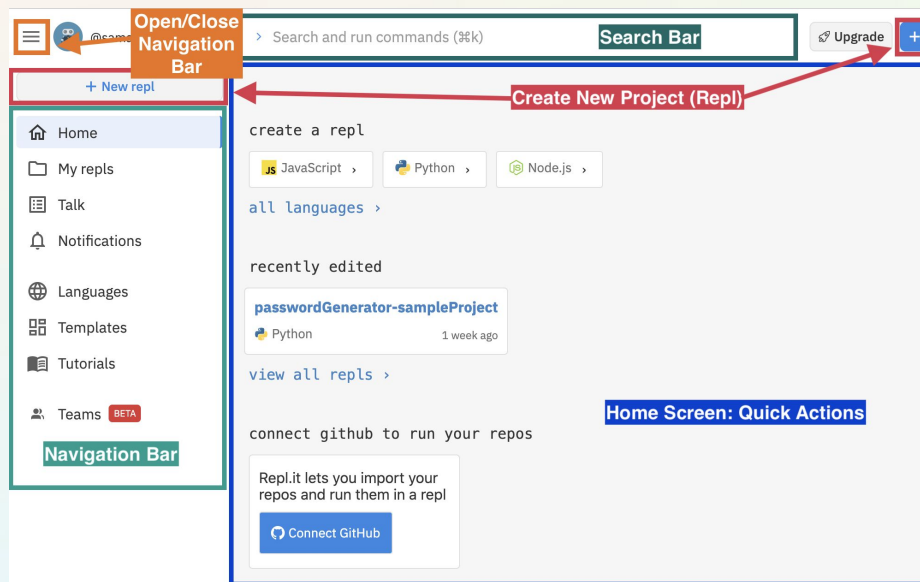
For this project we will be using the [Repl.it](https://repl.it) web editor. Repl.it is a free, collaborative, browser-based editor that supports multiple programming languages. This powerful tool can even allow you to code and talk with a group of friends all at the same time!

Create an Account (3-5 mins)

- ❑ **Sign up or login to Repl.it.** In order to save your work you will need to create an account. Follow the instructions on the sign up form to create an account. If you are under 13 you'll need your parent's email address to sign up.
- ❑ **Follow the instructions to create an account.** You may choose to sign up with your Google, GitHub, or Facebook account for faster login.

Explore the Repl.it Platform (3-5 mins)

Welcome to the Repl.it platform! Let's explore some of the key features available to get you started.



You may notice multiple mentions of GitHub throughout your Repl.it browser. **GitHub** is a popular software development manager. We will not use this tool in this activity, but to learn more check out this [GitHub tutorial](#).

- **Navigation Bar:** This column to the left of your screen allows you to access common actions that you might want to engage in, like create or view your Repls..
- **Search Bar:** Instead of using the navigation bar, you can also use the search bar to search and run commands.
- **New Repl:** There are two ways to create a new project. The first is through the navigation bar and clicking on the **+New repl** button or clicking the **+** button on the top right of the window.
- **Main Area:** In the center of your screen will vary depending on your view. When you first log in to Repl.it you will be defaulted to the home screen.

You can learn more about the Repl.it platform and Repls using this [resource](#).

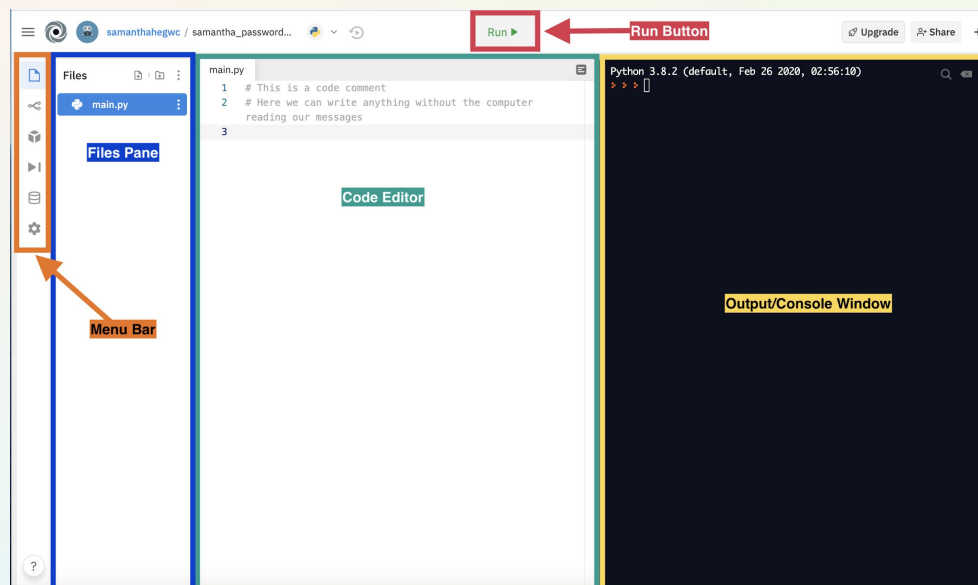
Stepc 3: Get Started with Repl.it (cont.)


Create a New Repl.it Project (2-4 mins)

- ❑ **Create a new repl.** Click the blue **+** button on the top right corner of your screen or the **+New repl** button in your navigation bar.
- ❑ **Select Python under the language option.** For this project we will be working in Python, but remember that for any future projects you can program in over 50 languages on Repl.it.
- ❑ **Name your project.** Give your project a descriptive name like **<yourName>_pwGenerator**. Typically projects should have no spaces and use **camelCase** or underscores to separate words.
- ❑ **Click Create Repl.**

Explore the Editor View (5-8 mins)

Let's take a look into the editor view for Repl.it. Now that we created a new project, we need to understand where to code, navigate between files and assets, and how to save and run your code!



- **Files Pane:** By default, this area will show you all of the files associated with the project. In this activity we will only be working in one file, main.py. You may find in other project you may want to include data files, images, or other Python files. To close this pane, click on the files icon  at the top of the menu bar.
- **Menu Bar:** This bar will allow you to change the view in the file pane. Some options include changing version control, adding packages, debugging, adding a database, and updating settings. In the settings you can customize the layout, theme, font size, and other text settings.
- **Code Editor:** This is where you will write your code!
- **Run Button:** After making changes to your code, click the run button at the top of the editor. You should see your result in the output/console window on the right.
- **Output/Console Window:** This displays any output in your code. All outputs will be visible, so if you click the run button multiple times, each result will be displayed here.

You may have noticed that there is no **save** button in Repl.it. As long as you have internet connection, all changes in your code will be saved automatically! Once you click the run button your Repl is saved, so make sure you always run your code before closing your editor!

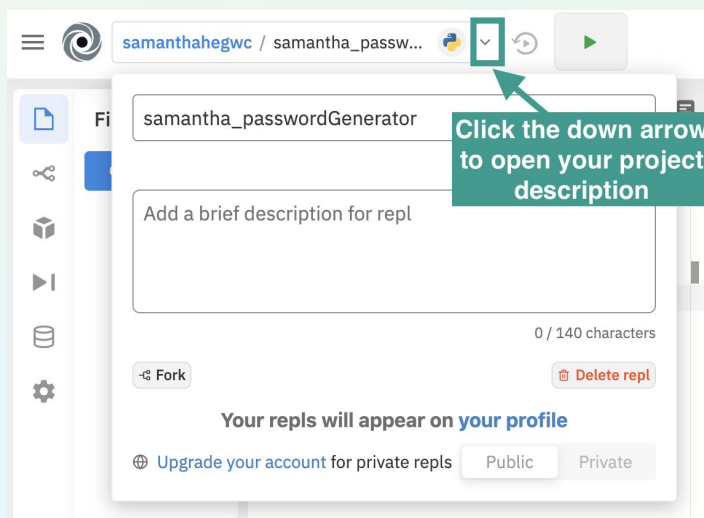
Stepc 3: Get Started with Repl.it (cont.)



It is always a good idea to also back up your code in case there is an error in your code or it does not save properly when you close Repl.it. You can learn more about how to use the version control option in the menu bar to set up a GIT repository to backup your work with this [resource](#). We will not be using this option for this project, but it is a valuable resource as you continue to build more projects with Repl.it!

Before we start coding our password generator, let's add a brief project description to our Repl.

- ❑ **Open the project description.** Locate your project name at the top left of your screen. Click the small down arrow to the right of the name. This should open a new window containing your project name and an area for a brief description.
- ❑ **Add a brief description.** Something you might want to include in your description may include:
 - ❑ **Overview:** How is it supposed to work?
 - ❑ **Instructions:** Are there any specific instructions needed to run your project?
 - ❑ **Attributions:** Did you get help from others or additional resources? Make sure you shout out these people and resources!



*This section does not need to be final now,
you can always go back and edit your description later.*

Step 4: Intro to Python (3-5 mins)



Python is a text-based programming language which means that all commands will need to be typed! Many programmers choose to use Python because it is easy to learn and understand since it mimics English-like commands. Python is an **open source** language, meaning that it is freely available for the public to use and modify as necessary. There are strict guidelines on how updates are accepted and implemented to the language, but essentially anyone can contribute to the evolution of the language!

The first thing that we will add to our project are some **code comments**. Code comments are lines of code that are not read by the computer. Programmers often use these comments to leave notes within their project that explain some lines of code and make their code more readable. This makes it easier for others to understand what the programmer was trying to accomplish in their project.

To add a single line code comment in Python, you start each line with the **#** symbol. This lets the computer know that the line of code is a comment and it doesn't read anything past the **#** symbol. Let's take a look at an example:

```
# This is a comment
# Here we can write anything without the computer
reading our messages
message = "hi"
```

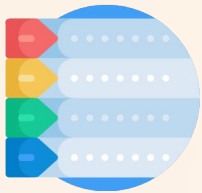
- ❏ **Take a moment to review the code snippet above. Which lines of code are comments? How do you know?**



Don't forget to check your ideas with the Reference Guide on pg 3.

Note: Since Repl.it restricts descriptions to only 140 characters, you can also use code comments at the top of your project to add more detailed description/instructions for your project.

Step 5: Meet Lists in Python (10-15 mins)



A **List** is an ordered data set that can be used to store different types of information. We will use **Lists** in our project to store all of the possible characters to choose from for our randomly generated password. **Lists** use indexes, or a number, to reference each element inside. This makes it easy to retrieve, add, delete, and modify information within the **List**. Indexes start at 0 and increase by 1, which is why we chose to start at 0 in our character pattern for the password in the example above! Let's take a look at an example:

PYTHON	DESCRIPTION
<pre>programmers = ["Ada", "Grace", "Katherine", "Roya"]</pre>	<ul style="list-style-type: none">→ programmers: The name of the variable that we store our List in.→ =: We use the equal sign to show assignment of an object/value to a variable.→ []: Square brackets are used to show the start and end of the contents in a List.→ "Ada", "Grace", "Katherine", "Roya": In this List we add the names programmers. We use either double or single quotation marks to indicate a String, or word, data type.→ ,: We use commas to separate each data/value stored in the List.

In this **List** indexes are assigned to each programmer name starting with the 0th index.

```
      0           1           2           3  
["Ada", "Grace", "Katherine", "Roya"]
```

We can see that Ada is at the 0th index in the **programmers** **List**, Grace is at the 1st index in the **programmers** **List**, and so on.

Practice Identifying the List Index (3-5 mins)

Let's take a moment to make sure that you understand the structure of **Lists** and how indexes are used to store and keep track of specific information in a **List**. In this practice project we have written all of the code. You will be running the project and answering the questions in the output/console window.

In this project we have a **List** with the following data:

```
wit = ["Ada", "Grace", "Violet", "Ayanna", "Mira1", "ENIAC"]
```

You will be asked a series of questions such as:

What is the WIT at index 2?

To answer, just type your response and press enter in the console. Remember that Python is case sensitive, so be sure that all of your responses start with a capital letter!

- ❑ **Click this [link](#) to open this List Practice Repl.it.** This is a practice project for you to test your knowledge of **Lists** and indices.
- ❑ **Click the RUN button at the top.**
- ❑ **Answer the questions in the console window.** This program will continue to ask you these questions infinitely. Answer at least 3-4 of these questions correctly before moving on to the next task.

Assigning Values in a List using the index (2-4 mins)

We can easily access information stored in lists by referring to the index of the data that we want. Let's take a look at the syntax to do this.

PYTHON	DESCRIPTION
<code>listName[index] = newValue</code>	<ul style="list-style-type: none"> → listName: This is the name of the variable storing the List of items. → []: We use square brackets to identify that we want to access information in our List. → index: This is the index, or number, of the data you want to access. → =: We use the equal sign to show assignment or reassignment to a new value. → newValue: This is the new data that you want to store!

Let's use our previous example to show how we can assign values to a specific index in a list.

```
programmers = ["Ada", "Grace", "Katherine", "Roya"]
```

- ❑ **What line of code would we write to change "Katherine" to "Violet" in our List?**



Don't forget to check your ideas with the Reference Guide on pg 3.

Step 6: Create All Possible Characters (10-15 mins)

We are finally ready to start coding our password generator, but where do we start? Let's think about the first thing that we need, the characters! We need to tell our program which characters are valid choices when generating a password. To do this we will create three **Lists** to separate the types of possible characters: letters, numbers, and special characters or symbols.

For simplicity we decided to combine both upper and lowercase letters in the same Lists. You may choose to make your pattern more complex by separating letters into two different **Lists**, one for lowercase letters and the other for uppercase letters.

Create List variables to store types of character possibilities (5-8 mins)

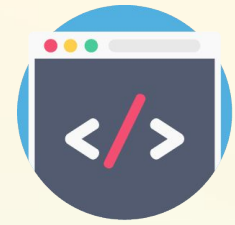
- ❑ **Open your password generator project on Repl.it.**
- ❑ **Add a code comment at the top to identify this section where you will be creating your Lists.** Your code comment can be simple as "Lists for possible types of characters". Remember that code comments must include a **#** symbol at the start of the line!
- ❑ **Create a variable to store a List of all possible types of letters, both upper and lowercase.** Be sure to include every letter of the alphabet in uppercase and lowercase separated by commas in your **List**. We named this variable **letters**, but you can create your own variable name. *Remember that Lists use square brackets **[]** to indicate the start and end of its contents and all information inside must be separated by commas! Since these are characters, each letter should be a String data type with either single or double quotation marks.*
- ❑ **Create a variable to store a List of all possible numbers.** Be sure to include every number separated by commas in your **List**. Remember that we will be reading these numbers as Strings, or words, so you will need to include either single **' '** or double **" "** quotation marks around each number. We named this variable **numbers**, but you can create your own variable name.
- ❑ **Create a variable to store a List of all possible special characters or symbols.** Feel free to include every symbol or just select a few symbols. Be careful not to include quotation marks (**'** or **"**) in your symbols since Python reads these as key characters that specify Strings. This can cause some confusion in your program. We named this **sChars**, but you can create your own variable name.



Don't forget to check your ideas with the Reference Guide on pg 4.

Meet the `print()` function (2 mins)

If you were to click the **Run** button you should notice that nothing happens. This is because we create variables but didn't actually tell the computer to do anything with that information. A great way to debug, or identify errors, in your program is to use the `print()` function in your code to test that your variables are holding the correct information. **Functions** are procedures or blocks of code that can be called to execute a task. Programmers often use functions to make their code more readable. The `print()` function is a built-in Python function that takes in an input and displays it in the console.



PYTHON	DESCRIPTION
<code>print(input)</code>	<ul style="list-style-type: none">→ <code>print()</code>: The name of the Python function that prints out a message to the console.→ <code>input</code>: This input can be in the form of a String or variable.

Test Your Code (3-5 mins)

Let's use the `print()` function to check that our Lists have been correctly implemented. For each print statement you should add them *underneath* all of your **List** variables.

- ❑ Add a `print()` statement to print out the content of your **letters** List.
- ❑ Add a `print()` statement to print out the content of your **numbers** List.
- ❑ Add a `print()` statement to print out the content of your **special characters** List.
- ❑ Click the **Run** button to view the contents of your Lists.

You should have:

- A **List** printed of all possible letters.
- A **List** printed of all possible numbers.
- A **List** printed of all possible special characters.

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
['!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '-', '+']
```

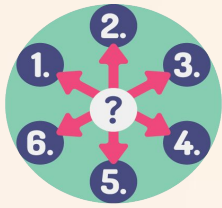
Not working the way you want it to? **Try these debugging tips:**

- Did you spell the function name correctly?
- Did you separate each character in your **List** with commas?
- Did you use either single or double quotation marks around each character?
- Do you use square brackets around the contents of each **List**?
- Do you have extra parentheses or brackets? You may notice that when you type a (symbol the editor automatically adds a closing bracket). This can cause you to add additional brackets by accident.



Don't forget to check your ideas with the Reference Guide on pg 4.

Step 7: Understand Randomness in Lists (5 mins)



It's finally time to build your randomly generated password. To store our final password we will create a password **List**. We will then assign each index to one of our random characters. This may seem like a lot but this can actually be done with just a few lines of code!

Meet the `random` library (2 mins)

A **library** is a collection of functions and variables. Programmers use libraries since many functions are already written for you to use! This built-in Python library will help us randomly select characters from our **Lists**. Before we can use any of the functions in a library, we first need to tell the computer to **import**, or load the library.

PYTHON	DESCRIPTION
<code>import random</code>	<ul style="list-style-type: none">→ import: This keyword lets the computer know to load the information from a Python library.→ random: This Python library is used to randomly generate numbers.

The **random** library contains many functions but we will be using the `choice()` function in our project. This function allows us to randomly select an item from a **List** or sequence. Let's take a look at the syntax of this function.

PYTHON	DESCRIPTION
<code>random.choice(listName)</code>	<ul style="list-style-type: none">→ random: This lets the computer know that we will be using a function from the random library.→ .: This symbol is often used to indicate calling a function from a library or another file.→ choice(): This is the name of the function we want to use.→ listName: The name of the List we want to select from.

Step 8: Generate the 1st random character (5-10 mins)



We are now fully equipped to start coding our randomly generated passwords! You may want to have your planned pattern for the password (on page 7) printed out or easily accessible for this step.

In a previous section we added `print()` statements to help debug our code and check our **Lists** were properly implemented. You may choose to *delete* or *comment* out these lines of code since they do not contribute to the overall objective of this project.

- ❑ **Add a new line of code at the top of your program, above your variables, to import the `random` library.** You may also add a comment above this to remind yourself what this line of code does. Typically programmers add any additional resources like importing libraries as the first few lines of code in their program.
- ❑ **Create a new password variable and assign a List of 10 empty values at the bottom of your code.** We named this variable `pw`, but you can create your own variable name. You may choose to create a List containing `0`s or empty Strings `""`. We will be replacing each of these values in the next steps.
- ❑ **Assign the 0th index of your password List to a random item in one of your characters List using the `random.choice()` function.** Use your password pattern to determine which List you should choose from for each index. For example, if you want a letter to appear first in your password, be sure to indicate the letters List as an input to the `random.choice()` function..
- ❑ **Add a line of code to print out your password List.** Before moving forward, we want to print out what our List looks like to make sure that we successfully retrieved a character and re-assigned the value at the 0th index in our List.



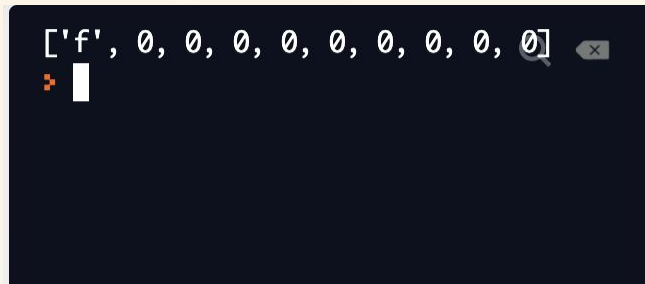
Don't forget to check your ideas with the Reference Guide on pg 5.

Step 8: Generate the 1st random character (cont.)

Test your Code (3-5 mins)

Let's test what we have written so far to make sure our program runs the way we want it to. Click the **Run** button to run your sketch. You should have:

- A password **List** printed where the 1st character is a random letter, number, or special character and the remaining characters are either 0s or empty Strings "" .



```
['f', 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Not working the way you want it to? **Try these debugging tips:**

- Did you spell the function and library name correctly?
- Did you separate each character in your **List** with commas?
- Did you use either single or double quotation marks around each character?
- Do you use square brackets **[]** when calling or assigning values to a **List**?
- Do you have extra parentheses or brackets? You may notice that when you type a (symbol the editor automatically adds a closing bracket). This can cause you to add additional brackets by accident.

Step 9: Create Your Random Password (5-10 mins)



Now that we understand how to generate the first random character, let's continue on and code the rest of the pattern of our password! When assigning the remaining indices of your password **List** you should do so **above** the print statement for your password.

- ❏ **Assign the remaining indices of your password **List** to a random item in one of your characters **List** using the `random.choice()` function.** You will have to repeat this step 9 times, assigning each remaining index from 1 to 9 a random character in one of the character type **List**.



Don't forget to check your ideas with the Reference Guide on pg 6.

Test your Code (3-8 mins)

Let's test what we have written so far to make sure our program runs the way we want it to. Click the **Run** button to run your sketch. You should have:

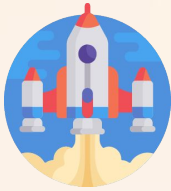
- A password **List** printed where each character is a random letter, number, or special character and follows the planned pattern of your password.

```
['s', 'V', 'F', 'z', 'S', 'C', '7', '-', 'W', 'F']
```

Not working the way you want it to? **Try these debugging tips:**

- Did you spell the function and library name correctly?
- Did you separate each character in your **List** with commas?
- Did you use either single or double quotation marks around each character?
- Do you use square brackets `[]` when calling or assigning values to a **List**?
- Do you have extra parentheses or brackets? You may notice that when you type a `(` symbol the editor automatically adds a closing bracket `)`. This can cause you to add additional brackets by accident.
- Are your lines of code in the correct location and sequence? Remember that order matters in program flow!

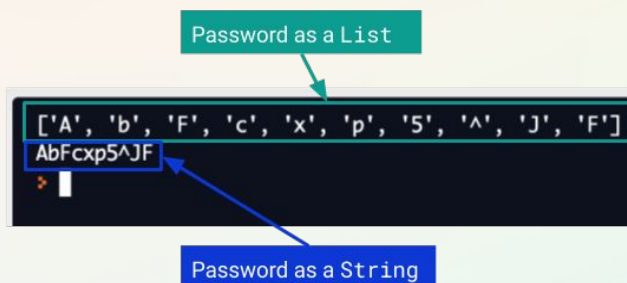
Step 10: Extensions (2-20 mins)



Use these extensions to add more function and complexity to your password generator. These challenges are meant to be more difficult, but don't be discouraged! Use the extension resources to learn more about the concepts covered in each challenge. It may also be helpful to look at the sample project provided and backwards engineer the steps by looking at the code.

Extension 1: Printing your password as a String (2-5 mins)

You may notice that by printing out your password **List** results in your password being shown with commas in between each character. If you want to represent your password as a String instead we can use the `join()` function.



Click [here](#) to see an example sketch of this extension

The `join()` function in Python allows you to take an iterable object, or structures like **Lists** that hold multiple information where each item can easily be accessed, and combines or *joins* each item with a **String**, or word.

PYTHON	DESCRIPTION
<code>stringName.join(itrObject)</code>	<ul style="list-style-type: none">→ stringName: This is a variable that holds a String, or word, type.→ .: This symbol is often used to indicate calling a function.→ join(): The name of the function that joins combines an iterable object and String into one String.→ itrObject: This is a variable that holds an iterable object, like a List.

- ❑ Use the `join()` function to combine an empty String `""` with the password **List** and store this in a **variable**. We named this variable `pwS`, but you can create your own variable name. You may also add a comment above this to remind yourself what this line of code does.
- ❑ Print out your password **String**.

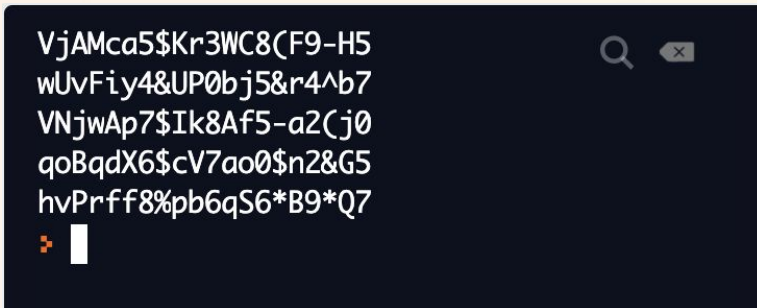
Extension Resources

Below are some helpful resources we used to create this Extension. These will help you get started, but remember that there are lots more resources only a search engine away!

- [join\(\)](#)
- [Iterable Objects](#)

Extension 2: Increasing the length of your password (10-15 mins)

Right now our password has a length of 10 characters, but what if we want to make our password even more secure? Most password managers will recommend you to create a password at least 15 characters in length or even better a 20 character long password!



Click [here](#) to see an example sketch of this extension

Make your password more complex by adding additional characters to your password **List** and assigning the remaining characters to random values.

- ❑ **Plan out the remaining pattern of your password.**
- ❑ **In your password **List** variable, add additional items to make your **List** the desired length.** To increase our password from 10 characters to 20, we added just 10 more **0**s to our **List** between the square brackets **[]**.
- ❑ **Assign the remaining indices of your password **List** to a random item in one of your characters **List** using the **random.choice()** function.**

Extension 3: Generate Passwords with Random Character Pattern (10-15 mins)

In our program you chose the pattern of your password, but what if we want to randomize the pattern for our passwords? We can use the `random` library to randomly select from any of our character lists. For this extension, you will need to use a **loop** and **conditionals** to generate our password.

```
3$+l0lv3Dq
%x6812%#9k
)1h)%K6c*q
5U@d4f$P7$
ek4114$491
> |
```

Click [here](#) to see an example sketch of this extension

In this extension, try using a loop to randomly assign each character in your password to a random type of character instead of assigning a specific pattern. Here are the basic steps you need to complete this extension:

- ❑ **Create a new password variable and assign a List of 10 empty values at the bottom of your code.**
- ❑ **Create a `for` loop with the `range()` function that iterates 10 times.** We use the variable `i` to store the iteration value of the `range` function but you can create your own variable name.
- ❑ **Inside your `for` loop, use the `random.randint()` function to randomly choose a number between 1 to 3 and store its result in a variable.** We will choose a random number to select which type of character should be selected. Since we have 3 types (letters, numbers, and special characters) we restrict our range to be only the numbers 1, 2, and 3. We named this variable `charType`, but you can create your own variable name.
- ❑ **Inside your `for` loop, add conditional statements that assign the character in your password List based on the character type randomly selected by the `randint()` function.** In our example we said that if `charType` was 1, we would randomly select a letter. If `charType` was 2, we would randomly select a number. Finally if `charType` was 3, we would randomly select a special character. Feel free to assign your own values to a different order of character types.

We will need to use the `i` variable from our `for` loop to assign each character in the password list. The `i` variable will update after each `for` loop iteration and hold values from 0 to 9.

- ❑ **Outside of your `for` loop, print out your randomly generated password!**

Extension Resources

Below are some helpful resources we used to create this Extension. These will help you get started, but remember that there are lots more resources only a search engine away!

- [For loops](#). You may also use a [while loop](#) for this example but we recommend a for loop since we know the specific number of iterations.
- [range\(\) function with for loops](#).
- [random.randint\(\)](#)
- [Conditional statements](#)

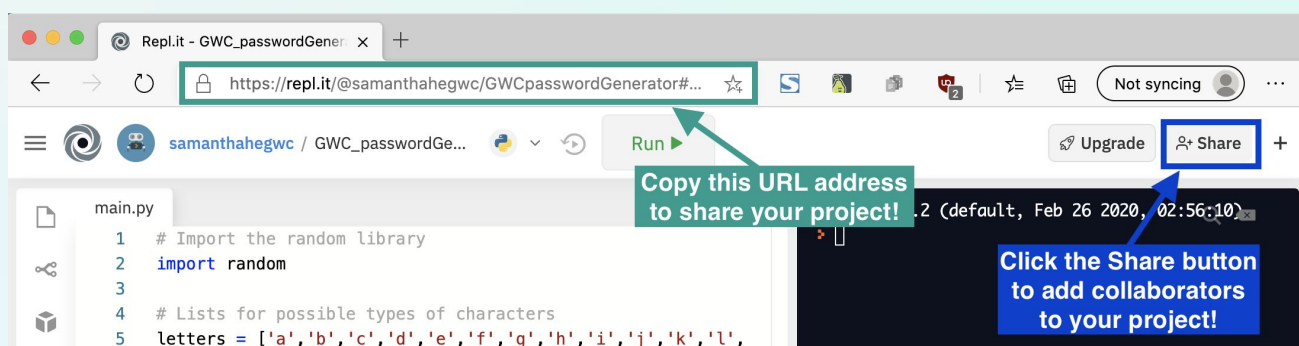
Step 11: Share Your Girls Who Code at Home Project (5-10 mins)

We would love to see your work and we know others would as well. Share your final project with us. Don't forget to tag [@girlswhocode](#) [#codefromhome](#) and we might even feature you on our account!

View Only Access (1 min)

Sharing your work on Repl.it is easy! Just copy and paste the URL address of your Repl project in the web address bar at the top! This will allow others to run your project, view your code, and “fork”, or duplicate, your project and remix on their own!

Be sure to share this link on your social media accounts and don't forget to tag [@girlswhocode](#) [#codefromhome](#) and we might even feature you on our account!



Step 11: Share your Girls Who Code at Home Project (cont.)

Adding Collaborators (2 mins)

If you want to work with a group of friends on a project, you can easily invite them to collaborate using the **Share** button at the top-right of the window. This should pop out a new window with two options for inviting others to collaborate on your project.

- **Invite by email or Repl.it username.** This option allows you to share your project with specific people by typing in their email address or Repl.it username if they already have an account with Repl.it. We recommend this option to ensure that you are sharing your project with the correct people!
- **Share invite link.** At the bottom of the window there is a unique invite link. You can copy and paste this link to friends which will allow them to access your project.

Note about collaborators: Remember that adding collaborators gives others edit access to your project. This will allow them to change your code, name, and description. **DO NOT share your invite link on social media!** Be selective on who you share these edit rights with.

Stay tuned for more Debug the Missing Code Part 2!





Girls Who Code At Home

Password Generator
Reference Guide

Meteor Catcher Game: Part 4 - Reference Guide



In this document you will find all of the answers to some of the questions in the activity. Follow along with the activity and when you see this icon, stop and check your ideas here.

Step 2: Plan Your Password

Create the Pattern of Your Password Generator (2-3 mins)

Rank <i>Rank the passwords from 1-5. 1: most secure to 5: least secure</i>	Password	# of characters <i>How many characters are in the password?</i>	Character Variation <i>Does the password include upper and/or lower case letters? Numbers? Symbols?</i>	Time for Hacker to Crack <i>Based on this website, how long would it take for a hacker to crack the password?</i>
<i>Example Password</i>	ku8@}:'\$	8	Lowercase letters, symbols, numbers	4 hours
4	hcVESx	6	→ Uppercase letters → Lowercase letters	4 hundred milliseconds
3	vWESp3Tt	8	→ Uppercase letters → Lowercase letters → Numbers	1 hour
1	Sg3Jpezyhv	10	→ Uppercase letters → Lowercase letters → Numbers	7 months
5	password1	9	→ Lowercase letters → Numbers	Instantly
2	jG/8ab{s	8	→ Uppercase letters → Lowercase letters → Numbers → Symbols	12 hours

Step 4: Intro to Python

```
# This is a comment  
# Here we can write anything without the computer reading our messages  
message = "hi"
```

In this example, the first two lines of code are code comments. We can tell because there is a # symbol at the start of the line. You may notice that the color of these lines of code are gray. The editor in Repl.it helps identify code comments by graying these lines of code, if you use a different editor code comments might be a different color. The only line of code that the computer will read is the one written on the third line! We will talk about what is written on the third line later in this activity.

Step 5: Meet Lists in Python

Assigning Values at a List Index

```
programmers = ["Ada", "Grace", "Katherine", "Roya"];  
programmers[2] = "Violet"
```

In our example we stored a **List** of names in a variable called **programmers**. In order to change "Katherine" to "Violet" we tell the computer that at index 2 of the **programmers** List, reassign the value to "Violet".

Step 6: Create All Possible Characters

Create List variables to store types of character possibilities

```
# Lists for possible types of characters
letters =
['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q',
'r','s','t','u','v','w','x','y','z','A','B','C','D','E','F','G','H','I',
'J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z']
sChars = ['!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '-', '+']
numbers = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
```

Test Your Code

Add these optional `print()` statements to check that your Lists have been implemented correctly.

```
# Lists for possible types of characters
letters =
['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r',
's','t','u','v','w','x','y','z',
'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S',
'T','U','V','W','X','Y','Z']
numbers = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
sChars = ['!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '-', '+']

# Print out Lists to check that content is correct
print(letters)
print(numbers)
print(sChars)
```

Step 8: Generate the 1st random character

In this example we implement a password layout where the first character is a letter.

```
# Import the random library
import random

# Lists for possible types of characters
letters =
['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p',
'q','r','s','t','u','v','w','x','y','z',
'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P',
'Q','R','S','T','U','V','W','X','Y','Z']
numbers = ['0','1','2','3','4','5','6','7','8','9']
sChars = ['!','@','#','$','%','^','&','*','(',')','-', '+']

# Print out Lists to check that content is correct
#print(letters)
#print(numbers)
#print(sChars)

# Create password List
pw = [0,0,0,0,0,0,0,0,0,0,0]

pw[0] = random.choice(letters)

print(pw)
```


Step 9: Create Your Random Password

In this example we implement a password layout where the first character is a letter.

```
# Import the random library
import random

# Lists for possible types of characters
letters =
['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p',
'q','r','s','t','u','v','w','x','y','z',
'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P',
'Q','R','S','T','U','V','W','X','Y','Z']
numbers = ['0','1','2','3','4','5','6','7','8','9']
sChars = ['!','@','#','$','%','^','&','*','(',')','-', '+']

# Create password List
pw = [0,0,0,0,0,0,0,0,0,0]

pw[0] = random.choice(letters)
pw[1] = random.choice(letters)
pw[2] = random.choice(letters)
pw[3] = random.choice(letters)
pw[4] = random.choice(letters)
pw[5] = random.choice(letters)
pw[6] = random.choice(numbers)
pw[7] = random.choice(sChars)
pw[8] = random.choice(letters)
pw[9] = random.choice(letters)

print(pw)
```