beanz
The Magazine for Kids, Code, and Computer Science

Roll a Story

BY BIANCA RIVERA / Programming / October 2020 Issue

Get ready to roll the dice with this fun programming exercise!

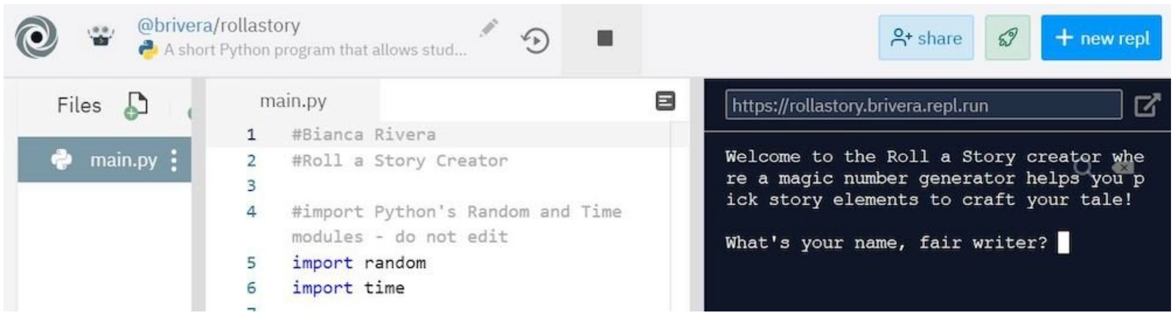
At some point in school you may have played a writing "game" in English class called Roll a Story. If you've never played it, it goes like this: a student takes two dice and rolls to have a short story genre and story elements (characters, setting and plot) selected. The student then writes an often outlandish and creative narrative story based on the selections "made" by the dice.

In this case, we are going to create a Python program that allows users to get randomly selected genres (either Mystery or Horror) and story elements to create their own mysterious or spooky story.

First, we need a place to host our program. Programmers can use the free site Repl.it to develop and host their programs and webpages right from their browser. Repl.it works on most operating systems and devices (even a cell phone!). Repl.it provides access to over 50 languages including the most popular ones such as JavaScript, Python (with Turtle), and HTML/CSS/JS.

Programs in Repl.it are called “repls”. You can create as many repls as you want and they can be shared with others. Other users can view and run programs as well as “fork” your program. Forking makes a copy that they can edit (without editing yours). You can even collaborate on a program with others in real-time.

Your repls will have an editor (where you type your code) and a console (where you see the output). You can type your code in one file or in multiple files. Repl.it auto-saves your work so you never lose your code! You can even see and restore your code history. All of these features are free!


@brivera/rollastory
A short Python program that allows stud...
share
new repl
Files
main.py
main.py
https://rollastory.brivera.repl.run

```python
#Bianca Rivera
#Roll a Story Creator

#import Python's Random and Time modules - do not edit
import random
import time
```

```
Welcome to the Roll a Story creator whe
re a magic number generator helps you p
ick story elements to craft your tale!

What's your name, fair writer?
```

Let’s start by creating a new account in Repl.it (or sign into your current one). Once you are signed in, paste the URL of my repl, “Roll a Story” program (linked below in “See More”). Hit enter and type your own name after the first hashtag in the file (you’ll see my name – #Bianca Rivera – replace it with your name). This will fork a copy and provide you a copy of the “Roll a Story” program. You can click on the pencil icon to rename the program and provide a description if you’d like.

Your screen will display the IDE (Integrated Development Environment) where you will see the program file, editor and console side by side. Your first file will be named main,py by default (the main.py file name doesn’t change). This particular Python program randomly rolls “dice” to pick story elements (character, setting and plot) from the Mystery or Horror genre.

If you look at the repl code, there's a lot going on but we will break it apart. Keep in mind there is plenty in this program you can edit to "make it your own" but some of this must stay as-is in order for the program to work.

Here's the code. There's a description below so be sure to scroll down. Or have your repl in one web browser window and this article in another as you read the description of how the code works.

```python
#Bianca Rivera
#Roll a Story Creator

#import Python's Random and Time modules - do not edit
import random
import time

#sets the number variables to None/null so the user gets new selections when they replay the game - do not edit
genreNum = None
charNum = None
setNum = None
plotNum = None

#welcomes the player to the game and stores their name in the name variable
def wel():
  global name
  #only the words within the quotation marks in the print statements can be edited
  print("Welcome to the Roll a Story creator where a magic number generator helps you pick story elements to craft your tale!")
  print()
  name = input("What's your name, fair writer? ")
  print("{}, that is a lovely name!".format(name))

#code that randomly selects a genre
def genre_roll():
  global genreNum
  global genreType
  genreNum = (random.randrange(1,3))

  #genre dictionary
  #only edit the name of the genre within quotation marks
  genreSwitcher={
    1: "MYSTERY",
    2: "HORROR"
  }

  genreType = genreSwitcher.get(genreNum, "invalid")

  #tells user their selections
```

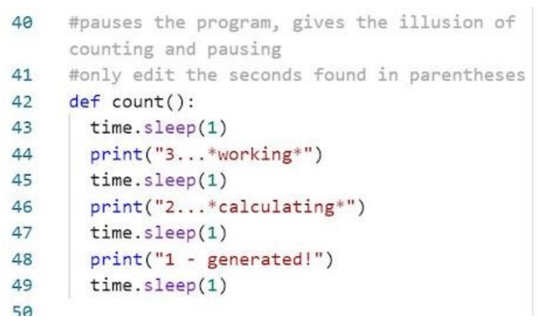
```python

#pauses the program, gives the illusion of counting and pausing
#only edit the seconds found in parentheses
def count():
  time.sleep(1)
  print("3...*working*")
  time.sleep(1)
  print("2...*calculating*")
  time.sleep(1)
  print("1 - generated!")
  time.sleep(2)

#tells the player which genre they will be writing about
def gen():
  print()
  print("Let's summon the magic number generator to help you pick a genre for your story!")
  count()
  print()
  print("The number generator picked a {}, which means you'll be writing about a story from the {} genre!".format(genreNum, genreType))
  if genreNum == 1:
    mys_roll()
  else:
    hor_roll()

#for Mystery Genre roll
#only edit the words found in the print statements between the quotation marks
def mys_roll():
  global charType
  global setType
  global plotType

  global charNum
  global setNum
  global plotNum

  charNum = (random.randrange(1,4))
  setNum = (random.randrange(1,4))
  plotNum = (random.randrange(1,4))

  #character dictionary
  charSwitcher={
    1: 'THIEF',
    2: 'DETECTIVE',
    3: 'FBI AGENT'
  }

  charType = charSwitcher.get(charNum, "invalid")

  #setting dictionary
  setSwitcher={
    1: 'A MUSEUM AT NIGHT',
    2: 'THE HIGHSCHOOL',
    3: 'AREA 51'
  }

  setType = setSwitcher.get(setNum, "invalid")

  #plot dictionary
  plotSwitcher={
    1: 'A PRICELESS JEWEL IS STOLEN',
    2: 'THE PRINCIPAL GOES MISSING',
    3: 'UNEXPLAINED PHENOMENA TAKING PLACE'
  }

  plotType = plotSwitcher.get(plotNum, "invalid")

#for Horror Genre roll
#only edit the words found in the print statements between the quotation
marks
def hor_roll():
  global charType
  global setType
  global plotType

  global charNum
  global setNum
  global plotNum

  charNum = (random.randrange(1,4))
  setNum = (random.randrange(1,4))
  plotNum = (random.randrange(1,4))

  #character dictionary
  charSwitcher={
    1: 'GHOST',
    2: 'VAMPIRE',
    3: 'WITCH'
  }

  charType = charSwitcher.get(charNum, "invalid")

  #setting dictionary
  setSwitcher={
    1: 'A CEMETARY',
    2: 'NEW ORLEANS',
    3: 'THE DEEP, DARK WOODS'
  }

  setType = setSwitcher.get(setNum, "invalid")

  #plot dictionary
  plotSwitcher={
    1: 'SPOOKY SOUNDS BEING HEARD',
    2: 'EERIE SIGHTINGS TAKING PLACE',
    3: 'A DEEP CHILL BEING FELT'
  }

  plotType = plotSwitcher.get(plotNum, "invalid")

#tells user their selections
#only edit the words found in the print statements between the quotation
```

```python
marks
def parts():
  print()
  print("Let's summon the magic number generator to help you pick a character for your story!")
  count()
  print()
  print("The number generator picked a {}, which means you'll be writing about a {}!".format(charNum, charType))
  time.sleep(2)
  print()
  print("Next, the magic number generator will pick a setting for you!")
  count()
  print()
  print("The number generator picked a {}, which means your setting will take place in {}!".format(setNum, setType))
  time.sleep(2)
  print()
  print("Last, the magic number generator will pick a plot for you!")
  count()
  print()
  print("The number generator picked a {}, which means the plot is based on {}!".format(plotNum, plotType))
  time.sleep(2)
  print()
  print("Lucky you, {}! You get to write a story from the {} genre, featuring a {}, set in {} based on the plot of {}!".format(name, genreType, charType, setType, plotType))
  print()

#calls the four functions needed to tell the story
#do not edit
wel()
genre_roll()
gen()
parts()

#asks if user wants to play again
#do not edit
play_again = 'y'
while True:
  play_again = input("Would you like to use Roll a Story again? Type 'y' for yes or 'n' for no. ")
  if play_again == 'y':
    genre_roll()
    gen()
    parts()
  elif play_again == 'n':
    print()
    print("Fare thee well, young writer!")
    break
  else:
    print("Hmm...I did not understand that. Type y for yes or n for no. ")
```

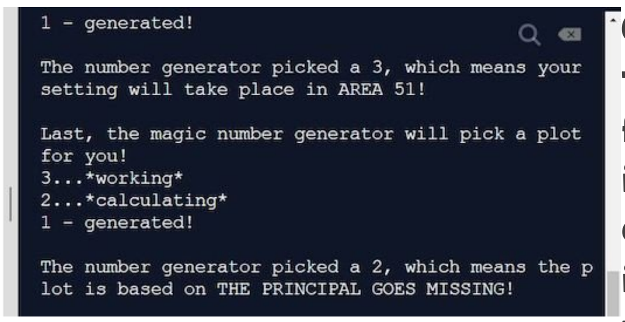
- The first thing we must do is to import two popular Python modules – time and random (lines 5 and 6 above). Python provides many built-in modules that provide programmers code libraries. What do you think time and random do in this program? Try looking it up!
- Four very important variables in this game are the Num variables (genreNum, charNum, setNum and plotNum) (lines 9-12 above). The program will use random to randomly select a number that is assigned to a specific genre, character, setting or plot. The game has a replay option at the end so in order to ensure players get new selections each time they play, the four number variables must start out as None, which is Python's version of Null.
- Next your will see a series of functions. You know they are functions when you see "def" printed in front of a variable with parenthesis and a colon at the end that looks like this: def function():. There are seven functions defined in this program. Can you list them all?
- Functions are an extremely important concept in Python and save programmers a lot of time. Functions are a block of reusable code that will run when called. The first function is named wel() and is located on line 15. Let's see how wel() works. Notice line 16 that says global name. Because the name variable has been defined in the wel() function, and we want to use it outside of this function, we need to tell Python that it should work everywhere within the program (global will do this). Next, the function provides a print statement, introducing players to the game. Anything within the print statement can be edited! Go ahead and change my wording but leave the parenthesis and quotation marks as-is! On line 20, you'll see the name variable being defined using the input function – did you notice that not only can we programmers create functions but Python has its own built-in functions? Input is one of them and we will use it to ask the player their name, which we can reference throughout the game. On line 21, we are using a Python string formatting method, .format(). By using the curly brackets {} as placeholders, we can use .format to pass the name variable to the string. Any name that the user has typed in will show up in the print statement on line 21!
- The next function, genre_roll(), makes use of the random module we imported! See genre_roll() on line 22. Notice the variable genreNum equals a number that is produced from the random module and that the range is between 1-3. Players only have two options for a genre –

Mystery or Horror, so why is the range 1-3? Because when it comes to the randrange method, Python includes the first number and up to, but not including, the last number. On line 28 is another important concept in Python that is a timesaver – a dictionary. Python's dictionaries pair up keys and values, like a real dictionary. In this example, key 1's value is Mystery and key 2's value is Horror. We are using random with genreNum to randomly select genre numbers that are assigned to genre types. Users will get a random genre selection each time they "roll" the dice. What other genres could we use in place of Mystery or Horror?

- On line 38 of the program, we see the function count(). Count is making use of the time library that we imported. In the code, you will see the word time.sleep and then a number in parenthesis. Sleep is a method of time and actually pauses a program for however many seconds you type in. How many seconds did I pause the program for? How could you make it pause even longer?



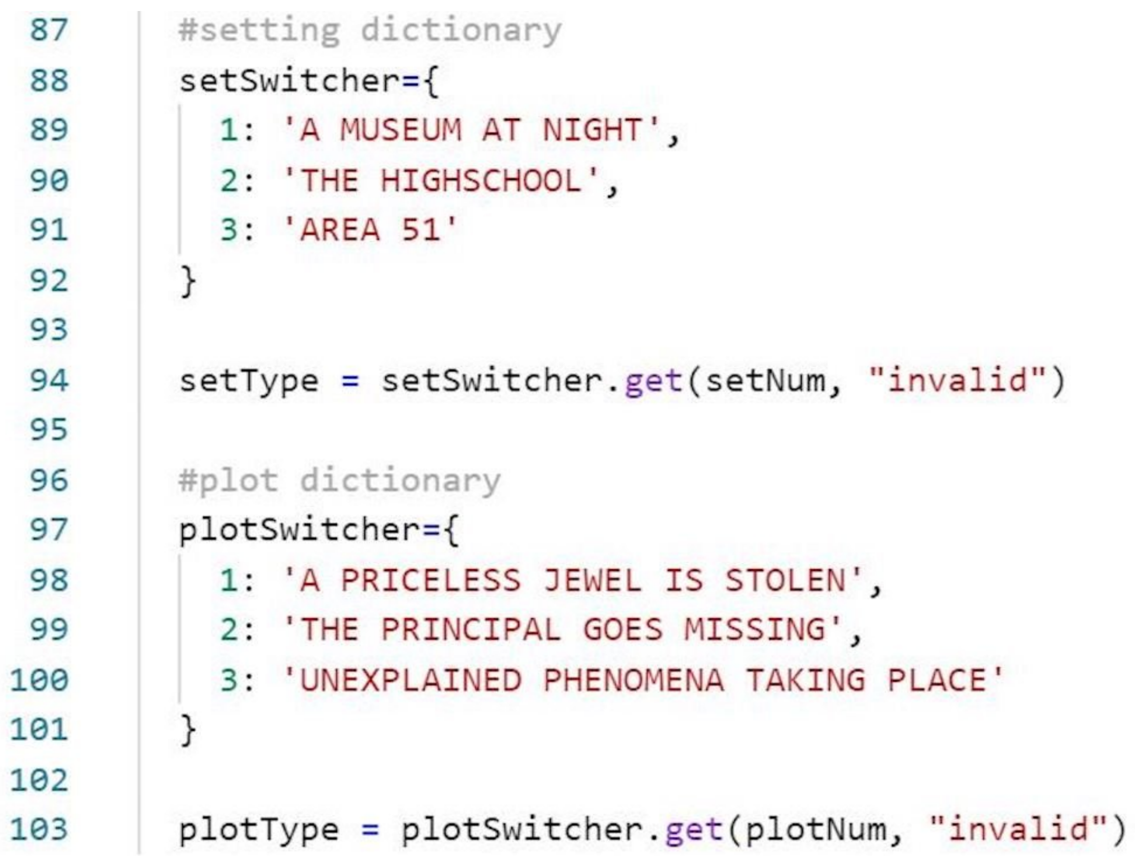
```
40  #pauses the program, gives the illusion of
    counting and pausing
41  #only edit the seconds found in parentheses
42  def count():
43    time.sleep(1)
44    print("3...*working*")
45    time.sleep(1)
46    print("2...*calculating*")
47    time.sleep(1)
48    print("1 - generated!")
49    time.sleep(1)
50
```



```
1 - generated!

The number generator picked a 3, which means your
setting will take place in AREA 51!

Last, the magic number generator will pick a plot
for you!
3...*working*
2...*calculating*
1 - generated!

The number generator picked a 2, which means the p
lot is based on THE PRINCIPAL GOES MISSING!
```

On line 47 is the gen() function and it makes use of another important programming concept, If... Else statements. On line 53, it states if genreNum == 1 then go to mys_roll(), else go to hor_roll(). Do you remember what number 1 was for and how we used it in the program? What was the number 2 for?

- On line 59 is the mys_roll() function and line 100 is the hor_roll() function. By now, you have seen all of the features of these



```
#setting dictionary
setSwitcher={
  1: 'A MUSEUM AT NIGHT',
  2: 'THE HIGHSCHOOL',
  3: 'AREA 51'
}

setType = setSwitcher.get(setNum, "invalid")

#plot dictionary
plotSwitcher={
  1: 'A PRICELESS JEWEL IS STOLEN',
  2: 'THE PRINCIPAL GOES MISSING',
  3: 'UNEXPLAINED PHENOMENA TAKING PLACE'
}

plotType = plotSwitcher.get(plotNum, "invalid")
```

functions within the previous functions, however these are longer and a little more complicated. You'll see variables set as global – do you remember why we do that? Random and randrange are used many times for the character, setting and plot variables – you'll see that they are used in tandem with the character, setting and plot dictionaries. Do you remember why that works? If you look at mys_roll() and hor_roll(), you will notice the code is pretty identical aside from the words found in the print statements. Those two functions are using the same concepts to randomly select a character, setting and plot from whatever genre was randomly selected.

- On line 141 is the last function, parts(). Believe it or not, this is the function that sets up the game. You will see print statements, the count() function being called, time.sleep() being used, and .format populating print statements with different variables. You have already seen how all of this works in the previous code.
- Starting on line 164, there are a series of 4 functions being called. What would happen if we deleted that code or commented it out (by putting hashtags in front of them)? The answer is…nothing! Remember what calling functions does – it runs the code found within those functions. If we delete or comment out any of those four lines, our code that provides the start of the story wouldn't run, the random genre selector wouldn't select a genre, the code that randomly picks a character, setting and plot

wouldn't produce selections, and the story that is found in the parts() function wouldn't print out! Remember, you can create as many functions as you'd like but they won't do anything if you don't call them!

- **Last but not least is code that makes it possible for players to replay the game if they don't like their randomly selected options. Remember in the beginning we had to set the four variables to None? This is so once the player is done with the game, the selections are cleared out and you get new selections. However, notice the words while True, if, elif and else are in the code – what are those doing? Those set up a while loop which will continue to run the functions genre_roll(), gen() and parts() as long as the user types the letter "y" for yes. If the user types in a 'n', the program prints off a goodbye statement. Do you see what that is? What happens if the user types in the word "yes" instead of a "y"?**

If you are not a fan of mysteries or horror, what could you do to easily edit the program to give players the chance to roll Science Fiction or Fantasy characters, settings and plots? What part of the code would you have to edit?

So, what if we want to add to the "Roll a Story" program and make it more complex? What could we do to improve this code? There is another extremely important concept in programming that would make this code more concise and you could easily add additional story elements to make the game more robust. That concept is called OOP – Object Oriented Programming, a type of programming paradigm. The program we created was an example of a Functional programming paradigm.

In functional programming, the code is split into functions. In effect, the function of the program is more important than the data. Very often, our goals as programmers should be to develop concise code divided into classes and objects that can be reused and easily managed. For example, had we used classes and objects in this program, it would allow us to skip on all of the nested if statements (which can become quickly confusing and easy to make mistakes). Although a little more difficult to create, the program would be easier to manage in the long run. OOP can also be used in other popular languages such as JavaScript, Java and C# and should be the next step in your programming journey.

Learn More

REPL.IT QUICK START GUIDE

https://docs.repl.it/misc/quick-start

ROLL A STORY PROGRAM

https://repl.it/@brivera/rollastory

ROLL A STORY WRITING ACTIVITY

https://www.teacherspayteachers.com/Product/FREE-Roll-a-Story-Writing-Activity-286634

DIGITAL WRITING ACTIVITY

https://www.thetechieteacher.net/2019/10/roll-story-digital-writing-activity.html

ROLL A STORY FREEBIE

https://crazyspeechworld.com/2014/06/roll-a-story-freebie.html

ROLL A DICE LITERARY FUN

https://msjordanreads.com/2012/08/11/roll-a-dice-literacy-fun/

ROLL THE DICE

https://repl.it/@amber192/Roll-the-dice#main.py

INTERACTIVE WRITING GAMES

https://classroom.synonym.com/interactive-games-narrative-writing-6140501.html

ROLL THE DICE WITH PYTHON

https://medium.com/@diskokarl/how-to-roll-dice-with-python-34865d83f53d

About the Author

Bianca Rivera
Bianca is a school librarian at East Islip School District where she leads a Technology Club for grades 3-5 students.