Girls' Programming Network

Markov Chains

Workbook 1

Tutors Only

This project was created by GPN Australia for GPN sites all around Australia!

This workbook and related materials were created by tutors at:

Sydney, Canberra and Perth

Girls' Programming Network

If you see any of the following tutors don't forget to thank them!!

Writers

Alex McCulloch
Maddy Reid
Renee Noble
Caitlin Mangan
Sheree Pudney

Testers

Vivian Dang
Annie Liu
Anton Black
Georgia Kirkpatrick-Jones
Olivia Gorton
Leesandra Fernandes
Josie Ang
Isabel Brison
Maddie Jones
Kassandra di Bona
Ashleigh Pribilovic

A massive thanks to our sponsors for supporting us!

ATLASSIAN

Part 1: Welcome message

Task 1.1: Print a message

At the top of the file use a comment to write your name!

We want to print a message to tell the user what our program does.

1. On the line after your name, use the `print` statement to display the following message:

```
I am a markov chain generator
```

2. Now **run your program** to see what happens!

Hint

Any line starting with # is a comment.

```
# Amy
```

You can print out a greeting like this:

```
print("Hello, world")
```

TUTOR TIPS

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 2:

- ☐ Add a comment with your name
- ☐ Print a message to the user

TUTOR TIPS

The code should look like this:

```
# <the student's name>
print("I am a markov chain generator")
```

Part 2: The first word

Task 2.1: Get the user to choose the first word

1. Use `input` to ask the user for the first word in our sentence, and store their answer in a variable called `current_word`.

2. Use a `print` statement to display the `current_word`.

3. When you **run your program**, you should see something like this:

```
I am a markov chain generator
What word do you want to start with? a
a
```

Hint

You can get information from the user using `input`:

```
name = input("What is your name? ")
```

This will put the what the user entered into the a variable called `name`.

TUTOR TIPS

- You need a space at the end of the **input** string, or the prompt will be squished together with the user's answer like this:
  `What word do you want to start with?oh no`
- Make sure they're printing the **variable** `current_word` and not the **string** `"current_word"`

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 3:

☐ Get your program to print the word the user enters

TUTOR TIPS

The code should look like this:

```
# <the student's name>
print("I am a markov chain generator")
current_word = input("What word do you want to start with? ")
print(current_word)
```

Part 3: What comes next?

For now we are going to use this text for our program!

one fish two fish red fish blue fish
this one has a little car

Task 3.1: Let's figure it out

Fill in this table with what words come after it in the example text

one	fish has
two	fish
red	fish
blue	fish
this	one
has	a
a	little
little	car
car	one
fish	two red blue this

Hint

The last word in the text (car) should connect to the first word (one) so that we don't just stop when we get to car

TUTOR TIPS

- Make sure they fill in car properly with the first word in the text

Task 3.2: What could come after current_word?

one fish two fish red fish blue fish
this one has a little car

Let's write an if statement that would figure out what words could come after the word "one"

1. Create a variable called next_word_options that should currently store an empty list. We will be adding to this list but only if the current word is "one"

2. Write an if statement that checks if the current word is the word "one"

3. Inside that if statement, create a variable called "next_word_options" and store the list of possible words that come after the word "one" in our text above into the list (you already worked this out in 3.1)

4. Print next_word_options outside the if statement

5. Run your program, enter "one" as your starting word, and make sure that it prints out the words that come after "one"

Hint

Have a look at the table you filled out in the last task to see what next_word_options should be.

Example of an if statement below. Remember to use == in your if line, and = to assign a value to a variable.

```
if name == "Alex":
    message = "I love your name"
```

Remember that a list looks like this:

```
animals = ["dog", "cat"]
```

TUTOR TIPS

(1) empty list looks like this [ ]
(5) should print out ["fish", "has"] - these are the words that come after "one" in text.
* If student inputs a word that is not "one", they will get a NameError! Error is expected and fixed later. Make sure spell "one" correctly and don't use capitals

Task 3.3: Now for all the words!

1. We need an if statement for all the different words in the first column of the table from 3.1 ! Let's make those now.

2. Make sure you just have 1 print statement at the end after the if statements like this:

Note: this is an example - if you use this exact code it **won't work**

```
if name == "Alex":
    message = "I love your name"
if name == "Lyndsey":
    message = "Your name is awesome"

print(message)
```

3. Run your program multiple times, entering other words from the text as your starting word. Make sure what prints is correct.

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 4:

- ☐ You have filled in the table in Task 3.1
- ☐ You have an if statement for the current word equal to "one"
- ☐ You have a variable named next_word_options that has a list of possible words
- ☐ You've run your program using "one" as your starting word
- ☐ You have an if statement for each word in the first column of the table from 3.1
- ☐ You've run your program using other words from the text as your starting word.

TUTOR TIPS

The code should look like this:

```
# <the student's name>
print("I am a markov chain generator")
current_word = input("What word do you want to start with? ")
print(current_word)

if current_word == "one":
    next_word_options = ["fish", "has"]
if current_word == "two":
```

```
    next_word_options = ["fish"]
if current_word == "red":
    next_word_options = ["fish"]
if current_word == "blue":
    next_word_options = ["fish"]
if current_word == "this":
    next_word_options = ["one"]
if current_word == "has":
    next_word_options = ["a"]
if current_word == "a":
    next_word_options = ["little"]
if current_word == "little":
    next_word_options = ["car"]
if current_word == "car":
    next_word_options = ["one"]
if current_word == "fish":
    next_word_options = ["two", "red", "blue", "this"]

print(next_word_options)
```

★ BONUS 3.4 Lowercase

What if the user puts in “One” or “ONE” or “Fish” or “FiSh” as their start word?

Case is important in Python (“One” is not the same as “one”), so we want to make the word the user enters lowercase too!

1. Use `.lower()` on the input we get from the user to convert any uppercase letters in the word they enter to lowercase

2. **Run your program** and make sure the output is correct even if the word you enter has some capital letters (like “oNE”)

Python has a shortcut for making things lowercase, here is an example:

```
>>> name = input("What's your name? ")
KeLlY
>>> name = name.lower()
>>> print(name)
kelly
```

TUTOR TIPS

The code should look like this:

```
# <the student's name>
print("I am a markov chain generator")
current_word = input("What word do you want to start with? ")
current_word = current_word.lower()
print(current_word)

if current_word == "one":
    next_word_options = ["fish", "has"]
if current_word == "two":
    next_word_options = ["fish"]
if current_word == "red":
    next_word_options = ["fish"]
if current_word == "blue":
    next_word_options = ["fish"]
if current_word == "this":
    next_word_options = ["one"]
if current_word == "has":
    next_word_options = ["a"]
if current_word == "a":
    next_word_options = ["little"]
if current_word == "little":
    next_word_options = ["car"]
if current_word == "car":
    next_word_options = ["one"]
if current_word == "fish":
    next_word_options = ["two", "red", "blue", "this"]

print(next_word_options)
```

Part 4: Making choices

Task 4.1: Import Random Library

To get access to cool random things we need to import random!

1. At the top of your file, below your name, add this line:
   `import random`

Task 4.2: Choose a random word!

1. After all the if statements, randomly choose a word from `next_word_options`, and put that word in a new variable called `next_word`
2. `print` out `next_word`
3. Delete where you print next_word_options
4. **Run your program** several times, using **'fish'** as the starting word each time. Keep going until you get 3 different words.

Hint

If you want to choose an option from a list, you can use `random.choice` like this:

```
food_options = ["Pizza", "Chocolate", "Pasta", "Dumplings"]
dinner = random.choice(food_options)
print(dinner)
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 5:

☐ The program randomly chooses a next word and prints it out

☐ When you run the program multiple times with the word "fish" you get 3 different words

TUTOR TIPS

The code should look like this:

```
# <the student's name>
import random
print("I am a markov chain generator")
current_word = input("What word do you want to start with? ")
print(current_word)

if current_word == "one":
    next_word_options = ["fish", "has"]
if current_word == "two":
    next_word_options = ["fish"]
if current_word == "red":
    next_word_options = ["fish"]
if current_word == "blue":
    next_word_options = ["fish"]
if current_word == "this":
    next_word_options = ["one"]
if current_word == "has":
    next_word_options = ["a"]
if current_word == "a":
    next_word_options = ["little"]
if current_word == "little":
    next_word_options = ["car"]
if current_word == "car":
    next_word_options = ["one"]
if current_word == "fish":
    next_word_options = ["two", "red", "blue", "this"]

next_word = random.choice(next_word_options)

print(next_word)
```

For Loops

Part 5: Even more words

Task 5.1: Now let's do that 100 times!

That was so much fun we're going to do it 100 times!

1. Before your first if statement, create a for loop that will run 100 times
2. Indent all the code after your for loop so it is inside the for loop (check out the hints below for indenting lots of lines quickly).
3. Run your code. You might have to make your Console window bigger and scroll through it to see all the 100 words you get!

A **for loop** that runs 100 times looks like this:

```
for i in range(100):
    # The thing you want to do 100 times goes here
```

Hint

When we put some code in an `if` or a `for` loop, we have to **indent** it. Indented lines have a tab at the start like this, only the indented things get repeated:

```
for blah in something:
    # THIS IS INDENTED
```

An easy way to do this is to select all the code you want to indent and then press CTL +] or TAB

Task 5.2: Too many lines !!!

Woah, that's hard to read!! Let's make it print on one line only!

When we use print("blah blah blah") it automatically adds an ENTER to end the line so the next line prints on a new line! We don't want that.

We can tell it to end the line with something different than ENTER. When we end the line with something different than ENTER, it will keep the next print in the same line!

The example below makes the print end with a space (so this will put a space after what it prints AND keep the next print on the same line!)

1. Change the ending symbol of your print to a space (end=" ") to make it print on one line! Don't forget to do it for when you print the first word too!
2. Run your code. Are your words now one after the other, with a space between them, and not on separate lines?

```
>>> print("blah blah blah", end=" ")
>>> print("BLAH", end=" ")
blah blah blah BLAH
```

Task 5.3: Almost … but not quite

Run your program a few times using 'one' as the starting word. What do you notice? Does it only contain the words 'fish' and 'has'?

Something's not quite right - we're always choosing from the first word ('one') so we're only getting words that normally come after the word 'one'!

Here's what the options for 'one' look like:

```
"one" = ['fish', 'has']
```

And here's what happens when we run our program starting with 'one'. We're only getting words that normally come after the word 'one':

```
I am a markov chain generator
What word do you want to start with? one
one has has has fish has fish has has has fish has fish fish
fish has has fish fish has has has fish fish fish fish fish has
fish fish fish fish fish has fish fish fish fish has has fish
has fish has fish has fish fish has has fish fish fish has fish
fish fish fish fish has has fish fish has has fish has has fish
fish has fish has has fish fish has fish has fish fish has fish
fish has has has fish fish fish fish fish has has fish has fish
has fish has fish
```

To fix the problem, we have to make sure we always look at the last word we printed. That means we have to update what the new current word is each time we print the next word.

1. At the end of your for loop after you print next_word, add a line to set `current_word` to be the `next_word` we just printed (make sure it is still indented in your for loop)

2. When you run your program you should look a little like one of these! It should be different every time!

```
I am a markov chain generator
What word do you want to start with? one
one has a little car one has a little car one fish red fish
blue fish red fish two fish this one fish two fish red fish red
fish two fish red fish this one fish this one has a little car
one has a little car one fish red fish blue fish blue fish this
one has a little car one fish red fish blue fish two fish red
fish blue fish blue fish this one has a little car one fish
this one fish two fish red fish red fish red fish two fish two
fish red fish red
```

```
I am a markov chain generator
What word do you want to start with? one
one blue fish blue fish red fish red fish blue fish red fish
blue fish red fish blue fish two fish red fish this one fish
two fish blue fish this one fish red fish two fish this one has
a little car one fish this one fish two fish this one fish red
fish red fish this one fish this one has a little car one has a
little car one fish red fish blue fish blue fish red fish blue
fish blue fish red fish this one has a little car one has a
little car one fish blue fish
```

Your program is writing the next word in the sentence based on the previous word to make a full sentence!

☑ CHECKPOINT ☑

If you can tick all of these off you can move to part 6!

☐ Your programs prints a silly sentence with 101 words (the starting word + 100 more)

☐ Each word is based on the previous word

TUTOR TIPS

The code should look like this:

```
# <the student's name>
import random
print("I am a markov chain generator")
current_word = input("What word do you want to start with? ")
print(current_word, end=" ")

for i in range(100):
  if current_word == "one":
      next_word_options = ["fish", "has"]
  if current_word == "two":
      next_word_options = ["fish"]
  if current_word == "red":
      next_word_options = ["fish"]
  if current_word == "blue":
      next_word_options = ["fish"]
  if current_word == "this":
      next_word_options = ["one"]
  if current_word == "has":
      next_word_options = ["a"]
  if current_word == "a":
      next_word_options = ["little"]
  if current_word == "little":
      next_word_options = ["car"]
  if current_word == "car":
      next_word_options = ["one"]
  if current_word == "fish":
      next_word_options = ["two", "red", "blue", "this"]

  next_word = random.choice(next_word_options)
  time.sleep(0.1)
  print(next_word, end=" ")
  current_word = next_word
```

★ BONUS 5.4: Not so fast!!

This would look cooler if the computer wrote our story 1 word at a time. At the moment the computer goes so fast, it looks like it appears all at once!

1. At the top of your file write `import time`
   This will let us use what we need to make our program sleep for a few seconds.

2. Before any `print`, add a line that says `time.sleep(0.1)`
   This will make our program 'sleep' for a tenth of a second! You can adjust it to any time you want.

3. In your print lines that are after time.sleep(0.1), add ,`flush=True` after end=" "
   (make sure it's inside the brackets)

We're using **flush** as print statements will usually automatically store every print until there's a new line character, and since we don't want it to wait, we want it to print every time, we use **flush** to clear the output storage by printing.

★ BONUS 5.5: How many words?

Currently we're always generating 100 words, but what if the user wants more or less? We can ask them how many we want...

1. After your input statement asking what word they want to start with, ask the user how many words they want to generate and store it in a variable called `generate`
2. Now, where you currently have `for i in range(100)` replace the 100 with our new variable
3. Check your work by running your code with different numbers

Hint

We will need to convert the input from a string to an int by using
`int(input("Question"))`

★ BONUS 5.6: Poetic Outputs

Currently our program outputs a lot of words all in one chunk, this makes it hard to read, it's also pretty boring. Lets use some extra prints to make our outputs look more like a poem.

To do this we will need to change some things

1. Before our `for` loop, make a new variable called `width_count` and set it to 1. We will be using this to set how many words can be in one line. It has to start at one because of us printing `current_word` before the loop

2. Now at the top of the code inside your for loop (before any of your ifs), check `if width_count = 10`

3. If it does, use `print()` so the next word is printed on a new line and reset `width_count` to zero

4. Lastly, at the bottom of the code inside our `for` loop add one to `width_count`

★ BONUS 5.7: Poetic Outputs pt. 2

You will need to have done Bonus 5.5 and 5.6 before you do this.

Let's make it so the user gets to choose how many words are in a line.

1. After your input statement asking how many words they want to generate, ask the user how many words they would like on each line and store it in a variable called `max_width`

2. Now, change the if statement you wrote in 5.6 to to check if `width_count == max_width`

Hint

We will need to convert the input from a string to an int by using `int(input("Question"))`

★ BONUS 5.8: No more fish

Let's change it up so we're no longer using the one fish blue fish text.

Find another text you want to use - you can use parts of a song or a book you like. To make this easier on yourself, you should aim to not have many words in your text.

1. Using some plain paper, break down your new text the same way we did for one fish blue fish text in 3.1 (ie for every word in your text, write down the words that come after it in your text).

2. Now, change all of your if statements so that they're checking the words from your new text.

3. Try running this and see if the responses you're getting are different!

TUTOR TIPS (BONUS 5.4, 5.5, 5.6, 5.7)

For 5.4 make sure they have a comma between end=" " and flush = True in the print
For 5.5 make sure they have used **int** to convert **generate** to number for use in loop
For 5.6 make sure they increment **width_count** still inside the for loop

The code should look like this:

```
# <the student's name>
import random
import time

print("I am a markov chain generator")
current_word = input("What word do you want to start with? ")
generate = int(input("How many words do you want to generate? "))
max_width = int(input("How many words would you like on each
line? "))
time.sleep(0.1)
print(current_word, end=" ", flush=True)

width_count=1
for i in range(generate):
  # Bonus 5.6 (replaced in 5.7)if width_count == 10:
  if width_count == max_width:
      print()
      width_count=0
  if current_word == "one":
      next_word_options = ["fish", "has"]
  if current_word == "two":
      next_word_options = ["fish"]
  if current_word == "red":
      next_word_options = ["fish"]
```

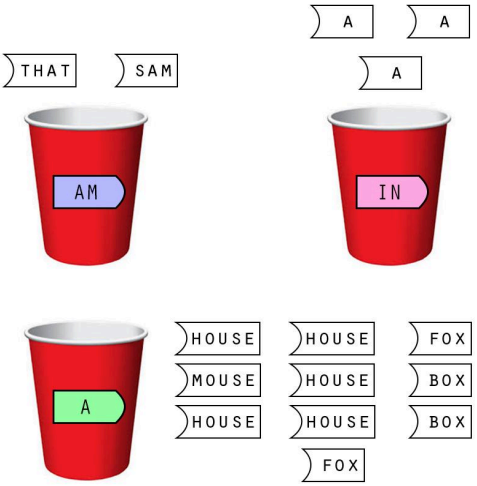
```
if current_word == "blue":
    next_word_options = ["fish"]
if current_word == "this":
    next_word_options = ["one"]
if current_word == "has":
    next_word_options = ["a"]
if current_word == "a":
    next_word_options = ["little"]
if current_word == "little":
    next_word_options = ["car"]
if current_word == "car":
    next_word_options = ["one"]
if current_word == "fish":
    next_word_options = ["two", "red", "blue", "this"]

next_word = random.choice(next_word_options)
time.sleep(0.1)
print(next_word, end=" ", flush=True)
current_word = next_word
width_count = width_count + 1
```

Part 6: That's a lot of ifs!

Our program works great but what if we wanted to use the Green Eggs and Ham text? That has over 30 unique words in it and we would have to write an if statement for each one… That's a lot of ifs!

Remember the cups game from the start of the day, let's do it with programming!



Instead of actual cups, our program will use a **Python dictionary** that tells us what's in the cup for each word. The label on each cup is the **key** of our dictionary. The matching **value** is a `list` of all the words that can come after it!

We'll give you all the cups/words for the text we've been using to get you started

one fish two fish red fish blue fish
this one has a little car

Task 6.1: Create the word cups

1. **The dictionary for our text looks like this:**

```
cups = {"one": ["fish", "has"],
        "two": ["fish"],
        "red": ["fish"],
        "blue": ["fish"],
        "this": ["one"],
        "has": ["a"],
        "a": ["little"],
        "little": ["car"],
        "car": ["one"],
        "fish": ["two", "red", "blue", "this"]}
```

2. **Copy and paste** the dictionary into your code after you print `current_word`. It is assigned to a **variable called `cups`.**

Hint: Understanding the cups dictionary

Each cup has a key-value pair (e.g. `'one': ['fish', 'has']`). The **key** (e.g. `'one'`) is the label on the cup and the **value** (e.g. `['fish', 'has']`) is a `list` of words inside the cup. Remember the words inside the cup are all the words that would possibly come next (after the label word).

Task 6.2: Let's look it up

Next we want to look up our `current_word` in our cups dictionary to find what could come next in the sentence we are making instead of using our if statements!

1. Delete or comment out all the if statements (I know it's a lot of code but trust me!)
2. Inside our for loop, look the `current_word` key up in the `cups` dictionary, and put the value in the variable called `next_word_options`.
3. **Run your program** to make sure that it still works the same as before

Hint

You can look at the **value** of a **key** in a dictionary using square brackets like this:

```
pet_names = {'dog': ['Spot', 'John'],
             'cat': ['Snowball', 'Shadow'],
             'mouse': 'Mr Squeaks'}
dog_name_suggestions = pet_names['dog']
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 7:

- ☐ You have a variable with a dictionary of the cups
- ☐ You have another variable containing a list of all possible next words
- ☐ You've run your program using a few different starting words and seen that it works the same as it did with the ifs

★ BONUS 6.3: Green Eggs and Ham

You can make your program work with any different texts now just by changing the cups dictionary!

Try changing the dictionary to the one in this link: girlsprogramming.network/markov-files

It should look like what we were doing this morning with Green Eggs and Ham

TUTOR TIPS

The code should look like this:
Note that the cups dictionary will be different if they've done Bonus 6.4

```
# <the student's name>
import random

print("I am a markov chain generator")
current_word = input("What word do you want to start with? ")
print(current_word, end=" ")

cups = {"one": ["fish", "has"],
        "two": ["fish"],
        "red": ["fish"],
        "blue": ["fish"],
        "this": ["one"],
        "has": ["a"],
        "a": ["little"],
        "little": ["car"],
        "car": ["one"],
        "fish": ["two", "red", "blue", "this"]}

for i in range(100):
  next_word_options = cups[current_word]
  next_word = random.choice(next_word_options)
  print(next_word, end=" ")
  current_word = next_word
```

Part 7: I don't know that word!

Task 7.1: What if current_word doesn't exist?

Run your program using a starting word that is **not** in the `cups` dictionary, like `"dog"`.

You should get an error... but don't worry! we'll fix that

Task 7.2: Check that current_word actually exists

We don't want an error if the user enters a starting word that is not in the `cups`!

1. **Use an `if` statement** before the loop to make sure the **for** loop only runs **if** `current_word` is `in` the `cups` dictionary.

2. Make sure you indent your **for** loop and everything in it inside of the **if**

3. **Run your program** again using a starting word that is not in the `cups` dictionary, such as `"dog"`. What happens now?

Hint

You can check if something exists in a dictionary using `if` ... `in` ... like this:

```
pet_names = {'dog': ['Spot', 'John'], 'mouse': 'Mr Squeaks'
             'cat': ['Snowball', 'Shadow']}
animal = input('Which animal do you have? ')
if animal in pet_names:
    print('I have some great name suggestions!')
```

☑ CHECKPOINT ☑

If you can tick all of these off you're done!

☐ Your program doesn't give an error, even if the user enters a word not in the cups dictionary

TUTOR TIPS

The code should look like this:

```
# <the student's name>
import random

print("I am a markov chain generator")
current_word = input("What word do you want to start with? ")
print(current_word, end=" ")
cups = {"one": ["fish", "has"],
        "two": ["fish"],
        "red": ["fish"],
        "blue": ["fish"],
        "this": ["one"],
        "has": ["a"],
        "a": ["little"],
        "little": ["car"],
        "car": ["one"],
        "fish": ["two", "red", "blue", "this"]}

if current_word in cups:
  for i in range(100):
    next_word_options = cups[current_word]
    next_word = random.choice(next_word_options)
    print(next_word, end=" ")
    current_word = next_word
```

★ BONUS 7.3:

If the user enters a starting word not in our cups dictionary, nothing happens!

Add an `else` statement to the `if` you added in Part 7.1. Use a `print` statement (remember your indenting!) so that if the user tries a word not in the dictionary, they will see a message like this:

```
Sorry, you can't use that as a starting word!
```

Run your program and enter a word not in the dictionary.

The word and message print on the same line - a bit messy! Make the message print on a new line by using **\n** (= newline) at the start of the message like this:

```
print("\nSorry, you can't use that as a starting word.")
```

TUTOR TIPS

The code should look like this:

```
# <the student's name>
import random

print("I am a markov chain generator")
current_word = input("What word do you want to start with? ")
print(current_word, end=" ")
cups = {"one": ["fish", "has"],
        "two": ["fish"],
        "red": ["fish"],
        "blue": ["fish"],
        "this": ["one"],
        "has": ["a"],
        "a": ["little"],
        "little": ["car"],
        "car": ["one"],
        "fish": ["two", "red", "blue", "this"]}

if current_word in cups:
  for i in range(100):
    next_word_options = cups[current_word]
    next_word = random.choice(next_word_options)
    print(next_word, end=" ")
    current_word = next_word
else:
  print("\nSorry, you can't use that as a starting word.")
```