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	Testers
Alex McCulloch	Vivian Dang
Maddy Reid	Annie Liu
Renee Noble	Anton Black
Caitlin Mangan	Georgia Kirkpatrick-Jones
Sheree Pudney	Olivia Gorton
	Leesandra Fernandes
	Josie Ang
	Isabel Brison
	Maddie Jones
	Kassandra di Bona
	Ashleigh Pribilovic

Part 0: Setting up

Task 0.1: Making a python file

Open the start menu, and type 'IDLE'. Select IDLE 3.5.

1. Go to the file menu and select 'new file'. This opens a new window.
2. Go to the file menu, select 'Save as' and save the file as 'markov_chain.py'

TUTOR TIPS

- **DOUBLE CHECK IT'S PYTHON 3**
- Make sure they saved it as **.py** or **they won't get syntax highlighting**
- Suggest they save it **somewhere they can find easily**, e.g. Desktop

Task 0.2: You've got a blank space, so write your name!

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

Hint

Any line starting with # is a comment.
```
# This is a comment
```

☑ CHECKPOINT ☑

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

☐ You should have a file called markov_chain.py

☐ Your file has your name at the top in a comment

☐ Run your file with the F5 key and it does nothing!!

TUTOR TIPS

The code should look like this:
```
# <the student's name>
```

Part 1: Welcome message

Task 1.1: Print a message

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

You can print out a greeting like this:
```
print("Hello, world")
```

TUTOR TIPS

You may need to remind them to use the **F5 key** to run the program

☑ CHECKPOINT ☑

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

☐ Print a message to the user
☐ Try running your code with the F5 key!

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 name the user enters into the 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

☐ Run your program a few times and input a different word each time

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 3.2: What could come after current_word?

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

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

2. Inside that if statement, create a variable called "next_word_options" and store the list of possible words that come after the word "one"

3. Print next_word_options outside the if statement

4. Run your program, enter "one" as your starting word, and make sure that it prints out ["fish", "has"]

Hint

Have a look at the table you filled out in the last task to see what next_word_options should be.
Remember that a list looks like this:

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

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! 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, entering other words from the first column of the table, and make sure what prints is correct.

Hint

Make sure you use **==** in your if line and **=** to assign a value to a variable.

☑ 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

☐ You've run your program using other words from the first column of the table as your starting word.

TUTOR TIPS

The code should look like this:

```python
# <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)
```

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":
```

```python
    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 such as `next_word`.
2. `print` out `next_word`
3. Delete where you print next_word_options

Hint

If I wanted to choose a random food for dinner, I could use the `choice` function in the Python `random` library, like this:

```
dinner = random.choice(["pizza", "chocolate", "nutella", "lemon"])
```

Task 4.3:

Run your program several times, using **'fish'** as the starting word each time, until you get 3 different words. **Write them down below!!**

Word #1: ……………………………………

Word #2: ……………………………………

Word #3: …………………………………....

☑ 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

☐ Fill in the next word after 'fish' for 3 different runs of your program

The code should look like this:

```python
import random

# <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"]

next_word = random.choice(next_word_options)

print(next_word)
```

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!

Put all your code about choosing the next word (that's your code from Part 3 and 4) into a for loop that runs 100 times.

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 the tab key

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 what to end the line with. This makes it end with 3 exclamation marks!

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

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!

Task 5.3: Almost … but not quite

Run your program a few times using 'one' as the starting word. What do you notice?

Something's not quite right. We're always choosing from the first word ('one')! We need to update the current_word so we pick from the next word!

Hint

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
```

Task 5.4: Get the right word

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

1. At a line at the end of your for loop (don't forget your indenting!) set `current_word` to be the `next_word` we just printed.

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:

```python
import random

# <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, 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)
    print(next_word, end=" ")
    current_word = next_word
```

★ BONUS 5.5: 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.

TUTOR TIPS

The code should look like this:

```python
import random
import time

# <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, 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
```

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 to get you started
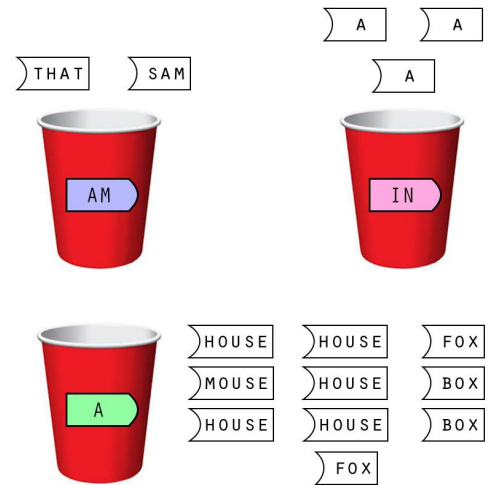
Task 6.1: Create the word cups

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

```python
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`.**
3. If you don't want the dictionary definition taking up so much room, you can select it, go to the top menu bar, click Format -> Format Paragraph.

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']
```

Task 6.3: What if current_word doesn't exist?

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

Which of the following errors do you see? (We'll fix that later!)
- A. **KeyError**   **<-- Tutors: It's this one.**
- B. IndexError
- C. TypeError

☑ 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

☐ You know what kind of error happens if you start with a word that's not in cups

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

```python
import random

# <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, 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: 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** 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:

```python
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:

```python
import random

# <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, 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 6.3:

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 3.5. 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:

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

The code should look like this:

```python
import random
# <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, 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.")
```