Girls' Programming Network

Scissors Paper Rock!

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

<table>
<tr><td>**Writers**</td><td>**Testers**</td></tr>
<tr><td>Sarah Mac</td><td>Catherine Murdoch</td></tr>
<tr><td>Renee Noble</td><td>Maddie Jones</td></tr>
<tr><td>Vivian Dang</td><td>Sheree Pudney</td></tr>
<tr><td>Courtney Ross</td><td></td></tr>
</table>

A massive thanks to our sponsors for supporting us!

Part 0: Setting up

Task 0.1: Making a python file

1. Go to https://replit.com/
2. Sign up or log in
   (we recommend signing in with Google if you have a Google account)

Task 0.2: Making a python file

1. **Create** a new project
2. Select **Python** for the template
3. Name your project **scissors_paper_rock**

TUTOR TIPS

Make sure the AI is turned off (the slides should prompt them)

Task 0.3: You've got a blank space, so write your name!

A main.py file will have been created for you!

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

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

2. Run your code using the ▶ **Run** button. It **won't** do anything yet!

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 1:

☐ You should have a file called main.py

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

☐ Run your file and it does nothing!

TUTOR TIPS

The code should look like this (no bonuses):

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

Part 1: Welcome Message

Task 1.1: Print a welcome and the rules

Welcome the player and print the rules!

Use a print to make it happen when you run your code:

```
------------------------------------------------------
Welcome to Human vs. Computer in Scissors, Paper, Rock!
------------------------------------------------------
Moves: choose scissors, paper or rock by typing in your
selection.
Rules: scissors cuts paper, paper covers rock and rock
crushes scissors.
Good luck!
------------
```

TUTOR TIPS

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

Hint

Want to print multiple lines at a time? You can use three sets of quotes instead of one, to make your strings go over multiple lines

```python
print("""
Print
Three
Lines
""")
```

☑ CHECKPOINT ☑

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

☐ Print a welcome

☐ Print the rules

☐ Try running your code!

Students may use multiple print statements or multi-line strings

The code should look like this (no bonuses):

```python
# <the student's name>
print("""
----------------------------------------------------
Welcome to Human vs. Computer in Scissors, Paper, Rock!
----------------------------------------------------
Moves: choose scissors, paper or rock by typing in your selection.
Rules: scissors cuts paper, paper covers rock and rock crushes
scissors.
Good luck!
-----------
""")
```

2. Who played what?

Task 2.1: Make the computer play the same move every time!!

Make a variable for the computer's move such as `computer_move`, set it to
`"scissors"`, `"paper"` or `"rock"`.

Task 2.2: Ask the human for their move

Use `input` to ask the human for their move and save their answer in a variable, name it
something like `human_move.`

It should look like this when you run your code:

```
-----------------------------------------------------
Welcome to Human vs. Computer in Scissors, Paper, Rock!
-----------------------------------------------------
Moves: choose scissors, paper or rock by typing in your
selection.
Rules: scissors cuts paper, paper covers rock and rock
crushes scissors.
Good luck!
-----------

 What is your move? scissors, paper or rock?
```

Task 2.3: Print out the moves

Print out the moves the computer and the human have played.

It should look like this when you run your code:

```
-------------------------------------------------------
Welcome to Human vs. Computer in Scissors, Paper, Rock!
-------------------------------------------------------
Moves: choose scissors, paper or rock by typing in your
selection.
Rules: scissors cuts paper, paper covers rock and rock
crushes scissors.
Good luck!
-----------

What is your move? scissors, paper or rock? scissors

 Computer Played: paper
 Human Played: scissors
```

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

☐ Set a move for the computer

☐ Ask the human to type in their move and store it in a variable

☐ Print out the human and computers moves

☐ Run your code!

TUTOR TIPS

The code should look like this (no bonuses):

```python
# <the student's name>
print("""
-------------------------------------------------------
Welcome to Human vs. Computer in Scissors, Paper, Rock!
-------------------------------------------------------
Moves: choose scissors, paper or rock by typing in your selection.
Rules: scissors cuts paper, paper covers rock and rock crushes
scissors.
Good luck!
-----------
""")
# 2.1, 2.2
computer_move = "scissors"
human_move = input("What is your move? scissors, paper or rock? ")
# 2.3
print("Computer Played: " + computer_move)
print("Human Played: " + human_move)
```

★ BONUS 2.4: Not so fast!!

This would look cooler if the computer paused before it said each line!

1) At the top of your file write `import time`
This will let us use what we need to use 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. **Try putting sleep between your print statements!**

Waiting for the next lecture? Try adding this bonus feature!!

1. At the start of the game ask the human to enter their name. Store it in a variable (maybe use `player_name`)

2. Change your other code so that every time it says "Human" it prints the player's name instead!

Remember you can add a variable to some text like this:
`"Hello " + player_name`

TUTOR TIPS

The code should look like this (with bonuses):

```
import time
# <the student's name>
print("""
--------------------------------------------------------
Welcome to Human vs. Computer in Scissors, Paper, Rock!
--------------------------------------------------------
Moves: choose scissors, paper or rock by typing in your selection.
Rules: scissors cuts paper, paper covers rock and rock crushes
scissors.
Good luck!
------------
""")
# 2.5
player_name = input("What is your name? ")
print("Hello " + player_name)

computer_move = "scissors"
human_move = input("What is your move? scissors, paper or rock? ")
# 2.4
time.sleep(0.1)
print("Computer Played: " + computer_move)
time.sleep(0.1)
print("Human Played: " + human_move)
```

3. Win, lose or tie?

Let's figure out who won the game!

Task 3.1: What are the different ways to win, lose and tie?

What are all the combinations of how the game could go? Finish this table (there is a printout for you to use).

Human Move 👧	Computer Move 💻	Who Wins? 🏆
scissors	scissors	draw
scissors	paper	human
scissors	rock	computer
paper	scissors	computer
paper	paper	tie
paper	rock	human
rock	scissors	human
rock	paper	computer
rock	rock	tie

Task 3.2: Store the combinations in a dictionary

Create a new dictionary called results, it will store all the data from the table above!

In this dictionary, each key should be a tuple of the human and computer move. The corresponding value will be who wins!

Hint

Make sure the keys are created using (human_move, computer_move), just like in the table above.

```
results = {("paper", "rock"): "human"}
```

Task 3.3: Get the value using the key!

Using our dictionary, store who wins in a variable such as winner. Use the combination of **human_move** and **computer_move** to create the key to get the value!

Then **print** out the winner to the screen!

Hint

Remember we can look things up in the dictionary with a *tuple*. Tuples go inside round brackets and can be multiple items.

If we had a dictionary of holidays by month and date like this:
```
holidays = {("December", 25): "Christmas", ("April", 25): "ANZAC Day",
            ("January", 1): "New Year's Day"}
```

we might look things up like this: `holidays[(month, date)]`
`holidays[("January", 1)]` would return `"New Year's Day"`

Hint

Remember that (human_move, computer_move) is not the same as (computer_move, human_move)! Order is important.

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 4:
☐ Create a dictionary containing every combination of moves
☐ Store who won in a variable

☐ Print out the winner

☐ Run your code and test different moves!

TUTOR TIPS

- Students may need help getting started with the syntax of the results dict, guide them through the first entry
- Students may struggle with the tuple being the key

The code should look like this (no bonuses):

```python
# <the student's name>
print("""
-------------------------------------------------------
Welcome to Human vs. Computer in Scissors, Paper, Rock!
-------------------------------------------------------
Moves: choose scissors, paper or rock by typing in your selection.
Rules: scissors cuts paper, paper covers rock and rock crushes scissors.
Good luck!
------------
""")
computer_move = "scissors"
human_move = input("What is your move? scissors, paper or rock? ")

print("Computer Played: " + computer_move)
print("Human Played: " + human_move)
# 3.2
results = {
    ("scissors", "scissors"): "draw",
    ("scissors", "paper"): "human",
    ("scissors", "rock"): "computer",
    ("paper", "scissors"): "computer",
    ("paper", "paper"): "draw",
    ("paper", "rock"): "human",
    ("rock", "scissors"): "human",
    ("rock", "paper"): "computer",
    ("rock", "rock"): "draw",
}
# 3.3
winner = results[(human_move, computer_move)]
print("The winner is " + winner)
```

★ BONUS 3.4: ROCK Rock rOcK!

Waiting for the next lecture? Try adding this bonus feature!!

We see that **"Rock"** is not the same as **"rock"** and our game only works when it's all in lowercase. Make your game work when players input a move with capital letters such as **"Rock"** or **"sCissors"**.

"FrOg".lower() will return **"frog"**. Try using **.lower()** on your variables to make sure the human players move is converted to lowercase!

TUTOR TIPS

- Students might forget to assign the result of .lower() to a variable
- Students may try and create a new string, show them how to use the lower function to override the variable

The code should look like this (with bonuses):

```python
# <the student's name>
print("""
-------------------------------------------------------
Welcome to Human vs. Computer in Scissors, Paper, Rock!
-------------------------------------------------------
Moves: choose scissors, paper or rock by typing in your selection.
Rules: scissors cuts paper, paper covers rock and rock crushes scissors.
Good luck!
------------
""")
player_name = input("What is your name? ")
print("Hello " + player_name)

computer_move = "scissors"
human_move = input("What is your move? scissors, paper or rock? ")
# 3.4
human_move = human_move.lower()

time.sleep(0.1)
print("Computer Played: " + computer_move)
time.sleep(0.1)
print("Human Played: " + human_move)

results = {
    ("scissors", "scissors"): "draw",
    ("scissors", "paper"): "human",
    ("scissors", "rock"): "computer",
    ("paper", "scissors"): "computer",
    ("paper", "paper"): "draw",
    ("paper", "rock"): "human",
    ("rock", "scissors"): "human",
    ("rock", "paper"): "computer",
    ("rock", "rock"): "draw",
}
winner = results[(human_move, computer_move)]
print("The winner is " + winner)
```

4. Winner, Winner!

It's time to tell the user who won the game!

Task 4.1: If it's a tie!

Use an `if`, `elif` and `else` statements to print out the winner!

You should print different messages based on whether:
- It's a tie
- The human won
- The computer won

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 5:
☐ Your if statement prints "it's a tie" if the moves are the same

☐ Your elif statement prints "human won the game"

☐ Your else statement prints "computer won the game"

☐ Run your code and test different moves!

★ BONUS 4.2: Name the winner!

Waiting for the next lecture? Try adding this bonus feature!!

Update your code so that instead of saying "The winner is human" refer to the human by name, using the name you collect in Bonus 2.5.

The code should look like this (No bonuses (except player name bonus, for bonus 4.2)**):**

```python
# <the student's name>
import time
print("""
--------------------------------------------------------
Welcome to Human vs. Computer in Scissors, Paper, Rock!
--------------------------------------------------------
Moves: choose scissors, paper or rock by typing in your selection.
Rules: scissors cuts paper, paper covers rock and rock crushes scissors.
Good luck!
------------
""")
player_name = input("What is your name? ")
print("Hello " + player_name)

computer_move = "scissors"
human_move = input("What is your move? scissors, paper or rock? ")
human_move = human_move.lower()

time.sleep(0.1)
print("Computer Played: " + computer_move)
time.sleep(0.1)
print("Human Played: " + human_move)

results = {
    ("scissors", "scissors"): "draw",
    ("scissors", "paper"): "human",
    ("scissors", "rock"): "computer",
    ("paper", "scissors"): "computer",
    ("paper", "paper"): "draw",
    ("paper", "rock"): "human",
    ("rock", "scissors"): "human",
    ("rock", "paper"): "computer",
    ("rock", "rock"): "draw",
}
winner = results[(human_move, computer_move)]
# 4.1
if winner == "draw":
    print("It's a tie")
elif winner == "human":
    # 4.2
    print(player_name + "won the game")
else:
    print("Computer won the game")
```

5. Smarter Computer

The computer keeps playing the same move! That's no fun! Let's make the computer choose a random move!

Task 5.1: Import Random Library

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

At the top of your file add this line:
```
import random
```

Task 5.2: Choose a random move!

Find your line of code where you set your computer move, improve this line by choosing a random move.

Choose a random move for the computer using `random.choice` from a list of `"scissors"`, `"paper"` or `"rock"`.

Hint

If I wanted to choose a random food for dinner I could use code like this:

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

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 6:

☐ The computer plays a random move every time.

☐ The line "Computer played: ...." prints different things out!

☐ Try different moves against the computer, does the the correct winner print?

The code should look like this (no bonuses):

```python
# <the student's name>
import random
print("""
--------------------------------------------------------
Welcome to Human vs. Computer in Scissors, Paper, Rock!
--------------------------------------------------------
Moves: choose scissors, paper or rock by typing in your selection.
Rules: scissors cuts paper, paper covers rock and rock crushes scissors.
Good luck!
------------
""")
# 5.2
computer_move = random.choice(["scissors", "paper", "rock"])
human_move = input("What is your move? scissors, paper or rock? ")
human_move = human_move.lower()

print("Computer Played: " + computer_move)
print("Human Played: " + human_move)

results = {
    ("scissors", "scissors"): "draw",
    ("scissors", "paper"): "human",
    ("scissors", "rock"): "computer",
    ("paper", "scissors"): "computer",
    ("paper", "paper"): "draw",
    ("paper", "rock"): "human",
    ("rock", "scissors"): "human",
    ("rock", "paper"): "computer",
    ("rock", "rock"): "draw",
}
winner = results[(human_move, computer_move)]
# 4.1
if winner == "draw":
    print("It's a tie")
elif winner == "human":
    # 4.2
    print("Human won the game")
else:
    print("Computer won the game")
```

★ BONUS 5.3: A picture says a thousand words!

Waiting for the next lecture? Try adding this bonus feature!!

Instead of printing "The human played paper" it would be much cooler to print a picture of a paper! Use ascii art to print images for what the human and computer played!

```
Human plays paper:


|         |
| I AM    |
|A SHEET  |
|OF PAPER|
|_____|
```

1. Go to this link: girlsprogramming.network/ascii And get the pictures for paper, scissors and rock

2. At the top of your code, store each of these ascii images as a string in different variables (maybe rock_pic, paper_pic, etc …)

3. Instead of just printing out the word the human or computer played, also print out the correct picture to match what they played. You might need to use an if statement to figure out which picture to print!

The code should look like this (with bonuses):

```python
# <the student's name>
import random
import time
rock_pic = <ascii art from website>
paper_pic = <ascii art from website>
scissors_pic = <ascii art from website>
# <the student's name>
print("""
THE INTRO TEXT
""")
player_name = input("What is your name? ")
print("Hello " + player_name)
computer_move = random.choice(["scissors", "paper", "rock"])
human_move = input("What is your move? scissors, paper or rock? ")
human_move = human_move.lower()
# 5.3
if computer_move == "paper":
    print(paper_pic)
elif computer_move == "scissors":
    print(scissors_pic)
else:
    print(rock_pic)
time.sleep(0.1)
print("Computer Played: " + computer_move)
# 5.3
if human_move == "paper":
    print(paper_pic)
elif human_move == "scissors":
    print(scissors_pic)
else:
    print(rock_pic)
time.sleep(0.1)
print("Human Played: " + human_move)

results = {
    ("scissors", "scissors"): "draw",
    ("scissors", "paper"): "human",
    ("scissors", "rock"): "computer",
    ("paper", "scissors"): "computer",
    ("paper", "paper"): "draw",
    ("paper", "rock"): "human",
    ("rock", "scissors"): "human",
    ("rock", "paper"): "computer",
    ("rock", "rock"): "draw",
}

winner = results[(human_move, computer_move)]
if winner == "draw":
    print("It's a tie")
elif winner == "human":
    print(player_name + "won the game")
else:
    print("Computer won the game")
```

6. Again, Again, Again!

We want to play Scissors-Paper-Rock more than once! Let's add a loop to play on repeat!

Task 6.1: How many games?

Find out how many games the user wants to play at the start of the game!
Put this after your welcome message!

Hint

Input returns a **string**. Make sure you **convert it to an int** and store it in a variable!
`int("57")` will give you back 57. You can use `int(...)` on a variable too!

Task 6.2: Loop time!

Create a for loop that runs as many times as the user asked for!
You'll need to use:
- A `for` loop
- `range(number_of_games)`

Use this line after you have asked how many games they want to play to start your loop:
`for i in range(number_of_games):`

Task 6.3: Indenting your code

Things we want to do every game need to be indented inside the loop.
We want to ask for a move and check the winner every round!

Hint

Indented lines have a tab at the start like this, they look this:

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

You can indent many lines at once by highlighting them and then hitting the tab key. Make sure you highlight the whole line though!

Task 6.4: GAME OVER!

After all the rounds are played, print out "GAME OVER!".
Make sure this is after your loop and doesn't print every round!

If you can tick all of these off you can go to the Extensions:

☐ Ask the user how many games they want to play

☐ Your game repeats the number of times the user asked for

☐ GAME OVER prints once, after all of the rounds!

TUTOR TIPS

The code should look like this (no bonuses):

```python
# <the student's name>
import random
print("""
-------------------------------------------------------
Welcome to Human vs. Computer in Scissors, Paper, Rock!
-------------------------------------------------------
Moves: choose scissors, paper or rock by typing in your selection.
Rules: scissors cuts paper, paper covers rock and rock crushes scissors.
Good luck!
------------
""")
number_of_games = int(input("How many games would you like to play? "))
# 6.2, 6.3
for i in range(number_of_games):
    computer_move = random.choice(["scissors", "paper", "rock"])
    human_move = input("What is your move? scissors, paper or rock? ")
    human_move = human_move.lower()

    print("Computer Played: " + computer_move)
    print("Human Played: " + human_move)

    results = {
        ("scissors", "scissors"): "draw",
        ("scissors", "paper"): "human",
        ("scissors", "rock"): "computer",
        ("paper", "scissors"): "computer",
        ("paper", "paper"): "draw",
        ("paper", "rock"): "human",
        ("rock", "scissors"): "human",
        ("rock", "paper"): "computer",
        ("rock", "rock"): "draw",
    }
    winner = results[(human_move, computer_move)]
    if winner == "draw":
        print("It's a tie")
    elif winner == "human":
        print(player_name + "won the game")
    else:
        print("Computer won the game")
print("GAME OVER!!!")
```

7. Extension: Keeping Score!

Why play lots of games if we're not even keeping count of who wins?? Let's keep score!

Task 7.1: Counter!

Before your loop create two variables which are going to be your human and computer counters. Start by setting them both to 0.

These will keep track of the human and computer scores throughout the game!

Task 7.2: Add 1!

Every time the computer or human wins we need to add one to the appropriate counter. If it's a tie, neither player gets a point!

Hint

You'll need to add to a counter inside your if/elif statements whenever someone wins!

Task 7.3: And the winner is!

After all the games are played we need to report the overall winner.

Print out how many games the human and computer won each.
Then print out who the overall winner was!

```
--------------------------------
GAME OVER!
Human won 5 games
Computer won 2 games
Human is the winner!!
--------------------------------
```

Hint

Use an `if` statement to compare the scores to calculate the overall winner!

The code should look like this (with bonuses):

```python
# <the student's name>
import random
import time
rock_pic = <ascii art from website>
paper_pic = <ascii art from website>
scissors_pic = <ascii art from website>
# <the student's name>
print("""
WELCOME MESSAGE GOES HERE
""")
number_of_games = int(input("How many games would you like to play? "))
player_name = input("What is your name? ")
print("Hello " + player_name)
# 7.1
human_counter = 0
computer_counter = 0
for i in range(number_of_games):
    computer_move = random.choice(["scissors", "paper", "rock"])
    human_move = input("What is your move? scissors, paper or rock? ")
    human_move = human_move.lower()

    PRINT OF ASCII ART GOES HERE

    time.sleep(0.1)
    print("Human Played: " + human_move)

     results = {
        DICTIONARY CONTENT IN HERE
     }

    winner = results[(human_move, computer_move)]
    if winner == "draw":
        print("It's a tie")
    # 7.2
    elif winner == "human":
        print(player_name + "won the game")
        human_counter += 1
    else:
        print("Computer won the game")
        computer_counter += 1

print("GAME OVER!!!")
print("Human won " + str(human_counter) + " games")
print("Computer won " + str(computer_counter) + " games")
if human_counter == computer_counter:
    print("Draw!")
elif human_counter > computer_counter:
    print("Human is the winner!")
else:
    print("Computer is the winner!")
```

Right now we play a set number of games. But can you figure out how you could change your program to keep playing until a player gets a certain number of points?

You might need to use a while loop, or a break, or something else you can think of!

TUTOR TIPS

Students may replace the for loop with a while loop like in this example:

```
human_counter = 0
computer_counter = 0
target = 5

while human_counter != target and computer_counter != target:
    ...
```

8. Extension: That's not a real move!

What happens if the human plays a wrong move, like Batman? Or does a typo, like "ppaer"? Test your code and find out!

We need to make our code more robust! If you haven't already, also make sure you also go back and do Task 3.4.

Task 8.1: Check the move is valid!

Create a `while` loop that runs until the user enters a valid move of "scissors", "paper" or "rock".

If the move isn't valid, ask the user for their move again!

The code should look like this:

```
while human_move != "scissors" and human_move != "paper" and human_move
!= "rock":
    human_move = input("What is your move? scissors, paper or rock? ")
    human_move = human_move.lower()
```

This should follow the initial time that a move is asked for

★ CHALLENGE 8.2: Game Over! Shut Down! ★

Sometimes the user might say they want to play a certain number of rounds, but has to leave before the rounds are finished.

Create an `if` statement that checks to see if the user entered "quit" as their move, and close the game down.

Don't forget to tell the user who the overall winner was!

The code should look like this:

```
while human_move != "scissors" and human_move != "paper" and human_move != "rock":
    human_move = input("What is your move? scissors, paper or rock? ")
    human_move = human_move.lower()
    if human_move == "quit":
```

9. Extension: Scissors, Paper, Rock, Lizard, Spock!

Let's add some more moves and play Scissors, Paper, Rock, Lizard, Spock! Follow the arrows in the picture to see who wins!
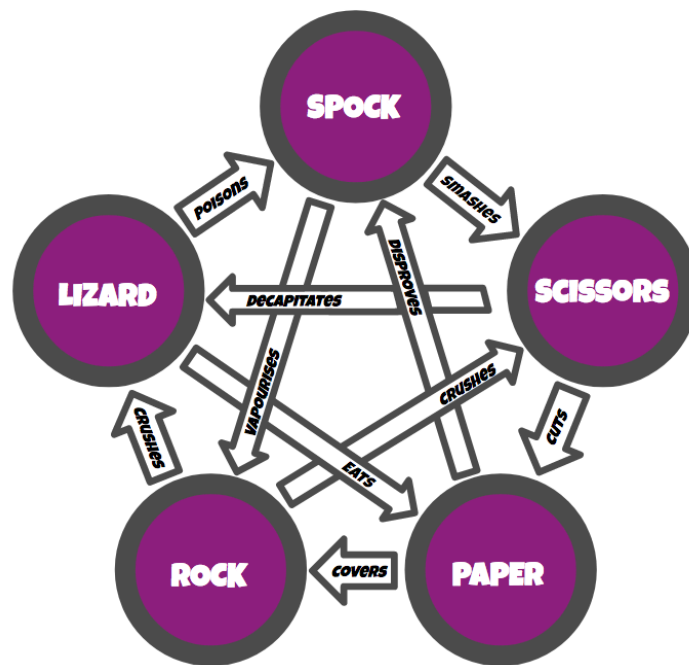
Task 9.1 Updated moves!

When you ask the user what move they want to play, include lizard and spock!

Make sure you give the computer the same options!

Task 9.2 Updated combos!

Update the moves dictionary to include all of the new combinations!
(There's a table on the printout you can use to figure out the options).

★ CHALLENGE 9.3: Too much dictionary!!

Woah that dictionary got big! It's got 25 combinations. But what if we dealt with all the ties before we got to looking up the winner.?

Use an if statement to deal with ties and eliminate all the ties from the dictionary.

Human Move 👤	Computer Move 🖥	Who Wins? 🏆
scissors	scissors	Tie
scissors	paper	Human
scissors	rock	Computer
scissors	lizard	Human
scissors	spock	Computer
paper	scissors	Computer
paper	paper	Tie
paper	rock	Human
paper	lizard	Computer
paper	spock	Human
rock	scissors	Human
rock	paper	Computer
rock	rock	Tie
rock	lizard	Human
rock	spock	Computer
lizard	scissors	Human
lizard	paper	Human
lizard	rock	Computer
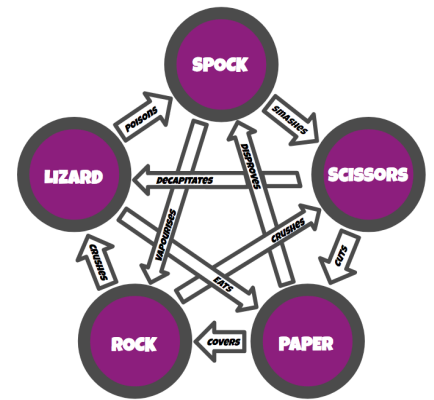
lizard	lizard	Tie
lizard	spock	Human
spock	scissors	Human
spock	paper	Computer
spock	rock	Human
spock	lizard	Computer
spock	spock	Tie

10. Extension: AI. The computer reads your mind!

Let's make the game more challenging by having the computer guess what move the user will choose next, based on what the user has chosen before!

Task 10.1 Create the dictionary

Create an empty dictionary called `ai` for the computer. Store the move the human last chose in another variable, such as `last_move`.

Task 10.2 Store what happens next!

In the dictionary, add the move that the human played last round as a `key` if it's not already in the dictionary. Then store the move that the human played this round in a `list` as the `value`.

Make sure this happens every round!

Task 10.3 Get the computer to choose!

Now that the computer knows what the human played last time, get it to guess what you'll play next!

If the move the human played last is in the `ai` dictionary, choose the computer's move from the list of values. Otherwise, select the `computer_move` from the standard lists of moves!

★ CHALLENGE 10.3: Pick the winner

Our computer move will now pick the more likely move to make it tie with the human. But we want our computer to win!

Rather than storing what the human played as the value in the dictionary, store the move that would win against the human instead.

11. Extension: Write your own rules!

Scissors-Paper-Rock-Lizard-Spock is boring. Everyone's already playing it! Let's make our own version!

Task 11.1: Ask the user for their moves!

Imagine your own game! Maybe it's called "Cat Mouse Dog" , or "Lightning Dragon Wolf Snake Wind"! Pretty much anything you can imagine! Maybe one move always wins! Maybe one always loses!

Here's how we write the game logistics for a few versions of the game:
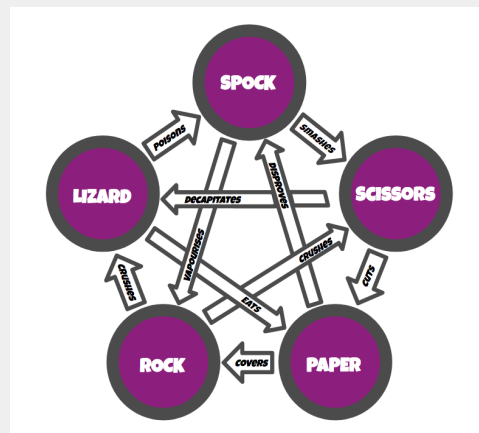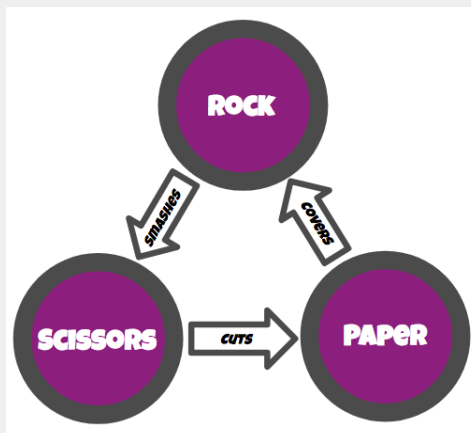
Draw up your idea for some game logistics here:

Task 11.1: Ask the user for their moves!

We might have lots of game ideas!! We don't want to have to write a big dictionary for each one! Let's make the computer do some of the work!

1. Use **input** to ask the user to list all the different possible moves on one line.
2. Then **split** those options into a list called **moves**!

Task 11.2: Manual winning!

Create a `for` loop (or two for loops!) that runs through every combination and have the user `input` if the winner was `human`, `computer` or `tie`.

Store the answers in a dictionary!

Task 11.3: Move it, move it!

Make sure that the computer and human are using the correct set of `moves`!

★ CHALLENGE 11.4: Work out the winners using for loops!★

Update your `for` loop so that if the user says that `move1` beats `move2`, the code knows that `move2` loses to `move1`.

We also know that if `move1` and `move2` are the same move, that it's a tie! Create another `for` loop that handles all the ties.

Store the answers in the dictionary!

★ CHALLENGE 11.5: Store the game mode!★

Putting in all the game logistics instructions takes a long time! We don't want to do it every game. If we could save and load the game modes we input it would be much better!

Use a file to store the game mode! At the start of a new game as the user if they want to create a new game mode or load one from a file!