



Girls' Programming Network

Scissors Paper Rock!

Extensions!

Congratulations!

You finished your Scissors Paper Rock game!

Here's a few extensions you can do to make your game even better!

Try one (or more!) of the extensions in this book!

Keeping Score

That's not a real move

Scissors Paper Rock Lizard Spock

AI: The computer reads your mind

Write your own rules

Where do I start?

You can do these in any order, so choose your favourites!

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 matches  
Computer won 2 matches  
Human is the winner!!  
-----
```

Hint

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

★ CHALLENGE 7.4: First to X

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!

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!

★ CHALLENGE 8.2: Game Over! Shut Down! ★

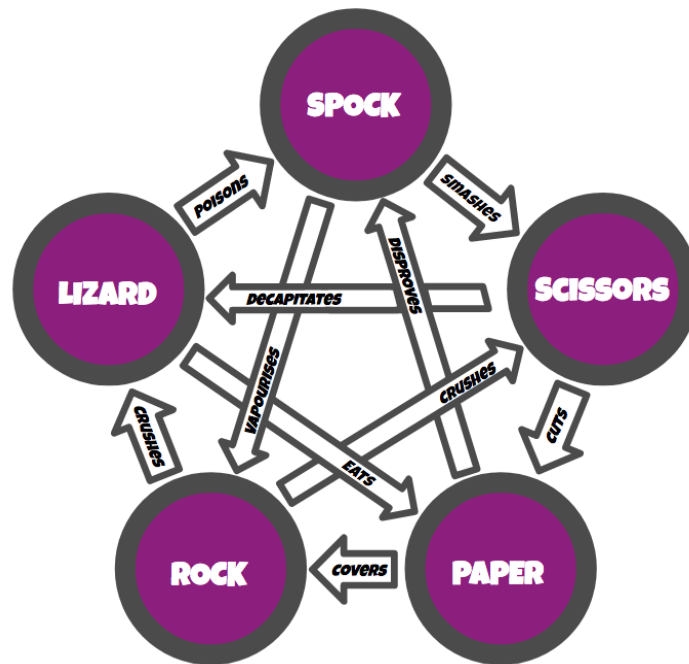
Sometimes the user might say they want to play a certain number of rounds, but has to leave before the rounds are finished.

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!

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!

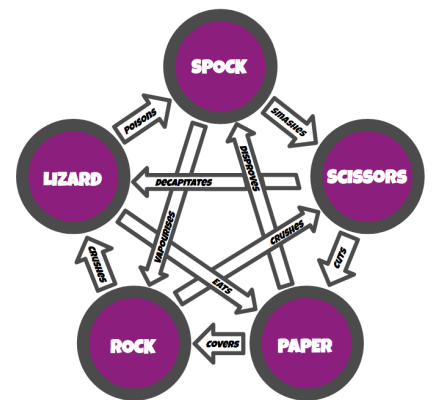
Update the moves dictionary to include all of the new combinations!
(There's a table on the printout you can use to figure out the options).

★ 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	
scissors	paper	
scissors	rock	
scissors	lizard	
scissors	spock	
paper	scissors	
paper	paper	
paper	rock	
paper	lizard	
paper	spock	
rock	scissors	
rock	paper	
rock	rock	
rock	lizard	
rock	spock	
lizard	scissors	
lizard	paper	
lizard	rock	
lizard	lizard	
lizard	spock	
spock	scissors	
spock	paper	
spock	rock	
spock	lizard	
spock	spock	



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.

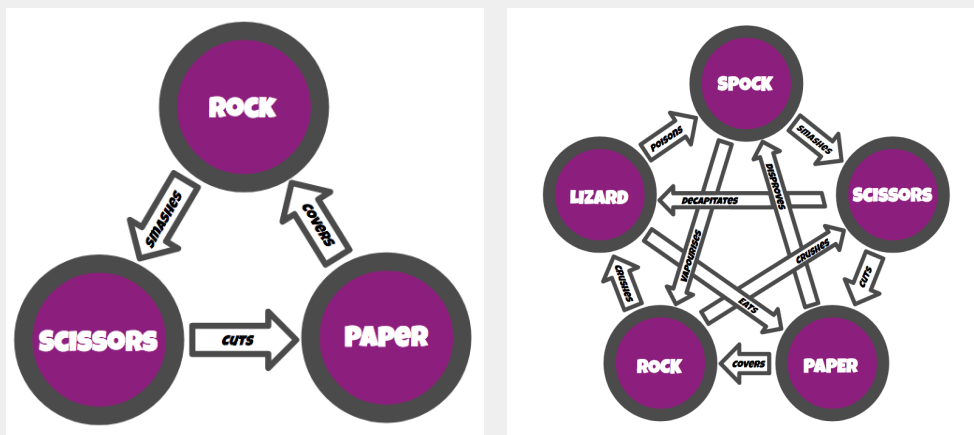
11. Extension: Write your own rules!

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!