Girls' Programming Network

Scissors Paper Rock!

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

Sarah Mac
Renee Noble
Vivian Dang
Courtney Ross

Testers

Pauline Kelly
Meike Moeckel

Part 0: Setting up

Task 0.1: Making a python file

Open the start menu, and type ‘IDLE’. Select IDLE 3.6.

Go to the File menu.
Select ‘New File’.
This opens a new window.
Go to the file menu.
‘Save As...’ and save the file as ‘SPR.py’

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!
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 1:

- ☐ You should have a file called SPR.py
- ☐ Your file has your name at the top in a comment
- ☐ Run your file with F5 key and it does nothing!!

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!
```

Don't want to type all that out? Go to this link: http://bit.ly/2nzHvVM

☑ 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!

2. Who played what?

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

Make a variable for the computer's move such as `computer_move`, set it to one of "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!
------------
```

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
```

☑ CHECKPOINT ☑

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!

★ BONUS 2.4: Personalise the game

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
```

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:

Human Move	Computer Move	Who Wins?
scissors	scissors	draw
scissors	paper	human
scissors	rock	
paper		

Task 3.2: Calculate and print the winner

Use `if` and `elif` statements to calculate the 9 different combinations above.

You should print out the winner inside your `if` and `elif` once you know the result!

Hint

You can check a particular combination of moves with code like this:

```
if computer_move == "paper" and human_move == "scissors":
      print("Human won the round!")
```

☑ CHECKPOINT ☑

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

- ☐ Compare every possible combination of moves
- ☐ Print out the winner
- ☐ Run your code and test different moves!
- ☐ Test when you input “ROCK” or “Rock” instead of “rock”, what happens?

★ BONUS 3.3: ROCK Rock rOcK!

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

We can use `word = word.lower()` to change what the user entered to lower case. Update your code so we’re always using the lowercase version of what your user entered!

4. Smarter Computer

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

Task 4.1: 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 4.2: Chose a random move!

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

Use chose a random move for the computer using `random.choice` from a list of "paper", "scissors" and "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 5:

- ☐ 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?

★ BONUS 4.3: 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: http://bit.ly/2nzHvVM 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!

5. Again, Again, Again!

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

Task 5.1: Loop time!

Create a while loop that runs forever, so we can play as much as we want!

You'll need to use:

- A `while` loop
- A `True` statement

The `while` loop will run as long as what comes after the `while` is true. The easiest way to do this is using a boolean `True`.
Use this line to make the game play on repeat

```
while True:
```

Task 5.4: 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:

```
while True:
    # THIS IS INDENTED
```

☑ CHECKPOINT ☑

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

☐ Create a while loop that constantly runs!

☐ Your game code is inside the while loop

☐ The game never ends!

6. Extension: How Many Games?

Instead of running infinite times, we now want to run as many as times as the user wants!

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!

Remember `int("57")` will give you back 57. You can use `int(..)` on a variable too!

Task 6.2: Loop time!

REMOVE the while loop you made in section 5. Instead, 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:

```
for i in range(number_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
```

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!

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 2 variables, these 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 over all winner.

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

```
---------------------------------
GAME OVER!
Human won 5 games
Computer won 2 games
```

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 what happens if the human doesn't write their move in lowercase letters and plays ROCK, Rock or ROcK? Test your code and find out!

There are a few little issues with our code so far:

- If the human inputs an incorrect move the program doesn't notice!
- If the human inputs a move which is not written in lowercase letters, they will lose the game. (Unless you already did the bonus task 3.3!)

We need to make our code more robust! Let's see what we can do to fix these issues!

7a. Inputting a move which is not case sensitive

To compare the human's move to the list `moves`, the strings need to look the same. We can make sure that the moves we are comparing have the same case as the moves in the list by calling the `lower()` function. For example, to make the variable `word` lowercase, you would write:

```
word = word.lower()
```

Notice the dot, this is important!

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! ★

Sometime 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!