Girls' Programming Network

Guess Who!

In this game we're going to choose a Guess Who character, and the computer is going to guess which one we've chosen!

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

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

Caitlin Macleod
Rachael Newitt
Imaina Widago
Alesiya Maynard
Courtney Ross
Jeannette Tran
Fiona Lin
Renee Noble
Alex McCulloch

Testers

Sheree Pudney
Claire Quinlan
Maddie Jones

Intro to Python

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 **guess_who**

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

A main.py file will have be 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!

Welcome Message

Task 1.1: Print a welcome 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:

```
------------------------------------------------------
Welcome to Guess Who!
------------------------------------------------------
Moves: Pick a person from the character sheet, and let the
computer guess who you're thinking of.

Type "yes" or "no" to answer the questions.

Good luck!
------------
```

Don't want to type all this out? Go to http://bit.ly/gpn-2018-4.

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

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

Task 1.2: Copy in the list of people

We need to create the list of all the people in our Guess Who game! This list will also contain a list of all their attributes.

Copy and paste the list from http://bit.ly/gpn-2018-4 and assign it to a variable called **`people`**.

Task 1.3: Hide that character!

In this game we're going to choose a character from our character sheet, and the computer is going to guess which one we've chosen!

Look at your character sheet and circle the character you want the computer to guess first. **This workbook will call the character the computer is trying to guess the `secret character`.**

★ Bonus 1.4: Who do you know? ★

Draw additional people on your character sheet **and add them to your list of characters**! They need to each have an eye colour, hair colour, and accessory.

You can add as many as you like, but make sure no one has exactly the same combination of hair, accessories and eye colour as someone else!

☑ CHECKPOINT ☑

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

- ☐ Print a welcome message
- ☐ You have a list of people.
- ☐ You have chosen a character for the computer to guess.
- ☐ Run your code!

Part 1: Selecting Attributes!

Lists

Task 2.1: What's your style?

Our people all have different attributes, in the order: name, eye colour, hair colour, and then accessory. We need to create lists of all the different options!

1. Create a list of all the different possible eye colours, and store it in a variable called `eye_colours`.
2. Do the same thing for all the different hair colours, and then the accessories! Call the variables `hair_colours` and `accessories`.

Make sure that each option that you included in `people` is also stored in the lists above!

Task 2.2: Creating looks

So the computer can start guessing, we need the computer to select an option from each of our `eye_colours`, `hair_colours` and `accessories` lists!

1. Select an item from `eye_colours`. Store it in a variable called `eye_guess`.
2. Do the same thing for `hair_colours` and `accessories`. Call the variables `hair_guess` and `accessory_guess`.

Hint

We can access items in a list individually. The below code will print out the second item in the `dinner` list:

```
dinner = ["pizza", "chocolate", "nutella", "lemon"]
selection = dinner[1]
print(selection)
```

Don't forget that lists start from 0!

Task 2.3: Do they look like this?

The computer needs to find out if the eye colour, hair colour and accessory they selected match the eye colour, hair colour and accessory of the `secret character`.

1. For the `eye_guess`, use `input` to ask the user if it matches the eye colour of the `secret character`. Store the answer in a variable called `eye_guess_answer`.

2. Do the same thing for `hair_guess` and `accessory_guess`. Store the answer in variables called `hair_guess_answer` and `accessory_guess_answer`.

★Bonus 2.4: Uppercase or lowercase

Sometimes users don't type exactly what we expect them to! If you're expecting a user to type `"yes"` or `"no"` but they type `"Yes"`, `"YES"` or `"NO"` your code may not recognise their answer correctly.

Make your game recognise user `input` if they enter versions of your expected input with different capitalisation.

Hint

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

☑ CHECKPOINT ☑

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

☐ You have a list called eye_colours

☐ You have a list called hair_colours

☐ You have a list called accessories

☐ The computer has selected an eye colour, hair colour and accessory to guess.

☐ The computer asks the user if their secret character has the eye colour, hair colour and accessory that the computer picked and stored the answers.

Part 2: Selecting Attributes!

Task 3.1: Splitting out people!

Now that we know whether the `secret character` has the attributes the computer guessed or not, we need to compare it to the list of `people`. Let's check if the first person matches!

1. Select the first person in the list of `people`. Store it in a variable called `person`.
2. Print out the `person` to see what they look like!

Task 3.2: Splitting out attributes

For each person, we need to get their eye colour, hair colour, and accessory!

1. For the `person`, get their name. Store it in a variable called `person_name`.

2. For the `person`, also get their eye colour, hair colour and accessory. Store it in variables called `person_eye`, `person_hair` and `person_accessory`.

☑ CHECKPOINT ☑

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

- ☐ Get the first person out of the people list
- ☐ You print the person to see what they look like
- ☐ You have each of the person's attributes in different variables!
- ☐ Try running your code!

Part 3: Narrowing it down!

If Statements

Task 4.1: What if?

We asked the user if our guess for eye_colour, hair_colour and accessory was right, and got a yes/no answer. Now we need to do something with it!

1. Create an if statement that checks if the eye_guess_answer was yes. If it was, print “I guessed eye colour right”

2. Add an elif statement that checks if the eye_guess_answer was no. If it was, print “I guessed eye colour wrong!”

Task 4.2: Character comparison

Now the computer has made an eye colour guess and got a Yes/No answer.
Now we need to make the computer work out which characters need to be eliminated!

Work out if the following characters need to be eliminated for this example:

```
Does the character have blue eyes? yes
```

Aleisha

☐ **Eliminate** ☐ **Keep**

What about this person?

Brittany

☐ **Eliminate** ☐ **Keep**

Here’s another example!

```
Does the character have brown eyes? no
```

Aleisha

☐ **Eliminate** ☐ **Keep**

Brittany

☐ **Eliminate** ☐ **Keep**

Hint

There's two different ways that a person can be eliminated:

1. Our guess was right, and the person does not have that feature
2. Our guess was wrong, and the person does have that feature

Task 4.3:

Now that we know all the different ways that a person can be eliminated, we can code it using `if` and `elif` statements!

We've already got an if-elif chain that does something when the eye_guess_answer is yes and does something else if eye_guess_answer is no. Now we're going to add an additional conditional to check to see if the person needs to be eliminated, depending on what eye_colour they have!

1. In the if statement where eye_guess_answer is yes, add another condition that checks if the person_eye is not equal to eye_guess. Then change the print statement to say ""I guessed eye colour right, but the person doesn't have that eye colour. We're going to eliminate them."

2. In the elif statement where eye_guess_answer is no, add another condition that checks if person_eye does equal eye_guess. Then change the print statement to say "I guessed eye colour wrong, but the person does have that eye colour! We're going to eliminate them"

3. Run your code! Using the examples in Task 4.2, does your code eliminate the right person?

Hint

In `if` statements, we can use the keyword `and` to check if multiple things are true:

```
if raining == True and umbrella == "I forgot it!":
```

```
        print ("Don't go outside!")
elif raining == False and umbrella == "I forgot it!":
        print("It's okay, it's not raining")
elif raining == True and umbrella == "I've got it!":
        print("Awesome! Let's go outside!")
```

Task 4.4: What if?

Now that we've got a person being eliminated correctly based on eye_colour, it's time to eliminate them based on hair colour and accessory!

Do task 4.1 and 4.3 again, but this time for hair_colour! Make sure you do this by adding additional elifs to your existing if statement!

Once you've tested that and got elimination working for hair_colour, do the same thing for accessory!

Hint

Why so many elifs???

We need to use and `if-elif-elif-elif-elif-elif` chain because we only want to add the character to the elimination list once! If we use several `if-elif` pairs then we might add the character to the elimination list of multiple times for different features!

If we try and eliminate them multiple times the computer will be confused because they are already eliminated.

☑ CHECKPOINT ☑

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

☐ If-elif statements list all the ways that a person can be eliminated, and print out when they are

☐ Try running your code!

Part 4: Eliminate! Eliminate!

For Loops

Task 5.1: Again, Again, and Again!

Now that we've checked to see if the attributes of one person matches what the computer guessed, we want to be able to check everyone in the people list! To do this, we're going to use a `for` loop.

1. Use a `for` loop to go through each person in the **`people`** list to check to see if they need to be eliminated.
2. Make sure that all the code from section 3 is inside the `for` loop!

Don't forget to get rid of the line that selects the first person in the people list!

Hint

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

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

Task 5.2: Make a list of things to eliminate

The computer needs to track all of the people that it knows isn't the correct answer. We're going to store this in a separate **`list`** for now.

1. Create an empty **`list`** and assign it to a variable called **`eliminate`** at the top of the code near the list of people.

Task 5.3: Make a list of things to eliminate

We need to add all the people that need to be eliminated to the **`eliminate`** list! In your `if` and `elif` statements that were created in section 3:

1. Every time there isn't a match, update your code so instead of printing something, we're going to add the **`person`** to the **`eliminate`** list.

Hint

You can add items to lists using the append statement:

```
dinner = []
dinner.append("pizza")
```

Task 5.4: Eliminate Them!

In another `for` loop, go through each person in the **`eliminate`** list and **`remove`** them from the **`people`** list. This way, the computer won't try to guess them.

1. Create a `for` loop that goes through each person in the **`eliminate`** list.
2. Remove each person from the list of **`people`**.

Hint

If I wanted to remove an element from a list I could use code like this:

```
dinner_options.remove("pizza")
```

☑ CHECKPOINT ☑

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

☐ Your code loops over every person in the people list

☐ Your code removes people already identified as eliminated from the list of available people.

☐ Try printing out your list of people before and after eliminating characters!

Part 5: Guess Them!

Task 6.1: Making the guess

It's time for the computer to guess who it is! The computer needs to ask the user if that's the `secret character` by using person's name.

1. Pick the first person from the list of people not yet eliminated. Store it in a variable called `guess`.
2. From `guess`, get the name of the person. Store it in a variable called `guess_name`.
3. Use `input` to ask the user if the computer guessed the name correctly. Store the answer in a variable called `answer`.

Hint

Remember that `people` is actually a list of lists! You may find it useful to `print` out `guess` and `guess_name` to help check that you're accessing the list correctly.

Task 6.2: That's correct!

If the computer guessed the right person, it's time to celebrate! Get the computer to `print` out a message about how great the computer is at this game, and how lovely it was to play with the user.

1. Create an `if` statement that checks to see if the guess was correct. If it was, `print` out a congratulations and thank you message.

Task 6.3: Wrong answer!

If the computer didn't guess correctly, the person they guessed should be removed from the list of people so they don't get guessed again.

1. Update the `if` statement you created in task 5.2 to have an `else`.
2. In the `else` statement, remove the guess from the list of `people` and `print` something like "Oh No!"

☑ CHECKPOINT ☑

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

☐ The computer selects the first person from the list of people, and guesses who!

☐ The computer responds to a correct guess by printing a congratulations message!

☐ The computer responds to an incorrect guess by removing the character from the list of possible people.

Part 6: Randomize it!

Task 7.1: Import random library

It's really boring that our computer only guesses the same eye colour, hair colour and accessory! Let's randomise what the computer picks.

At the top of your file add this line:

```
import random
```

Task 7.2: Pick a random look

Now we need to update how the computer makes its guesses!

1. Update the code where the computer selects the **eye_guess** so that it's randomly selected!
2. Do the same for **hair_guess** and **accessory_guess**!
3. Now randomly select the **person** to guess from the list of characters that remain in the game!

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 8:

- ☐ Import random
- ☐ Pick a random value from eye_colours
- ☐ Pick a random value from hair_colours
- ☐ Pick a random value from accessories
- ☐ Pick a random person to guess from people
- ☐ Try running your code!

Part 7: Over and over again!

While Loops

Task 8.1: Add the game loop!

Create a `while` loop that runs forever, so the computer can ask as many questions as it wants!

Think about what part of the game you want to repeat on a loop!
We suggest you loop over asking about hair colour, eye colour, accessory and name until the computer guesses the correct name.

But maybe you want your game to go differently!

Hint

You will need to indent all the code that you want looped!

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

Task 8.2: To infinity and beyond!

Whoops! It looks like we created an infinite loop - the game never ends! You can press `CTRL+C` to stop your program.

We need to `break` the loop!

1. Update the `if` statement that checks to see if the computer guess correctly to include a `break` statement when it has figured out the right person.

Task 8.3: Don't remove them twice!

Now that we are making more than one guess, we're going to be more careful when removing people from the list. If we try to remove someone from the list that isn't actually in the list, we'll get an error! (Try it yourself!)

To make sure we don't remove someone twice, we will have to empty the eliminate list when we are done with it. We're done with the list once we've looped through the people and removed everyone in the eliminate list from it.

Hint

Emptying a list is the same as making it into a new empty list.

★Bonus 8.4: Liar, Liar!

What if our user wasn't telling us the truth? If we get to the end of guessing and there's no more people to guess, what happens?

Run your code and see if you can make the computer run out of things to guess!

Hmmm….We should fix that. Let's add a check to check how many people are left in the list and yell at our user for trying to trick us if there is 0.

Make sure that this code is above where the computer makes its guess!

1. Create an `if` statement that checks the length of the `people` list.
2. If there is no one left, `print` out a message that says `"You're playing tricks on me! There's no one left :("`
3. Don't forget to `break` the loop if there is no one left

Hint

You can check the length of the list using `len()`.

```
len(["pizza", "chocolate", "nutella", "lemon"])
```

☑ CHECKPOINT ☑

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

- ☐ Your code runs without any problems.
- ☐ Guessing the right person ends the game.
- ☐ When the game is over, you break out of the loop.

Part 9: Extension: Smarter guessing!

Task 9.1: Process of elimination

Let's make our computer smarter! To do this, we're going to work through each of the lists for eye colour, hair and accessories from beginning to end.For example, our hair colours are:

```
hair = ["black","brown","red"]
```

Imagine if we ask our user if the hair colour is black, and they say no. Then we ask if it's brown, and our user says no.. what colour is the hair?

There's only one color left, so we know the hair colour must be red!

Let's change our code to get rid of `random.choice` and replace it with list indexes. Every time we get a hair colour from the list, we want to get the first option.

Go back and look at part 3 if you need a reminder about how list indexes work.

Task 9.2: We like to .remove() it, .remove() it!

Now we're getting the first item in the hair colour list every time. But, because our list is the same we always choose `"black"`.

To fix this we need to make sure we remove the bad items from the list, so we don't ask about it again.

Task 9.3: No questions asked

Just like the example before, if there's only one choice left or we've already guessed the secret character correctly, we don't need to ask whether it's the right one, we already know!

Change your code so that if there's only hair colour left in the list, we don't ask any more questions about hair colour.

Task 9.4: Off we go again!

Our hair colour guessing is excellent now, but we can definitely make the others better too. Go back and improve the guessing about eye colour and accessories to make them better as well.

☑ CHECKPOINT ☑

If you can tick all of these off, you have finished this part:

☐ Your guessing for eye colour, hair colour and accessories all work using list indexes

☐ When a guess is wrong, you remove it from the list

☐ When the lists are only one element long, you don't ask any more questions about that characteristic

Part 10: Extension: Read it in!

Task 10.1: Where have all the people gone?

1. Create an empty list of `people`.

You can `comment` out your list of `people` from earlier or delete it, whichever you prefer.

Task 10.2: Here they are!

1. Download the file `people.txt` from http://bit.ly/gpn-2018-4!
2. Make sure you save it in the same directory as your python file.

Task 10.3: Open sesame!

Use Python's *`with open`* to open the the text file.

Use this line just after you create your empty list to open your people file and read what it says.

```
with open('x.txt') as f:
      DO SOMETHING
```

Task 10.4: Let's loop again

So we can open the file, but how do we get the people out?

We make another loop of course! Use the code below inside your open statement to help you read in each of the lines in the file, one by one.

```
for line in f:
    line = line.strip()
    parts = line.split(“,”)
```

Hint

`with open` and the `for` loop both need to be indented. So if you're getting an error, make sure to check that your code is indented like below.

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

Task 10.5: Append your people!

Now we have each of the people in the file, we want to add them to our `people` list. Try to do this using `append()`.

Task 10.6: Find your features!

As we're reading in from a text file, we might find people with different features. For example, we might get someone with pink hair, or grey eyes, or they might have a bracelet! These options aren't in our current feature lists eye_colours, hair_colours and accessories.

Create a `for` loop for so you can create the list `eye_colours`. Go through each `person` in the `people` list, and `if` their eye colour isn't in the list `eye_colours`, add it!

Then do the same for `hair_colours` and `accessories`!

☑ CHECKPOINT ☑

If you can tick all of these off, you have finished!

- ☐ You are using "with open" to open a file
- ☐ You use a loop to read each line in the file
- ☐ All of the people are appended to your people list
- ☐ The lists eye_colours, hair_colours and accessories are created from the people list.
- ☐ Your code runs without any problems