Girls' Programming Network

Guess Who!

Workbook N

Create a program using classes, that picks a Guess Who character at random and gives you hints to help you guess it!

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

This workbook and related materials were created by tutors at:

Sydney and Canberra

Girls' Programming Network

If you see any of the following tutors don't forget to thank them!!

Writers

Jeannette Tran
Fiona Lin
Renee Noble
Alex McCulloch
Annie Liu
Bryony Lanigan
Heather Catchpole
Amanda Hogan
Anton Black

Testers

Vivian Dang
Courtney Ross
Rachel Alger
Libby Berrie
Caitlin Macleod
Sheree Pudney
Rashmica Gupta

A massive thanks to our sponsors for supporting us!

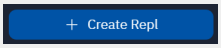
ATLASSIAN

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. Click 

in the top left hand corner to create a new Project.
2. Use the Python 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 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 

. 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!

Part 1: People are objects!

Classes

Task 1.1: Define a person

Define a class called Person with four attributes (**`name, eye_colour, hair_colour accessory`**).

When it's *initialised*, the Person object should receive these attributes and store them in its **`self`**.

Task 1.2: Make a new person object

Let's make a person object! You can make the person to be like you, your friend, or anyone you want! Once you have the class defined, you can set the variable **`character`** to be a new **`Person`**

Hint

To create a new object from your class you need to choose a variable name and the four attributes you want to add to that person, e.g.:

```
p = Person(<your_name>, <your_eye_colour>, <your_hair_colour>, <your_accessory>)
```

Task 1.3: Let's play!

Let's **`print`** out the information about the Person.

For example, if your name is Jane we might **`print`**:

```
I met a person called Jane!
Jane has green eyes, brown hair and has glasses.
```

Hint

You can access an object's attribute with the syntax:

```
object_name.attribute_name
```

For example, you could access the name attribute of the object character like this:

```
character.name
```

☑ CHECKPOINT ☑

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

- ☐ Create a new person class
- ☐ Make an init method to set its attributes
- ☐ Make a new person with attributes and store it in a variable
- ☐ Print out the details you stored in your person
- ☐ Try running your code!

Part 2: Importing our People!

Files

Task 2.1: Opening a file

We don't want to have to type all of our people, we want to use a file of people and make each one a separate Person object.

Download the file `'people.txt'`

Save it to the same place your code is running from (eg the Desktop)..

Each line in the file has the four attributes for one person, separated by commas, in this order: *name, eye colour, hair colour, accessory*.

Task 2.2: Make new people from the file

You'll need to:

1. Loop through each line in the file;
2. Remove the newline character `\n` from the end of each line;
3. `Split` the line by the commas;
4. Extract the four attributes from the comma-separated line (name, eye colour, hair colour, accessory)
5. Use the attributes to construct a Person object; and
6. Add each Person to a list so you can find them later.

Hint

When you know that an action is going to create more than one result, you can assign multiple variables at once using a comma between each value:

```
words = ['dog','woof']
animal, sound = words
print(animal)
>> dog
print(sound)
>> woof
```

Task 2.3: Print out the names of our people

Create a loop to go through all the Person objects in your list to print them each out.

Use print statements to print out the name, eye colour, hair colour and accessory of our characters. It should look like this for each person when you run your code:

```
Name is: Annie
Eye is: brown
Hair is: blue
Accessory is: glasses
```

Hint

Calling **`print()`** on the entire Person object won't display all the attributes. Python will just print out the memory address of the object:

```
>>> print(person)
<__main__.Person object at 0x00000167F73BF710>
```

You'll need to print out the attributes by accessing them individually, e.g.:

```
>>> print(person.name)
Jasmine
```

☑ CHECKPOINT ☑

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

- ☐ Open a file
- ☐ Read person objects and store them as a list
- ☐ Print out the names of the people
- ☐ Print out the features of the people
- ☐ Run your code!

Part 3: Checking a guess!

Task 3.1: Choose a random person to be the character to guess

Use `random.choice()` on your list of Persons to choose one person to be the character to guess, we will refer to this as `secret_character`.

Hint

Don't forget you will need to `import` the random module.

Hint

Debugging things that are random can be hard because you don't know what the correct answer is!

If you need to debug things later, consider **hardcoding** the `secret_character` to help you figure out the bug!

Task 3.2: Get a guess from the player

Use `input` to take the guess of a name from the player.

Methods

Task 3.3: Make a method to check the name against a guess

We're going to need a **method** to check if the name that has been guessed is the same as the `secret_character`.

1. Write a method inside the class **`Person`**, passing in the guessed name as a parameter
2. return True if the name matches the Person's name and False if it doesn't match.

Hint

Remember that the **first parameter** of every method must always be **`self`**

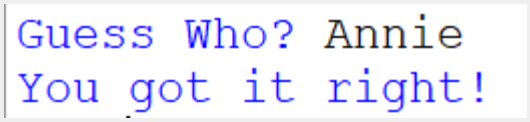
Hint:

To compare the guessed and actual names while ignoring case, convert them both to the same case before you compare them using String methods

Task 3.4: Output the result

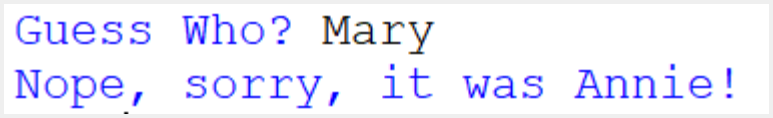
You should also congratulate the player if they have guessed it right:

```
Guess who? Annie
You got it right!
```



You should print out the correct name if they have guessed wrong, like below:

```
Guess who? Mary
Nope, sorry, it was Annie!
```



☑ CHECKPOINT ☑

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

- ☐ Choose a random character and store it in a variable
- ☐ Ask the player to guess a name
- ☐ Make a method to check if the Secret Character's name matches the guess
- ☐ Respond to the user
- ☐ Run your code and test different names

Part 4: Again, Again, Again!

So that was really hard. We had a 1 in 12 chance of guessing correctly. We want to play 'Guess Who' until we guess the correct person! Let's add a loop to guess on repeat!

Task 4.1: Loop time!

Use a **`while`** loop to make the game play on repeat.

```
while True:
    # The code here will be run repeatedly
```

Hint

We want to **repeatedly ask** the user for their guess, so put that part of your program **inside** the `while` loop.

Don't forget to indent your code!

Task 4.2: Stopping

We want our program to stop when we make the right guess!
After we congratulate the user on making the right guess, add a `break` to stop the loop.

`break` doesn't take any parameters; you can just put it on a line by itself like this:

```
        break
```

Task 4.3: Update what happens when the user guesses wrongly!

Update your `else` statement so that the user is no longer told who the correct person is when they get it wrong!

☑ CHECKPOINT ☑

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

- ☐ Create a while loop that lets your game keep going!
- ☐ Your game code is inside the while loop
- ☐ The game only ends when you guess the right person!

Part 5: Make some more checks!

We are just guessing blindly at the moment, which isn't very fun! Let's let the player get more information about the person before they have to guess who. We're going to need more methods to do this

Task 5.1: What's their eye colour?

Write a method to check whether the eye-colour guess matches the eye-colour of the Person.

Add your method inside the `Person` class, pass in the guessed eye colour as a parameter and return True if the name matches the Person's eye colour and False if it doesn't match.

It should allow you to write an interaction like this:

```
Guess their eye colour? brown
Yes
```

Hint

You wrote a method before to check if a guess matches the name of the Secret Character. Do the same here but for eye colour!

Task 5.2: What's their hair colour?

Now do the same thing that you did for 5.1, but this time, the method should check the Secret Character's hair colour!

Use your method to ask about the hair colour inside you `while` loop after asking about eye colour, but before guessing the name.

Task 5.3: What's their accessory?

Do the same thing again that you did for 5.1 and 5.2, but for the Person's accessory!

Use your method to ask about the hair colour inside you `while` loop after asking about hair colour, but before guessing the name.

☑ CHECKPOINT ☑

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

- ☐ Write a method to compare the eye colour guess
- ☐ Write a method to compare the hair colour guess
- ☐ Write a method to compare the accessory guess
- ☐ Test your methods
- ☐ Print out a menu to let the player choose the guess they wish to make
- ☐ Run your code!

Part 6: Picking a question!

Task 6.1: Make a Menu of options?

Now that you've written all your methods and updated our loop to ask all the questions. You can make a way for the player to choose what they'd like to guess.

Make something that works a little like this...

```
What would you like to guess? Type 1. to guess the Name, Type 2.
to guess the eye colour, Type 3. to guess hair colour or Type 4.
To guess the accessory.
```

Task 6.2: Use if statements for the different options

Show the menu to the player and then ask the different question based on the choice they make. e.g.

```
What would you like to guess? Type 1. to guess the Name, Type 2.
to guess the eye colour, Type 3. to guess hair colour or Type 4.
To guess the accessory. 3

What hair colour would you like to guess?
```

★ Bonus 6.2: Not an option!

Make it say something funny if they choose an option that is not on the list of options!

☑ CHECKPOINT ☑

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

- ☐ Ask the player for the person's feature they wantto guess
- ☐ Ask the player for their guess
- ☐ Run your code!

7. Extension: Printing our People

So far, we've had to deal with printing our people like this:

```
print(person.name, person.eyes, person.hair, person.access)
```

It would be more convenient if we could just call `print(person)`.

Classes have what's called **Magic Methods** -- you've used one already, `__init__` is a magic method. The magic methods allow you to set the functionality for things that lots of objects need to do. For example, you can use the `__add__` method to define what happens when someone tries to add something to the class.

We want to set the string value that is returned by the class so that printing works.

Task 7.1: Making `__str__()`

Create a new method `__str__()` and make it return a string for the format of the person you'd like to display when you `print(person)`

Task 7.2: Edit your code so that you use your new ~magic method!~

At the end when you print out the Secret Character print out the Person

8. Extension: How many questions?

Now, let's track how many (or how few) questions it takes you each game to guess correctly!

Task 8.1: Counter!

Initialise a **`counter`** variable before the loop starts. This will be your guess counter. Start by setting it to `0`.

Task 8.2: Add 1!

Every time the user makes a guess (a name guess or any other feature guess), add one to this **`counter`**.

Hint

You'll need to add to the counter inside each of your `if` or `elif` statements!

Task 8.3: How many questions?

At the end of the game, `print` out how many questions the user has asked.

9. Extension: Discarding People?

Up until now we have been using our pictures to keep track of which people are left to choose from. Wouldn't it be good if the game could keep track of that for us?

To do that we are going to need to add an attribute to our Person that is Boolean and tracks whether that person is discarded or not.

Task 9.1: Discarded!

Add an attribute **`self.discarded`** to your Person Class and set it to `False`

Task 9.2: The discard method

Write a method called **`discard`**. It should set the discarded attribute's value to `True`

Task 9.3: To discard or not to discard?

Every time the user makes a guess (a name guess or any other feature guess), loop through the list of people and if the person should be discarded from our list of potential people use the discard method to set their discarded value to True.

E.g. Hair colour was guessed to be brown and that was the right colour.

```
for person in people:
    if person.check_hair(hair):
        person.discard()
```

Task 9.4: Print the remaining people

After discarding all the invalid people, loop through and print out all the remaining people so that our player can see who is left and what guesses they can make.