**Chapter 4 – Game Setup and Character Creation**

In this chapter, we will look at setting up the correct environment so that our game can be played. With our game we want to implement a way of saving or loading data which and therefore we need to come up with a way to do this. We will also look at the character creation in terms of setting player names, classes and skills. We have touched upon parts of character creation in earlier but here will go in depth by looking at what sort of skills we want our roles to have.

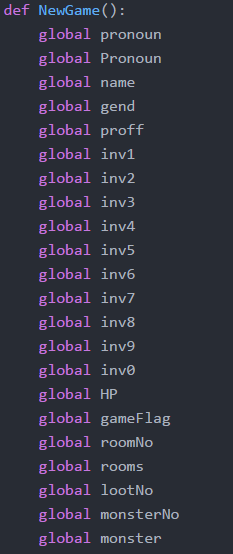
It is important to note that you don’t necessarily have to follow the same style of classes. You should create classes that fit the theme of the game you want to make and you can be as creative as you want.

It is good practice in programming to split up section of your code into functions. Each function will serve a purpose which makes it easy to manage different events and helps if you encounter a problem if you can narrow it down to which function causes the problem. This process is also known as debugging where you go through your program to eliminate as many ‘bugs’ as possible which cause your code to fail.

When a function is called, the code within that function is executed and once everything has been resolved the code returns to the point where the function was called. In python the word ‘def’ is used before the function name. The function name is followed by parenthesis which can sometimes include variable parameters. In our game we won’t need to be using theme because we will be using global variables for all things. Therefore, a function would look like this

def funtionname()

We will want a function that only handles the events that take place when the user wants to start a new game and thus we will call the function NewGame().



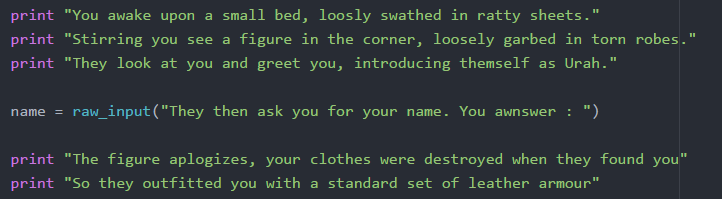
It is also important to note that any global variables we use in our function must be redeclared using the keyword ‘global’. These can be added when and if we need them as we develop the game.

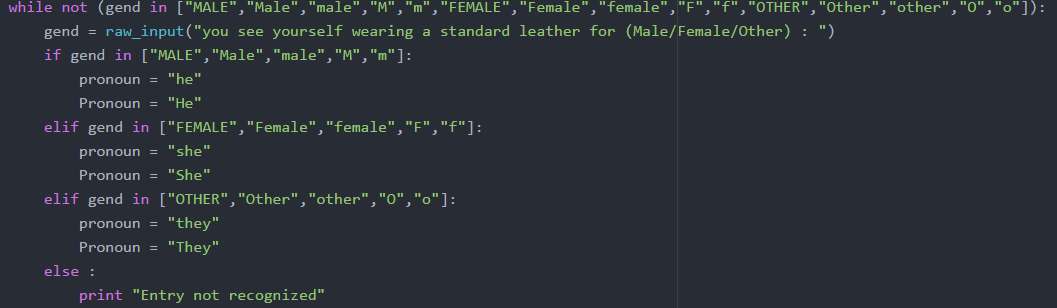
Here is the start of our function which includes some variables that we may not necessarily use now but may need later 🡪

It is important that the variables are declared globally which means that they must be outside any functions and we will show how to do this at the end of this chapter.

With the function declaration and prerequisites out of the way, we can start moving onto the content of the function.

We are going back to what we learnt in **chapter 1** and going to use print statements to set up our environment. Then we ask for the name and provide further setting. All the following should be very familiar from the first chapter:



The next part of this function will utilise everything we have learn in **chapters 1,2 and 3**. It may seem very complex at first but we will break down every part of it. Since we are telling a story, we want to use the correct pronouns in our story statements. If the character is male, we want to make sure that we use he, his etc. Therefore have written the following piece of code:

We will start with a while loop which will continuously repeat until we get a gender that the game will recognise. Before we start looking at the logic behind the while loop we need to establish our goal for the while loop.

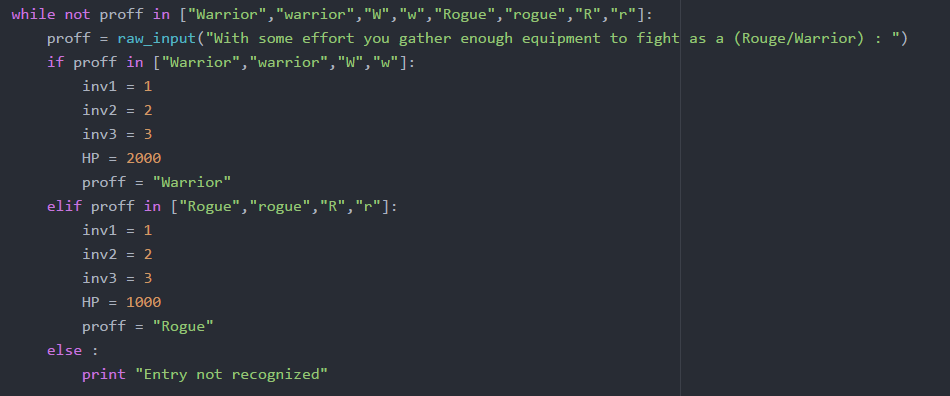
Our goal is that the while loop should continuously loop until we get a recognised gender. To do this we must first define what our recognised terms are so that we can make a comparison to it. This is done by the list in the first line. Now that we have our list we need to make sure the while loop continues if the user input doesn’t match any of the list entries.

A neat little feature that Python has is that we can is the keyword ‘in’ to see if a variable is contained within a list. Therefore we simply write ‘gend in list’ and this returns a true or a false based on whether it is in a list. This is very important as true or false statements dictate whether a loop is executed or not.

With the current set up, if the recognised term is used, the ‘gend in list’ statement will return true. There is a problem with this, we want to loop to execute when that is false but while loops only work when the condition in them is true. Luckily in many programming languages you can flip this by using a form of a ‘not operator’. In Python it is simply written as a not BEFORE the condition which is also shown in the above extract. The condition that needs to be flipped is placed within a set of brackets.

With that the looping done and within the loop we have a series of if statements which check if an assignment matches a gender and then assigns the correct pronoun. The first if statement checks if a male has been chosen by using the same principles on another list. If it is not a male, the female is check and then other is also checked. If is not any of the 3, then we print ‘entry not recognised’ and they are asked again.

The final portion of our function follows very similar to the previous part. Now we will ask the player what class they want to play as. Remember the classes themselves are up to your choice, you should choose roles that best suit the theme of your game. For this example, we will use two classes which are rogue and warrior.



Since our warrior is supposed to have higher health, we take this opportunity to assign a larger amount of HP to the warrior and a lower amount to the rogue. You may think it may be a good idea to create the skill set of each class too but later on we will look into those.

We also gave each inventory slot a number. Currently this means nothing as we have yet to assign each number with a property(e.g. 1 is a health potion) but we should give the player some starting items. Once again this is up to you as the designer whether you want to give your player some starting items.

With all of these components, we have completed our NewGame() function and we should call this function whenever the player wants to start again. Next we will take a closer look at how we will manage our inventory and we will create a function to show what is in your inventory.

EXTENSION

You may notice that checking for the user input is extremely robust as it does not take into account if the user enters ‘’waRRior”. However, there are ways to overcome this by having more rigorous checks. Can you think of a way of implementing this? (If you are struggling, a hint is provided in the code for this chapter on the Github link).