**Chapter 1 – Variables and I/O (input or output)**

Before we can start working on the mechanics of the game i.e the actual gameplay, it is important to understand what variables are and how they are implemented in Python.

We will also have a look at how to give information to an interface. An interface is what allows us to communicate with the computer so that our game knows what action the player wants to do. Initially we are going to have a text based interface which can be similar to playing through an interactive book.

Variables

In programming variables are considered to be one of the basic building blocks that are used to store information of different types. We can then write code that works with the variables to carry out a particular function which we will show in the example below:

*Example 1.1*

Since we are making a game with character that fights different enemies our player will have a health level. If our player drinks a potion of health or takes some damage, we would have to change the health number but the computer also needs a way of remembering this. This is where we use a variable to store the health. All variables need a name so that we can identify what it represents easily and this should be consistent throughout your game. In our version of the game we store health under the variable ‘HP’ which is short for hit points.

Let us set the base health of our character to be 100. To do this in our code we would write:

HP = 100

What this means is that the variable HP has the value 100 and if ever use the word “HP” in a calculation, the computer will know to use the number stored in it. Now we will show what happens if you want to make changes to that number and how the computer remembers it.

*Example 1.2*

So our character is in a fight with an enemy and has just lost some health by taking some damage. Let’s say we took 30 points of damage. Therefore, our new health would be 100 – 30 and our final health will be 70. *However there is a problem, can you figure it out?*

What if our health was not 100 before the fight started? If I fight two enemies in a row, I may have taken some damage fighting the first enemy and may not be at 100 health when fighting the second. Luckily the variable “#’HP’ will keep track of this. If the variable “HP” has the value of our current health, we can just minus 30 from the value in ‘HP’. In Python we can write this very easily:

HP - 30

The above line just simply just subtracts 30 from HP but we aren’t done yet. We have to make sure that we store this new value somewhere so that it can be reused in the future. Notice that since the result of the sum is still our health we can store it ‘HP’. What this will do is overwrite the previous value that was stored with the new value and is written as follows:

HP = HP - 30

Similarly, if the player drinks a potion of health, we can add a value to ‘HP’ and overwrite the pervious health value.

HP = HP + 50

We can also use a second variable to store the value of potion and then add the two variables like this:

Potion = 50

HP = HP + Potion

Input/Output

In this version of the game we will using a command prompt to interact with the game. This will be where you will create your character, do combat, check inventory and navigate through rooms. In order to do this, we need the game to recognise the command you enter and this is done by storing what the user writes into a string variable. Up until this point we have only been working with integer variables which are essentially variables that store whole numbers. A variable that stores a string essentially stores some text. So “hello” would be a string. We are going to learn how to ask for the user’s name and look how we can use it in the future. The command to simply output text is as follows:

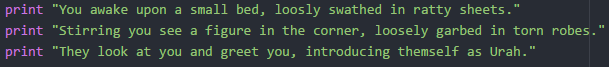
print “ ”

Inside the quotation marks is where we put the text we want to show.

*Example 1.3*

When the player starts a new game, we want to create a setting for the game. This will depend on the theme of your game but for our dungeon crawler we want a dark and mysterious environment.

We have created few print statements showing this:



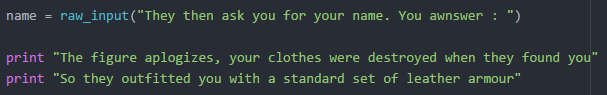
At this point we want to ask what the character’s name is but if we want the game to remember it was must store it in a variable. We can do this by using the function “raw\_input” which may look similar to a print but it also takes the next thing the user enters and assigns it to a variable:

Variable = raw\_input(“Enter a random word “)

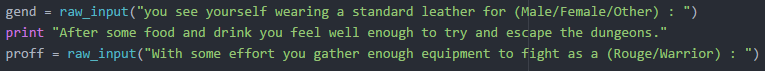
Anything that the user types and then presses enter will be stored in the variable name you give. This is particularly useful if you want to use pronouns in your story if your character is male or female which we will demonstrate later when we introduce “if” statements.

Example 1.4

Here is an example which continues the above by asking the player about their name and follows on the story:



I can also use it to store the gender and class of our character as shown below but a more in depth tutorial on character creation will be in another chapter:

When we move onto in depth character creation, we can see how the options above make a difference and we will also look at how to change the flow of the game based on the player’s choices.

MINI EXERCISE

1. Write a small program that asks for four lines of an address. Afterwards print that address using one single print statement.
2. Write a small program that asks for 5 numbers. You must then print the sum and average of those numbers.
3. Print a hollow square shape using ‘\*’ in print statements.