**Session 6**

**The world of Programming**

There are a huge number of programming languages, in this course you have been learning Python. The Zumo runs using the language C, hence the need for the libraries to translate our Python code into C for the Zumo. At last count there were over 1500 different languages which have been used to create a version of a popular song (<http://www.99-bottles-of-beer.net/>). Rosetta Stone (<http://rosettacode.org/wiki/Rosetta_Code>) is also a site to compare programming languages solving the same solution and it lists over 250 solutions for 99 bottle of beer song. So why are there so many?

Some language are created by large corporation (like Swift for Apple and Go for Google), others have been created for particular situations (like Javascript for browsers and PHP for web servers). Some were created to improve other language (like C++ or C# were designed to improve C) while others were created simply to make life easier for programmers (like Python to make life easier for beginners or Scala/Kotlin to make life easier for Java programmers). Some languages were created to solve a particular problem (like Erlang which solves multiple connections very well and is use to build exchanges) while other just grow in particular markets (like Java which is the main language used in business applications, while Ruby is used with the major web application Ruby on Rails). Some languages are useful for their speed (like C and Assembly) while others are useful because they are easy to change so are good for data crunching by data scientist like Python and R or game programmers changing their setting using mods like Lua.

Often you find major application use more than one language:

* Web applications use HTML, CSS and Javascript on the browser and Java or Python or Ruby or PHP and SQL on the back end.
* Games often use C, C++ or Assembly on their game engines and a scripting language like Python or Lua to control the huge number of settings.

Good programmers often have deep knowledge of a couple of languages but are able to write and edit code often in a dozen other languages. They do this by understanding the syntax and structure of each language, and applying their knowledge of programming.

**Syntax**

This refers to the key words and how to structure basic commands of a language. In this course the syntax of Python has been written in **bold** to help you learn. Terms like **if**, **while**, **def** etc are the syntax of Python. Syntax also means how to build commands, for example Python used indentation and white space to provide the structure around commands, while C uses semi-colons **;** at the end of each command and curly braces **{}** to create structure around functions and if statements.

**Structure**

This refers to how the language requires you to put the commands together. Python is a scripting language so it requires you to put your commands in the order that they are to be executed. Therefore it does require you to define your functions before you call them, using libraries makes this much easier to organise. C uses a similar structure while a language like Java requires everything to be in classes so beginners need to write over a dozen lines of code before you can start even the simplest program. However Java does not require you to write your code in any order since you need to compile it before being able to run any Java program. During the compile phase Java finds and checks all the commands to make sure they work. Java rarely fails at runs time but often will fail to compile if there are errors, while Python does no checks and will fail at run time if there are errors in your code.

Now that you know a bit about programming in Python it is often interesting to see how other languages solve the same problem. Often the solutions will be very similar while other language might seem quite strange at first sight, even within the one language. For example look at the solutions to 99 bottles for Perl, Javascript and Python where there are multiple solutions. Then for something really different look at LISP, Whitespace (you need to highlight this to see it) and Shakespeare (yes this does actually work).

**Collision Avoidance**

The world is moving towards self-driving cars. Companies such as Google and Uber have major investments in this area. Tesla already has a version in their cars now, it’s a feature called autopilot. Airplanes have had this feature also for many years.

Many governments are trialling this technology on both closed environments and on public roads. So how does this work, what does the programming look like?

The core process (algorithm) for this is simple. In pseudo code its

If the way is clear ahead

Move forward

Else

Do something else (turn or stop)

Putting this is python code it could be written in four lines

if is\_clear():

fwd()

else:

turn()

which uses three function that need to be written: is\_clear, fwd and turn. We could write this for the zumo and much of the work has already been done in the previous practicals. You already have a function in your movement library to move the zumo forward and you also have turning functions also in that library. The really difficult function to write is the is\_clear function. You don’t have anything like that yet.

**Motion Sensor**

The zumo has motion sensors attached to it. These sensor needs to be turned on and then they are available to read. To turn the sensor on use the library code

proxSensorInit()

and then wait for 2 seconds to ensure that it is ready. This only needs to be done once.

To read the sensors use the library code

proxSensorZumo()

which returns a **String** with four numbers separated by spaces. These numbers change depending on whether something is directly in front of the Zumo, or slightly to the left or right side. They also change whether something is close or far away.

To see this working create a file called sensor.py with the following code.

#! /usr/bin/env python3

from PiZumov1B import Zumo

from time import sleep

zumo = Zumo()

sleep(1)

zumo.proxSensorInit()

sleep(2)

while True:

print(zumo.proxSensorZumo())

This will produce 4 numbers continuously which are the results of the four sensors on the zumo.

**If you see 0 0 0 0 it means that the sensors have not been turned on. There could be an issue with your code but it is more likely that there is an issue with the zumo. Solving this issue is beyond the scope of this course, so if it happens call your lecturer over to fix this for you.**

Now we need to turn this into the function is\_clear so that we can use this information in a collision avoidance program.

The first thing you should notice is that we are using the function is\_clear in an if statement, so is\_clear must return either True or False, nothing else. It should return True if the sensor is providing the data which indicates that nothing is in front of the vehicle. There are a couple of ways to do this in Python but the simplest way is to use an if statement. So the is\_clear function should finish with:

if #the way is clear in front of the zumo:

return True

else:

return False

**In python this part could be done in just one line, but I will leave that for you to find out, it is beyond the scope of the course.**

So now we need to work out how to show that *the way is clear in from of the zumo.* This is done in three steps*.*

1. The first thing that you need to do is to capture the values that come from the sensor.
2. Next you need to get at the numbers which represent the sensor that you are interested in.
3. Finally you need to examine the information in some way to work out if it is showing you that something is in the way.

This all might take a few lines of code, though it could be done in less.

All your learning up to this point has been geared to being able to build this function, and since it is essential to your final project, your lecturer can not give you any more help. He will be able to help you with errors and troubleshooting, but he will not be able to give you how this code should work. That is your job.

The is\_clear function, in pseudo code is

def is\_clear():

#capture the information from the sensor

#get at the sensor that you are interested in (look at Session 3 and the Parser if you need

help)

#Convert the information so you can use it in an if statement (look at Session 1 and the   
 calculator for help)

if #the way is clear in front of the zumo:

return True

else:

return False

When you test run this code is should say False when you put your hand in front of the zumo and True when you take it away.

**Glossary**

|  |  |
| --- | --- |
| **Concepts** | **Meaning** |
| Command | These are key words used by the language that perform a function for the language. It is also possible to create your own commands to be used in your programming. Most languages have libraries of commands that have already been built for you to use as well as those you build yourself. |
| Functions | The most common way to create your own commands. These can accept data, process data in some way and can return the changed data to be used in other parts of the program. In a well structured program almost everything is done in small single purpose functions. |
| Arguments | These are bits of data that are used by functions, so that they have information to work with. |
| Parameters | These are the place holder variables that are used in function definitions. These are replaced by the arguments that are used when the function is run. Often the two terms are used interchangeably.  In Python Parameters can be given a default value, which means that an argument is not required when the function is called. |
| String | Data type, String refers to ordinary words. |
| Integers | Data type, int refers to whole numbers. |
| Float | Data type, Float used for decimals, but they are not accurate. |
| Variable | Way of representing data for the program to work on. |
| Libraries | These contain commands that have been developed and tested and are ready to use. Many of the libraries have been written by the people who originally developed the language while other libraries have been developed by companies or individuals that use the language (Google, Yahoo, Apache etc have all developed extensive language libraries for a wide variety of languages). You can also develop your own library of commands. |
| Assignment | Uses = to give a variable name a value |
| Expression | An operation which must be true or false |
| Comments | These are notes for people reading the code. |
| Array | Structure used to group data together, in Python there are many types including List, Set, Tuple, Dictionary etc. |
| Object methods | All Python objects have built in methods to make them more useful. These include |
| String concatenation | Adding to strings together to make one string. This can be done using the + sign. |
| Slicing | Python methods for getting inside sequences |
| Methods | Functions which are attached to objects and can be used to modify the object. In Python all the data types (Strings, Integers etc) are objects and all have methods attached. |
|  |  |
| **Python Commands** | **Meaning** |
| print | Python function which will show the data on the console |
| def | Python key word used to define your own functions |
| input | Python function which gets data from the keyboard |
| int | Python function which changes data to its integer value |
| if | Python keyword used to build decision making structures, must be followed by an expression which is either true or false |
| from … import | Python keywords used to bring in Python objects and function from other files for use in your current file. |
| while | Python keyword which creates a loop based upon the expression |
| True | Python keyword which always evaluates to true, there is also a keyword False. |
| for … in | Python keyword which creates a loop based upon the iterable structure that it is given. in is also a Python keyword to look inside Arrays and can be used in expressions |
| range | Python function which is used to create a sequence of numbers |
| .append | Python method which adds an element to a list |
| .split | Python method which changes a string to a list |
| enumerate | Python function which is used in for loops to provide both the data and the index of the data |
| len | Python function which finds the length of a sequence |
| assert | Python keyword which is followed by an expression, will throw an error if the expression is false |
| pass | Python keyword which is a placeholder, used to create function without filling in the exact implementation details yet |
| None | The value you turn from a function which does not contain a return statement. None evaluates to false but can also be tested for |
| with | Python keyword which activates the context manager attached |
| open | Python function which opens a text file. |
| as | Python keyword which creates an alias |
| .read | Python method on open which the entire file in as a string |
| continue  break | **continue** is a Python keyword which can be used in loops. It means stop processing this current iteration and start the next one. There is also a keyword **break** which can be used in loops, break means to stop the loop entirely and do no more processing on it. |
| str | Python function which changes data into a string |
| float | Python function which changes data into a float value |
| elif else | Python keywords used with if to build if chains. else can also work with some other keywords outside the scope of this course. |
| .splitlines | String method which splits on the end of line |
|  |  |
| **Best Practise** |  |
| DRY Principle | Don’t Repeat Yourself. When you need the same code in two places make a function, never copy and paste. This way if there are errors they will only be in one place. Makes you code much easier to maintain and change later on. |
| TDD | Test Driven Development a way of programming where you write your tests first. |
| Refactor | Once we have written our code and it works (with tests to prove it works) we can look at what we have done and make our code better. |
| Pseudo Code | Writing out in English inside comments what you want to do so that you can then turn this explanation into Python code. |
|  |  |