Turtle programming language is developed by Nigele McCoy. Turtle was developed for small tasks and is a scripting language. Turtle is derived from the python programming language. One of the key features of turtle, is that you do not have to write a lot code for tasks. The keywords are simple in nature, because they are short words. The syntax is very simple, because the requirements are few. Turtle is an interpreted high level language like python, and is object-oriented.

**The turtle interpreter runs through python**, this means you will need the latest version of python. You can download python 3.6 at [www.python.org](http://www.python.org). The interpreter will work fine on python 3.5.2 . The turtle programming language package will provide two interpreters, and a turtle interpreter class.

The two interpreters are not the same, one is cross platform and the other is Unix based. The turtle interpreter is cross platform but the turtle terminal interpreter is not. The turtle terminal interpreter is Unix based and receives a command line argument. This means that the interpreter can only be ran through the terminal. The interpreter class are for those that desire to pass many files into the interpreter and for the blending of python and the turtle language.

For any questions **contact** the developer at [turtle.language@gmail.com](mailto:turtle.language@gmail.com). **Documentation** will be provided at [www.turtlelanguage.blogspot.com](http://www.turtlelanguage.blogspot.com). You can follow the turtle programming language on Instagram at turtlelanguage.

The turtle programming language can be **downloaded** from [www.dropbox.com](http://www.dropbox.com).

Here is the link for the turtle programming language.

<https://www.dropbox.com/sh/lh3v8ioli99zvh5/AAAmAlJjZOAIzWc9I7IvnjKfa?dl=0>

The link is on the turtle blogspot as well. Updates for turtle will be available through

the same link.

The purpose of this booklet is to introduce you to the syntax, and features found in the first version of turtle. Being that turtle is derived from python, I expect those that are reading to have some experience in python or some other programming language. I will provide syntax example files along with the turtle package to provide greater clarity. Due to the size of this booklet you will want to look at the example files to make sure that your format is correct.

**Comments and Output**

Commenting in turtle, is like the syntax for commenting in C++.

The syntax:

// turtle code proper syntax

// turtle code proper syntax

//turtle code proper syntax

/ turtle code invalid syntax

In turtle **spacing is very important**. For commenting, there must be two forward slashes. As a rule, in turtle every keyword must have one space after it. That brings us to our first keyword which is **show**. The show keyword displays input.

Turtle is **case sensitive when it comes to keywords**. **All keywords are lowercase**.

The syntax:

show “hello” . output will be hello

show “ “ . output will be blank

show “100” . output will be 100

show 100 > 10 . output will be True

show a . output will be what a equals

show 100 \* 10 . output will be 1000

show “100” . output is indented but will show the same as the other statements

show “ ’genesis’ ” . output will be ‘genesis’

// wrong syntax

show . nothing to show. // these statements will not display

show “good” period missing

When using the **show** keyword always have some **type of content** to **display** and a **period** to end the statement. With all **statements,** they **must fit on one line** to be interpreted. The interpreter is very sensitive when it comes to strings. It is important to note, that the interpreter has been set up to **read any file extension** python can read. I would suggest that you **write your source code in python**. The reason for this is that double quotes outside python source code is formatted differently. This means other file types may not allow strings to display properly. Notice that **show can be indented when displaying digits and strings**, in most case indention in turtle will lead to some type of error.

**Variables and Input**

**The data types in turtle are derived from python**. In turtle, you have integer, float, string, Boolean, and complex. (If there are any other that I have forgotten include them as well) **The math operators are derived from python**.(\* / + - \*\* // %) The Strings that are surrounded **by single quotes in turtle, are not considered strings**. **Strings are only considered strings when they are surrounded by double quotes.** As turtle develops more this may change.

Variable declarations are **very simple**. You do not have to declare the data type. The other cool feature is that **you can change a variable data type** when re-assigning. The disadvantage of variable declaration in turtle’s first version is that you cannot do **many math operations** on variables. The other disadvantage is that **you cannot declare** a string variable that has **only special character(s)**. Example of special characters: ~:;<>?#$%!^&\*. These characters must be with non-special characters to appear properly. If you desire to have a variable that holds a special character you will have assign the variable by inputting it.

The syntax:

A = “turtle” will work properly

b = “genesis” will work properly

c = 100 + 20 will work properly c = 120

d = 100 > 30 will work properly d = True

// Assigning variables to variables

A = c data types do not matter when assigning variables

// These will not work properly

e = input error or unknown variable error will appear. nothing declared

f =”good” missing proper spacing

g = “ “ cannot declare empty strings

**Declaring variables always require that you leave the line below empty**. If you

do not provide an empty line beneath the declaration, the line underneath will not

be read. The only thing that should be beneath are **comments or an empty line**.

**Input is like declaring variables**. In turtle, you can **see the actual variable when**

**being prompted for its value**. It is important then to **name variables something that you can remember.**

The syntax:

Dog = ? to input you must use the question mark symbol.

Dog yorkie Dog will appear to the user.

// This will not work

Dog =? Will not work missing proper spacing.

Input is simple it only requires the question mark? As with declaring variables you must leave an empty line beneath the input. There is another way to create variables but we must first understand hashing and files.

**Hashing in turtle is another way of saying concatenation**. You can add strings together but with digits you can add or subtract with hashing. Hashing can be done on files. Hashing in higher version of turtle will have more features. **To hash variables, you must use a hashing statement**. The hashing statements always begins with the **# sign and the hashing variable to**. To finish the statement, you need to give the variable that is being hashed and the variable that will be added to the hashed variable. **Hashing statements always require an empty line beneath**.

The syntax:

A = “hello”

B = “ Chris”

# to a b **This is saying hash to variable a variable b**

show A . output will be hello Chris

// wrong syntax

# a to b wrong syntax

**As a rule, when it comes to hashing, hash characters with character and numbers**

**With numbers with numbers**. You will get an error if you mix data types. If you use

Numbers in a string be sure to add some other character when hashing. Example:

A = “100 people” This proper

B = “ went to the store”

# to A B output 100 people went to the store

If the 100 is by itself, when you try to hash it the interpreter may not read it correctly.

**Files**

Before we look at hashing to files, we need to understand how turtle works in

regards to files. In turtle, you do not have to write statements to open and close files.

**Turtle has objects that do all the back work for you**. These objects are known as

**blocks and will receive arguments**. **Blocks are objects and keywords**. In regards to

hashing files the same syntax with hashing variables apply. The only difference is that

you must have a file in existence and you must provide the path of that file in a string

variable.

The syntax:

File = "/Users/johndoe/Desktop/a.txt"

Text = "Hello world”

# to File Text

In this example, we are using Mac OS X path formatting but it works the same way on

windows. **Remember to leave an empty line beneath declarations and hashing**

**statements and blocks**. We will continue to look at file operations.

The syntax:

[r] file.txt reads a file

[w] file.txt creates a file to write to

[rw] file.txt reads a file and allows you to add more content to the file.

Notice how these statements have **brackets** around them. The statements that

have brackets around them are called **blocks**. Blocks normally **receive some type of**

**data** and does **some type of operation** with data received. In the case of reading,

writing, and appending to files it takes **one argument**. Arguments always begin after

the block and always require one space after the block. In the case of files, you only

need to give the **name of the file you want to read, write or append to**. If you want

read, write, or append to a file that is not in the same directory of the interpreter you

will need give **the path of that file** instead of the name. The last interesting thing that

you can do to a file is use the **get block**. This block gets the data from a file and

assigns it to variable. In this statement, you can create a variable. For now, in turtle

this is the last way you can create a variable. The get block unlike the [r], [w], and

[rw] block takes two arguments. **When you have two arguments, the arguments are**

**divided by a space, dash and a space.**

The syntax:

[get] /Users/johndoe/desktop/a.txt - info

With the **get block the first argument must be a path**. The second argument is the variable that **will hold the data**.

Now we can discuss directories and files. In turtle, you can **copy, move, and remove files**. Turtle allows you to **make** and **remove directories**. The last directory object is the take block. This block receives two arguments and **copies all the files from one directory** to **another directory**.

The syntax:

[copy] file.txt - /users/johndoe/desktop/file.txt

[move] /users/file.txt - /users/johndoe/desktop/file.txt

[remove] /users/file.txt //one argument only

The copy and move blocks requires two arguments. The **first argument is the file** or path of file and the second is the **location where the file is going to be copied or moved to**. I recommend that when working with directories whether you are in the

current directory or not **use paths for arguments**.

**Directories and Blocks**

The syntax:

[mkdir] /users/johndoe/desktop/games makes a directory

[rmdir] /users/johndoe/desktop/games removes a directory

[take] /Users/john/Desktop/ - /Users/john/Desktop/goo/

The **mkdir and rmdir block receives only one argument**. These take the path

where the directory is made, or deleted. The take block receives two arguments. The first argument is the **directory that the files are going to be copied from**. The second argument is the **directory that the files will be copied to**. The take block **can only receive paths as arguments**. One important thing to note is that **this block does not work properly on the Windows operating system**. The take block should work properly on Mac and Linux. Now that we have discussed directories, we shall discuss the remaining blocks in turtle’s first version. The next two blocks only receive one argument. The first block is the check block; its purpose is to check an expression to see if it is true or false. The check block will also show the expression along with the word true or false. The second block is the web block which opens a web page.

The syntax:

[check] 100 > 10 output will be 100 > 10 True

[web] [www.google.com](http://www.google.com) This will open a web browser and open [www.google.com](http://www.google.com)

// The wrong syntax for web block

[web] <http://www.google.com> will not work. You do not need the http header.

The last two blocks in turtle’s first version deal with the interpreter. The first is

the **output block**, and it copies all output to a file. The file will have a list of the output.

The output block receives one argument and that argument is to **receive the name** of

the file you want **to create to hold the output of the file**. The second block is the exit

block and it stops the interpreter. Be careful of using the exit block because the

interpreter runs very quickly. This means that the output may not be displayed

because the interpreter will stop before it can display output. I **recommend using the**

**exit block at the end of the program**. This will make sure that the whole program

runs. One thing to note is that the interpreter will stop once it reads the whole file.

The other thing about the exit block is that you once it is used then it **cannot be**

**reversed**. The **interpreter must be restarted**, after the exit block is used.

The syntax:

[output] file.txt One argument // a path can be provided as well

[exit] no argument

Before I end this booklet and I need to discuss how the interpreters works. The

interpreters only receive files. The terminal interpreter receives a command line

argument. The terminal interpreter only receives one command line argument. Due to

how python works on windows you use an **escape key to exit the interpreter**.

The reason for this, is so that you can see the output. On other operating systems, you

can see the output when the interpreter ends.

The other interesting feature of the interpreter is that it tells you where you

have went wrong. **The interpreter gives the line, and the issue that is found**. If the

interpreter finds an issue that it does not recognize, unknown variable will appear.

When the interpreter reads an error, the interpreter stops at that line and will read

no more from the file.

I hope that you enjoy the turtle programming language. If you have any

questions, problems, or suggestions please contact the developer at

[turtle.language@gmail.com](mailto:turtle.language@gmail.com).