TINKER ACADEMY

Mobile App Development

Handout 2: Lua Language Fundamentals

Note your Student ID. You will need to use it throughout the Course.

Connecting to the Network

1. Select “Cupertino Community Center” if your Student ID is divisible by 2. Else choose “Cupertino Community Center 3”. Alternatively choose “AndroidAP” (password “tinker2014”).
2. Open a browser (preferably Chrome, Safari or Firefox)
3. Type in “www.google.com” (without the quotes). Type in *exactly* as indicated. If nothing shows up, check again, did you include the 2 “dots”? Did you spell www google and com correctly.
4. You should be taken to a Cupertino Parks and Recreation. Accept the agreement and click the required buttons to activate the network.

Launching the Virtual Machine in Full Screen

1. Use the VirtualBox menu View->Switch to Fullscreen to switch the VM to fullscreen mode
2. Use the same VirtualBox menu View->Switch to Fullscreen to switch the VM back out of fullscreen mode

Shutting Down the Virtual Machine

1. Click on the red close window button (to the top left on a Mac, top right in Windows).
2. You will prompted with a confirmation message asking if you want to “Power Off” the machine. Click the button to confirm power off.
3. In a few minutes the VM will shut down and you should see the VirtualBox side panel with the “Tinker academy” VM indicating “Powered Off”

Restarting the Virtual Machine

1. Start VirtualBox
2. Click on the VM “TinkerAcademy” in the VirtualBox side panel
3. Click on the Start Button (Green Arrow) in the VirtualBox Toolbar. This will start the VM.
4. Once the VM startup you will be presented with a login screen.

Right Click in VM on Mac

1. Open System Preferences, Trackpad

2. Enable “Secondary Click”, Toggle the small arrow to the right and select “Click with two fingers”.

Updating the Course

1. Make sure you are on WiFi.
2. Click the “Setup” folder in “Nautilus” under “Bookmarks”
3. Double click “Course Update”. Choose “Run”. Notify an Instructor if you see a window popup with the message “update course failed”. *You messed up. No, just kidding :). We’ll fix it for you.*
   If you are doing this after class hours:
   Hop onto Skype, and request help in the class chat group.
2. Click the “Courses” folder under “Bookmarks”. Navigate to the TA-JAV-1 and locate the “Quiz0.odt” under “quiz0” (which is under “quiz”). Select the file.
3. Double click Quiz0.odt to open it in LibreOffice 3
4. Answer the 5 questions in the Quiz. Once you are done, navigate to the top to see the menu and select File->Save. Alternatively use Ctrl S

Submitting Homework

1. Make sure you are on WiFi.
2. Click the “Setup” folder in “Nautilus” under “Bookmarks”
3. Double click “Course Submit”. Choose “Run”. Notify an Instructor if you see a window popup with the message “submit course failed”.

Getting Ready to Program

Open StarterPack2.lua
We will use StarterPack2.lua to write a complete Lua program

Click on StarterPack2.lua under “Courses” “TA-GME-1” “starterpack” “starterpack2”. Right Click. Select “Open With Sublime Text 2”.

Click on “Terminal”.

Structure of StaterPack2.lua
StarterPack2.lua is a file containing lua code. Any file containing lua code is called a lua chunk.

Run the Program
Type in the following commands in the Terminal exactly as shown below

Line #	Command	What does the command do?
1	cd	Takes you to your HOME directory

<table>
<tr><td>2</td><td>source .bash_profile</td><td>Enables all shortcut keys</td></tr>
<tr><td>3</td><td>golua2</td><td>Shortcut key to take you directly to the starterpack2 directory
<table>
<tr><td>golua3
golua4
golua5
golua6
golua7
golua8</td><td>Shortcut to starterpack
starterpack3
starterpack4
starterpack5
starterpack6
starterpack7
starterpack8</td></tr>
<tr><td>goluahomework2
goluahomework3
goluahomework4
goluahomework5
goluahomework6
goluahomework7
goluahomework8</td><td>Shortcut to homework
homework2
homework3
homework4
homework5
homework6
homework7
homework8</td></tr>
</table></td></tr>
<tr><td>4</td><td>lua StarterPack2.lua</td><td>Runs the lua interpreter on the StarterPack2.lua program</td></tr>
</table>

You should see the following in the Terminal

```
Enter a number:
```

Type in an integer number between 1 and 10 and press Enter

You should see the factorial of the number displayed.

A factorial is a number multiplied by itself and all numbers less than it upto and including 1.

Structure of a Lua Program File

The program file containing the Lua program is a Lua chunk.

A chunk is simply a sequence of lua statements, one in each line.
(Lua needs no semicolons between the statements, unlike Java, C and C++)

A chunk can be as large as you wish.

Capitalization matters in Lua.

print is not the same as Print or PRINT or pRiNt

Here is the content of the StarterPack2.lua file

Line #	StarterPack2.lua
1	----------
2	---- StarterPack2.lua
3	---- Starter Pack for Mobile App Game Development
4	---- Copyright Tinker Academy 2014
5	----------
6	
7	function fact (n)
8	if n == 0 then
9	return 1
10	else
11	return n * fact (n - 1)
12	end
13	end
14	
15	print("Enter a number:")
16	a = io.read("*n")
17	print(fact(a))
18	
19	soccerBall = {}
20	soccerBall.speed=10
21	
22	my3FavSoccerTeams = {}
23	my3FavSoccerTeams[1] = “USA”
24	my3FavSoccerTeams[2] = “Brazil”
25	my3FavSoccerTeams[3] = “Argentina”
26	
27	-- TODO: add your code below
28	
29	

Lines 1-5 represent a File level comment. A File level comment describes the contents of the File.

Single line comments begin with 2 dashes (no space between the dashes, you can add more dashes for readability)

```
-- This is a comment
```

Line 6 is a blank line and will be ignored by the interpreter.

Lines 7 - 13 define a function.

What is a function?
A function is a reusable piece of code that takes some inputs, does some work and (usually) returns some output.
Once a function is defined in a program file, other parts of the program and even other programs can call it.
Functions are incredibly useful in programming.
Functions in lua are extremely powerful in Lua. They are more powerful than “methods” in Java as you will see later in the course.

Line 7 begins the function definition.

```
function fact (n)
```

Every function definition begins with the keyword function.
The keyword is followed by the name of the function.

Here the function name “fact”

The name of the function is followed by a pair of opening parenthesis (and a closing parenthesis).

The content between (and) is the input to the function. It contains a list of variable names separated by a comma (,). In the function above, the input is a single variable called “n”.

Line 13 ends the function definition with the keyword end.

Lines 8 - 12 define the body of the function. The body is another chunk (its a sequence of statements).

The function “fact” calculates the factorial of a number.

It is not important to understand exactly how the factorial is calculated. It is important to understand that the function uses the input n to calculate the factorial and return a value.

Lines 9 and 11 contain the keyword return that is used to return a value.

Line 14 is a blank line and will be ignored by the interpreter.

Lines 15-17 contain Lua statements.

Line 15 is the familiar print statement to print some text to the console.
The statement calls the function print using the text input. The print function is already defined and available for you when Lua was installed. It is part of Lua's standard library. The function prints the input to the console.

Line 16 defines a variable a (We will come back to it)

Line 17 is also the print statement but it does much more

1. First it calls the function fact with the value of the variable a (defined in Line 16)
2. Next it takes the value returned from the function (i.e the factorial of the number) and then calls the function print using the value as input.

Line 16 defines a variable named a and assigns a value to it

The variable a is assigned the value you entered when prompted.
For now, all you need to know is that

```
io.read("*n")
```

is a call to the function "read".

The statement

```
a = io.read("*n")
```

assigns the value returned by the function to the variable "a".

The function "read" waits for the user to enter a number in the Terminal, and returns the number.

Line 18 is a blank line and will be ignored by the interpreter.

Line 19 creates an empty table called "soccerBall".
Think of a table as a table in a spreadsheet with a header row, 1 data row and an unlimited

number of columns.

Visual model for “soccerBall”. “soccerBall” is currently empty.

Line 20 adds a header and data to the table. The header is “speed” and the value is 10.

speed						
10						

The header “speed” is called a key and 10 is the value for the key “speed”

Line 21 is a blank line and will be ignored by the interpreter.

Line 22 creates an empty table called “my3FavSoccerTeams”

Visual model for “my3FavSoccerTeams”. “my3FavSoccerTeams” is currently empty.

Lines 23 - 25 adds entries to this table. The Entries use numbers as keys starting from 1. You can think of this table as a list.

After the entries are added, the table will look like below

1	2	3				
USA	Brazil	Argentina				

Line 26 is a blank line and will be ignored by the interpreter.

Line 27 is a comment as you know by now
You will be adding more code below Line 27 :)

Variables

You were introduced to variables in the previous class. A Variable is a placeholder for a value. It has a name. The name can be any string of letters, digits or underscores. It should not begin with a digit. It should contain any spaces.

By convention a variable begins with a lowercase letter.

Capitalization matters in Lua.

Variables are assigned values. The code below assigns the value to the variable “b”.

```
b = 10
```

Variables such as the one above are called global variables.

You can use the variable later in the program. This is called accessing the variable.
The code below accesses the variable “b”.

```
c = b * 100
```

Lua is a very lenient language. Global variables can be accessed before they are assigned any value.

What will be printed to the console below?

```
print(b)
b = 10
```

The output will be the text “nil” to indicate that b has no value or “nil” value.

Types

Open the Lua program file “StarterPack2.lua” in Sublime Text 2.
Type in the following code on line 27 (after the TODO comment)

Line #	StarterPack2.lua
27 28 29 30 31 32	-- TODO: add your code below b = 10 print(b) b = “This is a string!” print(b)

33 34	

Run the program StarterPack2.lua.

The following gets printed to the console

```
10
This is a string!
```

In the code above you used one variable and initially assigned a number to it and then immediately assigned some text to it!

In a language such as Java, C or C++, you would need to first inform the compiler if the variable is a Number or text.

Lua (and Javascript and Python and most interpreted languages) allow this. These languages are called dynamically typed languages.

However Lua does keep track of if the variable currently has a number or text. Text is given a special name called “string” in Lua (and most other languages). In fact, it tracks much more.

- number
- string
- boolean
- nil
- function
- table

number
number is the type that represents number values.

string
string is a type that represents a sequence of characters.

boolean
boolean is a a very useful type that has only 2 possible values **true** or **false**

nil
nil is a special type. The type has only 1 value called nil. This value is given to a variable when it is not assigned a value.

function
function is a special type representing a function.
As mentioned functions are incredibly powerful in Lua.

table
table is a special type representing a table

Open the Lua program file “StarterPack2.lua” in Sublime Text 2.

number type

Type in the following code below

Line #	StarterPack2.lua
32	print(4)
33	print(type(4))
34	print(0.4)
35	print(type(0.4))
36	
37	
38	
39	

Line 32 will print the the number constant 4 to the console
Line 33 will print “number” to the console
Line 34 will print the number constant 0.4 to the console
Line 35 will print “number” to the console

string type

Type in the following code below

Line #	StarterPack2.lua
36	print(“first”)
37	print(type(“first”))
38	print(‘second’)
39	print(type(‘second’))
40	print(‘line 1\nline 2’)
41	print(‘line with “quotes”’)
42	
43	
44	
45	

Line 36 will print the string constant “first” to the console
Line 37 will print “string” to the console

Line 38 will print the string constant “second” to the console - note single quotes
Line 39 will print “string” to the console
Line 40 will print the 2 lines to the console
Line 41 will print text with quotes to the console

function and table type

Type in the following code below

Line #	StarterPack2.lua
42	print(type(fact))
43	print(type(soccerBall))
44	print(type(my3FavSoccerTeams))
45	
46	
47	
48	

Line 42 will print “function” to the console
Line 43 will print “table” to the console
Line 44 will print “table” to the console

Control Structures

Open the Lua program file “StarterPack2.lua” in Sublime Text 2.

if-then-else

Type in the following code below

Line #	StarterPack2.lua
45	function isEvenNumber(n)
46	if n % 2 == 0 then
47	return true
48	else
49	return false
50	end
51	end
52	
53	print(isEvenNumber(2))
54	print(isEvenNumber(3))
55	
56	
57	

Lines 45 to 51 define a function “isEventNumber” that has one input variable “n”.
The function uses the if-then-else control structure to calculate if the number is even or odd and returns true if the number is even and false if the number is odd.

Line 53 calls the function with an even number and prints the result (true) to the console.
Line 54 calls the function with an odd number and prints the results (false) to the console

An if-then-else tests the condition and executes the then-part if its true or its else-part accordingly.

The condition in the code above is

```
n % 2 == 0
```

n is a number.
% is a special character that is used to calculate the remainder after the division of the number to the left by the number to the right. We can identify if a number is even by comparing the remainder with 0. We use 2 double dashes (not 1) for comparing 2 values.

Remember if “n” is even it should be divisible by 2 which means the remainder is 0.

Open the Lua program file “StarterPack2.lua” in Sublime Text 2.

numeric for

Type in the following code below

Line #	StarterPack2.lua
55	function printList(list)
56	for i = 1, 3 do
57	print(“my #“ .. i ..” favorite team is “ .. my3FavSoccerTeams[i])
58	end
59	end
60	
61	printList(my3FavSoccerTeams)
62	
63	
64	
65	
66	
67	

Lines 55 to 59 define a function “printList” that has one input variable “list”.
The function uses the numeric for control structure to loop through the entries in the list and print the values.

The numeric for loop is defined below

```
for i = 1, 3 do
```

The loop will do the following

Loop #	Assign	Action
1	i = 1	Run statement on Line 57
2	i = 2	Run statement on Line 57
3	i = 3	Run statement on Line 57

Local Variables

We will be using a special keyword “local” when defining variables within functions.

Type in the following code below

Line #	StarterPack2.lua

62
63
64
65
66
67
68
69
70
	function printLocal()
	  local b = 10
	  print(“b=” .. b)
	end
	
	printLocal()

Lines 62 - 65 defines a function named “printLocal” that has no input and prints the value of the local variable b defined within the function.

Line 67 invokes the function.

Table properties

You will be using tables extensively as part of your mobile app/game programming.

As you can see there are 2 types of tables

Associate keys with values. For example, associate 10 with the key “speed” for the table “soccerBall”.

Lists. For example the list of my3FavSoccerTeams. The list starts at 1.

When used as the list, you will sometimes need to get the number of entries. This is done using the # operator

The code below will print the value 3 to the console.

```
print(#my3FavSoccerTeams)
```

That was a LOT we covered!

You made it this far! Awesome!

In the next class, we will be moving to creating Mobile Applications and Games using the Corona Game Engine.

Quiz 2: Lua Language Fundamentals

Open the Quiz

Make sure you are on WiFi.
Follow the instructions in “Updating the Course” in this Handout.
Open Quiz2.odt under “Courses” “TA-GME-1” “quiz” “quiz2”

Complete the Quiz

1. Attempt each question. Type in the answers in the “Answer:” box.
2. Save the file using File->Save or Ctrl-S

Submit the Quiz

Make sure you are on WiFi.
Follow the instructions in “Submitting Homework” in this Handout.

Homework 2: Lua Language Fundamentals

Overview

Open the Homework

Make sure you are on WiFi.
Follow the instructions in “Updating the Course” in this Handout.
Open Homework2.lua under “Courses” “TA-GME-1” “homework” “homework2”

Complete the Homework

You will need to refer to this handout (Handout2).

Homework
Open the Lua Source File “Homework2.lua” using Sublime Text 2

Create a function named "**addFactorials**"
The function takes **2 inputs n and m** and does the following

1. calculates the factorial of n (by calling fact(n))
2. calculates the factorial of m (by calling fact(m))
3. adds the 2 factorials together and prints the result to the console

If your code is correct, the program will print the following to the console

3
26

Make sure you save your program.

Test your program. If your program does not run successfully you will not get any credit.

Submit the Homework

Make sure you are on WiFi.
Follow the instructions in “Submitting Homework” in this Handout.