**3.1**

Because lua does not have switch statements, so in order to have multiple conditions a ifelse construct is required.

**3.2**

|  |
| --- |
| -- unconditional while loop  function whileloop() |
| while true do |
| print("loop") |
| end |
| end |
|  |
| -- unconditional repeat-until loop |
| function repeatloop() |
| repeat |
| print("loop") |
| until false |
| end |
|  |
| -- unconditional |
| function gotoloop() |
| ::loop:: |
| print("loop") |
| goto loop |
| end |
|  |
| function forloop() |
| for i = 1, math.huge do |
| print("loop") |
| end |
| end |

I prefer the while loop as it’s what I am most accustomed to while using c++

**3.3**

Agreed. As there are not many cases where the conditional check needs to be done after the body, most times you would check before the body. And in the seldom cases you do need to have the body run once, you could just rewrite the body before the loop

**3.4**

function concat(...)  
 local s = "";  
 for i,v in ipairs{...}  
 do  
 s = s .. v;  
 end  
 return s;  
end

**3.5**

Pros:

* The function is prebuilt

Cons:

* Has ‘i = 1’ as a built in default so you can’t change the starting value.
* Cant return the index for printing as you could do with your own function

function ArrayPrint(...)  
 for k,v in pairs(…)  
 do  
 print(k,v);  
 end  
end  
  
Tab = {7, 9, 78};  
ArrayPrint(Tab);  
print(unpack(Tab));

Output:  
1 7  
2 9  
3 78  
7 9 78

**3.6**

function PrintIgnoreFirst(...)  
 local tab = {...};  
 for i = 2, #tab do  
 print(tab[i]);  
 end  
end