Introduction to Learn Swift 2019 coding challenges!

Ready to start honing those swift skills you just learned? Then look no further as these extra challenges will not only help you practice but also help you recap on what you have learned so far.

So get those brain juices flowing and get those hands busy typing because its challenge time!

For the full playlist please use this link:
http://bit.ly/LearnSwift2019

For the current updated solutions please use this link:
http://bit.ly/LearnSwiftChallengesSolution

If you don't have a mac or are having problems with XCode you can use this online playground:
http://online.swiftplayground.run/

Challenge 1: Getting used to swift!

Video for the lesson can be found here: http://bit.ly/LearnSwift2019Challenge1

Let us set up our basic data, Mastery requires practice!

First, create and set variables with your ***lastName*** and ***firstName*** respectively.
Then, create a constant variable that is your ***gender***.
Third, create a variable that is your ***age***.
Finally, create a variable that is your ***cashOnHand***.

Print all of the information using the variables and constants you just created.

Challenge 2: Let's take that first step to a better you!

Video for the lesson can be found here: http://bit.ly/LearnSwift2019Challenge2

Use the basic variables that you created on challenge 1.

We should always strive to improve ourselves, and it is also the same in coding!

Add the respective data types for your variables.

firstName, ***lastName***, and ***gender*** should be ***String***.
age should be ***Int***.
cashOnHand should be a ***Double***.
In Addition, create and set a ***Bool*** variable named ***hasChildren***

print all of the information using the newly improved variables

Challenge 3: If life permits!

Video for the lesson can be found here: http://bit.ly/LearnSwift2019Challenge3

Use the variables you have set up on challenge 2.

We want to buy a game we always wanted but it can only be purchased if we can fulfill certain conditions! Why? because if life permits!

Add if statements after all the declarations.

Write an if statement with the logic that [***if you have children***] then ***print*** *"Being a parent is hard, my money goes to my children instead of games!".*

Additionally, write an if statement with the logic that [***if your Age is greater than 18***] then ***print*** *"Adulting is hard I can't buy the game because I need to pay bills".*

Finally, ***if none of these is true*** then ***print*** the statement *"I'm young and I can do what I want so gimme that game!"*

Challenge 4: Calculator app!?

Video for the lesson can be found here: http://bit.ly/LearnSwift2019Challenge4

Let's switch things up a bit to make it interesting and make our first calculator app!

For this challenge we will be using the Switch statement.

Create a ***String*** variable named **str*Operator*** and set up an initial value that you want for it.

Then create two ***Int*** variables named ***num1*** and ***num2***. You can assign any value you want for it.

Lastly, create a third ***Int*** variable called ***result***. You can initialize it with 0 or leave it be for now.

Use **str*Operator*** as the "*seed*" of our switch statement.

The first ***case*** should check if the value of operator is add (**"+"**). Create the logic by saving the resulting answer and saving it in the ***result*** variable. **print** the ***result*** before breaking out of the switch statement.

The next ***case*** should check if the value of operator is minus (**"-"**). Create the logic by saving the resulting answer and saving it in the ***result*** variable. **print** the ***result*** before breaking out of the switch statement.

The next ***case*** should check if the value of operator is multiply (**"*" and "x"**). Create the logic by saving the resulting answer and saving it in the ***result*** variable. **print** the ***result*** before breaking out of the switch statement.

The next ***case*** should check if the value of operator is divide (**"/"**). Create the logic by saving the resulting answer and saving it in the ***result*** variable. **print** the ***result*** before breaking out of the switch statement.

Finally, the ***default*** (catch-all) ***case*** should handle any wrong input of operators. If so, it should ***print*** *"Operator does not exist"*.

Challenge 5: Draw a Triangle app!?

Video for the lesson can be found here: http://bit.ly/LearnSwift2019Challenge5

For what reason to we create apps? To have fun ofcourse! This app should be able to draw a right triangle using the drawPixel as base.

For this challenge we will be using the For Loop statement.

Create a ***String*** variable named ***drawPixel*** and set up an initial value that you want for it.

Then create an ***Int*** variable named ***height***. You can assign any value you want for it (preferably 10 or below).

Additionally, create a ***String*** variable named ***tempRow*****,** you can initialize it as an empty string if you want.

First create a ***for loop*** with value ***columnPixel*** from 1 to the specified ***height***. First step would be to assign ***tempRow*** as an empty ***String*** (“”)

Then, create a ***nested for loop*** (loop inside the for loop) from 1 to the specified ***columnPixel.*** Inside this nested for loop append ***tempRow*** with the specified ***drawPixel***

Finally, ***outside the nested for loop*** you should ***print*** your final value for the ***tempRow***.

See the magic come alive!

As an example let’s assume that ***drawPixel*** is **“x”** and ***height*** is ***5***. The resulting output should be:
x
xx
xxx
xxxx
xxxxx

Challenge 6: Investment Forecast!?

Video for the lesson can be found here: http://bit.ly/LearnSwift2019Challenge6

For what reason do we create apps? To make life easier ofcourse! This app should be able to calculate investment forecasts based on the values you set.

For this challenge we will be using the repeat while statement.

Create and Use the ***Double*** value ***cashOnHand*** on challenge 2. Also Create a ***Double*** value called ***runningCash***.

Create a ***Double*** variable named ***percentGain*** and set a value you want for it.

Create an ***Int*** variable named ***yearsToInvest***. Set a value that you want for it, the longer the better.

Create an ***Int*** variable named ***yearsElapsed.*** Set a value the value to ***0***

Let's assume that a Broker has offered you to invest in the stock market with an a fixed gain of ***percentGain*** per year. You take the offer and use all of your ***cashOnHand*** to invest.

Use the ***repeat while Loop*** to calculate for your possible return of investment after your specified amount of ***yearsToInvest***.

First, set the value of ***runningCash*** to be the same value as ***cashOnHand.*** Let's also convert that ***percentGain*** to a real percentage by **dividing it by 100.**

Then, set up your ***repeat while*** loop, the code will loop while ***yearsElapsed*** is less than ***yearsToInvest***.

Inside your loop recalculate your ***runningCash*** where the new value is equal to the current value + its ***percentGain***.

Print the value of ***runningCash.***

Finally, ***yearsElapsed*** should add by 1 before the loop checks the ***while*** condition.

How rich will you be after all those years?

As an example let's assume that ***cashOnHand*** is **"2000"**, ***percentGain*** is "***10***" and ***yearsToInvest*** is "***5***". The resulting output should be:

2200.0
2420.0
2662.0
2928.2
3221.02

Challenge 7: Walking on all four

In order to grasp basic movement we should all go back to our toddler years. To a time where we are crawling on all fours. But this activity is not about that, this is about walking on all four directions! (North, South, East, West).

For this challenge we will be using basic functions without parameters.

Create a function called ***walkNorth*** and have it print the line ***"You walked North"***.

Then, create a function called ***walkSouth*** and have it print the line ***"You walked South"***.

Then, create a function called ***walkEast*** and have it print the line ***"You walked East"***.

Then, create a function called ***walkWest*** and have it print the line ***"You walked West"***.

Finally, call all the functions you have created.

Challenge 8: Walking on all four efficiently

Previously you have learned how to walk on all four directions using specific functions but as expected it's all vague and has no purpose. Just as a toddler learns to stand on and walk firmly using his feet we will move stand tall and move forward using our own two parameters!

For this challenge we will be using functions with parameters.

Create a function called ***walk*** which will take **two parameters** and will return a ***String*****.**

The first parameter should be named ***direction*** and will be of type ***String***, the argument label should be set as ***_.***

The second parameter should be named ***steps*** and will be of type ***Int,*** the argument label should be set as ***_.***

Inside the function return a string on which ***direction*** you have walked and how many ***steps*** did you take. Don't forget to type cast ***steps***

The format of the return statement should be:

return *"You have walked "* + ***String***(***steps***) + *" steps to the "* + ***direction***

Then, create a ***String*** called ***resultStr*** and assign to it the function and what data you want to display.

Finally, print the contents of ***resultStr.***

As an example let's assume that ***direction*** is **"***North***"** and ***steps*** is *5*. The resulting output should be:
You have walked 5 steps to the North

Challenge 9: What are those?!?

What are those?!? No I'm not talking about those bunny flip flops that you are wearing. I'm talking about real bunnies, or to be more specific, just pets in general!

To avoid further confusion we should just define what we want by the use of classes!

Thus, For this challenge we will be using classes with some functions mixed in.

First let's create a class called ***Pets*** and assign some variables to it.

It should have a ***name*** of type ***String*** and ***age*** of type ***Int.***

You can go ahead and assign default values to your variables, ***name*** should be **""** and ***age*** should be ***0***.

Next, let's create a function called ***feed()*** and have it print ***"\(name) has been fed".***

Then, create a function called ***clean()*** and have it print ***"\(name) has taken a bath".***

Then, create a function called ***play()*** and have it print ***"\(name) enjoyed playing with you".***

Then, create a function called ***sleep()*** and have it print ***"\(name) went to sleep".***

Save your class and move on to the next step.

Finally, create a variable named ***pet*** of type ***Pets*** and assign whatever you want for its ***name*** property.

Call your ***pet*** variable and call the **functions** to test out the app.

Challenge 10: Tamagotchi App?!?

*_ehem_* old people probably remember what a Tamagochi is so this should be no problem. But for those curious to what it is, it is essentially a virtual pet in a hand held pocket device that you feed, clean, nurse when sick, play with, and make it sleep. It's awesome but a lot of work at the same time!

It's hard to find those kinds of toys around now so for nostalgia's sake why don't we make one ourselves?

For this challenge we will be using subclasses with function overrides.

First let's create a class called ***Tamagotchi*** which will be a subclass of ***Pets*** (from challenge 9) and assign some variables to it.

It should have additional variables/attributes. Namely ***hunger*** of type ***Int, dirt*** of type ***Int***, ***boredom*** of type ***Int,*** and ***drowsiness*** of type ***Int***.

Assign default values to your variables, set all to 0 for now.

Next, override your ***feed()*** function and call its superclass. Have it set your ***hunger*** to **0** while also increasing your ***boredom*** by **20, *dirt*** *by* **20** and ***drowsiness*** by ***10*****.**

Then, override your ***clean()*** function and call its superclass. Have it set your ***dirt*** to **0** while also increasing your ***hunger*** by **20, *boredom*** *by* **20** and ***drowsiness*** by ***10*****.**

Then, override your ***play()*** function and call its superclass. Have it set your ***boredom*** to **0** while also increasing your ***hunger*** by **20, *dirt*** *by* **20** and ***drowsiness*** by ***10*****.**

Then, Then, override your ***sleep()*** function and call its superclass. Have it set your ***drowsiness*** to **0** while also increasing your ***boredom*** by ***20***, ***hunger*** *by* **20** and ***dirt*** by ***10*****.**

Next, create a function called ***check()*** have it print your 4 variables namely ***hunger, dirt, boredom,*** and ***drowsiness.***

Save your ***Tamagotchi*** class and move onto the next step.

Finally, create a variable named ***game*** of type ***Tamagotchi*** and assign whatever you want for its ***name*** property.

Call your ***game*** and call the ***methods*** to test out the app. Have fun and don't let the numbers reach too high!

Challenge 11: Update the Tamagotchi App?!?

Your Tamagotchi App has been getting rave reviews lately but a lot of people are having a hard time setting up their virtual pets and some even say that it lacks substance!. Why don't we prove them wrong and update our app to match their needs right? ;)

For this challenge we will be using setting up initializers for our Pets and Tamagotchi class while also applying some conditional statements to make our app more engaging.

For the Pets class

First, set up a basic initializer for your ***Pets*** class. Make an initializer with no arguments and have it set up a basic ***name,*** you can set the value to whatever you want.

Next, create an initializer take takes the ***name*** argument and set it as the initial ***name***.

Finally, Test out your ***Pets*** class to see if it works.

For the Tamagotchi class

First, set up a basic initializer for your ***Tamagotchi*** class. Remember to call your ***super init*** to have it set up your basic ***name***. Additionally, set ***boredom*** to **60** so your users can start playing with their virtual pets immediately after initializing.

Next, create an initializer that takes the name argument and set it by calling your ***super init*** class. Additionally, set ***boredom*** to **60** so your users can start playing with their virtual pets immediately after initializing.

Then, modify the ***feed(), clean(), play(),*** and ***sleep()*** function that you have created, make it so that it checks if the value is **0** before deciding the appropriate course of action.

For the ***feed()*** function check if ***hunger*** is **0**, if it is then print out ***"\(name) is already full".*** Else just call the ***super*** class and do the same operations as before.

For the ***clean()*** function check if ***dirt*** is **0**, if it is then print out ***"\(name) is already clean".*** Else just call the ***super*** class and do the same operations as before.

For the ***play()*** function check if ***boredom*** is **0**, if it is then print out ***"\(name) is already done playing".*** Else just call the ***super*** class and do the same operations as before.

For the ***sleep()*** function check if ***drowsiness*** is **0**, if it is then print out ***"\(name) has already slept".*** Else just call the ***super*** class and do the same operations as before.

Next, modify the ***check()*** function. Let the values print same as before but add these if statements after the printing of values.

if ***hunger*** is **>= 60** then print out ***"\(name) is hungry".***

if ***dirt*** is **>= 60** then print out ***"\(name) is dirty".***

if ***boredom*** is **>= 60** then print out ***"\(name) is bored".***

if ***drowsiness*** is **>= 60** then print out ***"\(name) is sleepy".***

Finally, save your progress and test out your new and improved ***Tamagotchi*** app.

Challenge 12: Missing Link

Has anybody noticed that our Tamagotchi app seems to be missing something? Did anybody also notice that there was a variable that was set in both Pets and Tamagotchi classes but was never used? It's the age variable isn't it?.. I'll tell you why, it's because I forgot all about it! Haha just kidding *shifty eyes*

Jokes aside it's actually because we will use that variable to apply what we learned on lesson 13 and 14 so if you haven't watched that yet now is the time ;)

For this challenge we will be using **optionals** and **properties.**

For the Pets class

Update your ***Pets*** class and delete the variable ***age.*** This is because we won't be using this placeholder value anymore.

For the Tamagotchi class

First, create a variable named ***ageInDays*** of Type ***Double***.

Set the value of ***ageInDays*** to 0 on both of your existing initializers (***init***)

Next, let's change our ***age*** variable to an optional ***Double*** type. Make this variable a computed property. Let's assume that the number of times your pet slept is his number of days in age. Thus, have the computed property of ***age*** be equal to ***ageInDays*** divided by ***30***. So we can get your pet's ***age*** in months term.

Additionally, edit your ***sleep()*** method so that every time it is called ***ageInDays*** will be increased by 1.

Next, create a method called ***getAge()*** and have it print ***"\(name) is \(age) months old".*** Be sure to unwrap your ***age*** variable here.

Then, initialize you pet and do your usual routine (***feed(), play(), clean(), sleep()***) and repeat it as many times that you like. Remember to use any of the ***loops*** that you have learned so far.

Finally, check your pet's ***age*** by calling the method ***getAge()***

Challenge 13: Spring Cleaning!

Something seems wrong here. That's right we are repeating the set up of our values in our different initializers. It looks messy and should not be necessary so we'll do some spring cleaning!

We will apply what we learned on lesson 15 so if you haven't watched that yet now is the time ;)

For this challenge we will be using **designated** and **convenience** initializers**.**

For the Tamagotchi class

First, try to notice that seems to be the repeated codes in our initializers. Did you notice it? Our set up for **boredom** and **ageInDays** is repeated in our 2nd initializer where we take the name parameter and assign it. Thus, it looks kinda messy so lets **delete the 2nd initializer** as a start.

Next, create a convenience initializer that takes the ***name*** argument and set it the ***name*** variable in your class (***self***). Ofcourse in order to initialize our other values we should class our own class's basic initializer (***self.init***).

Finally, test your code to see if everything works as intended.

Now our class certainly looks cleaner and much more efficient now!

Challenge 14: Continuous improvement!

There seems to be a bit of a problem with the memory consumption of your Tamagotchi app! Your users said that some of the variables are taking too much space in memory! Why don't we improve our app then!

We will apply what we learned on lesson 16 so if you haven't watched that yet now is the time ;)

For this challenge we will be using **arrays.**

Take a moment to look at our current variables and see what you think can be bundle up. Done yet? It's actually our properties (***hunger, dirt, boredom, drowsiness***) instead of taking the space of 4 variables we can just make a single variable (array) of our current properties!

First, let's just agree where to map our variables. Thus, ***hunger*** is **0**, ***dirt*** is **1**, ***boredom*** is **2**, and ***drowsiness*** is **3**.

Next, let's create an ***Array*** variable named ***properties*** of type ***Int*** and initialize it to all 4 zeroes.

Then, update all the corresponding variables on all of the initializers and methods, make sure to add comments to remind you what map of the array is what.

Finally, test your app to see if it still works ;)

Challenge 15: Final hurdle!

There seems to be another problem with your Tamagotchi app! The properties are so ambiguous that you have a hard time keeping track on which property is which. Why don't we fix that then!

We will apply what we learned on lesson 17 so if you haven't watched that yet now is the time ;)

For this challenge we will be using **dictionaries.**

Take a moment to look at our current set up and see what's wrong. Done yet? It's actually our properties (***hunger, dirt, boredom, drowsiness***) they are mapped as ***hunger*** is **0**, ***dirt*** is **1**, ***boredom*** is **2**, and ***drowsiness*** is **3**. It's hard to remember and doesn't make much sense. Thus, instead of using an array we will be using a ***dictionary*** to map our ***properties*** variable

First, let's delete our ***Array*** variable named ***properties*** as we won't be needing it anymore.

Next, let's create a ***Dictionary*** variable named ***properties*** of type ***String:Int***.

Next, edit your ***init*** and add the initialize your dictionary by mapping out the values (***hunger, dirt, boredom, drowsiness***) all values are **0** except for ***boredom*** that is **60**

Then, update all the corresponding variables on all of the initializers and methods, you might need to force unwrap the values to use shorthand operations and printing. no need to add comments to remind you which property is which anymore :)

Finally, test your app to see if it still works ;)

Congratulations you have completed all challenges for LearnSwift2019 :)