Step by Step Guide for creating the Quiz App

Step 1: Start by linking the code with the design

- Create an `IBOutlet` from my Qestion label as well as my progress view. inside the `UIViewController`
  - name the Question label `questionLabel`
  - name the Preogress view `progressView`
- Do the same for the true and false buttons. Named them `trueButton` and `falseButton`.
- Add IBAction when eiher the true or false button gets pressed.
  - Add the true button and false buttons, name it `answerButtonPressed`.

```
class ViewController: UIViewController {

    @IBOutlet weak var progressView: UIProgressView!
    @IBOutlet weak var questionLabel: UILabel!
    @IBOutlet weak var trueButton: UIButton!
    @IBOutlet weak var falseButton: UIButton!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }

    @IBAction func answerButtonPressed(_ sender: UIButton) {
    }
}
```

Step 2: Creating an array `quiz`

```
let quiz = [
  "Four + Two is equal to Six",
  "Five - Three is greater that One",
  "Three + Eight is less tahn Ten"
]
```

- Track which question the user is currently reading by creaating a variable `questionNumber`

```
questionNumber = 0;
```

- Update the `questionLabel.text`

```
questionLabel.text = quiz[questionNumber]
```

- Create a new function `updateUi`

```
func updateUI () {
  questionLabel.text = quiz[questionNumber]
}
```

Step 3

Remove the `print` stetements from the function `answerButtonPressed` and add the following:

```
if userAnswer == actualAnswer {
    sender.backgroundColor = UIColor.green
} else {
    sender.backgroundColor = UIColor.red
}
```

Right now the `backgroudColor` stays on the button, to clear it follow the code:

```
func updateUI () {
    questionLabel.text = quiz[questionNumber].text
    trueButton.backgroundColor = UIColor.clear
    falseButton.backgroundColor = UIColor.clear
}
```

The clearing of the button happens, so fast that the user is unable to see the color of the button `green` or `red` .

```
Timer.scheduledTimer(timeInterval: 0.2, target: self, selector: #selector(updateUI), us
```

Step 4

Complete the `progressBar`

```
progressBar.progress = Float(questionNumber) / Float(quiz.count)
```

You need to add `+1`, so the progress bar can represent that we are currently working on qestion 1.

```
progressBar.progress = Float(questionNumber + 1) / Float(quiz.count)
```

Step 5: Start implementig the MVC Design Pattern

- Right click on the `Question.swift` and select `New Group from Selection`
- Rename the folder `Model`
- Right click on the `Main.storyboard` and select `New Group from Selection`
- Rename the folder `View`
- Right click on the `ViewController.swift` and select `New Group from Selection`
- Rename the folder `Controller`

Step 6

- Right click on the `Model` folder, and select `New File...`, name the file `QuizBrain` which it will handle the logic.
- Move the `quiz` array from the `Main` to `QuizBrain`

```
struct QuizBrain {
    let quiz = [
        Question(q: "A slug's blood is green.", a: "True"),
        Question(q: "Approximately one quarter of human bones are in the feet.", a: "Tr
        Question(q: "The total surface area of two human lungs is approximately 70 squa
        Question(q: "In West Virginia, USA, if you accidentally hit an animal with your
        Question(q: "In London, UK, if you happen to die in the House of Parliament, yo
        Question(q: "It is illegal to pee in the Ocean in Portugal.", a: "True"),
        Question(q: "You can lead a cow down stairs but not up stairs.", a: "False"),
        Question(q: "Google was originally called 'Backrub'.", a: "True"),
        Question(q: "Buzz Aldrin's mother's maiden name was 'Moon'.", a: "True"),
        Question(q: "The loudest sound produced by any animal is 188 decibels. That ani
        Question(q: "No piece of square dry paper can be folded in half more than 7 tim
        Question(q: "Chocolate affects a dog's heart and nervous system; a few ounces a
    ]
}
```

- Move the `questionNumber` from the `Main` to `QuizBrain`
- Create a new variable `quizBrain` in the `UIViewController`

```
var quizBrain = QuizBrain()
```

- Create a function `checkAnswer`, the parameter is a String in the `QuizBrain`

```
func checkAnswer(userAnswer: String) {
}
```

- Remove the and replace the code:

```
// Remove the code
let actualAnswer = quiz[questionNumber].answer


// Add this code
quizBrain.checkAnswer(userAnswer: userAnswer)
```

- Add the `!` to the end of

```
let userAnswer = sender.currentTitle!
```

Step 7

- Modify the `checkAnswer` function to allow for an output `Bool`

```
func checkAnswer(userAnswer: String) -> Bool{
  if userAnswer == quiz[questionNumber].answer {
      // User got it right
      return true
  } else {
      // User got it wrong
      return false
  }
}
```

- Create a var `userGotItRight`

```
let userGotItRight = quizBrain.checkAnswer(userAnswer: userAnswer)
```

- Remove the `userAnswer == actualAnswer` from the if statement and type `userGotItRight`

- Create a new function `getQuestionText` with an output of `String` inside the `QuizBrain`

```
func getQuestionText () -> String  {
  return quiz[questionNumber].text
}
```

Inside the `Main` edit the `updateUI` function:

```
questionLabel.text = quizBrain.getQuestionText()
```

- Create a new function `getProgress()` inside the `QuizBrain`

```
func getProgress () -> Float{
    let progress = Float(questionNumber + 1) / Float(quiz.count)
    return progress
}
```

- Update the `updateUI` function in the `Main`

```
progressView.progress = quizBrain.getProgress()
```

- Cut out the quiz progression functionality from the `main`

```
if questionNumber + 1 < quiz.count {
    questionNumber += 1
} else {
    questionNumber = 0
}
```

- Add the `quizBrain.nextQuestion()` to the `Main`

```
quizBrain.nextQuestion()
```

- Mark the function `nextQestion` with mutating keyword

```
mutating func nextQuestion() {
  if questionNumber + 1 < quiz.count {
      questionNumber += 1
  } else {
      questionNumber = 0
  }
}
```

Step 8

- Add a `label` and place it under the `Stack View`
- Modify the properties of the label
  - Change the text color to white
- Create an `IBAutlet` that links the label to the `ViewController`
- Name the label `scoreLabel`
- Go under the `/Controller/ViewController` under the `updateUI()` type the following

```
scoreLabel.text = "Score: \(quizBrain.getScore())"
```

- Create a variable `score` . Place it under the `questionNumer` in the `quizBrain`

```
var score = 0;
```

- Increase the score by 1, if the user got the answer right. Place it in the `checkAnswer` function

```
mutating func checkAnswer(userAnswer: String) -> Bool{
  if userAnswer == quiz[questionNumber].answer {
      // User got it right
      score += 1
      return true
  } else {
      // User got it wrong
      return false
  }
}
```

- Create a new function `getScore` that returns an Integer

```
func getScore() -> Int {
  return score
}
```

- Reset the `score` to 0 when the quiz restarts.

```
mutating func nextQuestion() {
  if questionNumber + 1 < quiz.count {
      questionNumber += 1
  } else {
      questionNumber = 0
      score = 0
  }
}
```

Step 9: Change button opacity

Add the code to the `keyPressed` function

```
//Code should execute after 0.2 second delay.
  DispatchQueue.main.asyncAfter(deadline: .now() + 0.2) {
    //Bring's sender's opacity back up to fully opaque.
    sender.alpha = 1.0
}
```