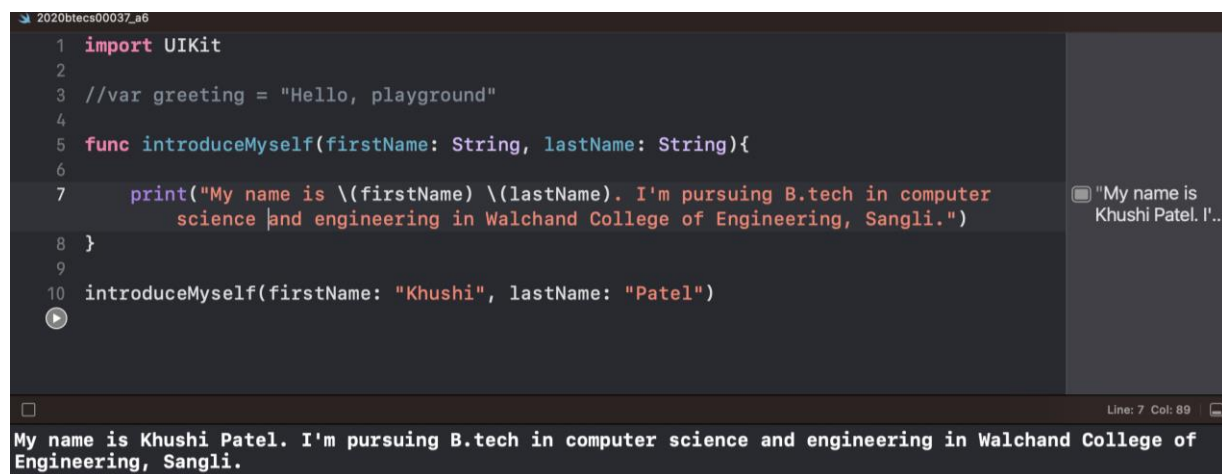


Name: Khushi Nitinkumar Patel
PRN: 2020BTECS00037

Assignment no: 6

Exercise - Create Functions

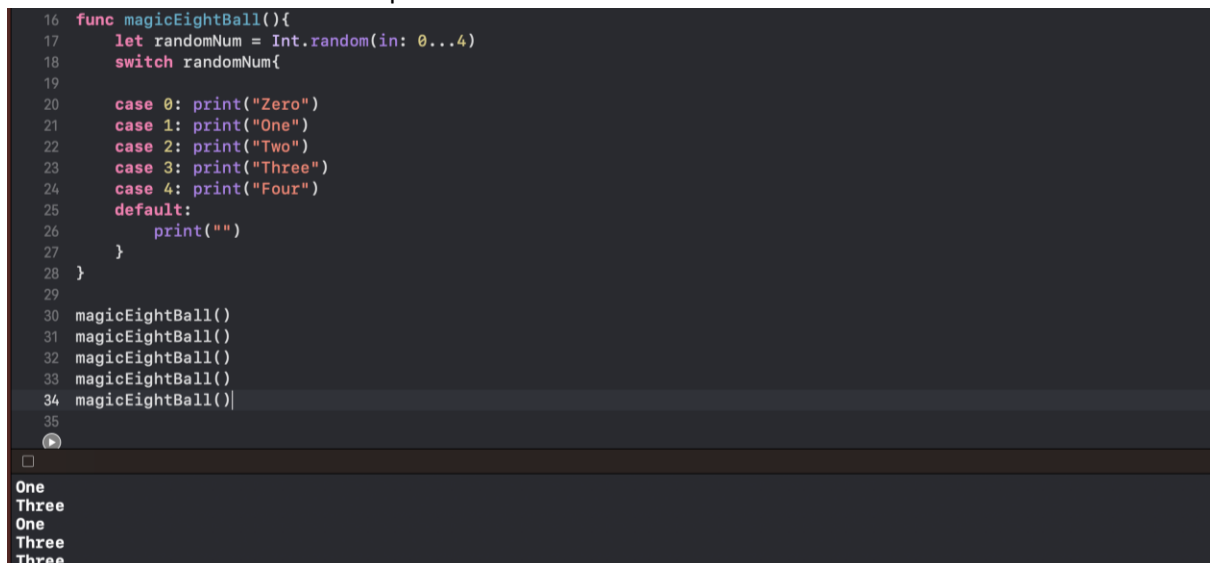
1. Write a function called `introduceMyself` that prints a brief introduction of yourself. Call the function and observe the printout.



```
1 import UIKit
2
3 //var greeting = "Hello, playground"
4
5 func introduceMyself(firstName: String, lastName: String){
6
7     print("My name is \(firstName) \(lastName). I'm pursuing B.tech in computer
8         science and engineering in Walchand College of Engineering, Sangli.")
9 }
10 introduceMyself(firstName: "Khushi", lastName: "Patel")
```

My name is Khushi Patel. I'm pursuing B.tech in computer science and engineering in Walchand College of Engineering, Sangli.

2. Write a function called `magicEightBall` that generates a random number and then uses either a switch statement or if-else-if statements to print different responses based on the random number generated. `let randomNum = Int.random(in: 0...4)` will generate a random number from 0 to 4, after which you can print different phrases corresponding to the number generated. Call the function multiple times and observe the different printouts.



```
16 func magicEightBall(){
17     let randomNum = Int.random(in: 0...4)
18     switch randomNum{
19
20         case 0: print("Zero")
21         case 1: print("One")
22         case 2: print("Two")
23         case 3: print("Three")
24         case 4: print("Four")
25     default:
26         print("")
27     }
28 }
29
30 magicEightBall()
31 magicEightBall()
32 magicEightBall()
33 magicEightBall()
34 magicEightBall()
```

One
Three
One
Three
Three

App Exercise - A Functioning App

3. As you may have guessed, functions are key to making your app work. For example, in every exercise dealing with step count until now, you have simply assigned a number of steps to a `steps` variable. This isn't very realistic seeing as the number of steps you take increments one at a time and continues changing throughout the day.

A reoccurring process like this is a perfect candidate for a function. Write a function called `incrementSteps` after the declaration of `steps` below that will increment `steps` by one and then print its value. Call the function multiple times and observe the printouts.

var steps = 0

```
36 //3
37 var steps = 0
38 func incrementSteps(){
39
40
41     steps+=1
42     print(steps)
43 }
44
45 incrementSteps()
46 incrementSteps()
47 incrementSteps()
48 incrementSteps()
```



1
2
3
4

4. Similarly, if you want to regularly provide progress updates to your user, you can put your control flow statements that check on progress into a function. Write a function called `progressUpdate` after the declaration of `goal` below. The function should print "You're off to a good start." if `steps` is less than 10% of `goal`, "You're almost halfway there!" if `steps` is less than half of `goal`, "You're over halfway there!" if `steps` is less than 90% of `goal`, "You're almost there!" if `steps` is less than `goal`, and "You beat your goal!" otherwise. Call the function and observe the printout.

```
let goal = 10000
```

```
//4
50 let goal = 10000
51 let steps = 850
52 func progressUpdate(){
53     if steps < (goal*10/100){
54         print("You're off to a good start.")
55     }
56     else if steps < goal/2{
57         print("You're almost halfway there!")
58     }
59     else if steps < (goal*90)/100{
60         print("You're over halfway there!")
61     }
62     else if steps < goal{
63         print("You're almost there!")
64     }
65     else{
66         print("You beat your goal!")
67     }
68 }
69
70 progressUpdate()
71
```

```
You're off to a good start.
```

Exercise - Parameters and Argument Labels

5. Write a new introduction function called `introduction`. It should take two `String` parameters, `name` and `home`, and one `Int` parameter, `age`. The function should print a brief introduction. I.e. if "Mary," "California," and 32 were passed into the function, it might print "Mary, 32, is from California." Call the function and observe the printout.

```
77
78 //5
79
80 func introduction(name: String, home: String, age: Int){
81     print("\(name), \(age), is from \(home).")
82 }
83
84
85 introduction(name: "Khushi", home: "Mumbai", age: 20)
```

☐

Khushi, 20, is from Mumbai.

6. Write a function called `almostAddition` that takes two `Int` arguments. The first argument should not require an argument label. The function should add the two arguments together, subtract 2, then print the result. Call the function and observe the printout.

```
88
89 func almostAddition(_firstNumber: Int, secondNumber: Int) {
90
91     print(_firstNumber + secondNumber)
92     print(_firstNumber - secondNumber)
93 }
94
95 almostAddition(_firstNumber: 5, secondNumber: 3)
```

☐

8
2

7. Write a function called `multiply` that takes two `Double` arguments. The function should multiply the two arguments and print the result. The first argument should not require a label, and the second argument should have an external label, `by`, that differs from the internal label. Call the function and observe the printout.

```
99 func multiply(firstNumber: Double, by secondNumber: Double){
100     print(firstNumber * secondNumber)
101 }
102
103
104 multiply(firstNumber: 5, by: 4)
105
```

20.0

App Exercise - Progress Updates

8. In many cases you want to provide input to a function. For example, the progress function you wrote in the Functioning App exercise might be located in an area of your project that doesn't have access to the value of `steps` and `goal`. In that case, whenever you called the function, you would need to provide it with the number of steps that have been taken and the goal for the day so it can print the correct progress statement.

Rewrite the function `progressUpdate`, only this time give it two parameters of type `Int` called `steps` and `goal`, respectively. Like before, it should print "You're off to a good start." if steps is less than 10% of goal, "You're almost halfway there!" if steps is less than half of goal, "You're over halfway there!" if steps is less than 90% of goal, "You're almost there!" if steps is less than goal, and "You beat your goal!" otherwise. Call the function and observe the printout.

Call the function a number of times, passing in different values of `steps` and `goal`. Observe the printouts and make sure what is printed to the console is what you would expect for the parameters passed in.

```
108 func progressUpdate(steps: Int, goal: Int){
109     var percent = (Double(steps) / Double(goal)) * 100
110     if percent <= 10{
111         print("You're off to a good start.")
112     }
113     else if percent <= 50{
114         print("You're almost half way there")
115     }
116     else if percent <= 90{
117         print("You're over half way there")
118     }
119     else if percent < 100{
120         print("You're almost there!")
121     }
122     else{
123         print("You beat your goal!")
124     }
125 }
126
127 progressUpdate(steps: 6000, goal: 10000)
128
```

You're over half way there

9. Your fitness tracking app is going to help runners stay on pace to reach their goals. Write a function called `pacing` that takes four `Double` parameters called `currentDistance`, `totalDistance`, `currentTime`, and `goalTime`. Your function should calculate whether or not the user is on pace to hit or beat `goalTime`. If yes, print "Keep it up!", otherwise print "You've got to push it just a bit harder!"

```
136
137 //9
138
139 func pacing(currentDistance: Double, totalDistance: Double, currentTime: Double, goalTime: Double){
140
141     var speed = totalDistance / goalTime
142     var currentSpeed = currentDistance / currentTime
143     if currentSpeed >= speed{
144
145         print("Keep it up!")
146     }
147     else{
148
149         print("You've got to push it just a bit harder!")
150     }
151
152 }
153 }
```

Exercise - Return Values

10. Write a function called `greeting` that takes a `String` argument called `name`, and returns a `String` that greets the name that was passed into the function. I.e. if you pass in "Dan" the return value might be "Hi, Dan! How are you?" Use the function and print the result.

```
124
125 //10
126
127 func greeting(name: String) {
128
129     print("Hi \(name)! How are you?")
130
131 }
132
133 greeting(name: "Khushi")
```

Line: 129 Col: 38

Hi Khushi! How are you?

11. Write a function that takes two `Int` arguments, and returns an `Int`. The function should multiply the two arguments, add 2, then return the result. Use the function and print the result.

```
164
165 //11
166
167 func numbers(num1: Int, num2: Int)->Int{
168
169     return num1*num2+2
170
171 }
172
173
174 print(numbers(num1: 4, num2: 6))
```

26

App Exercise - Separating Functions

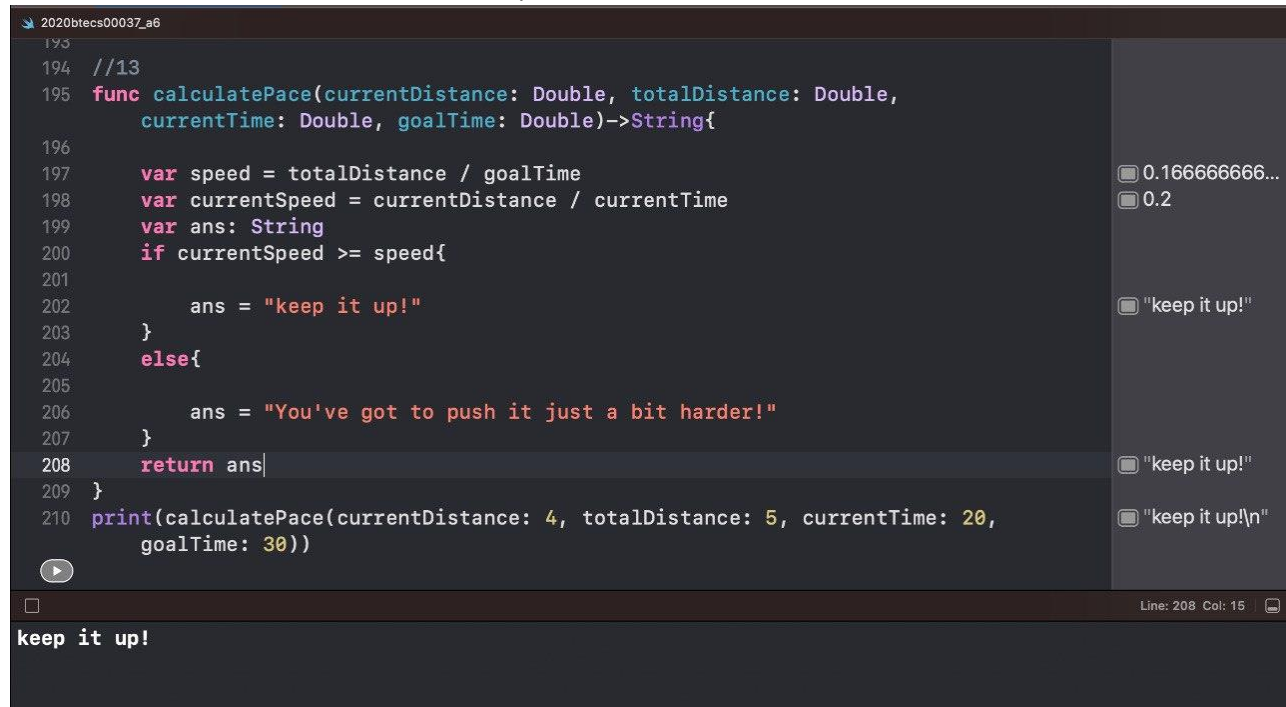
12. One principle that can help in debugging and maintaining code is abstraction. For example, in your fitness tracking app some of your existing functions have been written to both perform a calculation and print a message. But it's very possible that you'll decide to change either the calculation or the message in the future. It will be easier to go back and change this if you separate the calculation from the message.

As an example, write a function that only does a portion of what your previous `pace` function did. This function will be called `calculatePace`. It should take three `Double` arguments called `currentDistance`, `totalDistance`, and `currentTime`, and should return a `Double` that will represent the time at which the user will finish the run based on the user's current distance and time. call the function and print the return value.

```
175
176 //12
177
178 func calculatePace(currentDistance: Double, totalDistance: Double, currentTime:
    Double, goalTime: Double){
179
180     var speed = totalDistance / goalTime
181     var currentSpeed = currentDistance / currentTime
182
183     if currentSpeed >= speed{
184
185         print("keep it up!")
186     }
187     else{
188
189         print("You've got to push it just a bit harder!")
190     }
191 }
192 calculatePace(currentDistance: 4, totalDistance: 5, currentTime: 20, goalTime: 30)
```

☐ keep it up!

13. Now write a function called `pacing` that takes four `Double` arguments called `currentDistance`, `totalDistance`, `currentTime`, and `goalTime`. The function should also return a `String`, which will be the message to show the user. The function should call `calculatePace`, passing in the appropriate values, and capture the return value. The function should then compare the returned value to `goalTime` and if the user is on pace return "Keep it up!", and return "You've got to push it just a bit harder!" otherwise. Call the function and print the return value.



```
193
194 //13
195 func calculatePace(currentDistance: Double, totalDistance: Double,
196                   currentTime: Double, goalTime: Double)->String{
197     var speed = totalDistance / goalTime
198     var currentSpeed = currentDistance / currentTime
199     var ans: String
200     if currentSpeed >= speed{
201
202         ans = "keep it up!"
203     }
204     else{
205
206         ans = "You've got to push it just a bit harder!"
207     }
208     return ans
209 }
210 print(calculatePace(currentDistance: 4, totalDistance: 5, currentTime: 20,
211                   goalTime: 30))
```

0.166666666...
0.2
"keep it up!"
"keep it up!"
"keep it up!\n"

Line: 208 Col: 15

keep it up!
