



Lesson 1

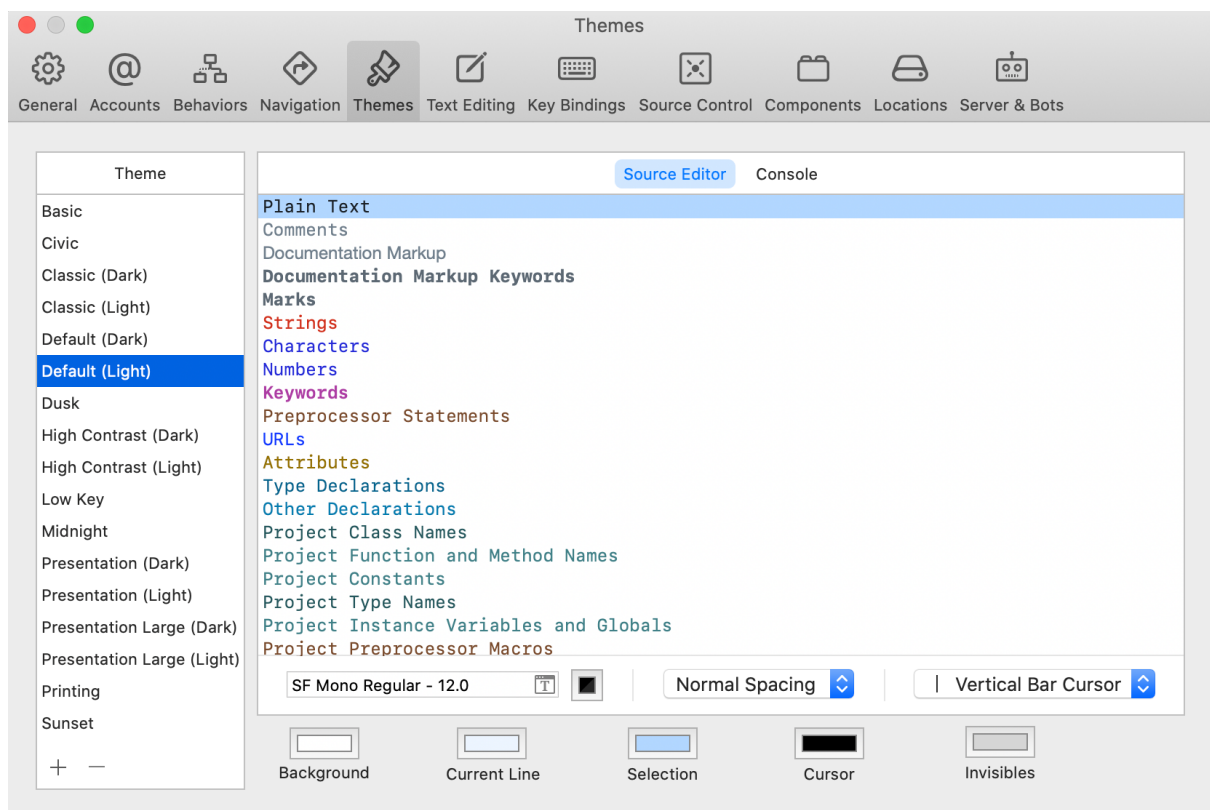
PDF Slides for Lesson 1:

 Lesson1_slides.pdf <https://drive.google.com/file/d/16cnGPSKcZqfVXnCVKm1r1E0zEmvkPXDk/view?usp=drivesdk>

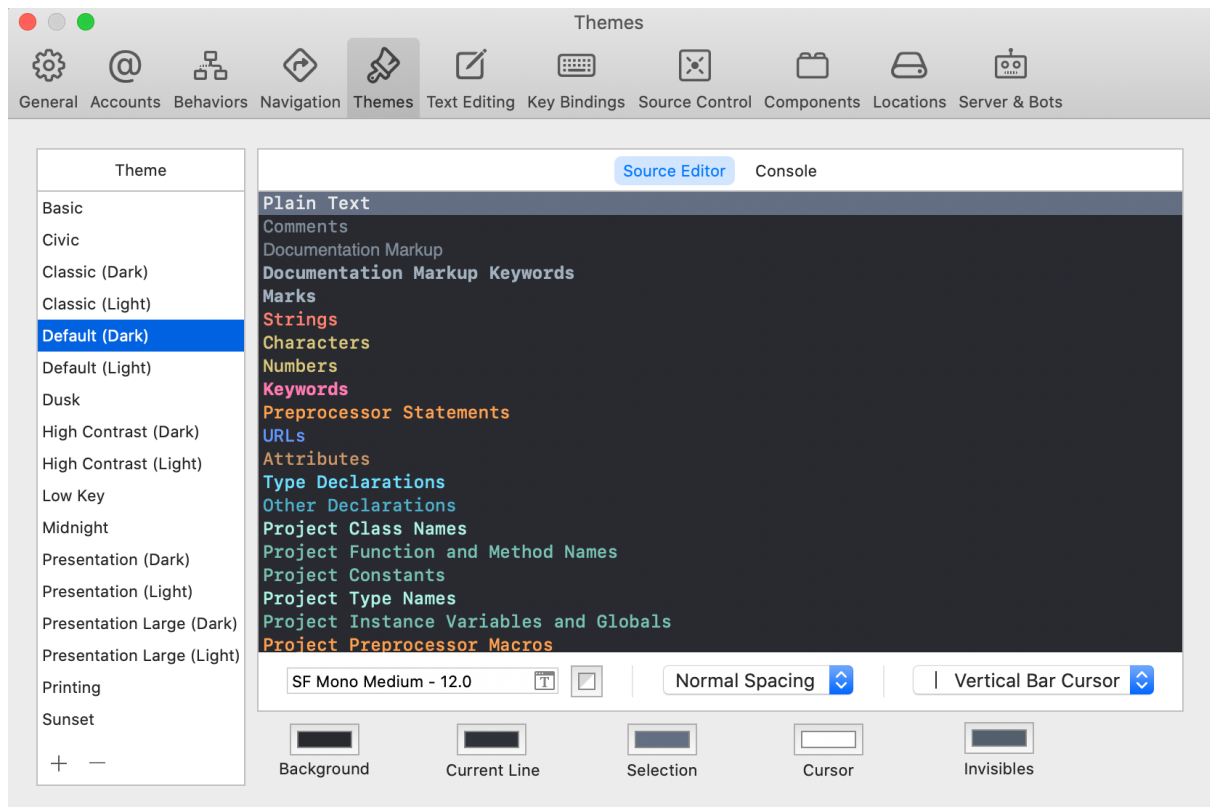
Change Xcode color mode

In Xcode you have the ability to change your display settings which will adjust the colours and fonts used in the application. One way to do this is by changing your **colour themes**. There are a wide range of themes available but most people tend to use either the default Light or Dark themes.

This is what Xcode looks like in the default **Light** theme.



This is what Xcode looks like in the default **Dark** theme.



It is completely up to you as to which theme you use as this is a purely personal choice, however some developers prefer to use Dark theme as they find it easier to read Swift code on a darker screen.

Please note that throughout this course we'll be using the default Dark theme in our screen captures and videos however the application works exactly the same in either mode, the colours of windows and elements are just slightly different.

Instructions on how to change to a dark theme are provided below for reference.

How to change your colour themes

If you want to change your colour theme, simply follow these steps.

1. Launch Xcode. Go to **Xcode > Preferences > Fonts & Colors (tab)**
2. You will see there are a wide range of themes to choose from. Click on each one to see the colours and fonts for that theme.

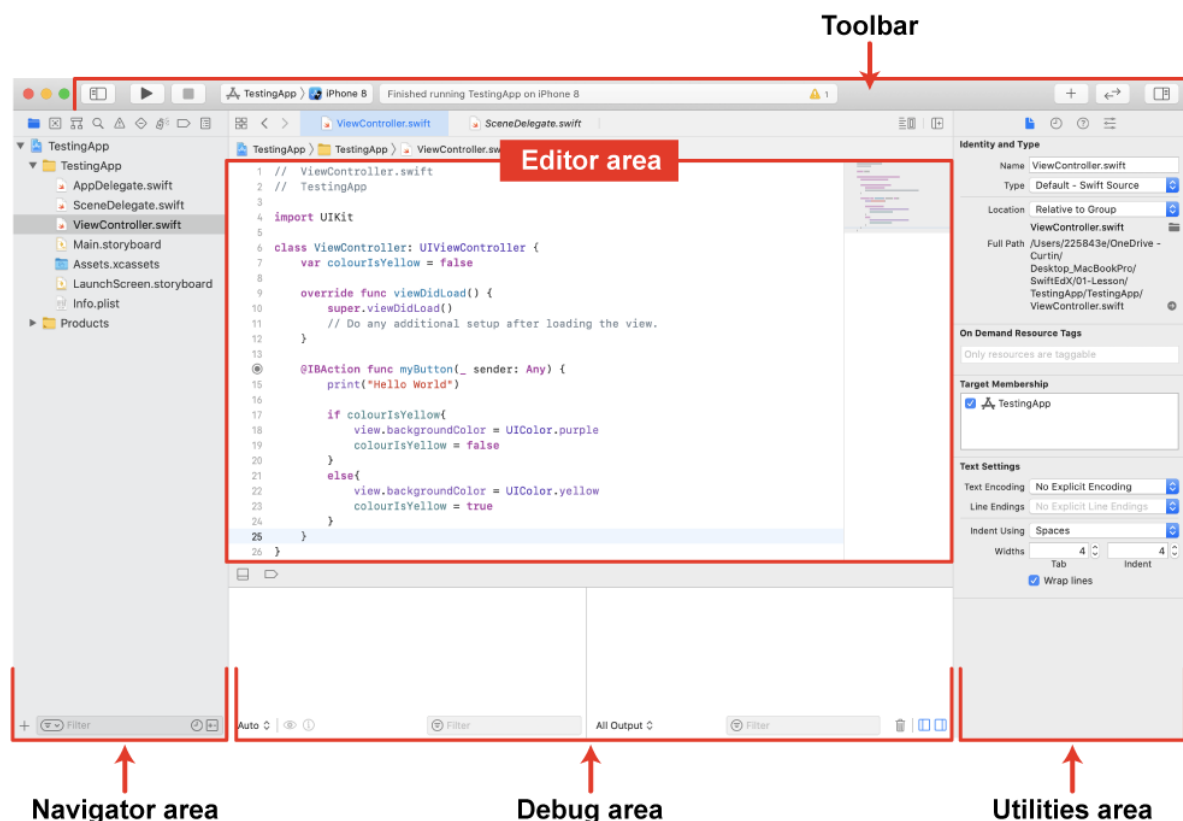
3. Scroll down and select **Default (Dark)** or choose whichever theme you like the best.
4. Close the **Preferences** window once you have selected your preference.

Next we're going to examine the Xcode interface and explore the main elements.

How Xcode interface works

The Xcode interface is quite complex and is made up of a single window which integrates code editing, user interface design, files and library objects, testing, an iOS simulator and debugging tools. If you select a file in one section, an appropriate editor will open in another area. If you select a symbol or object in another section - its documentation or related properties will then appear in another window.

After you create a new Xcode project, you will see the following main window. Let's examine the interface in detail.



1. **Toolbar** – the Toolbar area is used to build and run your app, view the progress of any running tasks and can be used to show and hide different

areas of the main window.

2. **Navigator** – the Navigator area lets you quickly access different parts of your project, including files.
3. **Editor** – If you select a file in the Navigator area, it will open in the Editor area. All the coding happens in the Editor window.
4. **Utilities** The Utilities area in Xcode displays the Inspector pane and the Library pane. It is used to view and edit the properties of an object which you've selected in the Editor areas.
5. **Debug** – The debug area opens automatically when you build and run an app. You can use it to control the execution of your code, inspect variables, view console output and interact with the debugger.

Each area provides you with access to a range of different options or settings and we'll explain these as we go through them.

Project Vs Playground

Projects vs Playgrounds - What is the difference?

During this course you will be asked to work with both Projects and Playgrounds, so it's important that we know the difference. In summary, an Xcode Project allows you to create your finished apps ready for the App Store, whereas the Xcode Playground is more of a testing ground.

Projects allow you to store assets for your app development, along with describing the relationships between these assets. These could include things such as source code, libraries, images and frameworks.

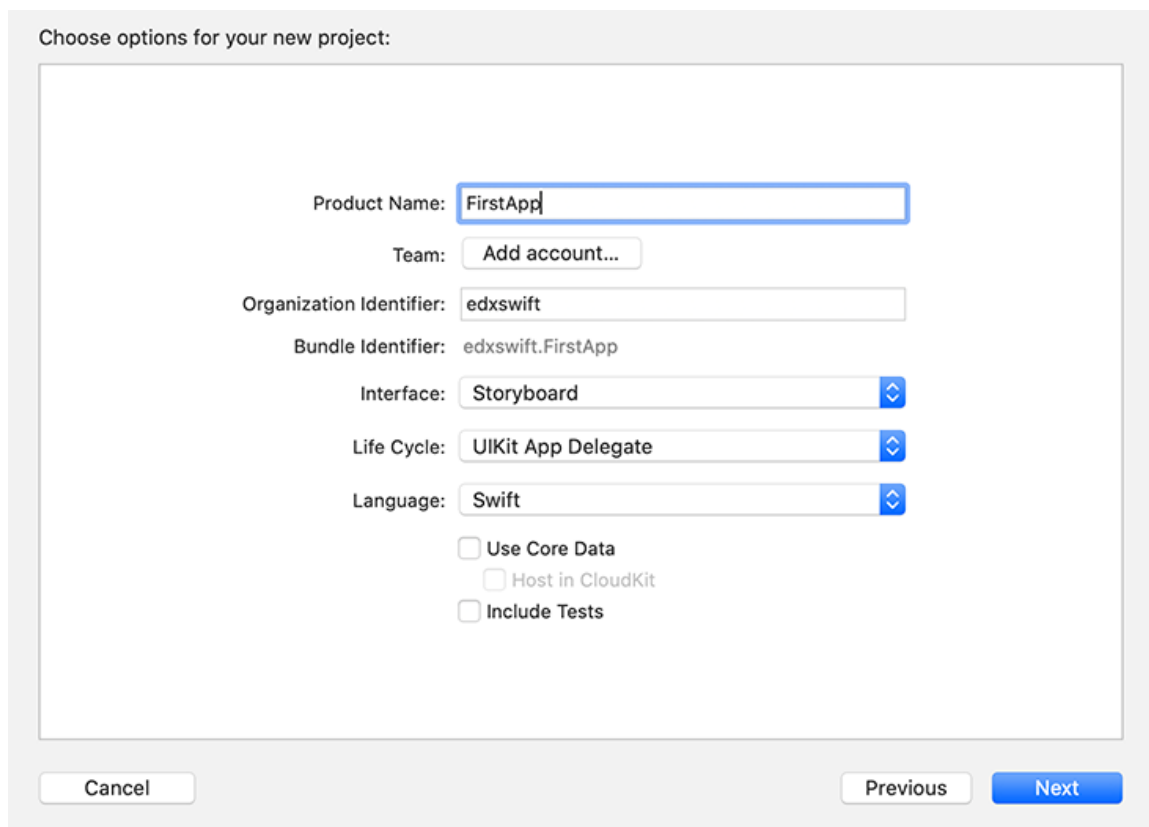
Playgrounds are an interactive environment within Xcode that allows you to develop and test your Swift code in real-time. By clicking a 'Play' button at any stage of your development, you're able to check how your code is working. However, fully-fledged apps can't be made in Playgrounds.

You will learn more about each of these environments in more detail as the course progresses.

Creating the first IOS App

The step by step instructions for this exercise are provided below. You may like to print these out for easy reference.

1. You can launch Xcode in one of the following ways:
 - Use your **Spotlight Search** to search for 'Xcode' or
 - Go to **Applications > Xcode**.
2. Select **Create a new Xcode Project** (from the startup window) or go to **File > New > Project**.
3. Choose a template for your new project. Select **iOS > Single View App** and click **Next**.
4. Choose the following options for your new project:



Choose options for your new project:

Product Name: FirstApp

Team: Add account...

Organization Identifier: edxswift

Bundle Identifier: edxswift.FirstApp

Interface: Storyboard

Life Cycle: UIKit App Delegate

Language: Swift

☐ Use Core Data

☐ Host in CloudKit

☐ Include Tests

Cancel Previous Next

5. Now you'll be prompted to save your project to your Mac. Save it to your desktop and click **Create**.
6. The Xcode main window will then appear. It consists of 3 main sections: the Navigator area (on left hand side), the Toolbar area (at the top) and the Utility/Inspector area (on the right hand side).

7. In the Navigator area, double click on the **ViewController.swift** file to open it. The file will open up into the middle of the screen - into the Editor area.
8. In the Navigator area, double click on the **Main.storyboard** file to open it. A new window will appear which looks similar to a mobile device interface.
9. In the Utility area, click on the **Show the Assistant editor** icon. This will display the Assistant editor next to your main storyboard file. This is where you will see (and edit) your Swift code.



10. Resize your windows if need be so you can easily see **both** the Assistant editor and the main storyboard side by side.
11. Let's add a button to our main storyboard (interface). Click on the Library icon and a library of objects will appear.



12. Scroll down the objects in the Library until you see the **Button** object. Click and drag the button onto the middle of the storyboard interface.
13. Select the button on the storyboard and in the Utility area click on **Show the Attribute Inspector**.



14. In the attributes that appear, change the Text Color to **Black Color**.
15. Scroll down and select Background to **Custom ...** Select a custom background colour from the colour palette.
16. Ensure your button is still selected, hold down the Control key and drag your mouse across into the **ViewController.swift** file in the Editor area. A line will appear. Drag this line in between the two curly brackets at the bottom of the screen and release it.
17. A small window will pop up. Keep the default settings but enter a name for your button. Call it **myButton** and click **Connect**.

You'll see the following Swift code appear in the Editor window **between** the two curly brackets.

```
@IBAction func myButton(_ sender: Any) {  
  
}
```

18. Now it's time to program the button. Enter the following code between the curly brackets **after** the myButton code and between the curly brackets.

```
@IBAction func myButton(_ sender: Any) {  
  
    if colorIsGreen{  
        view.backgroundColor = UIColor.red  
        colorIsGreen = false  
    }  
  
    else {  
        view.backgroundColor = UIColor.green  
        colorIsGreen = true  
    }  
  
}
```

19. The code within your ViewController.swift editor should look like this at the end.

```
//  
// ViewController.swift  
// FirstApp  
//  
// Created by SWIFT MOOC on 6/2/19.  
// Copyright © 2019 DM. All rights reserved.  
//  
  
import UIKit  
  
class ViewController: UIViewController {  
  
    var colorIsGreen = false  
  
    override func viewDidLoad() {  
        super.viewDidLoad()  
        // Do any additional setup after loading the view, typically from a nib.  
    }  
}
```

```

@IBAction func myButton(_ sender: Any) {

    if colorIsGreen{
        view.backgroundColor = UIColor.red
        colorIsGreen = false
    }

    else {
        view.backgroundColor = UIColor.green
        colorIsGreen = true
    }

}

}

```

20. From the Toolbar area, select the type of device you want to build your application for. Select **iPhoneXR** from the list.
21. On the toolbar click the **Run** (Play) button to run your application on the iOS Simulator. The iOS Simulator should launch and your app will display within it. Click on your button to see it in action.



Terminal

1. You can launch Terminal in one of the following ways:
 - Use your **Spotlight Search** to search for 'Terminal' or
 - Go to **Applications > Utilities > Terminal**.
2. Type **swift** and press **Return**. This will let us start programming in Swift.
3. Type **print ("Hello World")** and press **Return**. The text "Hello World" will appear.
4. To exit Swift and quit the Terminal follow these steps:
 - Type **:exit** and press **Return**
 - Go to **Terminal > Quit Terminal**

Xcode Playground

1. Launch Xcode in one of the following ways:
 - Use your **Spotlight Search** to search for 'Xcode' or
 - Go to **Applications > Xcode**.
2. Select **Get started with a playground** (from the startup window) or go to **File > New > Playground**.
3. Choose a template for your new playground. Select **iOS > Blank** and click **Next**.
4. Now you'll be prompted to save your Playground to your mac. Save it to your desktop (or wherever you like), call it '**FirstPG**' and click **Create**.
5. The playground window will now be visible. Select everything in the editor window **below** the first line of code 'Import UIKit' and **delete it**.
6. Now we have an empty playground and we'll write our very first Swift program using Playgrounds. In the editor window type:

```
print("Hello World")
```

and hit **Return**.

7. Your playground window should now look like this.

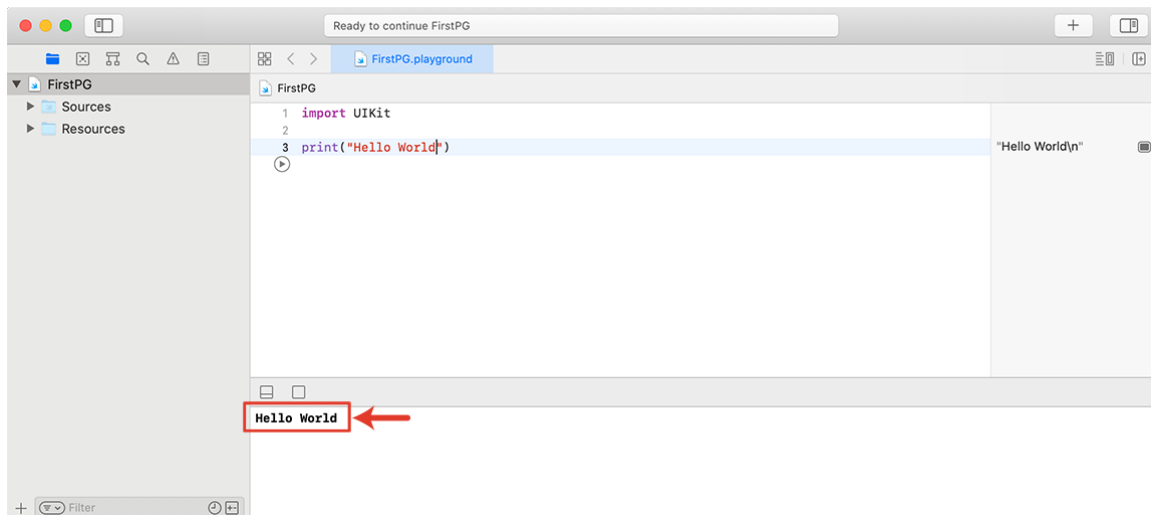
```
import UIKit

print("Hello World")
```

8. Click the blue **Play** button below your line of code to run the program.



The text 'Hello World' should appear in the results area down the bottom of your screen.



Constant and Variable

- To Create a variable use `var`
 - To Create a constant use `let`
1. Launch Xcode and open up your Playground file named '**FirstPG**'.
 2. First we're going to create a variable called greeting. In the editor window add the following four lines of code:

```
var greeting = "Hello Swift Developers"
print(greeting)

greeting = "Glad you are here"
print(greeting)
```

3. Your playground editor window should **now** look like this.

```
import UIKit

print("Hello World")

var greeting = "Hello Swift Developers"
print(greeting)

greeting = "Glad you are here"
print(greeting)
```

4. Click the blue **Play** button below your line of code to run the program.



The text 'Hello World', "Hello Swift Developers" and "Glad you are here" should appear in the results area down the bottom of your screen.

5. Now we're going to create a constant. In the editor window add the following lines of code:

```
let mothersName = "Judith"
print(mothersName)
```

6. Click the blue **Play** button below your line of code to run the program.



The text "Judith", should now appear in the results area down the bottom of your screen.

Data Types

Types

| Name | Symbol | Purpose | Example |
|----------------|--------|-------------------|---------|
| <u>Integer</u> | Int | For whole numbers | 4 |
| <u>Double</u> | Double | Decimal numbers | 4.5 |
| <u>Boolean</u> | Bool | True or False | true |
| <u>String</u> | String | Text | "Hello" |

- Syntax: `var (keyword) <variable_name>: <data type> = <variable value>`

1. Launch Xcode by going to **Applications > Xcode** .
2. Select **File > New > Playground > iOS > Blank** and save your Playground with the name '**DataTypesPG**'.
3. Delete all the code **below** the first line 'import UIKit'.
4. We're going to create a range of variables now and assign their data types and value.

In the editor window add the following lines of code so that your playground editor window looks like this.

```
import UIKit

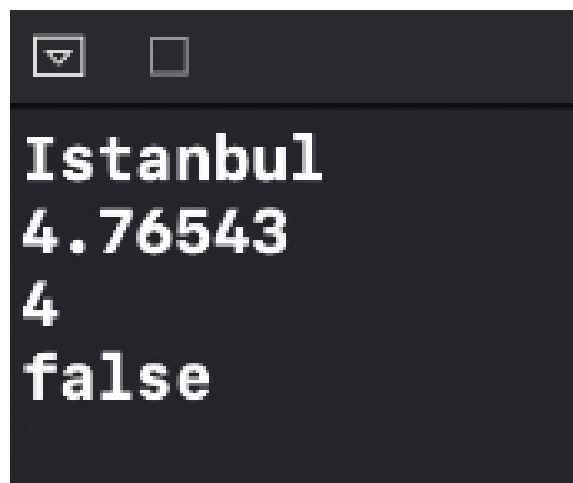
var cityName: String = "Istanbul"
var fastestSpeed: Double = 4.76543
var numberOfChildren: Int = 4
var personIsHome: Bool = false

print(cityName)
print(fastestSpeed)
print(numberOfChildren)
print(personIsHome)
```

5. Click the blue Play button below the last line of code in the editor to run the program.









Your variables will appear in the results section at the bottom of the screen.



Operator

Types of Operator

|  Operator |  Symbol |  Purpose |  Example |
|--|--|---|---|
|--|--|---|---|

|  Operator |  Symbol |  Purpose |  Example |
|--|--|---|---|
| <u>Assignment</u> | = | Assign a value to a variable | <code>Greeting = "hello"</code> |
| <u>Addition</u> | + | Add the values together | <code>25 + 30 = 55</code> |
| <u>Subtraction</u> | - | Subtract the values | <code>22 - 13 = 9</code> |
| <u>Multiplication</u> | * | Multiply the values together | <code>4 * 7 = 28</code> |
| <u>Division</u> | / | Divide the values | <code>12 / 4 = 3</code> |
| <u>Remainder</u> | % | Returns the remainder of dividing the number | <code>12 % 5 = 2</code> |

1. Launch Xcode by going to **Applications > Xcode**.
2. Select **File > New > Playground > iOS > Blank** and save your Playground with the name '**OperatorsPG**'.
3. Delete all the code **below** the first line 'import UIKit'.
4. We're going to create a range of variables, assign values to them and then apply some operators to these variables.

In the editor window add the following lines of code so that your playground editor window looks like this.

```
import UIKit

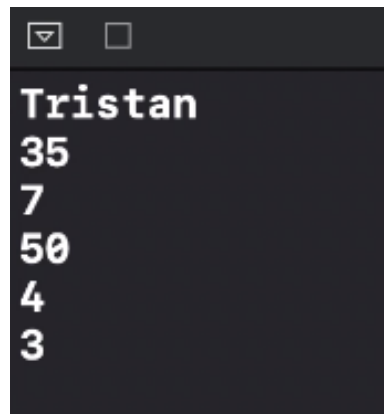
var name = "Tristan"
var numSum = 22+13
var numDiff = 10-3
var numProduct = 5*10
var numDiv = 16/4
var numRem = 18%5

print(name)
print(numSum)
print(numDiff)
print(numProduct)
print(numDiv)
print(numRem)
```

5. Click the blue Play button below the last line of code in the editor to run the program.



Your variables will appear in the results section at the bottom of the screen and should display the following.



```
Tristan
35
7
50
4
3
```

Extra Resources


- In Apple Books:



Source Code:

Ace5584/IOS-Dev-Notes

Contribute to Ace5584/IOS-Dev-Notes development by creating an account on GitHub.

 <https://github.com/Ace5584/IOS-Dev-Notes/tree/main/Course%201/L1>

Ace5584/**IOS-Dev-Notes**



 1  0  0  0

Contributors Issues Stars Forks

