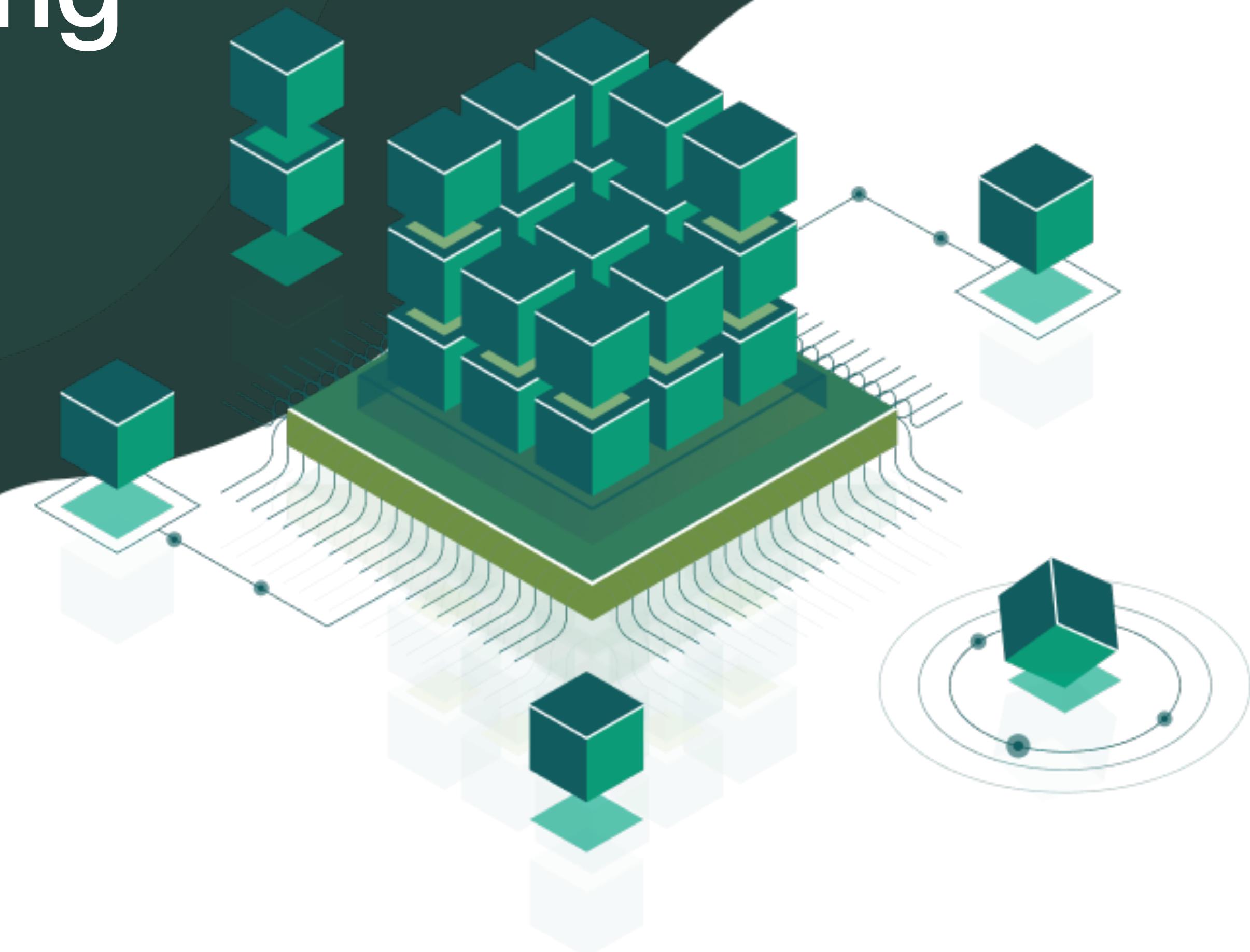


# Debugging, Profiling



# Debugging

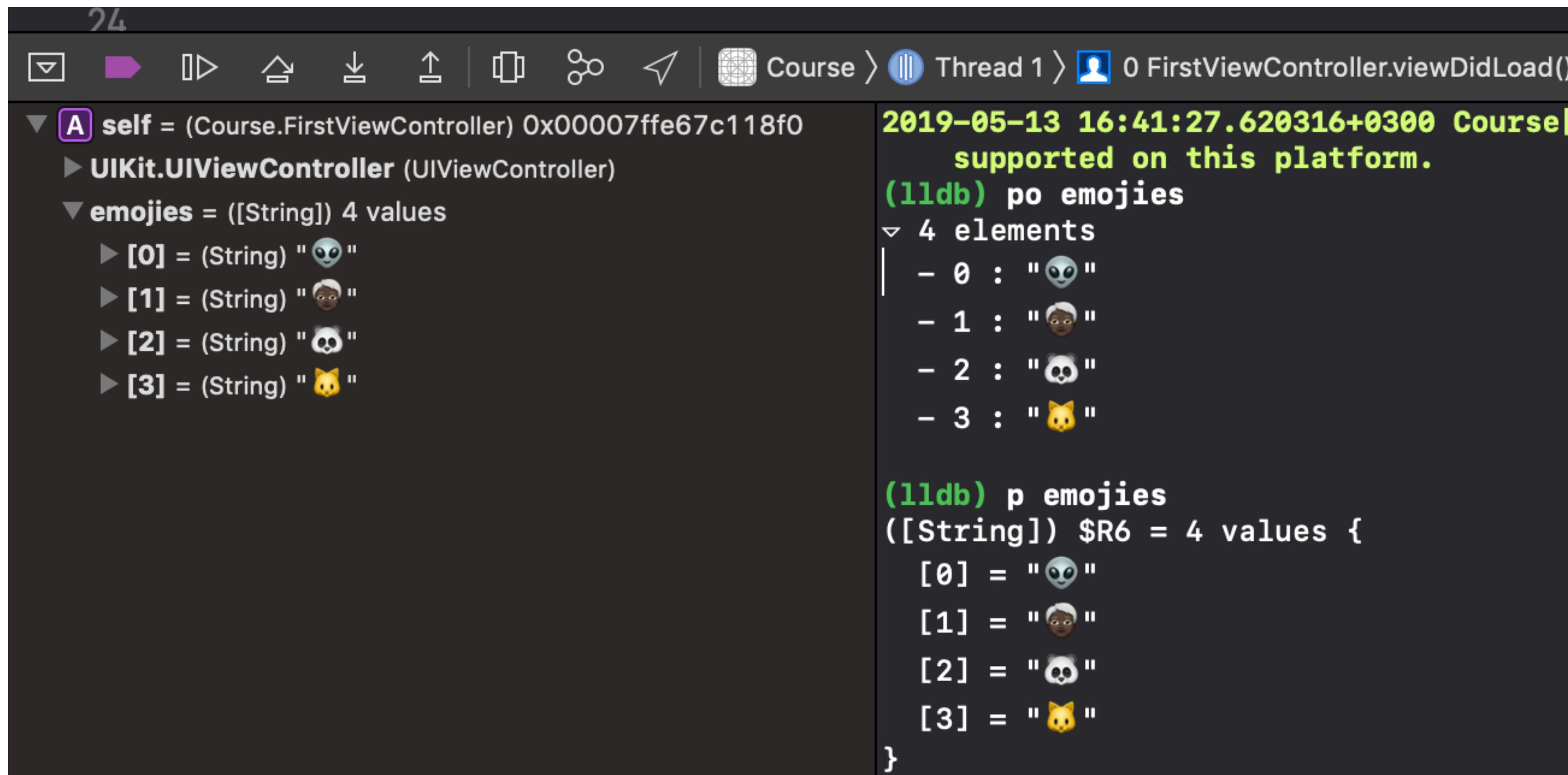
The screenshot shows a Xcode interface with the following details:

- Top Bar:** Shows "Course" and "iPhone 6".
- Project Navigator:** Shows "Course PID 42798" with sections for CPU (27%), Memory (48.4 MB), Disk (Zero KB/s), and Network (Zero KB/s).
- Code Editor:** Displays Swift code for "FirstViewController". A purple arrow points to line 18, which contains three consecutive calls to `emojis.removeLast()`. A green bar highlights this line, labeled "Thread 1: breakpoint 1.1".
- Call Stack:** Shows the call stack from "FirstViewController.viewDidLoad()" up to "UIApplicationMain".
- Breakpoint:** A purple arrow points to the line `emojis.removeLast()` at index [1].
- Variables:** Shows the variable `emojis` with four values: "\ud83d\udcbb", "\ud83d\udcbe", "\ud83d\udcbb", and "\ud83d\udcbb".
- Output:** Shows a log message: "2019-05-13 14:43:49.462545+0300 Course[42798:447448] libMobileGestalt MobileGestalt.c:890: MGIsDeviceOneOfType is not supported on this platform."
- Bottom Bar:** Includes "Filter" and "Auto" buttons.

# Console debugging

There are two common commands to show object's description in console:

- `p`
- `po`



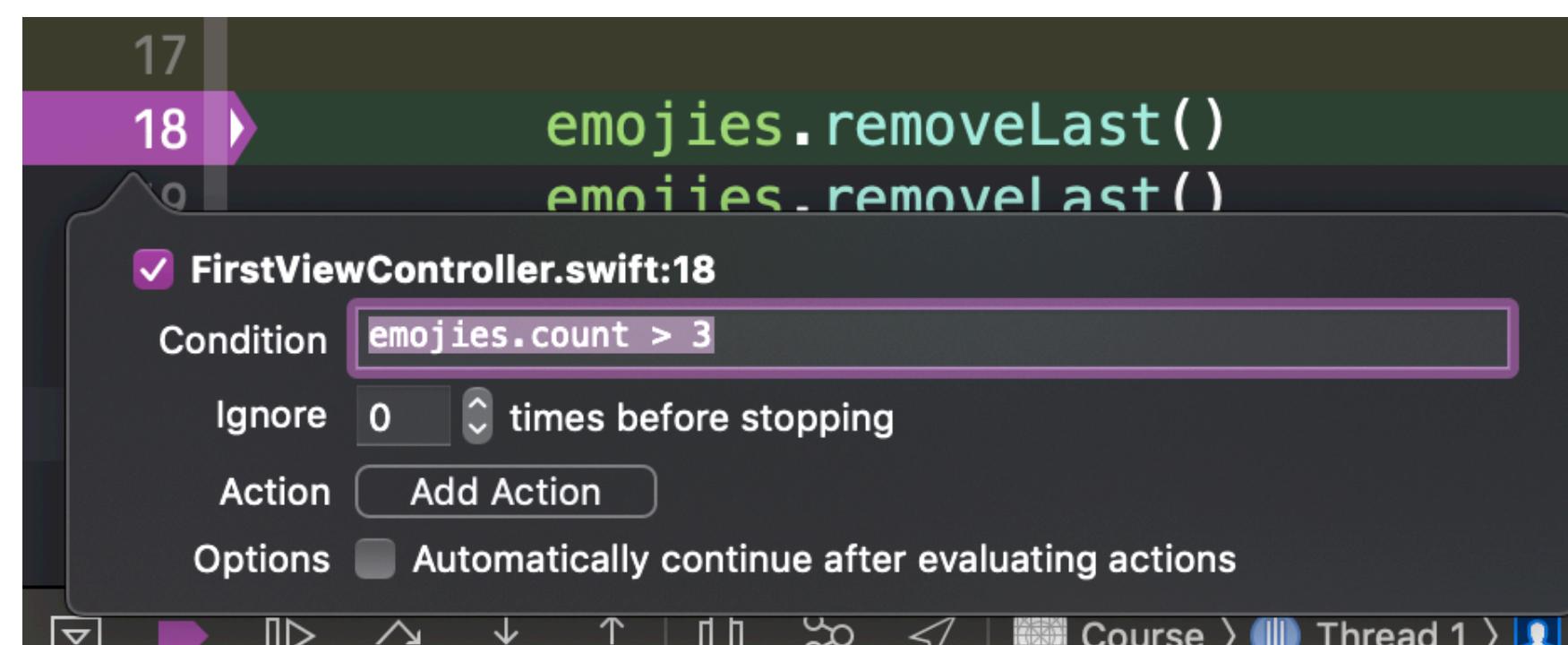
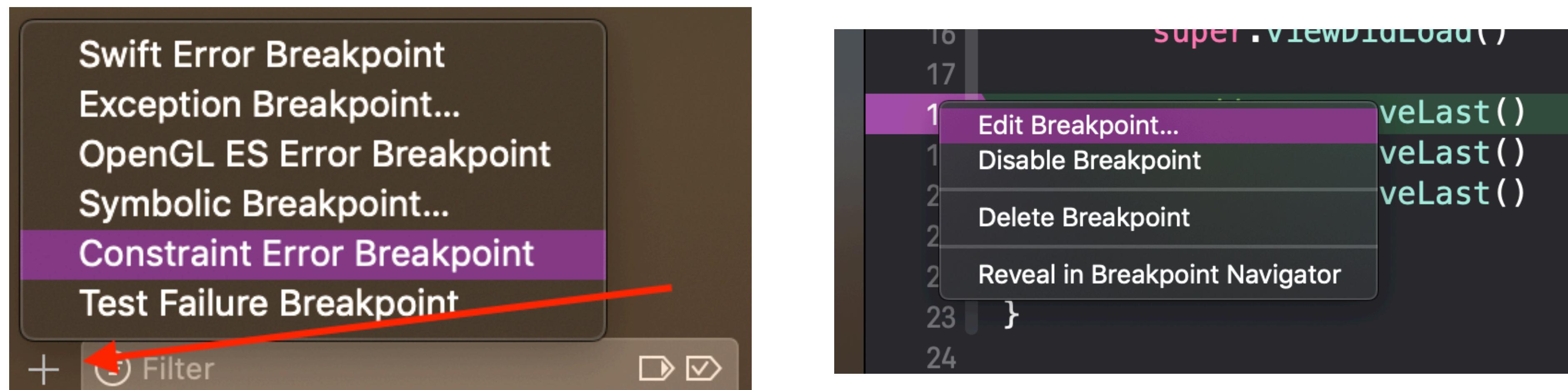
The screenshot shows the Xcode debugger interface with the following details:

- Top Bar:** Shows the date and time (2019-05-13 16:41:27.620316+0300), the target (Course), the thread (Thread 1), and the current function (0 FirstViewController.viewDidLoad()).
- Memory Dump:** A tree view showing the variable `self` of type `(Course.FirstViewController)` at address `0x00007ffe67c118f0`. It also lists `UIKit.UIViewController` and the array `emojies`.
- Output Area:** Displays the results of the `po` command:

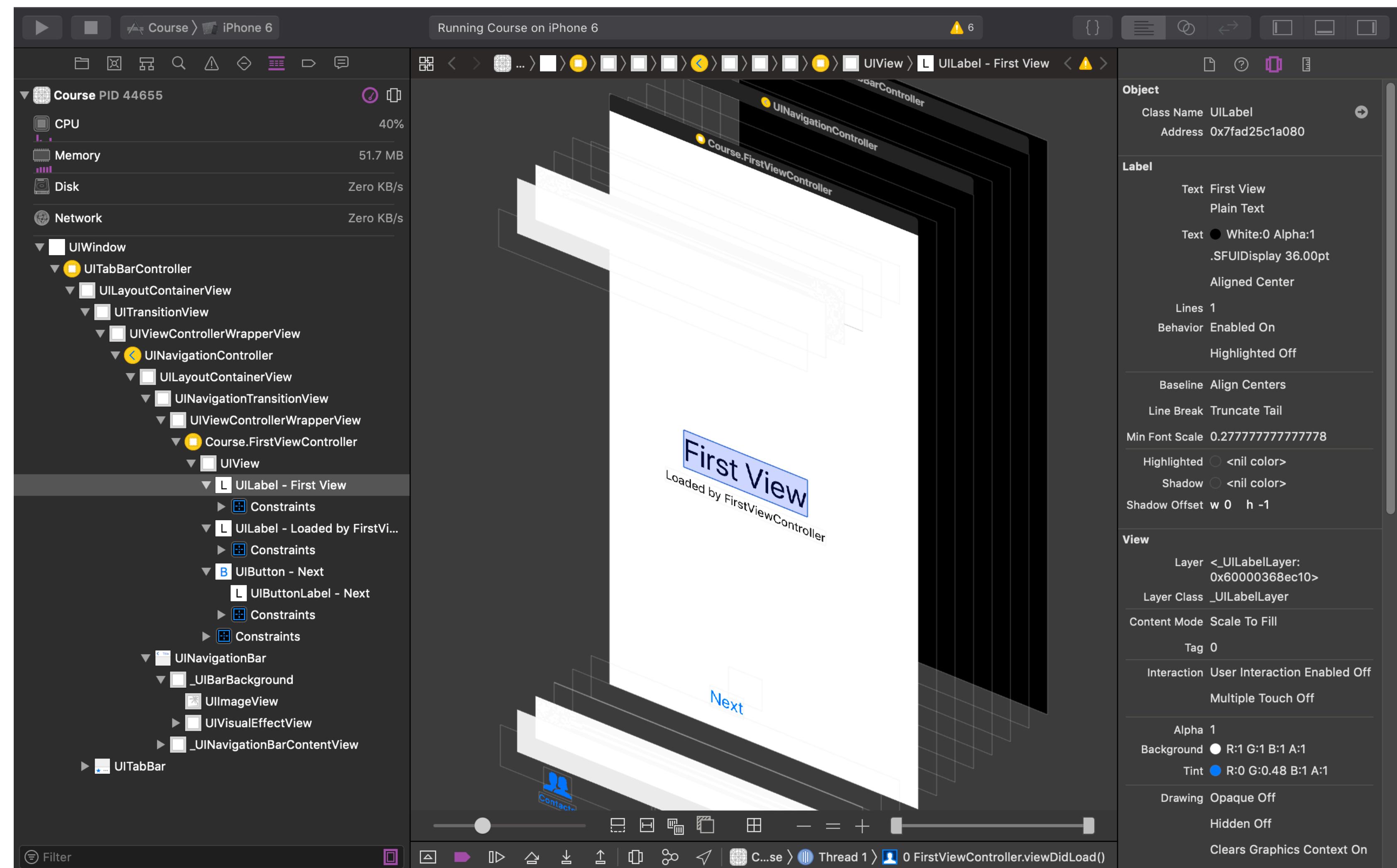
```
2019-05-13 16:41:27.620316+0300 Course[4]
    supported on this platform.
(lldb) po emojies
↳ 4 elements
|- 0 : "👽"
|- 1 : "👳"
|- 2 : "🐼"
|- 3 : "🐱"
```
- Bottom Area:** Displays the results of the `p` command:

```
(lldb) p emojies
([String]) $R6 = 4 values {
    [0] = "👽"
    [1] = "👳"
    [2] = "🐼"
    [3] = "🐱"
}
```

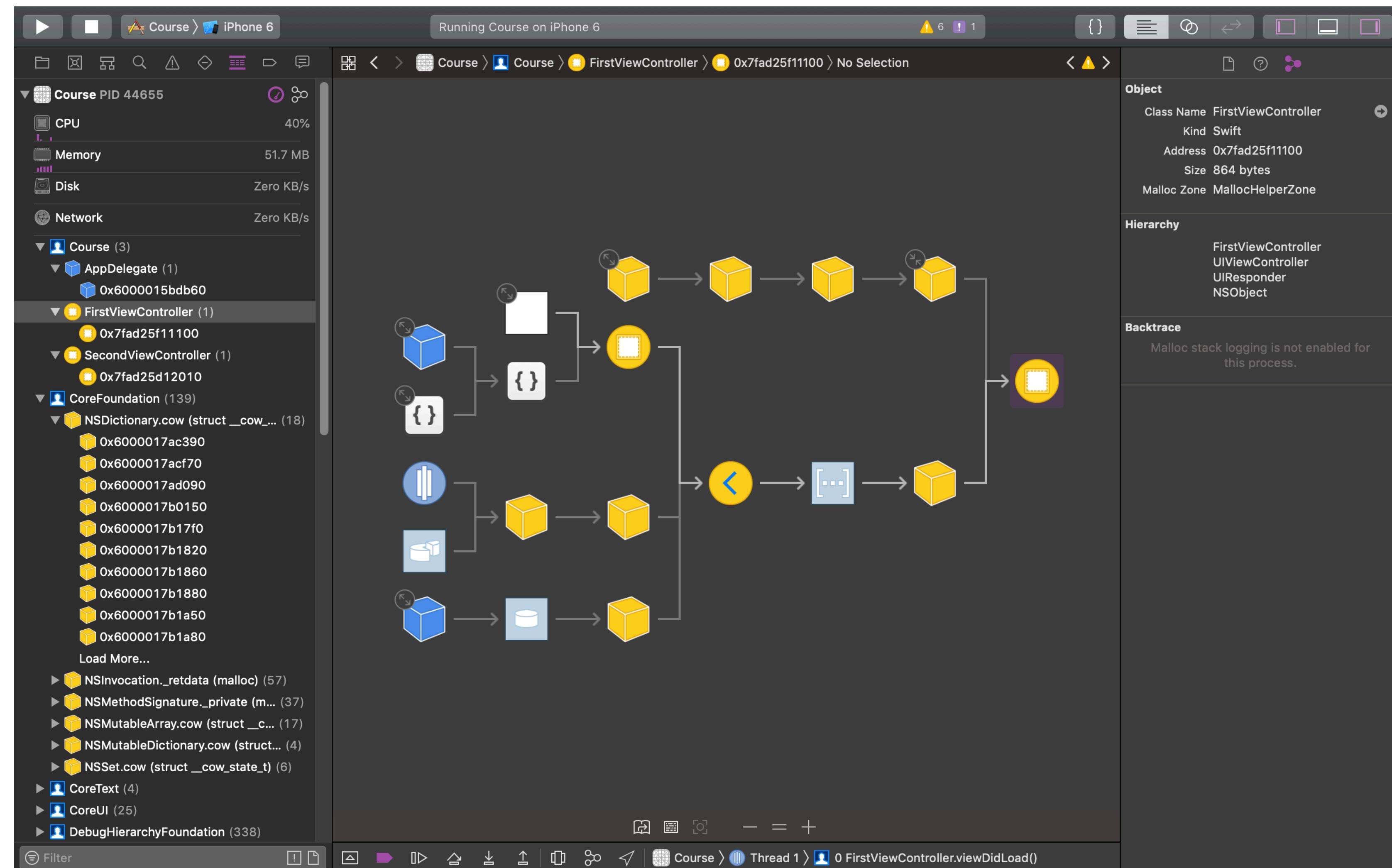
# Edit & Custom breakpoints



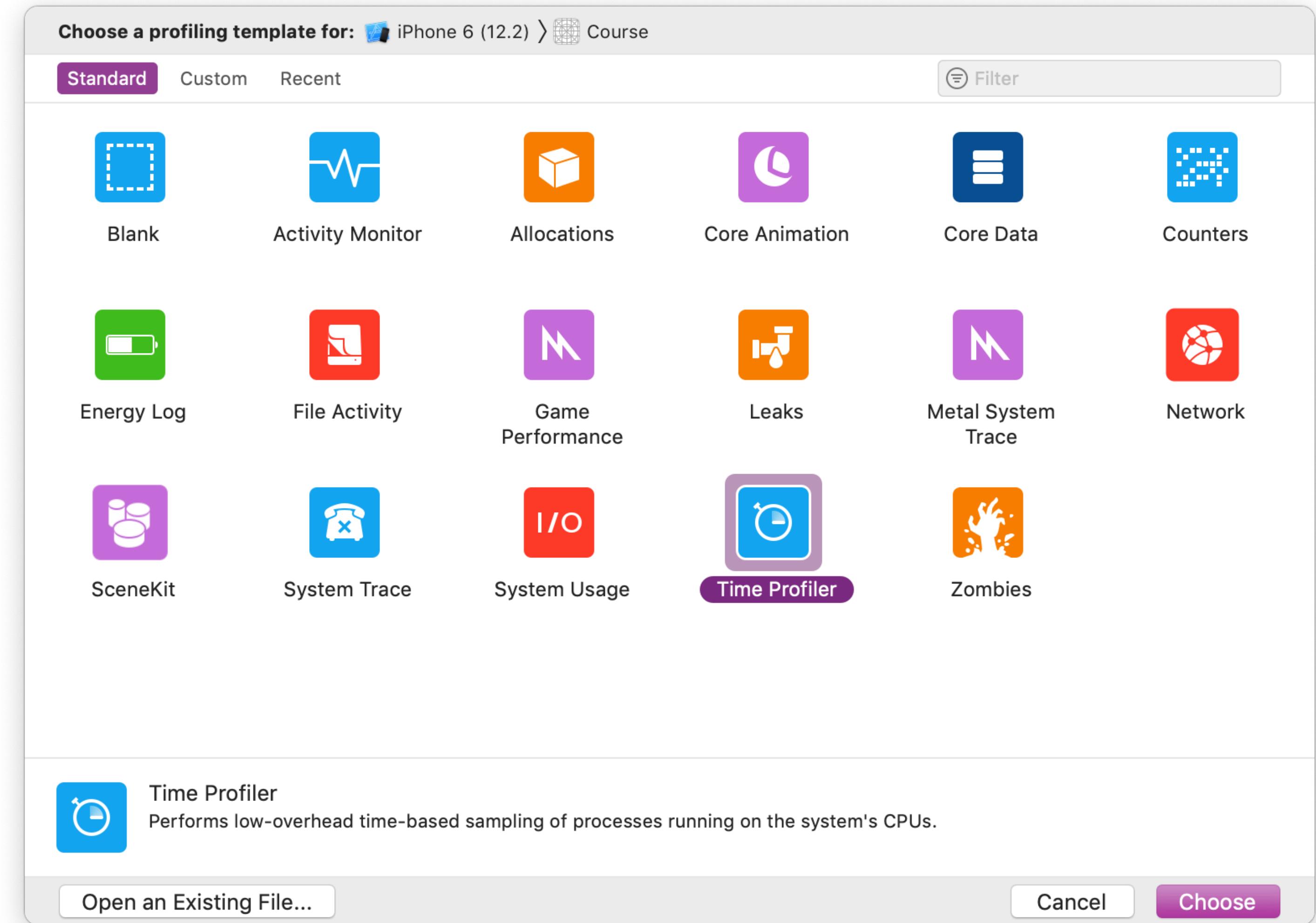
# View debugging



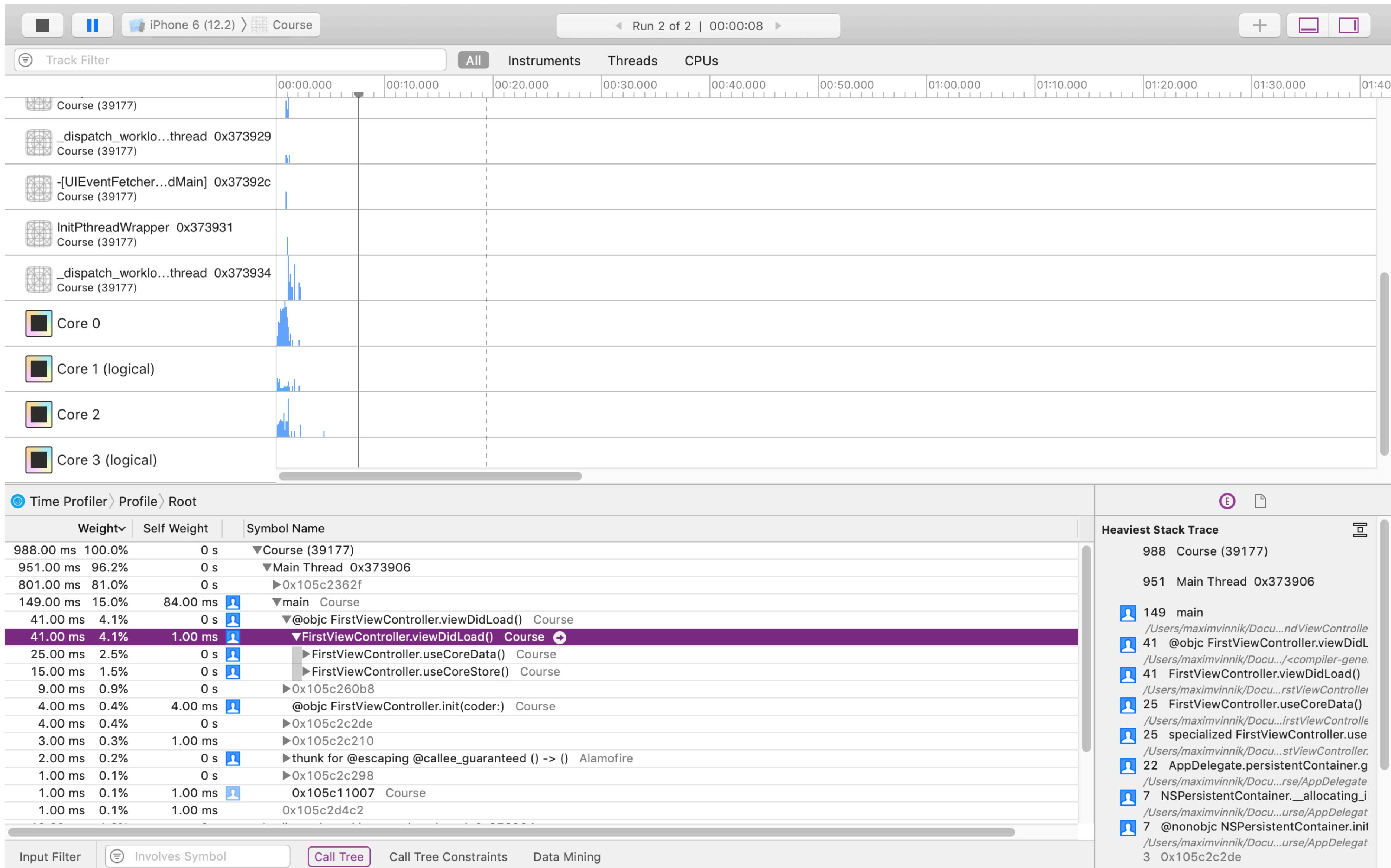
# Memory debugging



# Instruments

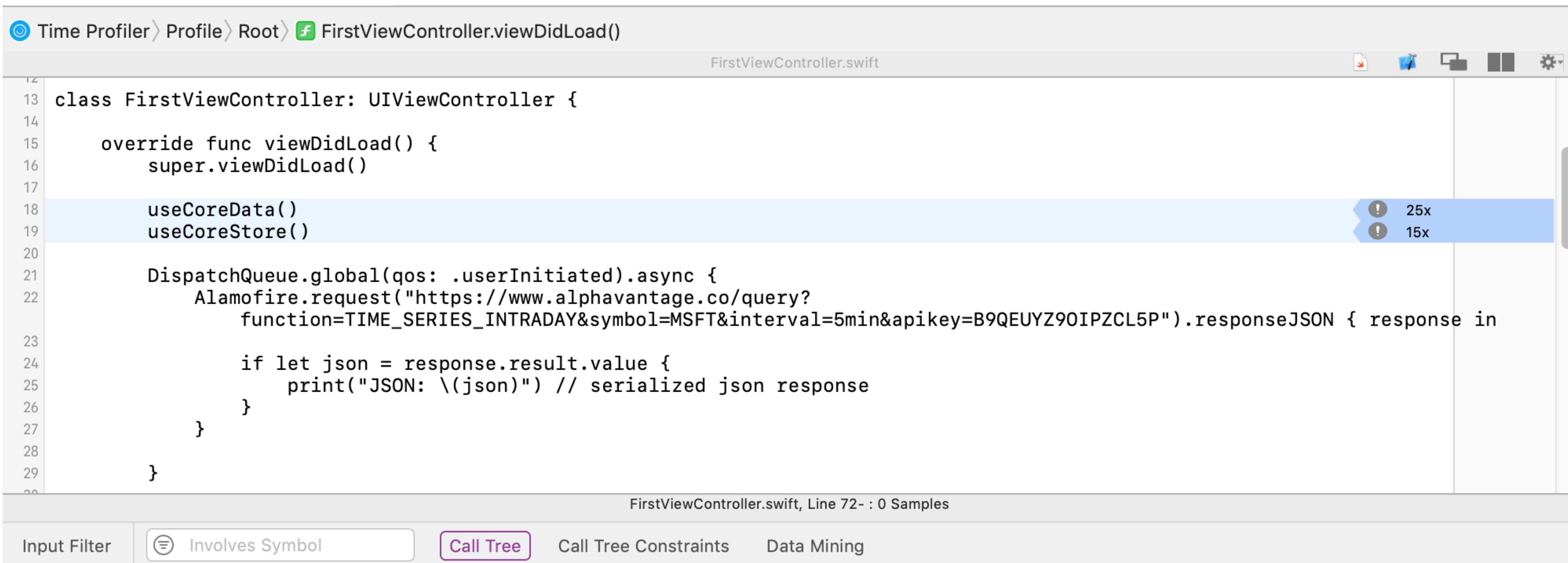


# Time Profiler



# View code in Instruments

By double clicking on some line in call tree, instruments will open corresponding code if it is available. Also some lines will be highlighted, showing some measured value.



```
Time Profiler > Profile > Root > FirstViewController.viewDidLoad()
FirstViewController.swift

12
13 class FirstViewController: UIViewController {
14
15     override func viewDidLoad() {
16         super.viewDidLoad()
17
18         useCoreData()
19         useCoreStore()
20
21         DispatchQueue.global(qos: .userInitiated).async {
22             Alamofire.request("https://www.alphavantage.co/query?
23                 function=TIME_SERIES_INTRADAY&symbol=MSFT&interval=5min&apikey=B9QEUYZ90IPZCL5P").responseJSON { response in
24
25                 if let json = response.result.value {
26                     print("JSON: \(json)") // serialized json response
27                 }
28             }
29         }
30     }
}

FirstViewController.swift, Line 72- : 0 Samples

Input Filter  Involves Symbol  Call Tree Call Tree Constraints Data Mining
```

# Simulator Debugging

Simulator itself has some debug options you can turn on

