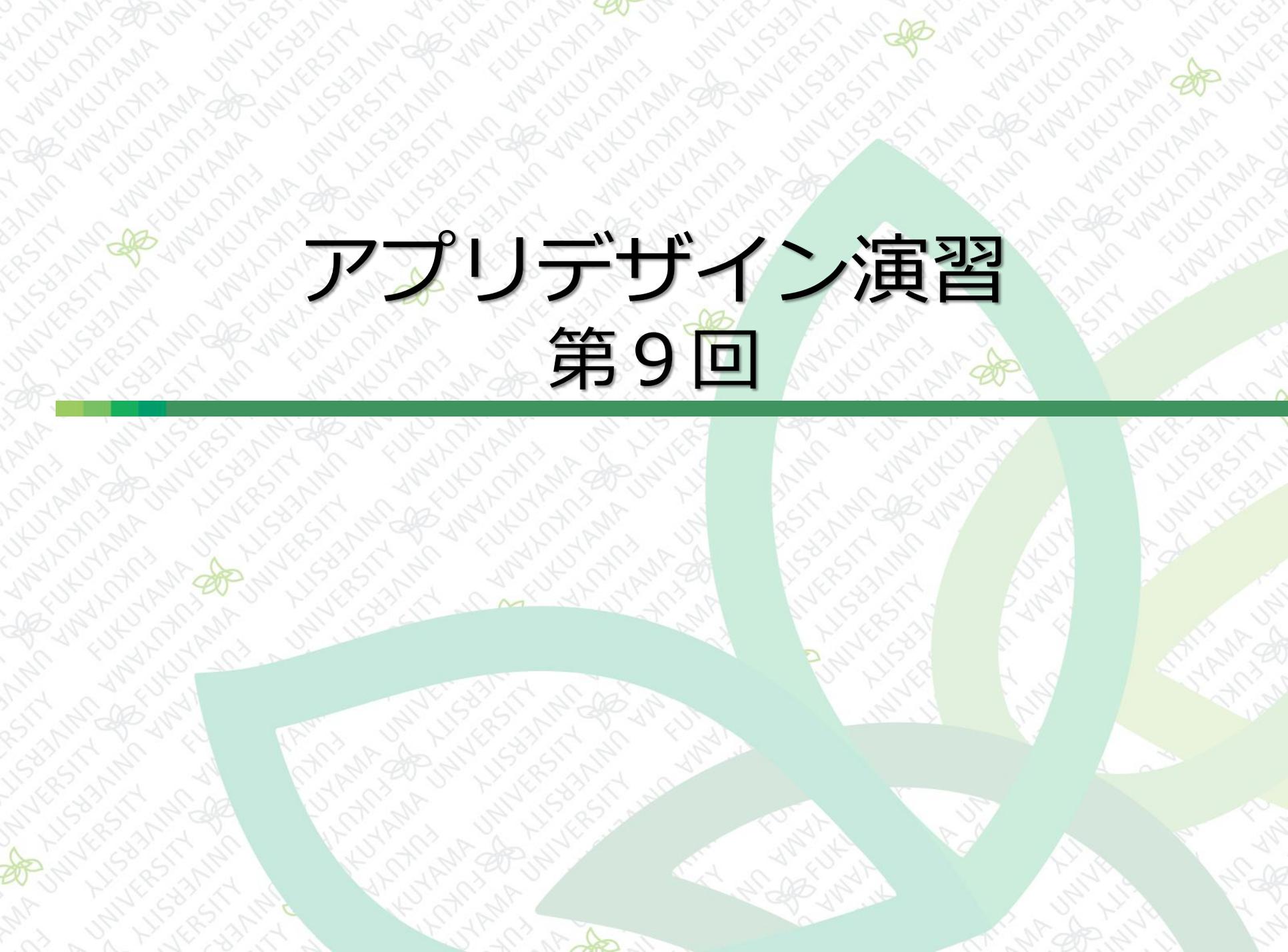


アプリデザイン演習 第9回



今日の予定

■ 準備

■ グループわけ

■ MacBookの取り扱い

■ Macの基本的な使い方

■ Xcode、Playground

■ Swiftをかじってみる

MacBookの取り扱い

- 使わないときは、MacBookは棚に入れておく

- 棚は施錠してある

- 鍵は・・・にある（秘密にしておくこと）

- 使用後は、必ず元に戻して施錠すること

- 演習中は

- 1グループに1台

- 棚から出して各グループが設置する

- 使い終わったら、元に戻しておく

- ◆ ACアダプター&ケーブルも元のところに

MacBookの取り扱い

■ 利用可能なMacBookは3種類

■ MacBookAir (15インチ) … 2台

◆ M3チップ

■ MacBookAir (13インチ) … 2台

◆ M1チップ

■ MacBookPro (13インチ) … 5台

◆ 一部、古いものもある

Macの基本的な使い方

■ Macの使い方（1）

■ タッチトラックパッド

◆ クリック

◆ 副クリック(右クリック) ・ ・ [control]+クリック

■ ファイル操作（Finder）

◆ アプリケーションフォルダ

■ ドック（Dock）

■ ブラウザ（Safari）

■ システム環境設定

Macの基本的な使い方

■ Macの使い方（2）

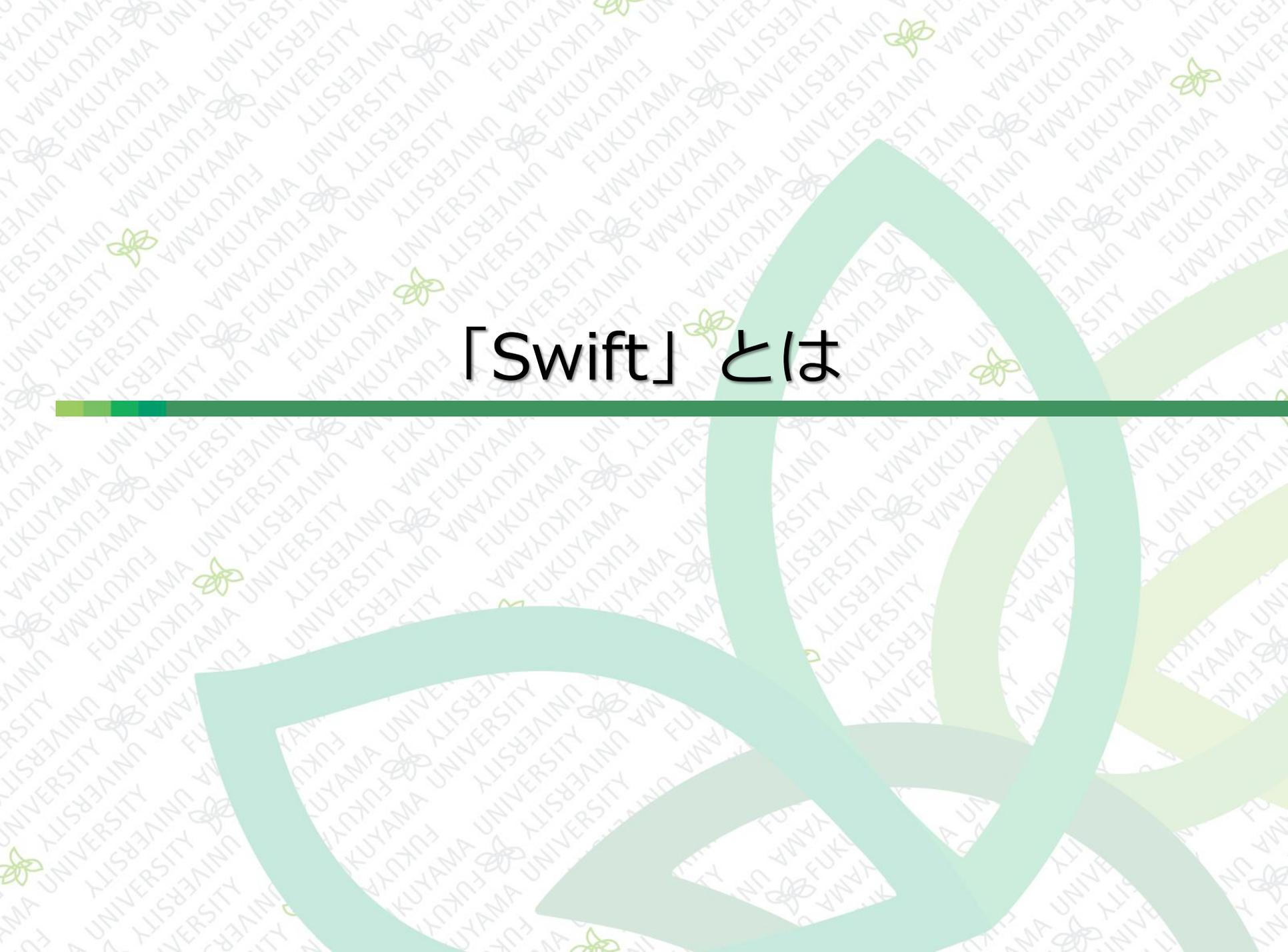
■ 日本語入力

- ◆ [英数]キー：半角英数モード
- ◆ [かな]キー：日本語入力モード

■ ショートカットキー

キーの組み合わせ	動作
[command] + [C]	コピー
[command] + [X]	切り取り
[command] + [V]	貼り付け
[option] + [¥]	バックスラッシュ
[fn] + [delete]	カーソルの右側の文字を削除
[shift] + [command] + [3]	画面全体のスクリーンショットを撮る
[shift] + [command] + [4]	画面の一部のスクショを撮る

「Swift」とは



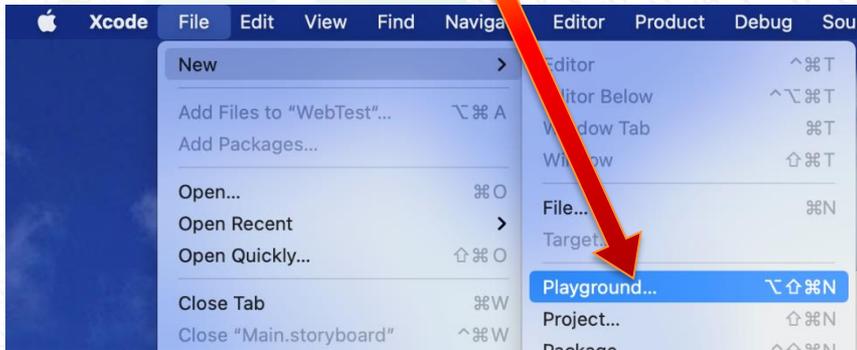
Swiftとは

- Appleが開発したプログラミング言語
- 特徴は
 - 速い
 - モダン（現代的）
 - 安全
- 安全機能がついている
 - nil（null、ヌル）に対してチェックが厳しい

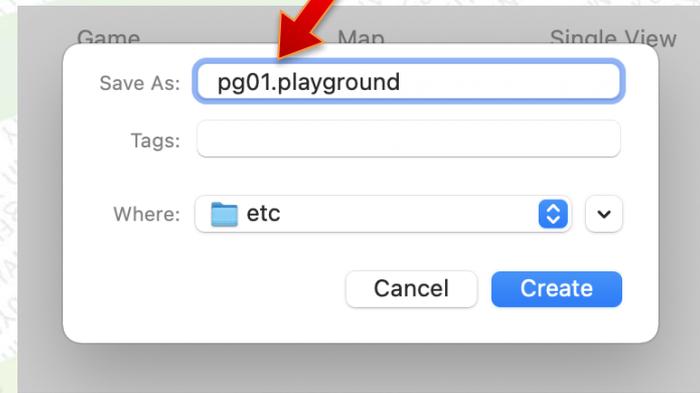
次のスライドから、簡単なSwiftのプログラムを打ち込んで
どんな感じか見ていきます・・・

Xcodeを起動、Playgroundを使う

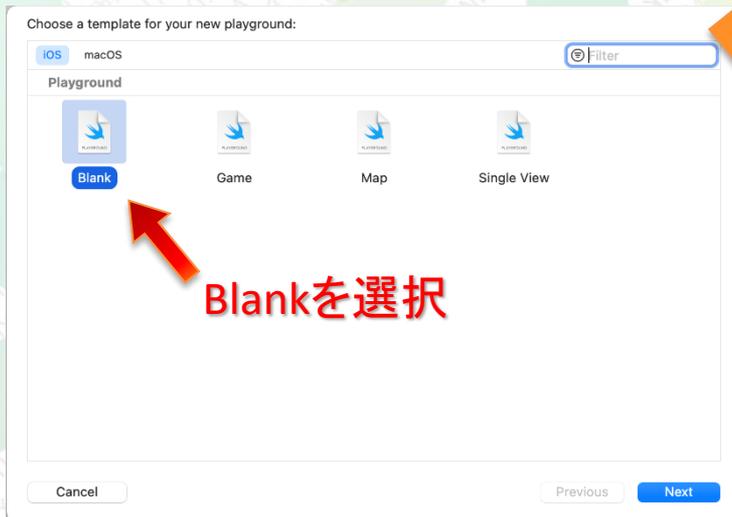
Playgroundを選択



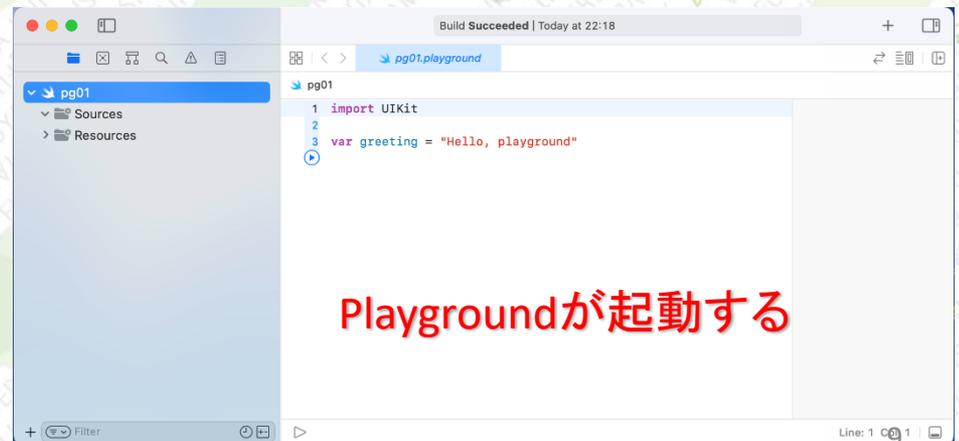
名前をつける
「pg01.playground」とか



Blankを選択



Playgroundが起動する



Xcodeを起動、Playgroundを使う

Ready to continue pg01

```
pg01
1 import UIKit
2
3 var greeting = "Hello, playground"
4
5 print(greeting)
6 1+1
```

"Hello, playground"

print(greeting)
1+1
と入力してみる

ここには途中の結果が表示される

Ready to continue pg01

Developer/Platforms/iPhoneOS.platform/Library/Developer/CoreSimi
iOS.simruntime/Contents/Resources/RuntimeRoot/System/Library/Pr
UIKitCore.framework/UIKitCore (0x118a3ba50) and /Applications/X
Platforms/iPhoneOS.platform/Library/Developer/CoreSimulator/Pro
iOS.simruntime/Contents/Resources/RuntimeRoot/System/Library/Pr
TextInputUI.framework/TextInputUI (0x13740a8d8). One of the two
undefined.

```
pg01
1 import UIKit
2
3 var greeting = "Hello, playground"
4
5 print(greeting)
6 1+1
7
```

"Hello, playground"

"Hello, playground\n"

2

Line: 6 Col: 4

Hello, playground

▶をクリックすると、実行される

基本的な計算、変数、定数

以下をPlaygroundで入力してみよう（エラーは修正してみる）

- ・ 計算式（算術演算子：+,-,*,/,%）

```
7 1+1
8 8-3
9 99*99
10 256/16
11 100%3+100*3-100
```

このスライド以降、Macで
入力する人を、スライドご
とに順番に交代すること

- ・ 変数は「var」をつけて宣言します

```
13 var score = 100
14 score = 200
15 score = 100+200; score = score*2
```

みんなで確認しな
がら進める

- ・ 定数は「let」をつけて宣言します

```
17 let heisei = 1989
18 heisei = 2000
```

Cannot assign to value: 'heisei' is a 'let' constant

定数は変更
できない

- ・ 変数の「型」を指定して宣言します（var 変数名：型）

```
20 var itemCount:Int = 10
21 itemCount = itemCount + 50
```

```
23 let height:Float = 1.72
24 let weight = 64.5
25 let BMI = weight/(height*height)
```

異なる型どうしの演算
は厳しくチェックされる

基本的な計算、変数、定数

- 主な変数の型は `int`, `Float`, `Double`, `Bool`, `String` など
- `Bool` 型は `true` か `false` の値を保持します

```
27 var isOK:Bool = false
28 isOK = true
```

• `String` 型

```
30 let helloStr1 = "みなさん"
31 let helloStr2 = helloStr1 + "こんにちは"
32 let myWeight = 64.5
33 let myStr = "私の体重は" + myWeight + "kgです。"
34 //let myStr = "私の体重は\" + myWeight + "kgです。"
```

型が異なるので
型変換が必要

• 厳密な型のチェック、型変換（キャスト）

```
36 let inputString = "100"
37 let answer = inputString*5
38 //let answer = Int(inputString)!*5
39 let intValue = Int(123.45)
```

型変換が必要

```
41 let price:Int = 100
42 let pay = price*1.08
43 //let pay = Float(price)*1.08
44
45 let count = 5
```

priceはInt型、
1.08はFloat型かDouble型
なので、payの型が定まらない...

```
46 let myMessage:String = "リンゴが" + count + "個"
47 //let myMessage:String = "リンゴが" + String(count) + "個"
```

型変換が必要

条件分岐

• if文

```
7 // if文
8 var score = 100
9 if 80 < score {
10     print("GOOD!")
11 } else {
12     print("BAD!")
13 }
```

• Switch文

```
15 // switch文
16 var dice = 1 // 数値を変更してみよう
17 switch dice {
18 case 1:
19     print("振り出しに戻る")
20 case 2,5: // 2か5が出たらもう一回サイコロを振る
21     print("もう一回振る")
22 default:
23     print("出た目の数だけ進む")
24 }
```

繰り返し

• for文

```
26 // for文
27 for i in 0...3 { // 0,1,2,3で実行する(3まで)
28     print(i)
29 }
30 for i in 0..<3 { // 0,1,2で実行する(3は含まない)
31     print(i)
32 }
33 for _ in 0...3 { // 繰り返しの変数が必要ない場合
34 }
```

• while文

```
36 // while文
37 var d = 0;
38 while (d<100) {
39     d += 7
40 }
41 print("答えは\u{d}")
```

配列

• 配列の作成

```
7 // 配列(Array)
8 var intArray1 = [1,2,3] // 整数の配列
9 var strArray1 = ["ABC","DEF","GHI"] // 文字列の配列
10 // 型を指定して配列を作る
11 var intArray2:[Int] = [1,2,3] // 整数の配列
12 var strArray2:[String] = ["ABC","DEF","GHI"] // 文字列の配列
13 // 個数を指定して配列を作る
14 var intArray3 = Array(repeating:0, count:3) // 0を3個
15 var strArray3 = Array(repeating:"AB", count:5) // "AB"を5個
16 // 空の配列を作る
17 var emptyArray1:[String] = []
18 var emptyArray2 = [String]()
```

配列

• 配列の操作（1）

```
19 // 要素の個数を調べる
20 var intArray4 = [1,2,3,4,5]
21 var cnt = intArray4.count
22 // 要素の値
23 print(intArray4[0],intArray4[2])
24 // すべての要素を参照する
25 var strArray6 = ["AB","CD","EF"]
26 for val in strArray6 {
27     print("要素=\(val)")
28 }
```

• 配列の操作（2）

```
30 // 配列の最後に要素を追加する
31 strArray6.append("GH")
32 // 指定の位置に要素を挿入する
33 strArray6.insert("XY", at:1) // 添字1のところに挿入
34 // 指定の位置の要素を削除する
35 strArray6.remove(at:1) // 添字1のところの要素を削除
36 // 要素をすべて削除する
37 strArray6.removeAll()
```

配列

• 配列の操作（3）

```
38 // 配列の並び替え
39 var intArray5 = [4,3,1,5,2]
40 var sortArray5 = intArray5.sorted(by: { $0 < $1 }) // 昇順
41 print(sortArray5)
42 var sortArray6 = intArray5.sorted(by: { $0 > $1 }) // 降順
43 print(sortArray6)
```

関数

• 関数の定義、呼び出し

```
7 // 関数
8 func showHello1()
9 {
10     print("こんにちは")
11 }
12 showHello1() // 関数の呼び出し
```

• 関数の引数 (1)

```
13 // 関数の引数(1)
14 func showHello2(name:String)
15 {
16     print("\(name)さん、こんにちは")
17 }
18 showHello2(name:"Suzuki") // 関数の呼び出し
```

• 関数の引数 (2)

```
19 // 関数の引数(2)
20 func calcBMI(height:Double, weight:Double)
21 {
22     let m = height/100.0
23     let BMI = weight/(m*height)
24     print(String(format:"BMIは%5.2fです",BMI))
25 }
26 calcBMI(height:171.0,weight:64.5) // 関数の呼び出し
```

関数

• 関数の戻り値（1）

```
27 // 関数の戻り値
28 func getHello(name:String) -> String{
29     let message = "\(name)さん、こんにちは"
30     return message
31 }
32 let hello = getHello(name:"Tanaka")
```

• 関数の戻り値（2）

```
33 // 関数の戻り値（戻り値が複数ある場合）
34 func calcTax(price:Double) -> (Double,Double){
35     let tax = price*0.08
36     let includingtax = price*1.08
37     return (tax,includingtax)
38 }
39 let (tax,includingtax) = calcTax(price:300)
40 print("消費税は\(tax)円")
41 print("税込価格は\(includingtax)円")
```

その他の機能

- 辞書 (Dictionary)
- タプル型
- オptional型
- 構造体
- クラス

など . . .