iPhoneレクチャー1回目

溝口健太
村上ろまん

1回目からswiftをやります。

なお基本構文の模様

まだUIは触りません

変数宣言のやり方

- let
  定数を扱う
  途中で値を変えるとコンパイラに怒られるよ

- var
  変数を扱う
  初期化をしない場合は明示的な型名の宣言が必要
  -ex ○ var hoge:String / var hoge = “hoge”
  × var hoge

変数宣言のやり方

余談

変数名は日本語,絵文字でも可

- var 犬 = “ニャー(⌒▽⌒)”

print(犬)

- var 🚴 = “road bike”

print(🚴)

セミコロンの扱い

今まで

int hoge = 3;

swift

var hoge = 3

セミコロンは一行に複数の処理を行う時のみ付ける
ただしセミコロン付けてもきちんと動くで

複数の変数宣言のみなら,(コンマ)でもおk

コンソーラに関して

- print(“hogeと表示”)

let hoge = 5

- print(“hoge = \(hoge)”)

\は alt+¥ で出るで

文字列の扱い

文字の連結が非常に楽

var hoge = “Hello ”

hello += “World!”

print(hoge)

~~printlnでもおk 改行するだけ~~
swift2.0より廃止。printで改行がデフォ

練習

要素数5の適当な値の中から５よりも大きいものをコンソールに表示させてください

for文 for in文 while文 好きなループ使ってください

~~実行は時間かかるので⌘Bで~~

解答例(for文)

```
let value = [2,4,6,8,10],score = 5

for var i = 0;i < value.count;i++ {
    if value[i] > score{
        print(value[i])
    }
}
```

解答例(for in文)

```
let value = [2,4,6,8,10],score = 5

for num1 in value{
    if num1 > score {
        print(num1)
    }
}
```

三項条件演算子

hoge ? num1 : num2

You know what this mean?

練習

任意の真偽値の変数を宣言してその値(または文字列)によって変化(四則演算etc)した値をコンソールに表示してください。値と演算方法はご自由に

解答例

値の場合

```
let hoge = true,num1 = 30
var num = num1 + (hoge ? 50:10)
print("num is \(num)")
```

解答例

文字列の場合

```
let hoge = true,moji1 = "abc"
var moji = moji1 + (hoge ?
    "def":"ghi")
print("moji is \(moji)")
```

辞書配列(連想配列)の扱い

聞いたこと/使ったことある？

辞書配列(連想配列)の扱い

配列

hoge[5] = [1,2,3,4,5]

辞書配列(連想配列)の扱い

連想配列

let hoge = [“れんげ”:7,”ほたる”:11,”なつみ”:13,
”こまり”:14]

今回の例では””内(String)がkey値,value値に数値(Int)が対応

イメージ れんげ→7 ほたる→11etc

key値を呼び出すことによってそれに対応したvalue値を取り出す
↑ここが大事

辞書配列(連想配列)の扱い

普通の配列は配列の順番が決まっている

-ex 順番 ① １ ② ２ ③ ３ ④ ４ ⑤ ５

連想配列の場合順番は決まっていない!

-ex 順番 ② １ ① ２ ④ ３ ③ ４ ⑤ ５ もありうる

辞書配列(連想配列)の扱い

実際に書いて確認してみよう！

Try it!

辞書配列(連想配列)の扱い

ちなみに連想配列だとkey値のみvalue値のみの取り出しも(わかると思うけど)可能です

for key in hoge.keys{

で確認したかったらやってみてください

要は使い所考えて配列は使ってください

練習

配列 hoge = [“れんげ”:7,”ほたる”:11,”なつみ”:13,”こまり”:14]の任意のkeyからvalue値を１つコンソールに表示してみよう

取り出した値はoptional型(値がnilである可能性がある認識でおk)になってしまうので取り出した値がnilでないか条件をつけましょう

ヒント

1行の方

- Optional型で変数の後に！をつけることにより変数が絶対にnilではないと明示できる

条件が必要な方

- 条件として別の変数に要素を入れてnilでないのを確認

解答(1行)

```
var hoge = ["れんげ":7,"ほたる":11,"なつみ":13,"こまり":14]
print(hoge["ほたる"]!)
```

Optional型には!の他に?もあるので興味があったら調べてみ

解答

```
var hoge = ["れんげ":7,"ほたる":11,"なつみ":13,"こまり":14]

var f:Int
if let nonon = hoge["ほたる"]{
    f = nonon
}else{
    f = 0
}
print(f)
```

タプルに関して

タプルとは・・・

-ex hoge = (3,5,7,”hg”,7.3)

hoge = (name:”コーラ”,price:100,tax:0.08,priceWithTax:108)

タプルに関して

配列との違いは？

異なる型をまとめられる
要素の追加/削除ができない

練習

配列 item = ("ジュース", 100, 0.08, 108)の中の

1. １番目と４番目の合計をコンソールに表示
2. 2番目と３番目の合計をコンソールに表示

型変形 -ex Int() Double()

関数宣言のやり方

```
func hoge(num1:Int){
    print("\(num1)")
}

hoge (2)
```

引数1つ 戻り値なし

関数宣言のやり方

```
func hoge(num1:Int)->Int{
    return num1
}
let num = hoge (2)
print(num)
```

引数1つ 戻り値あり

関数宣言のやり方

```
func hoge(num1:Int,num2:Int)->Int{
    return (num1+num2)
}
let num = hoge (2,3)
print(num)
```

引数2つ 戻り値あり

関数宣言のやり方

引数が２つ以上の関数の場合２つ目以降変数名
をきちんとつけよう

```swift
func hoge(num1:Int,num2:Int)->Int{
    return (num1+num2)
}
let num = hoge(2,num2:3)
print(num)
```

objcも長くてキモいけどswiftもそこそこキモいよ

外部引数に関して

関数の引数に仮引数とは別にラベル（外部名）をつけると、コードを読む人（自分含）がわかりやすくなるよ

ラベルをつけて呼び出しを行うと、2つ以上の関数の際に引数名が必要ありません(どっちが書いてて分かりやすいかで決めておk)

外部引数に関して

```
func hoge(num1 n1:Int,num2 n2:Int)->(Int){
    return (n1+n2)
}
let num = hoge(num1:2,num2:3)
print(num)
```

外部引数に関して

n1=num1に　　省略パターン

```
func hoge(num1 n1:Int,num2:Int)->(Int){
    return (n1+num2)
}
let num = hoge(num1:2,num2:3)
print(num)
```

swift1.2だと#で仮引数をそのまま外部引数に出来て分かりやすかったけどswift2.0では廃止

練習

任意の文字列を２つ引数にもつ関数
hogeにて文字列連結の処理を行い、
その結果をコンソールに表示

解答例

```
func hoge(moji1 m1:String,moji2:String)->(String){
    return (m1+moji2)
}
let str = hoge(moji1:"sore",moji2:"java")
print(str)
```

m1はmoji1にした方が分かりやすいのでなるべく同じに

練習

変数num1,num2にはそれぞれInt型で50,40の値が入っている。ここで、num1の値がnum2より大きい場合加算、小さい場合減算する関数を作成し、結果をコンソールに表示してください

解答例

```swift
func add(num1 num1:Int,num2:Int)->(Int){
    return num1+num2
}

func subtract(num1 num1:Int,num2:Int)->(Int){
    return num1-num2
}

let num1 = 50,num2 = 40;var sum = 0
if(num1 > num2){
    sum += add(num1:num1,num2:num2)
}else{
    sum -= subtract(num1:num1,num2:num2)
}
print(sum)
```

関数を引数に

関数を引数にすることもできます。

うまく使えば結構便利なので頭の片隅でも覚えとくといいよ

関数を引数に

```
func hoge(a a:String)->(String){
    return a+"java"
}
```

```
func addStr(function function:(String)->(String))->(String){
    return hoge(a:"sore")
}

print(addStr(function:hoge))
```

イメージ

addStr(function:hoge)でfunctionの関数決定

addStrで関数hogeに文字列を引数に与える

hoge()の呼び出し

練習

先ほどの例 変数num1,num2にはそれぞれInt型(ry

を変更して引数に関数を指定してコンソールに表示してください

解答例

```swift
func add(num1 num1:Int,num2:Int)->(Int){
    return num1+num2
}

func subtract(num1 num1:Int,num2:Int)->(Int){
    return num1-num2
}

func calculate(num1 num1:Int,num2:Int,function:(Int,Int)->Int)->(Int){
    return function(num1,num2)
}

let num1 = 50,num2 = 40;var sum = 0
if(num1 > num2){
    sum = calculate(num1:num1,num2:num2,function:add)
}else{
    sum = calculate(num1:num1,num2:num2,function:subtract)
}
print(sum)
```

時間あったらやる

任意の文字列を持つ変数にループを１０回繰り返して任意の文字を追加させて、その結果をコンソールに表示させてください

解答例

```
func hoge(allStr allstr:String)->(String){
    count++
    if(count != 1){
        allStr += "a"
    }
    return allStr
}

func addStr(function function:(String)->(String))->(String){
    return hoge(allStr:allStr)
}

var count = 0,allStr="sore"

for num in 0...9{
    print(addStr(function:hoge))
}
```

戻り値に関して

戻り値は１つだけじゃなくてもおk

使用用途に応じて使い分けよう

戻り値に関して

```
func hoge()->(String,Int){
    return ("hironaka",0)
}

let human = hoge()
print("\(human.0)'s communication ability \(human.1)")
```

どっちも同じ処理だけど見やすさ/分かりやすさは？

```
func hoge()->(name:String,communication_Ability:Int){
    return ("hironaka",0)
}

let human = hoge()
print("\(human.name)'s communication ability \(human.communication_Ability)")
```

戻り値に関して

コード書き忘れたけど、print(human)

にするとタプルで値を取り出せるYO