SCIENCES SORBONNE UNIVERSITÉ

# Swift, control structures

Fabrice.Kordon@lip6.fr

LIP6

# As an introduction…



- **Goal, control of instructions**
  - Based on manipulated data
- **Classical structures**
  - If/then/else
  - loops
  - switch

# If/then/else



```swift
import UIKit

let str = "Rominet"
if str == "Rominet" {
    print ("Titi is in danger!!!")
} else {
    print ("All is OK")
}

var grade = 13
if note < 10 {
    print("Exam failed")
} else if (grade >= 10) && (grade < 16) {
    print("Exam passed")
} else {
    print("You are better than most people")
}
```

```
Titi is in danger!!!
Exam passed
```

# «enumerative» loops



## very «Ada-oriented»

- Index implicitly declared
- read-only access in the loo baby
- Different from dictionaries/arrays iterators

```
for i in 1 ... 5 {
    print("i = \(i), ", terminator: "")
}
print("Let's have a look at planets")
let terrestrial = [ "Mercury", "Venus", "Earth", "Mars"]
for p in terrestrial {
    print ("\(p), ", terminator: "")
}
print ("Let's have a look at animals")
let dictionnaire = ["Spider": 8, "Ant": 6, "Cat": 4]
for (animal, feet) in dictionnaire {
    print("This \(animal) has \(feet) feet")
}
```

```
i = 1, i = 2, i = 3, i = 4, i = 5, Let's have a look at planets
Mercury, Venus, Earth, Mars, Let's have a look at animals
This Cat has 4 feet
This Ant has 6 feet
This Spider has 8 feet
```

# «while» loops



## tes may be before or after the loop body

```
// Test at the beginning of the loop body
var x = 3
while x < 5 {
    print ("x = \(x)")
    x+=1
}

// Test at the end of the loop body
var y = 3
repeat {
    print ("y = \(y)")
    y+=1
} while y < 5
```

```
x = 3
x = 4
y = 3
y = 4
```

# Switch



```swift
enum Terrestrial {case Mercury, Venus, Earth, Mars }
var p = Terrestrial.Earth

switch p {
case .Earth :
    print("You can leave on it")
case .Mars :
    print ("It is really too cold");
default: // All possibles values for the type must be covered
    print ("It is really to hot")
}
```



```
You can leave on it
```

# As a conclusion…



## That's quite simple is'n't it?

- Notions found in other languages

## Considers feed back on «safe programming»

- Strong typing + dynamic typing
- Sound syntax
- etc.

## This leads to safe structures

- Bodies surrounded by «{« and «}»
- Covering of types (switch)
- Protection of indexes
  - **No more «C-ike loops»**

# As a conclusion…



## That's quite simple is'n't it?

- Notions found in other languages

## Considers feed back «programming»


- Strong typing + dynamic typing
- Sound syntax
- etc.

## This leads to safe structures

- Bodies surrounded by «{« and «}»
- Covering of types (switch)
- Protection of indexes
  - **No more «C-ike loops»**