**1. Các khái niệm cơ bản :**

- Kiểu dữ liệu nguyên thủy : Int, Double, Boolean..

- Kiểu số nguyên không dấu : UInt64, UInt8

- Bí danh typealias BigNumber : Double

let number : BigNumber = 1000

print(“\(number)”) → 1000.0

→ Tạo ra 1 kiểu dữ liệu mới tên là BigNumber dựa trên kiểu gốc là Double

- assert(number >= 10, “Số đã cho không thể lớn hơn 10”) : So sánh number với 10, nếu >= 10 thì in ra thông báo kia

- Thay assert bằng precondition, thì nếu điều kiện không đúng, ngoài việc thông báo lỗi ra thì nó sẽ dừng luôn không chạy các lệnh phía sau nữa

→ precondition(number >= 0, “Số đã cho không thể nhỏ hơn 0”)

**2. Các toán tử cơ bản**

- Toán tử đơn : boolean

- Tuple : Danh sách các biến

→ let (width, height) = (600, 800)

- Toán tử số học : +, - , \* , /, %

- Toán tử so sánh : ==, !=, >=, >, <=, <

- So sánh 2 tuple ta so sánh từng phần tử trong mỗi tuple đó

VD (1, ‘cat’) < (3, ‘dog’) = true do 1 < 3 và c < d

- Toán tử 3 ngôi

let isHD = (width, height) > (1280,720) ? “isHD” : “notHD”

→ So sánh a > b, nếu đúng trả về isHD , sai trả về notHD

- Toán tử ??

var age : Int? = nil

let myAge = age ?? 18

→ Gán myAge = age, nếu age nil thì gán mặc định là 18

- Vòng for :

+ for index in 1…5 : lặp từ 1 đến 5

+ for index in 1…<5 : lặp từ 1 đến 4 (5-1)

- Kiểm tra 1 số có nằm trong khoảng …7 (<=7)

let range = …7

range.contains(5) → true

- Khi không biết mảng bao nhiêu phần tử có thể dùng 0… (>=0)

let animals = [“dog”, “cat”,”fish”]

for anim in animals[0…] hoặc animal[…2] → tất cả các phần tử có index >= 0 hoặc tất cả các phần tử có index <= 2

**3. String và character**

- Viết string trên nhiều dòng dùng 3 dấu nháy: “”” string “” như python

Let str = “””

This is my project

“Welcome to you”

“””

- Tạo 1 string rỗng : let str = “” hoặc let str = String()

- Kiểm tra 1 string rỗng không : str.isEmpty

- Duyệt các ký tự trong 1 string

for character in str{

}

- Có thẻ chèn thêm phép toán vào trong string thông qua ‘\(a \* a)’

- Thêm icon trong app, có thẻ chèn các mã icon unicode : \u{2602} : cái ô

let umbrela = “\u{2602}”

print(umbrela)

- Xem phần tử đầu, phần tử cuối của mảng str

+ str[str.startIndex] : xem vị trí phần tử đầu

+ str[str.index(before: str.endIndex)]) : xem vị trí phần tử cuối

+ str[str.index(after: str.startIndex)]) : xem vị trí phần tử sau phần tử đầu

- Xem phần tử thứ n kể từ đầu

→ str[str.index(str.startIndex, offsetBy: n)]

- Insert một kí tự vào vị trí nào đó trong chuỗi : str.insert(character, at : position)

- Insert một string vào vị trí nào đó : str.insert(contentOf : string, at : position)

- Substring :

From : str.startIndex

To : str.index(str.startIndex, offsetBy: 5)

→ Lấy từ 0 đến 5 :   
let subString = str[from…to]

- Clone String2 từ string1, string 1 thay đổi thì string 2 vẫn không đổi  
let subString2 = String(string1)

- Kiểm tra tiền tố của chuỗi str có bắt đầu là str1 không  
let str = “Hello Tú Nguyễn”  
str.hasPrefix(“Hello”)

- Kiểm tra hậu tố : str.hasSuffix() – có phân biệt hoa thường  
→ Muốn không phân biệt hoa thường, có thể dùng .uppercased() để in hoa hết rồi so sánh

**4. Arrays**

- Tạo mảng : var a = [“dog”,”cat”]

+ Tạo mảng trống : var b = [String]()

+ Tạo mảng 10 phần tử mặc định gtri 5 : var array = Array(repeating: 5, count: 10)

- Thêm vào mảng :

+ a.append(“first”)

+ a += [“snack”, “elephant”]

+ a.insert(“giraffe”, 0) : thêm vào vị trí số 0

- Cập nhật mảng : a[0] = “sloth”

- Đếm số phần tử : a.count

- Xóa phần tử : a.remove(at : 0)

- Lấy phần tử từ 1 đến 3 : print(arr[1…3])

- Lấy danh sách phần tử gồm cả index và value  
for (index, value) in arr.enumerated(){  
 print(“index = \(index) và value = \(value)”)

}

**5. Sets**

- Là một dạng dữ liệu kiểu mảng nhưng không quan tâm đến vị trí sắp xếp của nó

- Khởi tạo : var animalSet: Set<String> = [“Dog”, “Cat”, “Fish”]

- Đếm số phẩn tử : animalSet.count

- Check rỗng : animalSet.isEmpty

- Thêm phần tử : animalSet.insert(“elephant”)

- Sắp xếp theo alpha: animalSet.sorted()

- Xóa phần tử, trả về thông tin phần tử đã xóa được

+ Xóa phần tử biết rõ : animalSet.remove(“Dog”)  
+ Xóa tại vị trí : animalSet.remove(at: position)

- Kiểm tra trong tập có phần tử Dog không : animalSet.contains(“Dog”)

- Lấy ra các phần tử : for animal in animalSet { print(“\(animal)”) }

- Giao của 2 tập hợp : set1.intersection(set2)  
let set1: Set = [1,2,3]  
let set2: Set = [3,4,5]  
let resultSet = set1.intersection(set2)  
print(\(resultSet))

- Hợp của 2 tập hợp : set1.union(set2)

- Hiệu của 2 tập hợp : s1.substracting(set2) - Các phần tử chỉ có trong 1, không có trong 2

- Gộp 2 tập hợp và lấy các phần tử khác nhau : set1.symmetricDifference(set2)

- Kiểm tra tập hợp 2 có là cha của 1 hay không : set2.isSuperset(of: set1)

- Kiểm tra tập hợp 1 có là con của 2 hay không : set1.isSubset(of: set2)

- Kiểm tra tập 1 và tập 2 có khác nhau hoàn toàn không : set1.isDisjoint(with: set2)

**6. Dictionary**

- Kiểu dữ liệu dạng (key, value)

- Khởi tạo :

+ Dictionary rỗng : var emptyDict: [String: String] = [:]

+ var user:[String, String] = [“name”:”Tú”, “email”:[tunguyen1601@gmail.com](mailto:tunguyen1601@gmail.com)

- Lấy giá trị : print(user[“name”])

- Update giá trị : user.updateValue([tunguyen@gmail.com](mailto:tunguyen@gmail.com), forKey: “email”), trả về giá trị email cũ trước khi update → oldEmail

- Duyệt bằng for : for {key, value} in user{ // do something }

- Duyệt chỉ lấy key :   
for eachKey in user.keys{ print \(eachKey)}

- Duyệt chỉ lấy value :   
for eachValue in user.values { // do something }

- Lấy allKey đưa vào mảng : var allKeys = [String](user.keys)

7. Vòng lặp

- For :   
+ for index in 1…10{ // nếu có sử dụng index}  
+ for \_ in 1…<10 { // nếu index không dùng thì cho thành \_ } : duyệt từ 1 → 9   
+ for index in stride(from: 1, to: 12, by: 3 ) = for(int i = 1; i < 12; i+=3)  
→ by này có thể là số thực vd by: 0.5

3. Optionals

- var name : String? = nil

→ Tạo biến name kiểu String cho phép gán gía trị nil (null) cho nó – có thể nil được

-