1.

**enum** Animal {

**case** People

**case** Tiger

**func** eat() -> String {

**switch** **self** {

**case** .People:

**return** "I eat everything!"

**case** .Tiger:

**return** "I eat pork"

}

}

}

**let** animal = Animal.People

print(animal.eat())

2.

**class** Animal {

**func** Elephant() {

}

**func** Tiger() {

}

**func** Horse() {

}

}

**class** eat: Animal {

**override** **func** Elephant() {

print("I eat vegetables")

}

**override** **func** Tiger() {

print("I eat pork")

}

**override** **func** Horse() {

print("I eat vegetables")

}

}

**let** animal = eat()

animal.Elephant()

animal.Tiger()

animal.Horse()

3.

String

4.

Class 只會容納存放目標實體位址 （遠端實體的概念）

且 class可以被多個其他的 class所繼承,

Calss適用於有較多的資料要處理和傳遞

Struct 會存放實體的所有內容 （實體存於本地）

Struct適用於固定的面積 / 幾何 數學運算

5.

Instance method:

實體方法(instance method)是屬於一個特定型別(類別、結構或列舉)的實體，可以用來存取和設置實體屬性或是提供實體需要的功能，將class、struc、enum實體化，指定給一個個體，並需要放在這個個體{}內部的function稱為Instance method，且每個實體有屬性（self），如果有需要修改可使用mutating(不可為常數)

Type method

型別方法(type method)為定義在不屬於實體的特定型別(類別、結構與列舉)上。，而是屬於這個型別(類別、結構或列舉)本身的方法，與實體方法一樣使用點語法(dot syntax)呼叫。

宣告型別方法時需要在func前加上關鍵字static。而以類別來說，還可以將static替換成class來允許子類別覆寫(override)父類別的類別方法。

6.

當一個物件在初始化的當下可以順便帶入, 該物件該有的變數，以便建立物件，在一個 class / struct在初始化時便會呼叫 init() 這一個 function，利用 init() 賦予 class 後續的 code 運行所需的所有變數初始值

7.

在Instance method且每個實體有屬性（self），使用這條規則的主要場景是實例方法的某個參數名稱與實例的某個屬性名稱相同的時候。在這種情況下，參數名稱享有優先權，並且在參考屬性時必須使用一種更嚴格的方式。這時你可以使用self屬性來區分參數名稱和屬性名稱

在實例方法中修改值型別(Modifying Value Types from Within Instance Methods)

如果有需要修改可使用mutating(不可為常數)

在變異方法中給self賦值(Assigning to self Within a Mutating Method)

在Type method（body）中，self指向這個型別本身，而不是型別的某個實例。對於結構和列舉來說，可以用self來消除靜態屬性和靜態方法參數之間的歧義

8.

