1. INTRODUCTION TO OPTIONALS

Swift has a special data type called **optional**.

An **optional variable** is a variable whose value **can be nil**. (nil = nothing)

An optional is a unique feature of the Swift programming language. Many other languages do not have this concept because most modern languages use the concept of **assigning default values to unassigned variables**.

Swift, on the other hand, does not have this concept - and therefore, how Swift handles **null values** or **unassigned variables** is different!

Read more about optionals: https://stackoverflow.com/a/24026093

2. DEFAULT VALUES OF VARIABLES

Many programming languages have the concept of "default values" for variables.

For example, in Java, suppose you have these variables:

```
int x;
double y;
String z;
```

Because you did not give an initial value to x, y, or z, Java will automatically assign them a default value.

In Java, ints default to 0, doubles default to 0.0, and Strings default to "".

Thus, the output of the following code is:

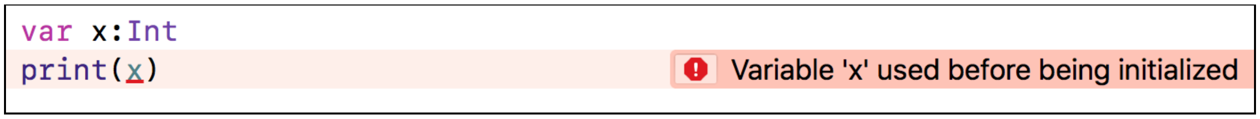
```
System.out.println(x)       // 0
System.out.println(y)       // 0.0
System.out.println(z)       // ""
```

Swift, however, does NOT have a concept of "default values".

For example:

```
var x:Int;
print(x)              // produces an error!
```

Results of running the code in XCode:



```
var x:Int
print(x)                                 Variable 'x' used before being initialized
```

The error: “Variable x used before being initialized” means:

- You want me to print out the value of “x”.
- But, you didn’t tell me what “x” is supposed to be!
- Therefore: ERROR!

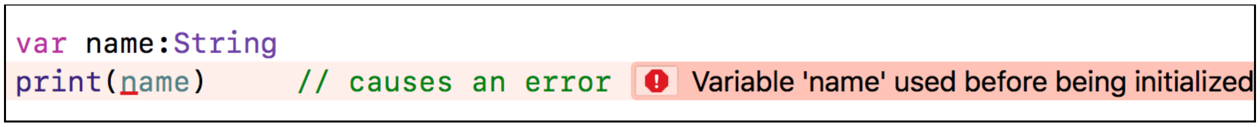
This example shows you a core principle of Swift:

ALL VARIABLES MUST BE ASSIGNED A VALUE BEFORE USING IT!

(unless you use an Optional!)

Example: Declaring a String variable

This will NOT work:

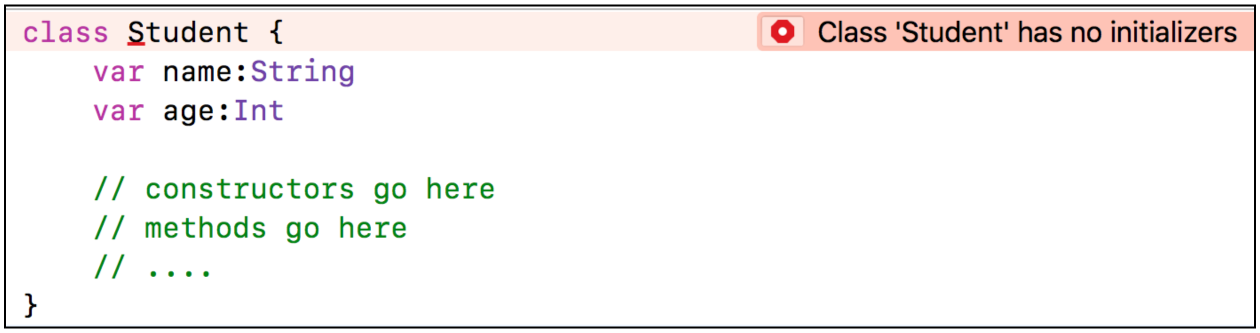


```
var name:String
print(name)      // causes an error   Variable 'name' used before being initialized
```

But this WILL:

```
var city:String = "Paris"
print(city)
```

This behaviour of Swift can lead to many interesting side effects, such as when you declare a class:



```
class Student {                                    Class 'Student' has no initializers
    var name:String
    var age:Int

    // constructors go here
    // methods go here
    // ....
}
```

In the example above, we declare a Student class with 2 properties: **name**, **age**.
In many programming languages, you do not have to assign an initial value for the properties.

However, in Swift, all variables need to be assigned values before they can be used. Thus, the above code produces an error.

On the other hand, giving your properties an initial value is okay:

```
class Student {
   var name:String = "Peter"
   var age:Int = 25

   // constructors go here
   // methods go here
   // ....
}
```

(No errors are produced by this code!)

-----

3. OPTIONALS

Although Swift does not have a default value for variables, Swift does recognize that many times you may not know what the initial value of a variable should be.

Swift provides a variable type called “**optional**”.
This data type means: “maybe the variable will have a value….and maybe it will not.”

An optional is declared with a **?** symbol.

Example: Optional Integer Data Type:

var x:Int?
Int? Means:

- In the future, I MIGHT assign a value to x
- But…I might not !

var y:String? Means:

- I might assign **y** a value in the future
- Or, I might just leave **y** unassigned. (I haven’t decided yet)

4. EVERY VARIABLE HAS AN OPTIONAL TYPE

Every data type in swift as a non-optional and optional version.

```
// non-optional int
var x:Int

// optional int
var x:Int?

// non-optional Student
var s:Student

// optional Student
var s:Student?
```

As a developer, you need to make a decision about whether your variables are optional or non-optional.

Use non-optional types when:

- You know that the variable will always have a value

Use optional types when:

- You aren't sure if the variable will always have a value.
- If there is ever a situation when the variable might be set to null, then use an Optional type.