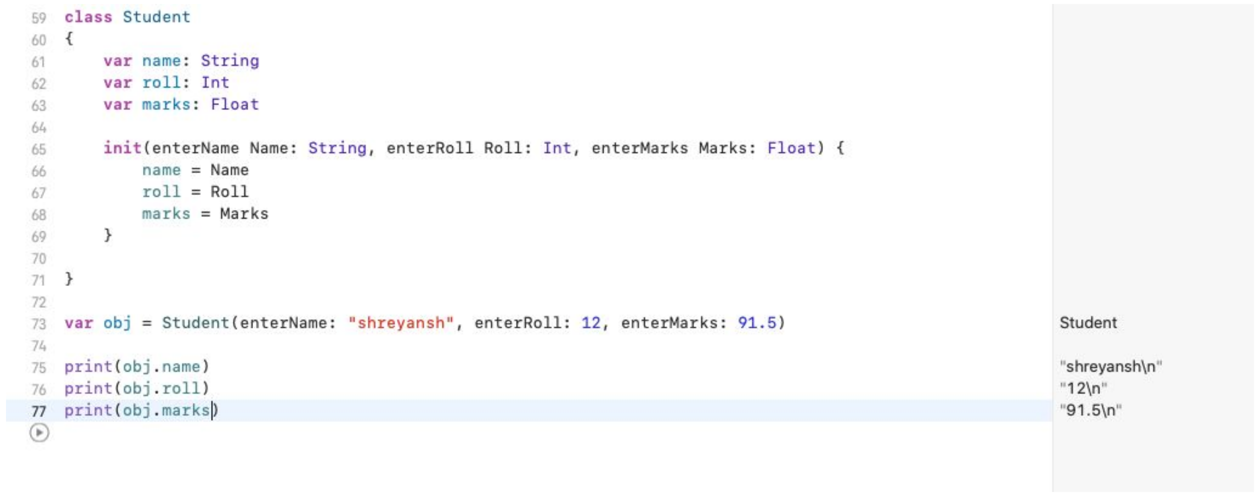
Swift Basic 2 Solution

Name : Shreyansh Raj Keshri
Date : 18/02/2020

Initializers

1. Implement the parameterised initialisation with class or struct.



```swift
class Student
{
    var name: String
    var roll: Int
    var marks: Float

    init(enterName Name: String, enterRoll Roll: Int, enterMarks Marks: Float) {
        name = Name
        roll = Roll
        marks = Marks
    }

}

var obj = Student(enterName: "shreyansh", enterRoll: 12, enterMarks: 91.5)

print(obj.name)
print(obj.roll)
print(obj.marks)
```

```
Student

"shreyansh\n"
"12\n"
"91.5\n"
```

2. Write all the Rules of initialiser in Inheritance

Rule 1: A designated initializer must call a designated initializer from its immediate superclass.

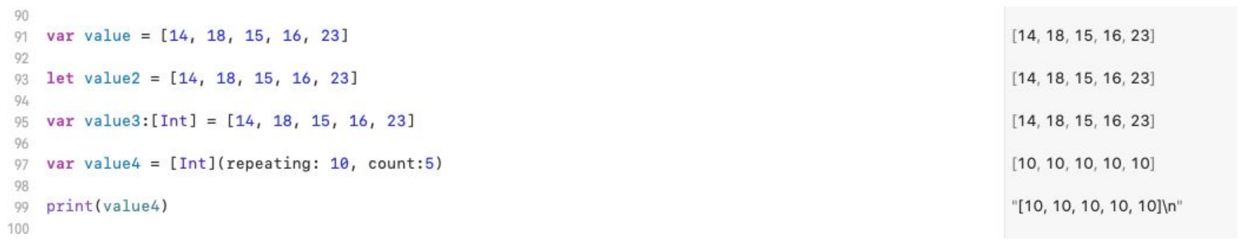
Rule 2: A convenience initializer must call another initializer from the sameclass.

Rule 3: A convenience initializer must ultimately call a designated initializer

3. Using convenience **Initializers**, write-down the **Initializers** for MOVIE class having basic attributes like title, author, publish_date, etc.
4. Declare a structure which can demonstrate the throwable Initializer

Array

1. Create an array containing the 5 different integer values. Write are at least 4 ways to do this.



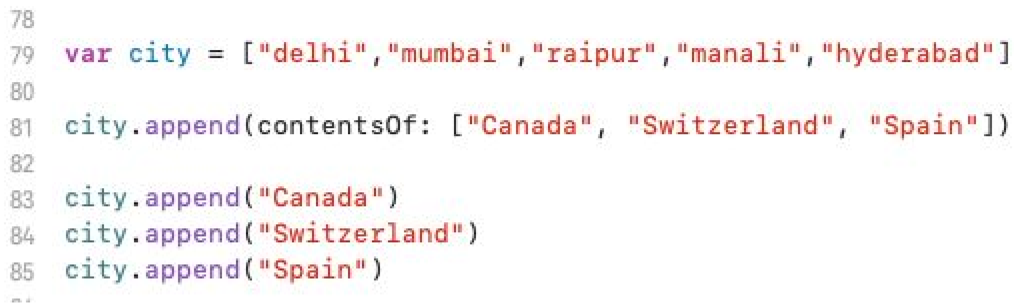
```
var value = [14, 18, 15, 16, 23]

let value2 = [14, 18, 15, 16, 23]

var value3:[Int] = [14, 18, 15, 16, 23]

var value4 = [Int](repeating: 10, count:5)

print(value4)
```

```
[14, 18, 15, 16, 23]
[14, 18, 15, 16, 23]
[14, 18, 15, 16, 23]
[10, 10, 10, 10, 10]
"[10, 10, 10, 10, 10]\n"
```

2. Create an immutable array containing 5 city names.

let city = ["delhi","mumbai","raipur","manali","hyderabad"]

3. Create an array with city 5 city names. Later add other names like Canada, Switzerland, Spain to the end of the array in at least 2 possible ways.



```
var city = ["delhi","mumbai","raipur","manali","hyderabad"]

city.append(contentsOf: ["Canada", "Switzerland", "Spain"])

city.append("Canada")
city.append("Switzerland")
city.append("Spain")
```

4. Create an array with values 14, 18, 15, 16, 23, 52, 95. Replace the values 24 & 48 at 2nd & 4th index of array

var value = [14, 18, 15, 16, 23, 52, 95]

value[2] = 24
value[4] = 48

Set

1. Given the following sets:

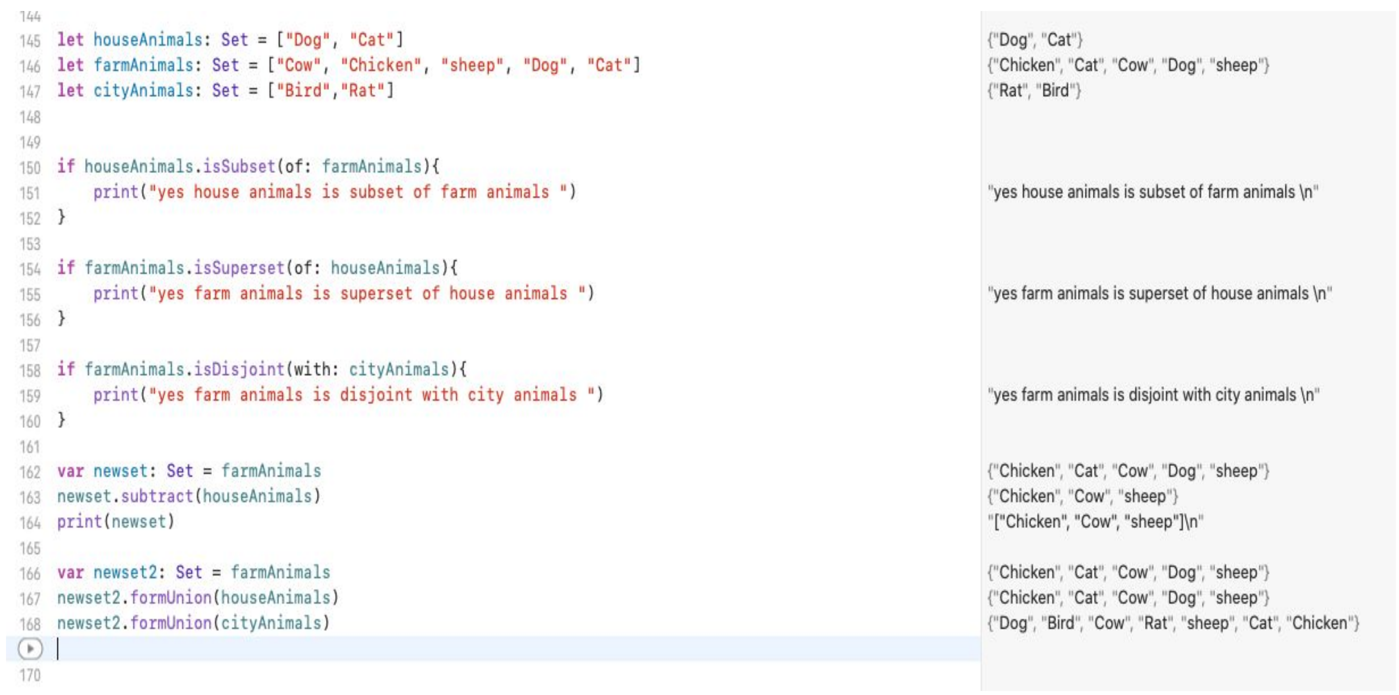
let houseAnimals: Set = ["🐶", "🐱"]

let farmAnimals: Set = ["🐮", "🐔", "🐑", "🐶", "🐱"]

let cityAnimals: Set = ["🐦", "🐭"]

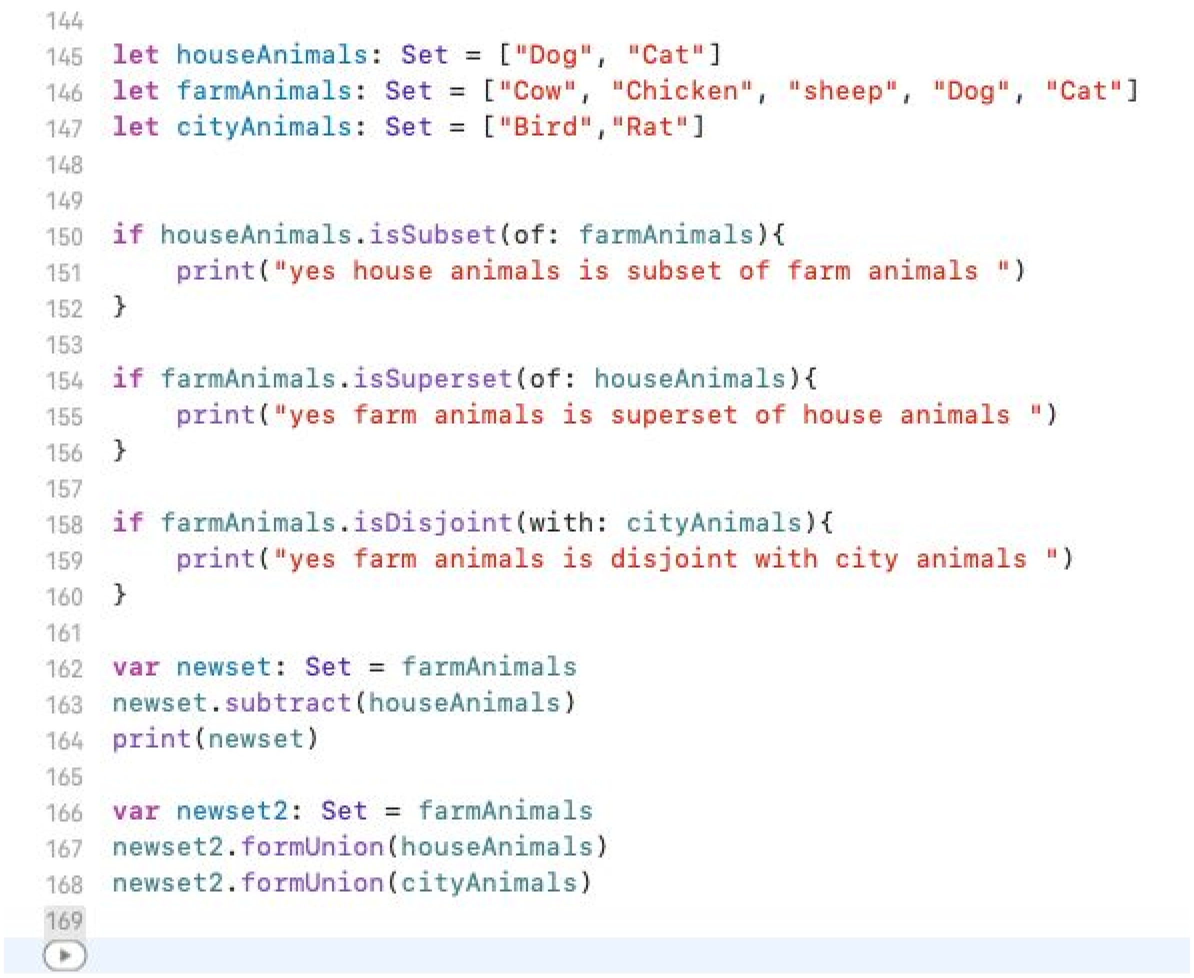
Use set operations to...

1. Determine whether the set of house animals is a subset of farm animals.
2. Determine whether the set of farm animals is a superset of house animals.
3. Determine if the set of farm animals is disjoint with city animals.
4. Create a set that only contains farm animals that are not also house animals.
5. Create a set that contains all the animals from all sets.



```swift
let houseAnimals: Set = ["Dog", "Cat"]
let farmAnimals: Set = ["Cow", "Chicken", "sheep", "Dog", "Cat"]
let cityAnimals: Set = ["Bird","Rat"]


if houseAnimals.isSubset(of: farmAnimals){
    print("yes house animals is subset of farm animals ")
}

if farmAnimals.isSuperset(of: houseAnimals){
    print("yes farm animals is superset of house animals ")
}

if farmAnimals.isDisjoint(with: cityAnimals){
    print("yes farm animals is disjoint with city animals ")
}

var newset: Set = farmAnimals
newset.subtract(houseAnimals)
print(newset)

var newset2: Set = farmAnimals
newset2.formUnion(houseAnimals)
newset2.formUnion(cityAnimals)
```

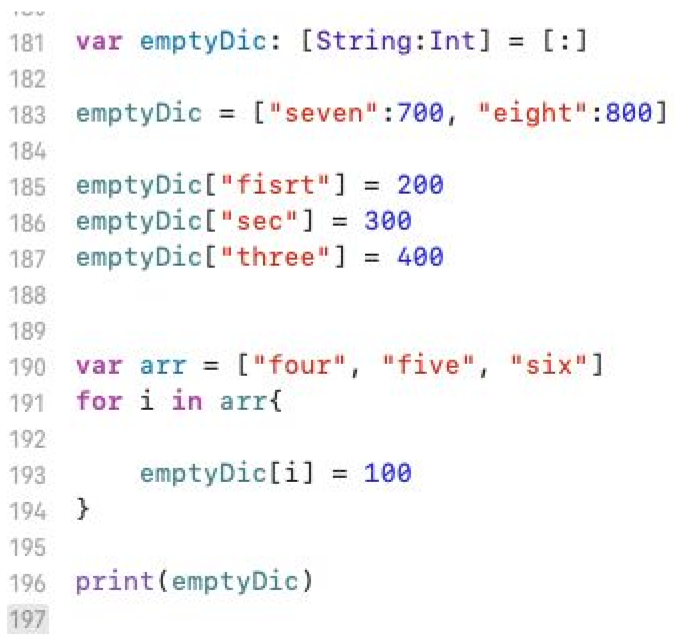
```
{"Dog", "Cat"}
{"Chicken", "Cat", "Cow", "Dog", "sheep"}
{"Rat", "Bird"}

"yes house animals is subset of farm animals \n"

"yes farm animals is superset of house animals \n"

"yes farm animals is disjoint with city animals \n"

{"Chicken", "Cat", "Cow", "Dog", "sheep"}
{"Chicken", "Cow", "sheep"}
"["Chicken", "Cow", "sheep"]\n"

{"Chicken", "Cat", "Cow", "Dog", "sheep"}
{"Chicken", "Cat", "Cow", "Dog", "sheep"}
{"Dog", "Bird", "Cow", "Rat", "sheep", "Cat", "Chicken"}
```



```swift

let houseAnimals: Set = ["Dog", "Cat"]
let farmAnimals: Set = ["Cow", "Chicken", "sheep", "Dog", "Cat"]
let cityAnimals: Set = ["Bird","Rat"]


if houseAnimals.isSubset(of: farmAnimals){
    print("yes house animals is subset of farm animals ")
}

if farmAnimals.isSuperset(of: houseAnimals){
    print("yes farm animals is superset of house animals ")
}

if farmAnimals.isDisjoint(with: cityAnimals){
    print("yes farm animals is disjoint with city animals ")
}

var newset: Set = farmAnimals
newset.subtract(houseAnimals)
print(newset)

var newset2: Set = farmAnimals
newset2.formUnion(houseAnimals)
newset2.formUnion(cityAnimals)

```

Dictionary

1. Create an empty dictionary with keys of type String and values of type Int and assign it to a variable in as many ways as you can think of (there's at least 4 ways).

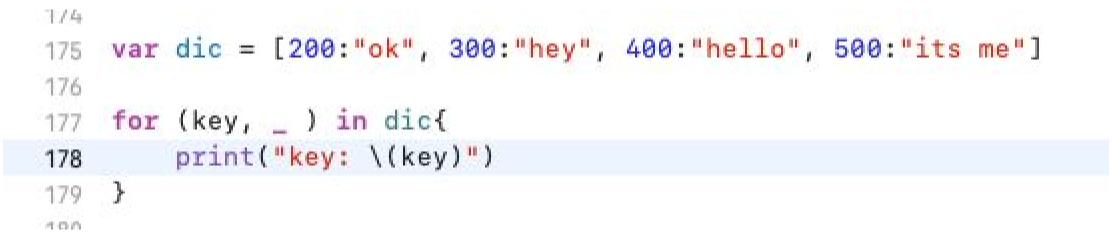


```
var emptyDic: [String:Int] = [:]

emptyDic = ["seven":700, "eight":800]

emptyDic["fisrt"] = 200
emptyDic["sec"] = 300
emptyDic["three"] = 400


var arr = ["four", "five", "six"]
for i in arr{

    emptyDic[i] = 100
}

print(emptyDic)
```

2. Create a mutable dictionary named secretIdentities where the key value pairs are "Hulk" -> "Bruce Banner", "Batman" -> "Bruce Wayne", and "Superman" -> "Clark Kent".

var secretIdentities: Dictionary = ["Hulk" : "Bruce Banner", "Batman" : "Bruce Wayne", "Superman" : "Clark Kent"]

3. Create a nesters structure of Key-value pair.
4. Print all the keys in the dic



```
var dic = [200:"ok", 300:"hey", 400:"hello", 500:"its me"]

for (key, _ ) in dic{
    print("key: \(key)")
}
```

Subscript

1. What is subscript ? Write down the declaration syntax.

Subscripts are used to access information from a collection, sequence and a list in Classes, Structures and Enumerations without using a method. Its used to store and retrieve the values with the help of index without the use of separate method.

Syntax :

```
 subscript (<parameters>) -> <return type> {

    // the getter is required
    get {
       // used for subscript value declarations
    }

    set(newValue) { // the setter is optional
       // definitions are written here
    }
}
```

2. Create a simple subscript that outputs true if a string contains a substring and false otherwise.