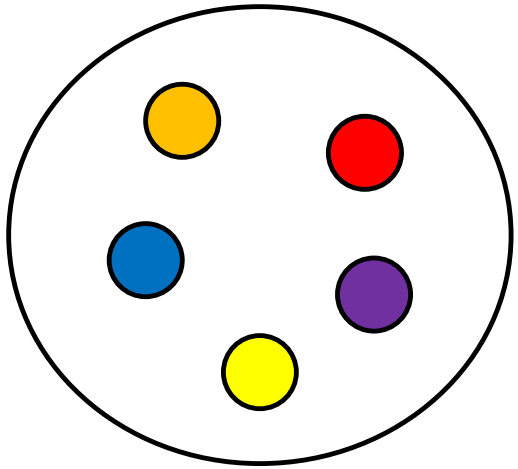
Sets and Maps

Kit Eason
www.kiteason.com
@kitlovesfsharp

Sets

- A collection of any type
- Only ever contains one copy of any value
- Must implement comparison
- ‘Adding’ means creating a new set with additional value
- ‘Adding’ a duplicate value will be ignored



Constructing a Set

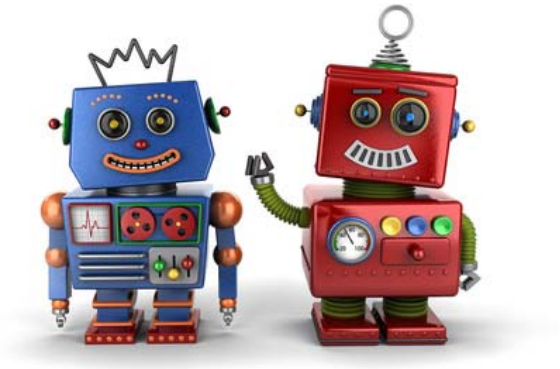
- **Call the Set constructor**
  - `let mySet = Set(mySeq)`
- **Provide a sequence, array, list to a Set.of… function**
  - `let mySet = mySeq |> Set.ofSeq`
  - `let mySet = myList |> Set.ofList`
  - `let mySet = myArray |> Set.ofArray`
- **Input collection doesn't need to be unique – set still will be**

Sets Stay Unique

- **This…**
  - `let myUniqueSeq = mySeq |> Seq.distinct`
  - `let mySet = Set(myUniqueSeq)`
- **…could simply be this:**
  - `let mySet = Set(mySeq)`
- **…or this**
  - `let mySet = mySeq |> Set.ofSeq`

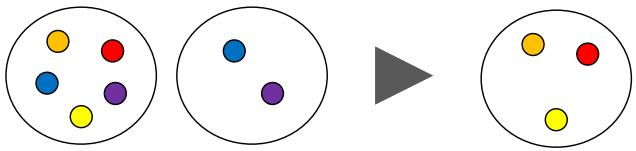
The Set Module

- **Set module has functions like Array, Seq and List modules**
  - Set.map
  - Set.iter
  - Set.filter
- **Set instances have some useful members**
  - mySet.Count
  - mySet.Contains
  - mySet.IsEmpty



The Set Module – Set Specifics

- Set.difference
- Set.union
- Set.unionMany
- Set.isSubset/Set.isProperSubset
- Set.isSuperSet/Set.isProperSuperset



Set.difference

- Can also use – (minus) operator

Set.union

- Can also use + operator

Set.unionMany

- Seq.reduce (+) |> Set.ofSeq

Set.isSubset

Set.isProperSubset

Set.isSuperset

Set.isProperSuperset

Hidden Mutability

- Accept a bit of mutability
- Hide it from callers
- Hide the mechanics

```
let TrackSomething =
    let internalSet = ref Set.empty
    (fun (argument : string) ->
        internalSet := Set.add argument !internalSet
        !internalSet)
```

Maps

- An immutable dictionary
- Create from a sequence of key-value pairs

```
let moons =
   Map([("Mercury",0); ("Venus",0); ("Earth",1); ("Mars",2)])
```

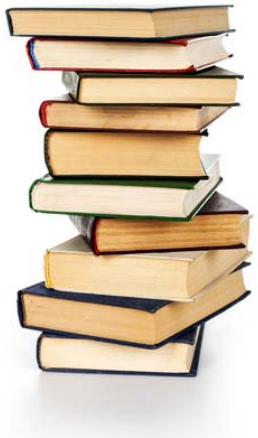
```
let moons =
   [("Mercury",0); ("Venus",0); ("Earth",1); ("Mars",2)]
   |> Map.ofSeq
```

- Look up values using square-brackets index

```
let ourMoons = moons.["Earth"]
```

- Functional-style adds

```
let moons' = moons.Add("Antichthon",1)
```

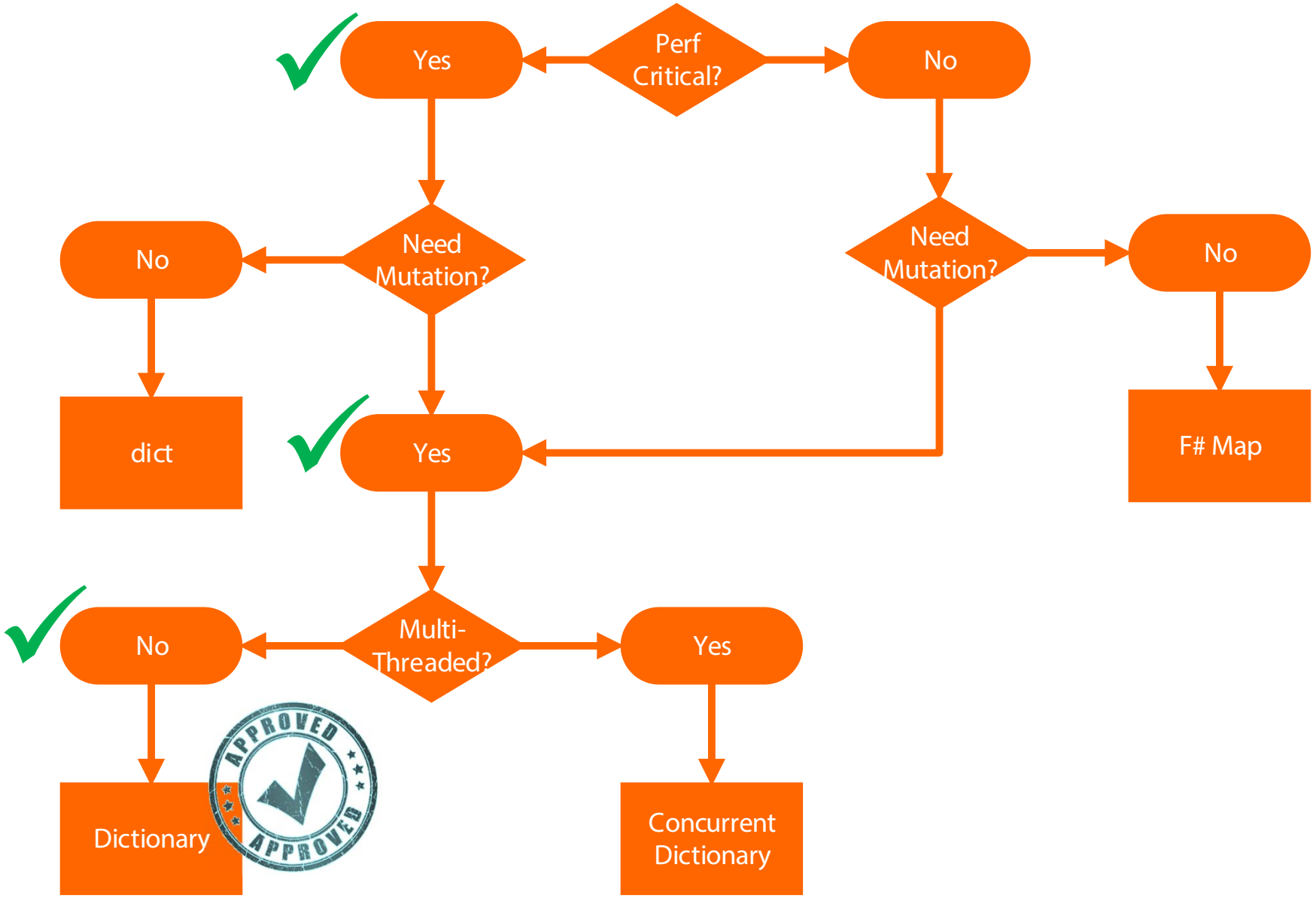


Which Lookup to Use?

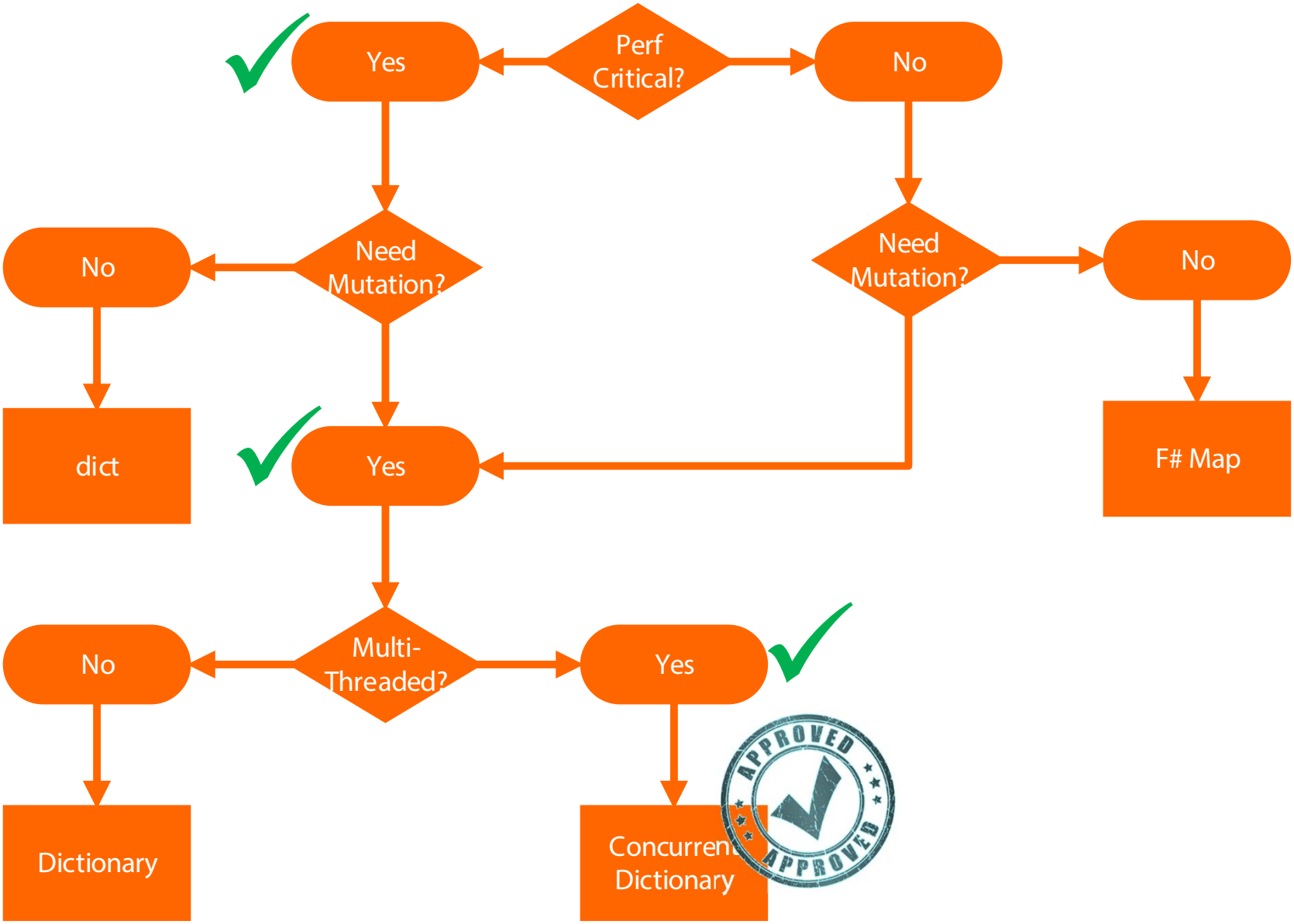
- .NET Dictionary
- .NET ConcurrentDictionary
- F# dict
- F# Map

- .NET Dictionary faster than F# Map
- …for creation and retrieval

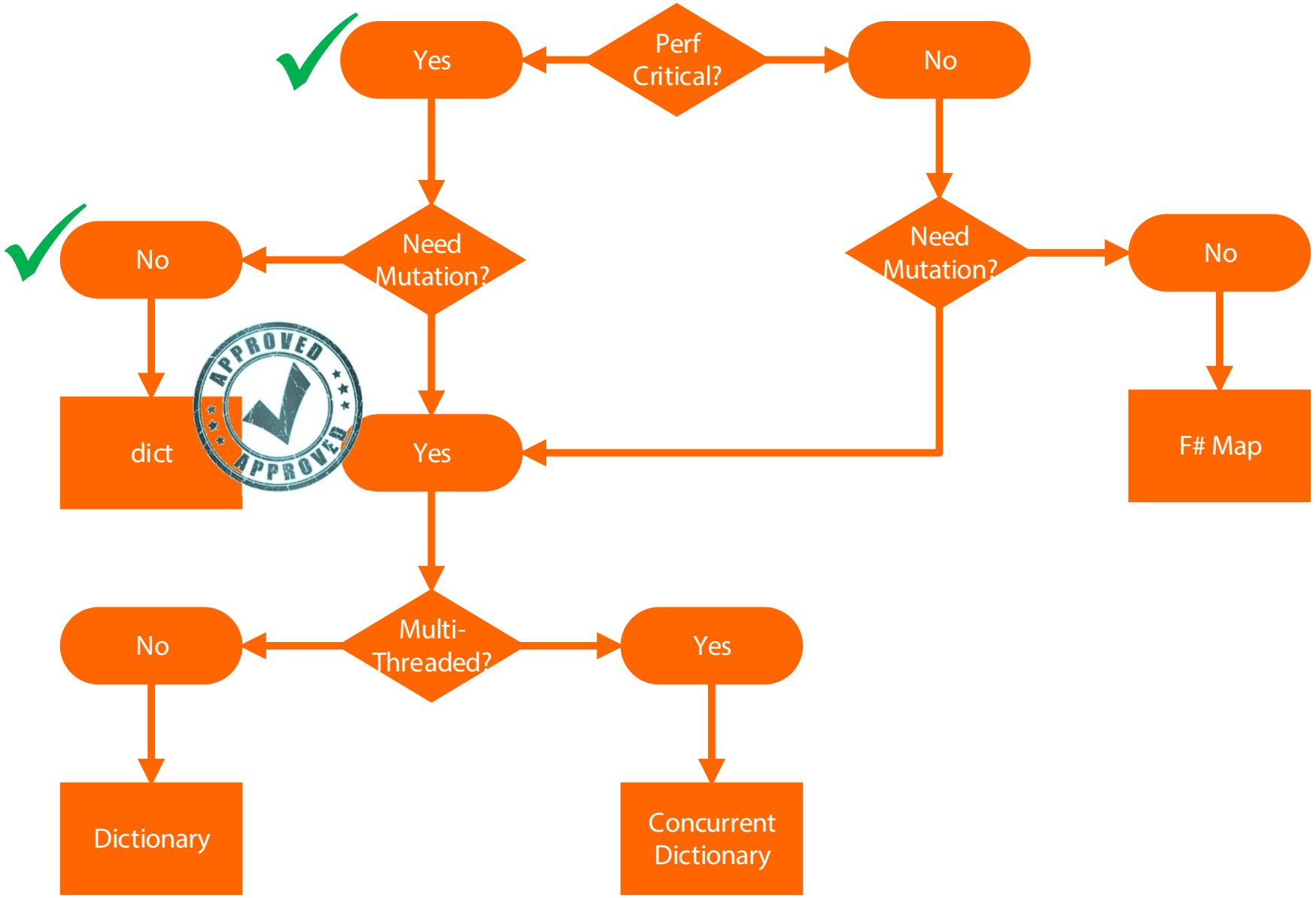
Which Lookup to Use?



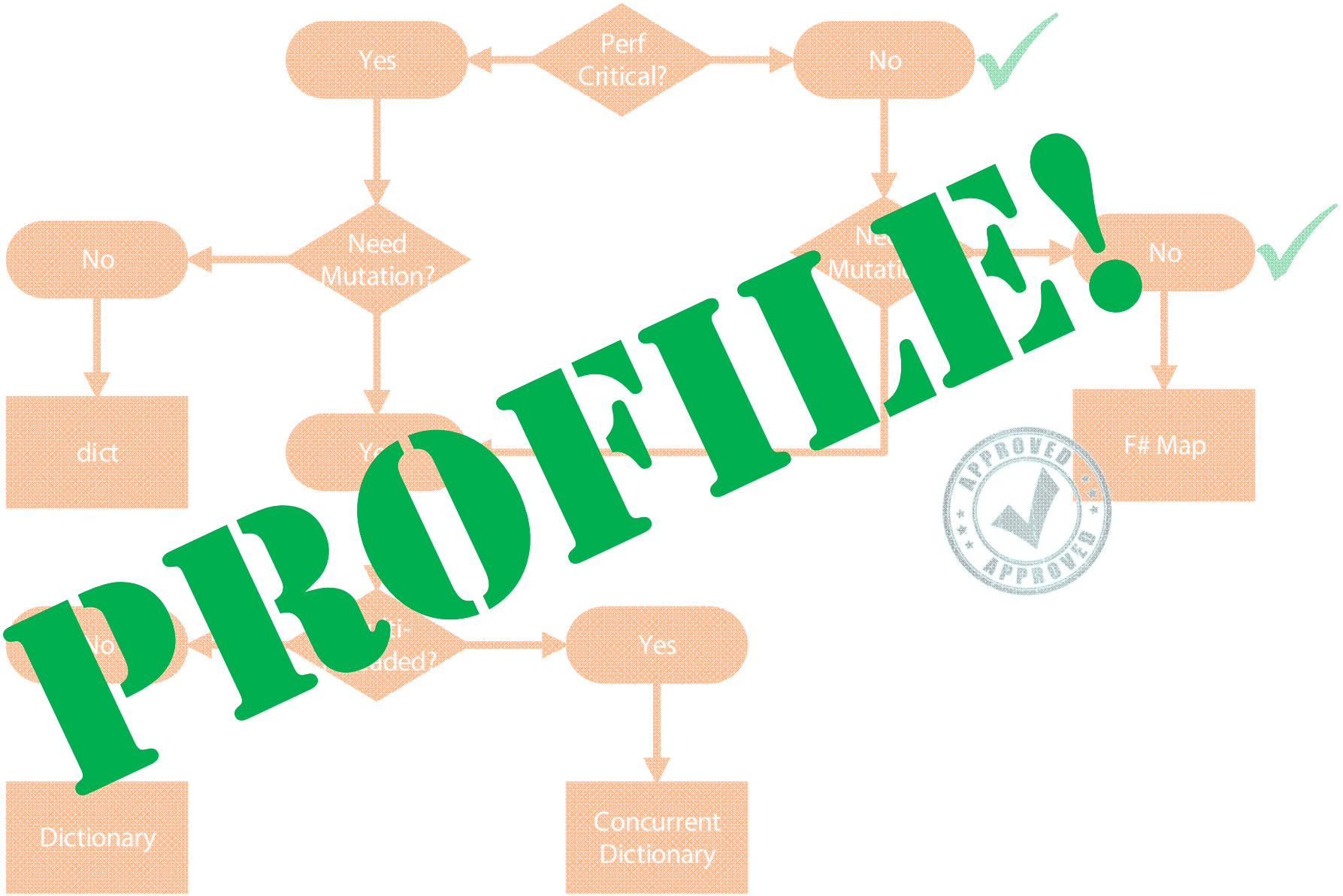
Which Lookup to Use?



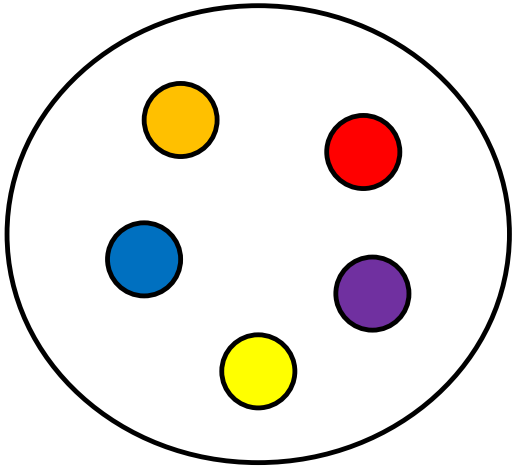
Which Lookup to Use?



Which Lookup to Use?



Summary - Sets



- **A unique collection**
- **Create with Set() or Set.of…**
  - Set.ofArray, Set.ofList, Set.ofSeq
- **No need to supply a *unique* collection**
- **Set.difference, Set.union, Set.isSubset, Set.isSuperset**
- **Hidden mutability**
  - Bind a value
  - Initialize Set in ref cell
  - Return a function which updates set and returns something

Summary - Maps

- **An immutable dictionary**
- **Create with Map() or Map.of…**
  - Map.ofArray, Map.ofList, Map.ofSeq
- **Add returns a new Map with element added**
- **Not as fast as Dictionary or dict**