```
var myVariable: Int          UInt
                String       Character
                Double
                Float
                Bool
```

Core Swift Data Types

```
var myVariable: Int
                String
                Double
                Float
                Bool
```

UInt

Character

Collection Types

Array

Dictionary

Set

Core Swift Data Types

```
var myVariable: _____________  // what else?
```

What about a *Date* type? Or a *Time* type?

Complex Data Types

```
var myVariable: _____________  // what else?
```

What about a *Date* type? Or a *Time* type?

***URL* / *File* / *Button* / *Document* / *Image* (etc.)**

Complex Data Types

```
// We'll import other frameworks for additional types
import _____________

var myVariable: _____________  // what else?
```

What about a *Date* type? Or a *Time* type?
***URL* / *File* / *Button* / *Document* / *Image* (etc.)**

Complex Data Types

Constants: Why They're Important

var to make a variable

```
var myVariableMessage = "Hello"
// later, try to change it
myVariableMessage = "Bye"
```

let to make a constant

```
let myConstantMessage = "Hello"
// later, try to change it
myConstantMessage = "Bye"
```

var to make a variable

```
var myVariableMessage = "Hello"
// later, try to change it
myVariableMessage = "Bye"      // OK
```

let to make a constant

```
let myConstantMessage = "Hello"
// later, try to change it
myConstantMessage = "Bye"     // ERROR!
```

Cannot assign to value: 'myConstantMessage' is a 'let' constant

```
const int minutesInAWeek = 10080;
```

Creating Constants in Other Languages

Additional keywords are often required

```
final int minutesInAWeek = 10080;
```

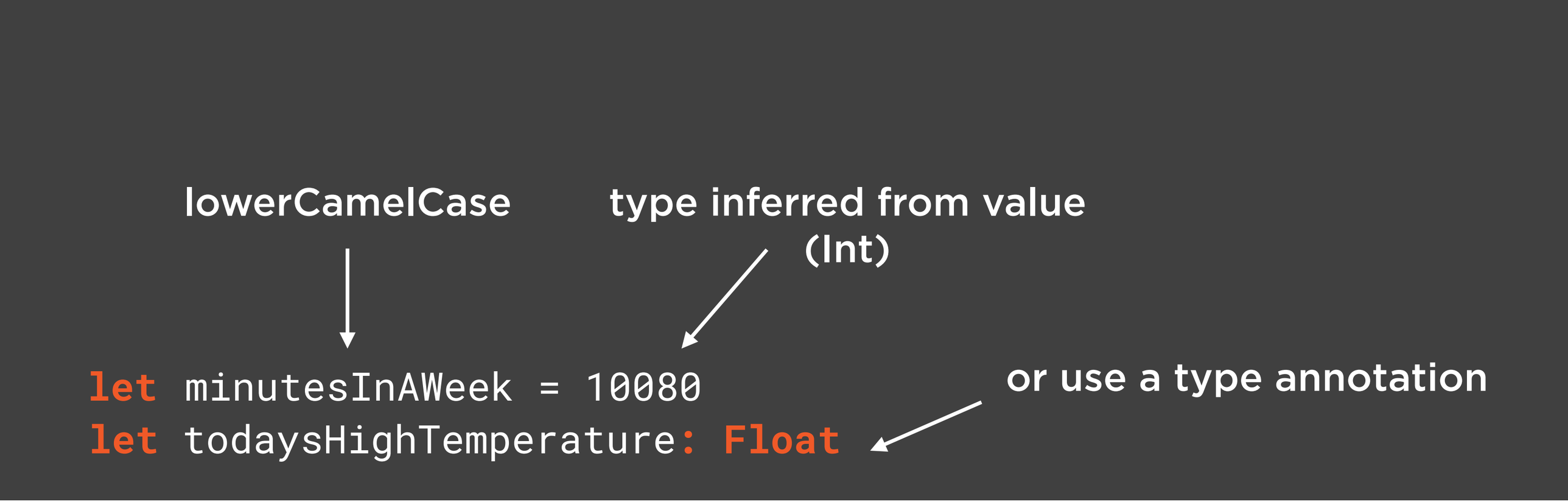
Creating Constants in Other Languages

Additional keywords are often required

```
readonly int minutesInAWeek = 10080;
```

Creating Constants in Other Languages

Additional keywords are often required



Creating Constants in Swift

```
// constants with fixed compile-time values
const float PI = 3.1415927;
const int MAX_WIDTH = 3840;
const int MAX_HEIGHT = 2160;
```

Constants aren't just

Additional keywords are often required

“But does this actually make a difference?”

Skeptical Programmer

```
// make an integer variable
int myInteger = 40;
// make a floating-point variable
double myDouble = 2.5;

// now add them together
Console.WriteLine(myInteger + myDouble);
> 42.5
```

Some Implicit Conversion (Coercion) is Common

Many languages perform some automatic "behind the scenes" conversion

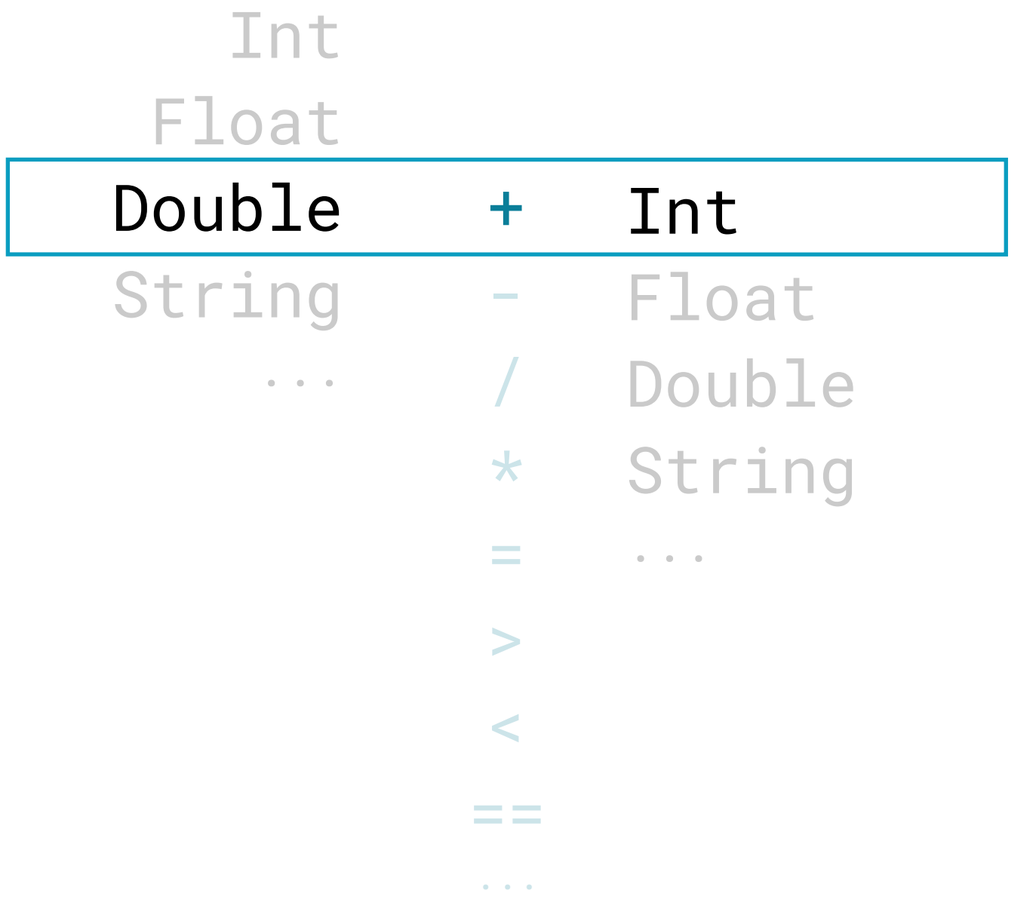
```
let numberOfEdges = 4                    // Int
let defaultTemperature = 72.6            // Double
let message = "The message is: "         // String
var registeredUser = true                // Bool
var firstInitial: Character = "S"        // Character

// etc..
```

Converting Data

When it works, when it doesn't... and when "we won't know until we try"

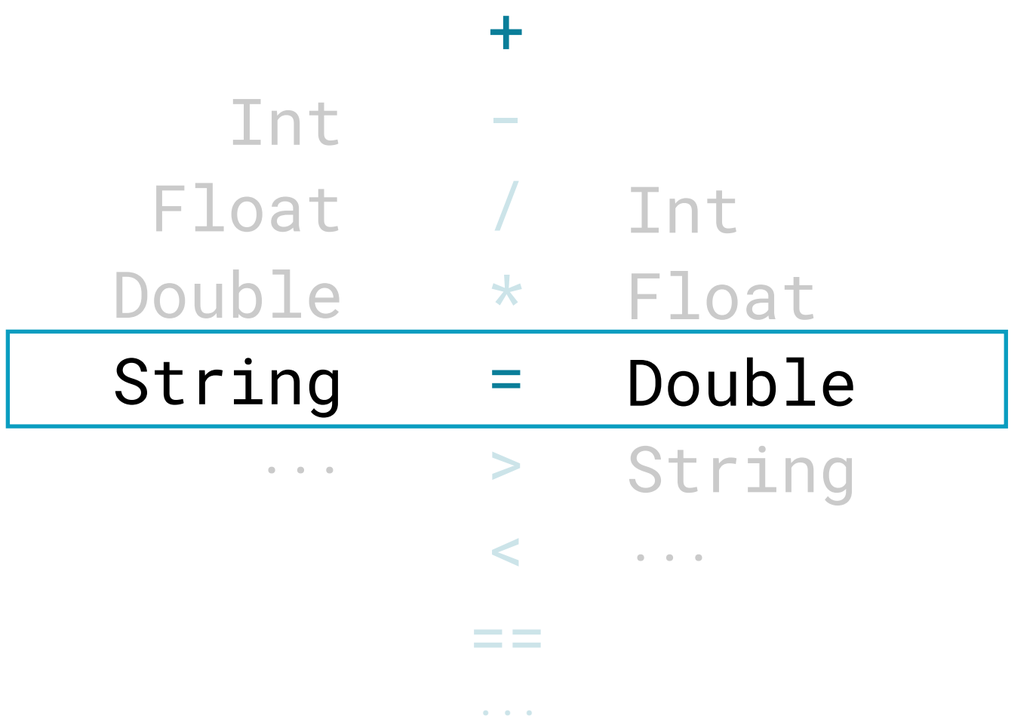
Interactions Between Types



Interactions Between Types

	+	Int
Int	-	Float
Float	/	Double
Double	*	String
String	=	...
...	>	
	<	
	==	
	...	

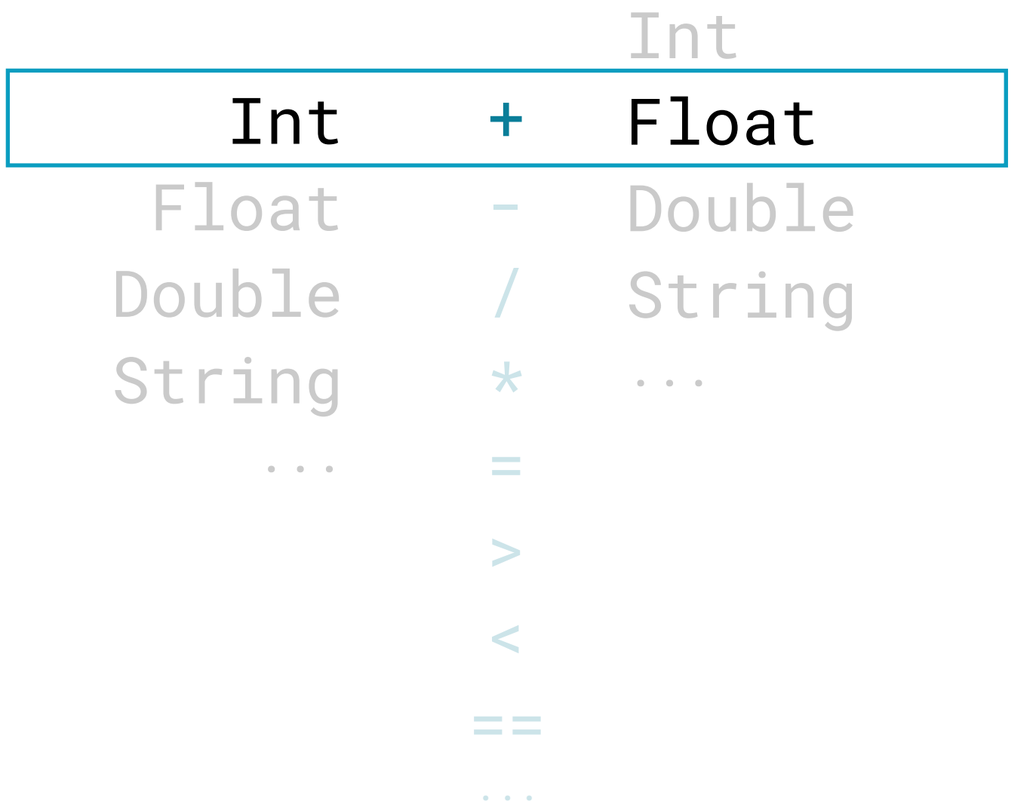
Interactions Between Types



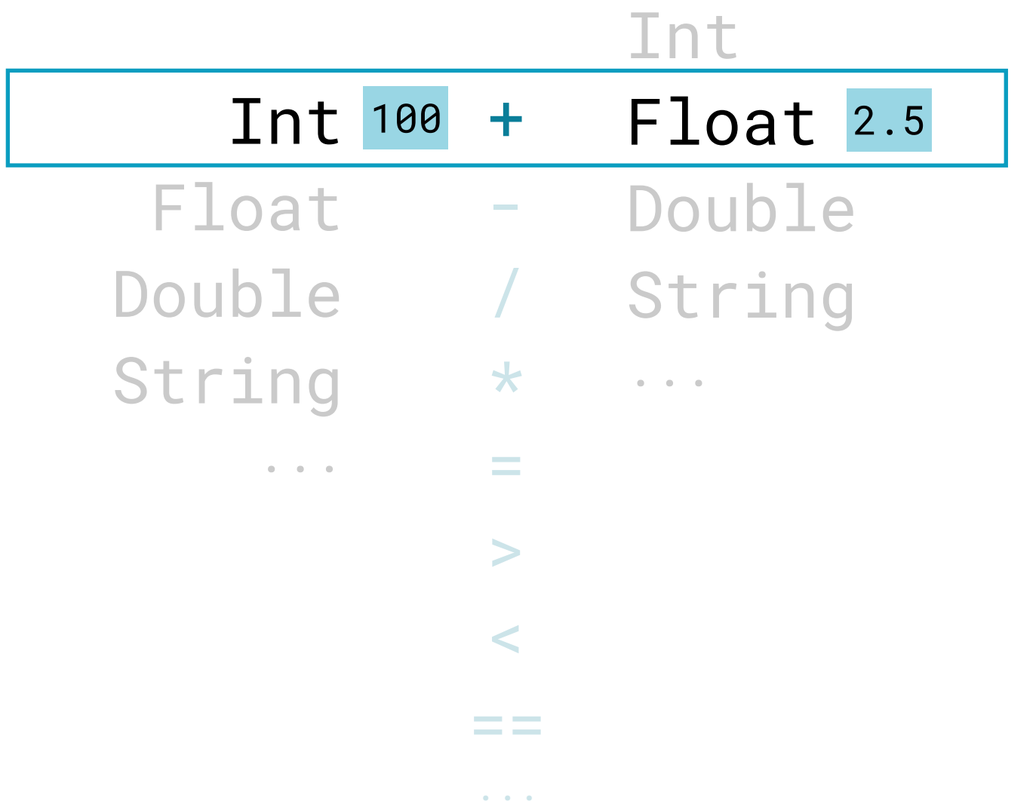
Interactions Between Types

	+	
Int	-	
Float	/	
Double	*	Int
String	=	Float
...	>	Double
	<	String
	==	...
	...	

Interactions Between Types



Interactions Between Types



as Ints = 102

as Floats = 102.5

```
// make a Float from an Int
let myFloat = Float(someInteger)

// make a String from a Float
let myString = String(myFloat)

// make a Double from a String
let myDouble = Double(myString)

// make an Int from a Float
let myInt = Int(myFloat)
```

Basic Conversion Syntax

```
// make a Bool
var hasLoggedIn = false

// attempt to convert to a Double?
let myDouble = Double(hasLoggedIn)
```

1/3: Not All Conversion Makes Sense

```
let myFloat: Float = 3.5

// convert to Int
let myInt = Int(myFloat)
print(myInt)

> 3
```

2/3: Conversions Can Still Lose Information

```
let myInt = 12345678912345678 9
```

3/3: Not All Conversions Succeed

Some conversions will always work...

```
let myInt = 12345678912345678 9

// an Int will always convert to a String
let myString = String(myInt)
print(myString)
```

3/3: Not All Conversions Succeed

Some conversions will always work...

```
let myInt = 12345678912345678 9

// an Int will always convert to a String
let myString = String(myInt)
print(myString)

> 123456789123456789
```

3/3: Not All Conversions Succeed

Some conversions will always work...

3/3: Not All Conversions Succeed

Other conversions may or may not work depending on the value

```
let myString = "21"
```

3/3: Not All Conversions Succeed

Other conversions may or may not work depending on the value

```
// this String will convert to an Int value
let myString = "21"


// ...
let myResult = Int(myString)
```

3/3: Not All Conversions Succeed

Other conversions may or may not work depending on the value

```
// this String will convert to an Int value
let myString = "21"


// ...
let myResult = Int(myString)
```

3/3: Not All Conversions Succeed

Other conversions may or may not work depending on the value

```
// ...but this String will not convert to an Int
let myString =


// ...
let myResult = Int(myString)
```

3/3: Not All Conversions Succeed

Other conversions may or may not work depending on the value

```
// ...but this String will not convert to an Int
let myString = "lkjhg"


// ...
let myResult = Int(myString)
```

3/3: Not All Conversions Succeed

Other conversions may or may not work depending on the value

```
// ...but this String will not convert to an Int
let myString = "lkjhg"


// ...
let myResult = Int(myString)
```

3/3: Not All Conversions Succeed

Other conversions may or may not work depending on the value

```
// ...but this String will not convert to an Int
let myString = "lkjhg"


// ...
let myResult = Int(myString)
```

so the result will be either an **Int**... or nothing.

3/3: Not All Conversions Succeed

Other conversions may or may not work depending on the value

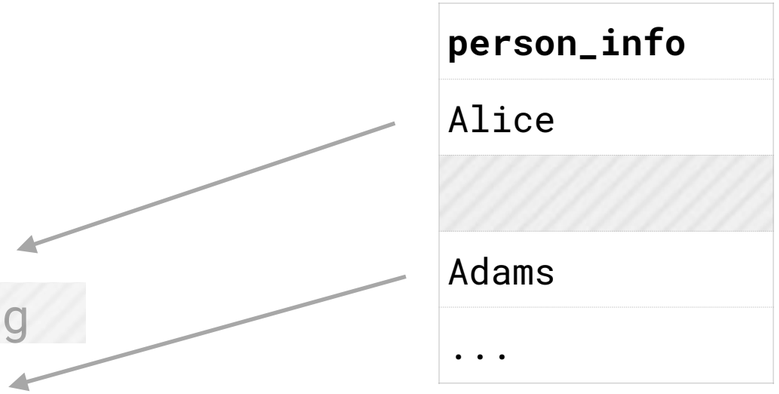
Well-Defined Data Doesn't Ensure a Value

```
// ...
let firstName: String
let middleName: String
let lastName: String
let age: Int
let avatar: Image
// ...
```

person_info
Alice
Adams
...

Well-Defined Data Doesn't Ensure a Value

```
// ...
let firstName: String
let middleName: String
let lastName: String
let age: Int
let avatar: Image
// ...
```



Well-Defined Data Doesn't Ensure a Value

```
// ...
let firstName: String
let middleName: String
let lastName: String
let age: Int
let avatar: Image
// ...
avatar = fetchUserAvatar()
```

person_info
Alice
Adams
...

Well-Defined Data Doesn't Ensure a Value

```
// ...
let firstName: String
let middleName: String
let lastName: String
let age: Int
let avatar: Image
// ...
avatar = fetchUserAvatar()
```

person_info
Alice
Adams
...

```
// ...but this String will not convert to an Int
let myString = "lkjhg"


// ...
let myResult = Int(myString)
```

3/3: Not All Conversions Succeed

Other conversions may or may not work depending on the value

```
string myString;
int myInteger;
bool myBoolean;
float myfloat;
// etc.
```

Default Values Are Common in Other Languages

```
string myString;   // set to ""
int myInteger;     // set to 0
bool myBoolean;    // set to false
float myfloat;     // set to 0.0
// etc.
```

Default Values Are Common in Other Languages

The Problem of Default / Placeholder Values

The Problem of Default / Placeholder Values

```
// Traveler information

var firstName: String
var middleName: String
var lastName: String

var email: String
var secondaryEmail: String

var daysUntilNextTrip: Int

// ...
```

The Problem of Default / Placeholder Values

```
// Traveler information

var firstName: String      ..............  "Grace"
var middleName: String
var lastName: String

var email: String
var secondaryEmail: String

var daysUntilNextTrip: Int

// ...
```

The Problem of Default / Placeholder Values

```
// Traveler information

var firstName: String      ..............  "Grace"
var middleName: String
var lastName: String       ..............  "Murray"

var email: String
var secondaryEmail: String

var daysUntilNextTrip: Int

// ...
```

The Problem of Default / Placeholder Values

```
// Traveler information

var firstName: String      ..............  "Grace"
var middleName: String     ..............  ""
var lastName: String       ..............  "Murray"

var email: String
var secondaryEmail: String

var daysUntilNextTrip: Int

// ...
```

The Problem of Default / Placeholder Values

```
// Traveler information

var firstName: String ..... "Grace"
var middleName: String ..... ""
var lastName: String ..... "Murray"

var email: String
var secondaryEmail: String

var daysUntilNextTrip: Int

// ...
```

The Problem of Default / Placeholder Values

```
// Traveler information

var firstName: String ..... "Grace"
var middleName: String ..... ""
var lastName: String ..... "Murray"

var email: String
var secondaryEmail: String

var daysUntilNextTrip: Int

// ...
```

The Problem of Default / Placeholder Values

```
// Traveler information

var firstName: String ..... "Grace"
var middleName: String ..... ""
var lastName: String ..... "Murray"

var email: String
var secondaryEmail: String

var daysUntilNextTrip: Int ..... 0

// ...
```

The Problem of Default / Placeholder Values

```
// Traveler information

var firstName: String ..... "Grace"
var middleName: String ..... ""
var lastName: String ..... "Murray"

var email: String
var secondaryEmail: String

var daysUntilNextTrip: Int ..... 0

// ...
```

so the next trip is today?

The Problem of Default / Placeholder Values

```
// Traveler information

var firstName: String ..... "Grace"
var middleName: String ..... ""
var lastName: String ..... "Murray"

var email: String
var secondaryEmail: String

var daysUntilNextTrip: Int ..... 0

// ...
```

so the next trip is today?

or... we haven't calculated it yet?

The Problem of Default / Placeholder Values

```
// Traveler information

var firstName: String ..... "Grace"
var middleName: String ..... ""
var lastName: String ..... "Murray"

var email: String
var secondaryEmail: String

var daysUntilNextTrip: Int ..... 0

// ...
```

so the next trip is today?

or... we haven't calculated it yet?

or... there is no next trip?

Swift Optionals

How to define type-safe values
when there might be no value at all

Optional Values Are Type-Safe

But if Defined as Optional, May Have No Value at All

```
var daysUntilNextTrip: Int    // an Int
```

Optional Values Are Type-Safe

But if Defined as Optional, May Have No Value at All

```
var daysUntilNextTrip: Int?  // an Optional Int
```

Optional Values Are Type-Safe

But if Defined as Optional, May Have No Value at All

```
var daysUntilNextTrip: Int?   // an Optional Int
var secondaryEmail: String    // a String
```

Optional Values Are Type-Safe

But if Defined as Optional, May Have No Value at All

```
var daysUntilNextTrip: Int?   // an Optional Int
var secondaryEmail: String?   // an Optional String
```

Optional Values Are Type-Safe

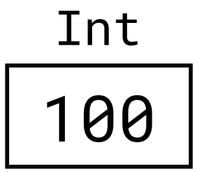
But if Defined as Optional, May Have No Value at All

nil

Using Optionals

Optionals

Optionals



Optionals

`myOptional`

Optionals

`myOptional`

Optionals

nil ?

`myOptional`

Optionals

nil

or not nil ?

`myOptional`

Optionals

nil ?

or **not nil**

(there is a value)

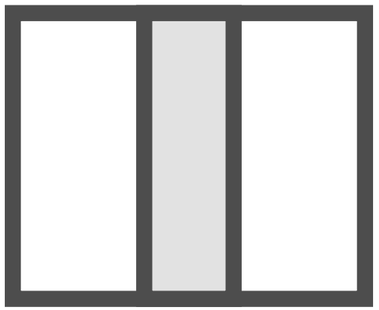
`myOptional`

Optionals

nil ?

or **not nil**

(there is a value)



myOptional

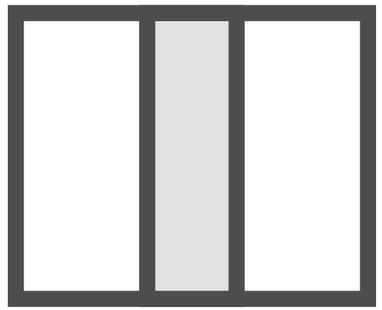
unwrapping
an optional

Optionals

nil ?

or **not nil**

(there is a value)



myOptional

Int

100

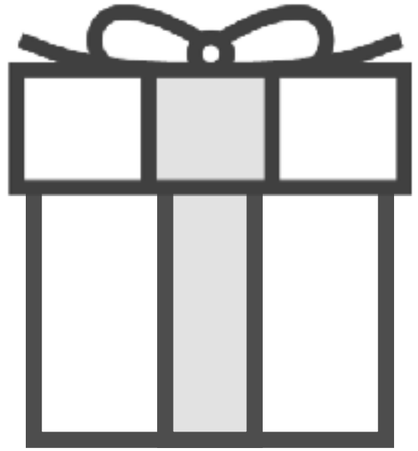
unwrapping
an optional

Optionals

nil

or **not nil**

(there is a value)



myOptional

Int

100

unwrapping
an optional

Creating and Using Swift Arrays

```
let firstMode = "Ionian"
let secondMode = "Dorian"
let thirdMode = "Phrygian"
// etc.
```

Creating and Using Swift Arrays

`musicalModes`

Creating and Using Swift Arrays

Swift Collections

Array **Dictionary** **Set**

Swift Collections

0	"Ionian"
1	"Dorian"
2	"Phrygian"

Array

an **ordered collection** of items

Dictionary

Set

Swift Collections

0	"Ionian"
1	"Dorian"
2	"Phrygian"

Array

an **ordered collection** of items

"UPS"	"United Parcel Service"
"FedEx"	"Federal Express"
"USPS"	"United States Postal Service"

Dictionary

a collection of **key/value pairs**

Set

Swift Collections

0	"Ionian"
1	"Dorian"
2	"Phrygian"

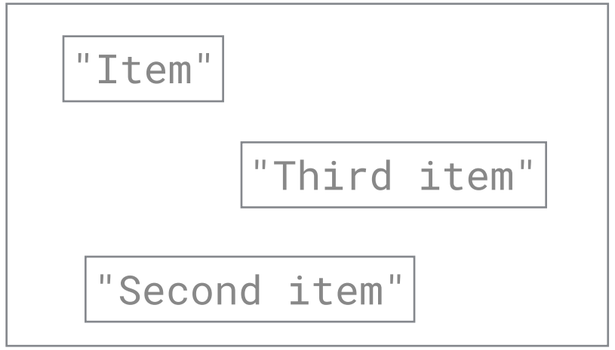
Array

an **ordered collection** of items

"UPS"	"United Parcel Service"
"FedEx"	"Federal Express"
"USPS"	"United States Postal Service"

Dictionary

a collection of **key/value pairs**



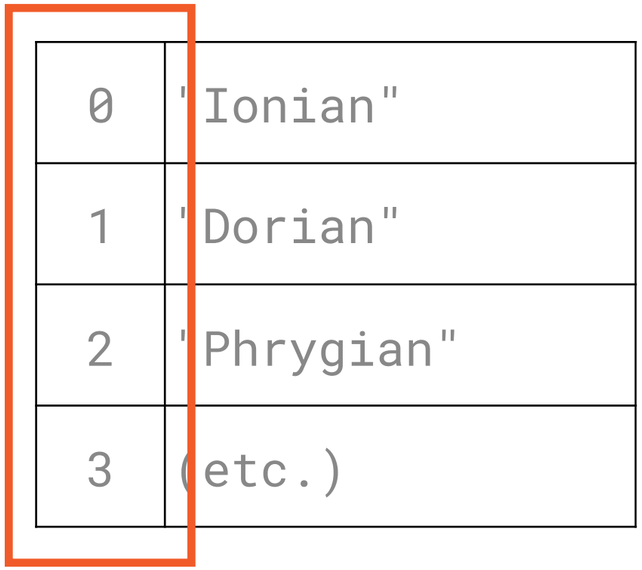
Set

an **unordered collection** of items

Swift Arrays

0	"Ionian"
1	"Dorian"
2	"Phrygian"
3	(etc.)

Swift Arrays



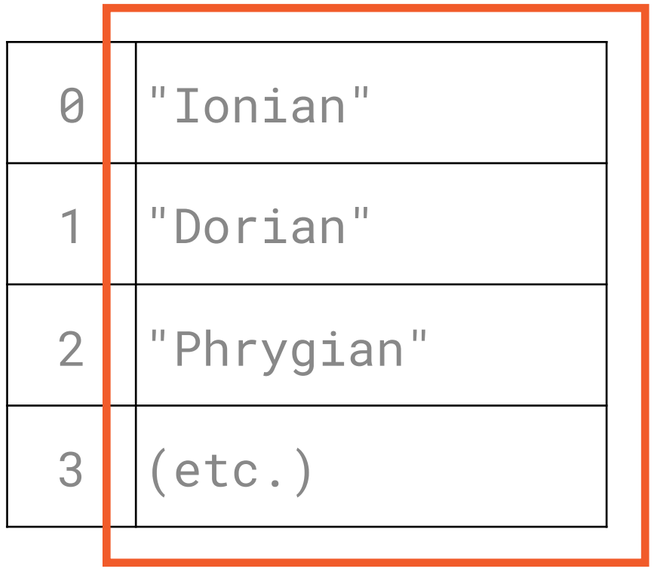
Zero-based

Swift Arrays

0	"Ionian"
1	"Dorian"
2	"Phrygian"
3	(etc.)

Zero-based

Swift Arrays



Zero-based

Type-safe

Swift Arrays

0	"Ionian"
1	"Dorian"
2	"Phrygian"
3	(etc.)

Zero-based

Type-safe

Swift Arrays

0	"Ionian"
1	"Dorian"
2	"Phrygian"
3	(etc.)

Zero-based

Type-safe

Mutable or **Immutable**
(using var) **(using let)**