

Unit 1 — Lesson 2:

Constants, Variables, and Data Types

Constants and variables

Associate a name with a value

Defining a constant or variable

- Allocates storage for the value in memory
- Associate the constant name with the assigned value

Constants

Defined using the `let` keyword

```
let name = "John"
```

Defined using the `let` keyword

```
let pi = 3.14159
```

Can't assign a constant a new value

```
let name = "John"  
name = "James"
```



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

Variables

Defined using the `var` keyword

```
var age = 29
```

Can assign a new value to a variable

```
var age = 29
```

```
age = 30
```

```
let defaultScore = 100
var playerOneScore = defaultScore
var playerTwoScore = defaultScore

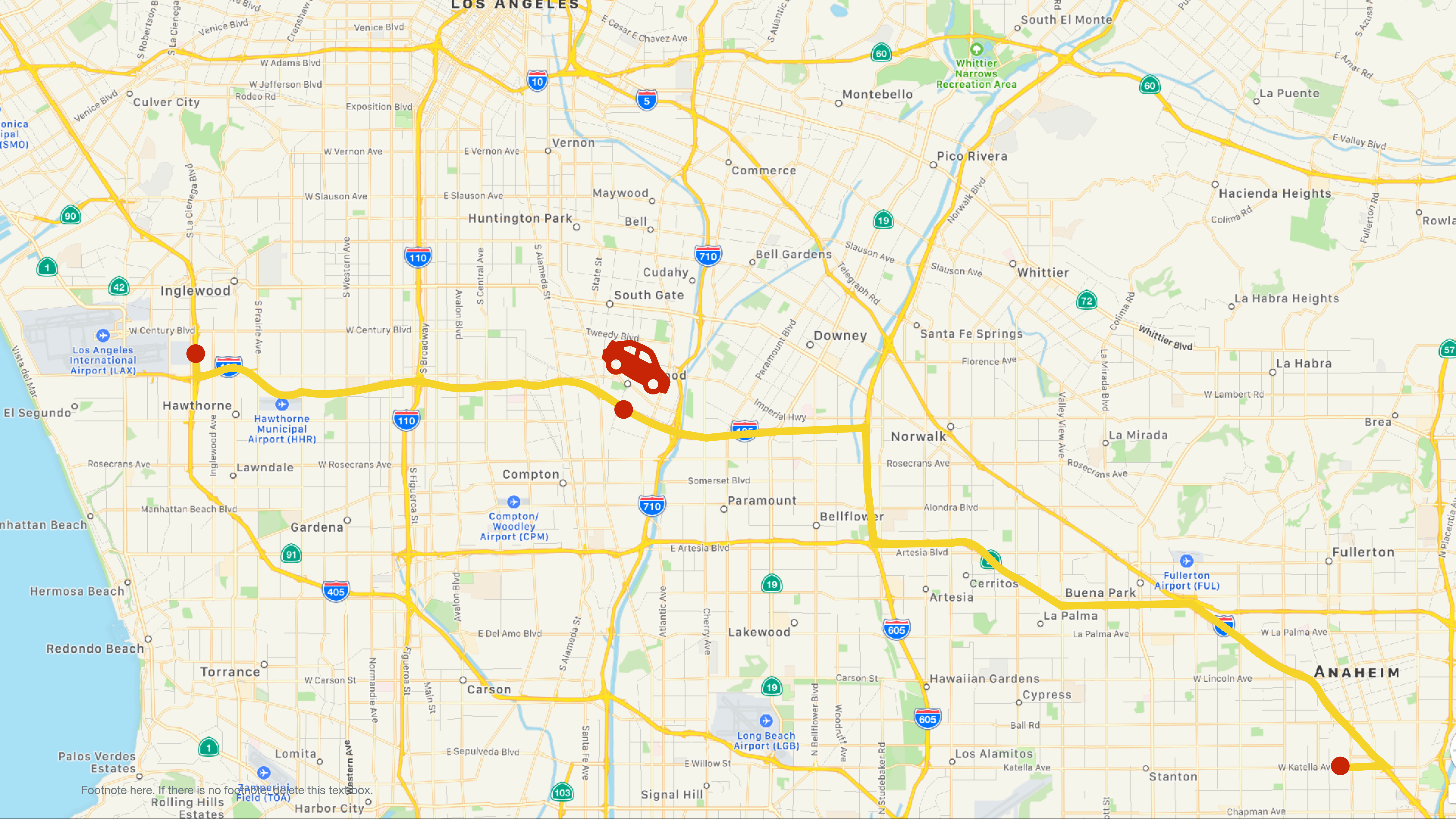
print(playerOneScore)
print(playerTwoScore)

playerOneScore = 200
print(playerOneScore)
```

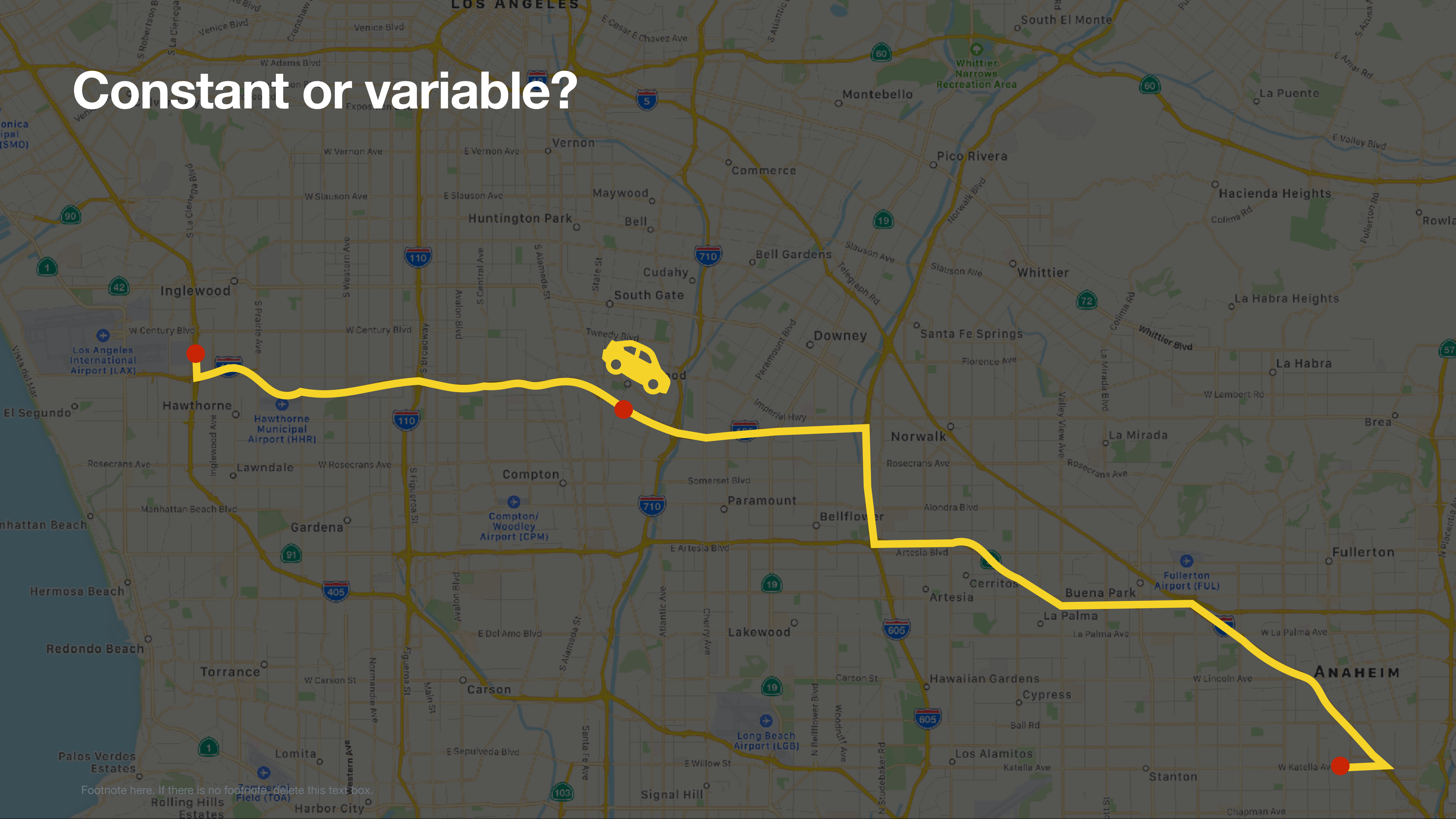
100

100

200



Constant or variable?



Constant or variable?

Starting location



Constant or variable?

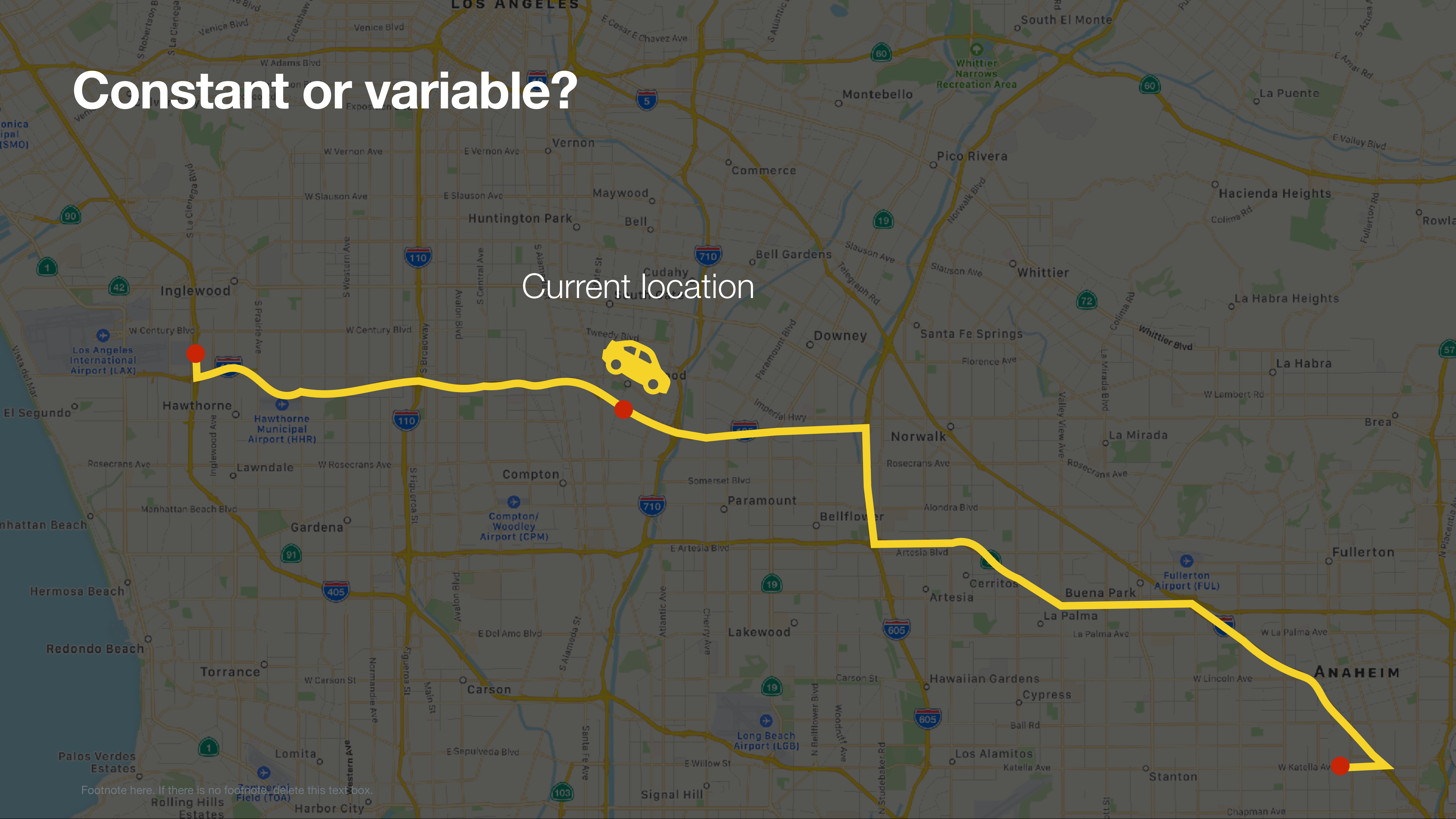


Destination

Footnote here. If there is no footnote delete this text box.

Constant or variable?

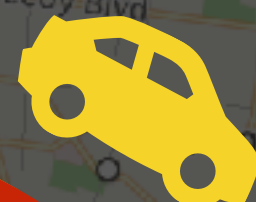
Current location



Footnote here. If there is no footnote delete this text box.

Constant or variable?

Distance traveled



Constant or variable?



Remaining distance

Footnote here. If there is no footnote delete this text box.

Naming constants and variables

Rules

No mathematical symbols

No spaces

Can't begin with a number

```
let π = 3.14159
let 一百 = 100
let 🎲 = 6
let mañana = "Tomorrow"
let anzahlDerBücher = 15 //numberOfBooks
```

Naming constants and variables

Best practices

1. Be clear and descriptive

`n`

`firstName`

2. Use camel case when multiple words in a name

`firstname`

`firstName`

Naming constants and variables

Best practices

1. Be clear and descriptive

✗ `n`

✓ `firstName`

2. Use camel case when multiple words in a name

`firstname`

`firstName`

Naming constants and variables

Best practices

1. Be clear and descriptive

✗ `n`

✓ `firstName`

2. Use camel case when multiple words in a name

✗ `firstname`

✓ `firstName`

Comments

```
// Setting pi to a rough estimate
```

```
let  $\pi$  = 3.14
```

```
/* The digits of pi are infinite,  
so instead I chose a close approximation.*/
```

```
let  $\pi$  = 3.14
```

Types

```
struct Person {  
    let firstName: String  
    let lastName: String  
  
    func sayHello() {  
        print("Hello there! My name is \(firstName) \(lastName).")  
    }  
}
```



```
struct Person {  
    let firstName: String  
    let lastName: String  
  
    func sayHello() {  
        print("Hello there! My name is \(firstName) \(lastName).")  
    }  
}  
  
let aPerson = Person(firstName: "Jacob", lastName: "Edwards")  
let anotherPerson = Person(firstName: "Candace", lastName: "Salinas")  
  
aPerson.sayHello()  
anotherPerson.sayHello()
```

Hello there! My name is Jacob Edwards.

Hello there! My name is Candace Salinas.

Most common types

	Symbol	Purpose	Example
Integer	<code>Int</code>	Represents whole numbers	<code>4</code>
Double	<code>Double</code>	Represents numbers requiring decimal points	<code>13.45</code>
Boolean	<code>Bool</code>	Represents true or false values	<code>true</code>
String	<code>String</code>	Represents text	<code>"Once upon a time..."</code>

Type safety

```
let playerName = "Julian"  
var playerScore = 1000  
var gameOver = false  
playerScore = playerName
```



Cannot assign value of type 'String' to type 'Int'

```
var wholeNumber = 30  
var numberWithDecimals = 17.5  
wholeNumber = numberWithDecimals
```



Cannot assign value of type 'Double' to type 'Int'

Type inference

```
let cityName = "San Francisco"  
let pi = 3.1415927
```

Type annotation

```
let cityName: String = "San Francisco"  
let pi: Double = 3.1415927
```

```
let number: Double = 3  
print(number)
```

3.0

Type annotation

Three common cases

1. When you create a constant or variable before assigning it a value

```
let firstName: String  
//...  
firstName = "Layne"
```


Type annotation

Three common cases

2. When you create a constant or variable that could be inferred as two or more different types

```
let middleInitial: Character = "J"  
var remainingDistance: Float = 30
```

Type annotation

Three common cases

3. When you add properties to a type definition

```
struct Car {  
    let make: String  
    let model: String  
    let year: Int  
}
```


Required values

```
var x
```




Type annotation missing in pattern

Required values

```
var x: Int
```


Required values

```
var x: Int  
print(x)
```

 Variable 'x' used before being initialized

Required values

```
var x: Int
```

```
x = 10
```

```
print(x)
```

10

Numeric literal formatting

```
var largeUglyNumber = 1000000000  
var largePrettyNumber = 1_000_000_000
```

Unit 1 — Lesson 2

Lab: Constants and Variables.playground



Open and complete the exercises in Lab – Constants and Variables.playground

