1. The Basics

Constants and Variables

Exercise 1.1

Declare a floating-point variable that represents temperature and assign the current temperature to it. If you don't have a thermometer, guess the temperature. :-)
Then assign a different temperature to it. Pick one at random, it doesn't matter.

Exercise 1.2

Declare a constant integer value that represents the number of seconds in an hour and assign that number to it on the same line.
Then try to assign a different value to the constant. Why doesn't it work?

Exercise 1.3

Declare an integer variable with an explicit type annotation and then one without.

Exercise 1.4

Declare a constant integer value that represents the number of wheels of a car. *Don't* assign it a value on the same line.

On the next line, assign a value to the constant. Why does this work?

Exercise 1.5

Declare the constant π using the name π, which you can type by pressing `alt-p`.

Exercise 1.6

Declare a variable using an emoji in the name.

Exercise 1.7

Print a variable using the `print()` function.

Exercise 1.8

What is the maximum value that can be stored in an Int16? Write the code that prints the maximum value of the Int16 type.

Exercise 1.9

What type is the constant pi in the example below? Why?

```
let pi = 3 + 0.141592654
```

Exercise 1.10

What happens if you try the following code? Why?

```
let myNumber: UInt = -17
```

Exercise 1.11

What happens if you try the following code? Why?

```
let bigNumber: Int16 = 32767 + 1
```

Exercise 1.12

Why does the following code not work? What do you need to add to it to make it work, if we absolutely want to store this value as an integer, i.e. 3, but we don't want to change the type of the variables?

```
let pi = 3.141592654
let approximatePi: Int = pi
```

Exercise 1.13 EXTRA CREDIT

The following code will print `true`, which means that valueA and valueB are equal. Why are they equal? (The << is the bitshift left operator.)

```
let valueA: Int16 = -0x8000
let valueB: Int16 = 0x4000 << 1

print(valueA == valueB)
```

Comments

Exercise 1.14

There are two types of comments. Single-line and multiline comments. Write one of both.

Exercise 1.15

In Swift multiline comments can be nested. Write a nested multiline comment.

Semicolons

This is not really an exercise, but try to avoid using semicolons. If you have to use a semicolon, you are probably doing something wrong. :-) It's generally preferred code style to write each statement on its own line.

Tuples

Exercise 1.16

Assign a tuple with two values in it to a constant named `player`. The values could represent the number of a hockey player and the name of the hockey player. For example, Igor Larionov whose number used to be 8.

Exercise 1.17

OK, now you have a `player` tuple. Decompose (i.e. split) the number and the name into two constants named `number`and `name`. This could be done in at least three ways. Try all three.

Optionals

Exercise 1.18

Can a constant have an optional type? If you're not sure, try it and see what happens.

Exercise 1.19

Why doesn't this work? What needs to be added to `value` on the second line for this to work?

```
let value: Int? = 17
let banana: Int = value
```

Exercise 1.20

If `value` in the previous exercise had been nil, what would have happened if you force unwrapped `value`?

```
let value: Int? = nil
let banana: Int = value!
```

Exercise 1.21

What would be a better way to assign `value` to the `banana` constant?