

Summary – Data Types, Assignment, and Tracing

The four kinds of value we have met, how to tell them apart by how they are written, what the arrow and = really do, and how to trace a few lines of code.

Four kinds of values

Every value a program keeps is one of a few kinds. Swift calls a kind a **type**. Here are the four from the grid, and how to recognise each one by how it is written.

Value	Kind	Swift type	How you can tell
7	a whole number	<code>Int</code>	digits only, no decimal point
7.5	a number with a decimal point	<code>Double</code>	it has a decimal point
<code>"7"</code>	text	<code>String</code>	it is in quotation marks
<code>false</code>	a yes-or-no answer	<code>Bool</code>	it is <code>true</code> or <code>false</code>

Quotation marks change everything. 7 is a number, so 7 + 7 is 14. `"7"` is text, so `"7" + "7"` is `"77"`: text is joined, not added.

Assignment

A line that stores a value in a name is an **assignment statement**, or just an assignment. The AP exam writes it with an arrow. Swift writes it with =, the **assignment operator**. They mean the same thing: **work out the value on the right, then store it in the name on the left.**

```
a ← 5
```

```
var a: Int = 5
```

Neither one means “equals” in the math sense. That is why this line makes sense to a computer, even though it is nonsense as an equation:

```
a ← a + 3
```

The right side is worked out first, using the value a holds now. The result is then stored back in a, and the old value is gone.

Whole numbers divide into whole numbers

In Swift, 7 / 2 is 3, not 3.5. When both numbers are `Int`, the answer is an `Int`, and whatever is after the decimal point is dropped, not rounded. Write 7.0 / 2 to get 3.5.

Copies, not links

Example 1

Trace this code one line at a time, keeping a column for each name.

```
a ← 5
b ← a
a ← a + 3
DISPLAY(b)
DISPLAY(a)
```

Line	a	b
a ← 5	5	
b ← a	5	5
a ← a + 3	8	5

`b ← a` copies the value in `a` into `b` **once**. After that, nothing links them. Changing `a` later does not change `b`. It displays 5 and then 8.

The same code in Swift

Example 2

```
var a: Int = 5
let b: Int = a
a = a + 3
print(b)
print(a)
```

`a` changes on the third line, so it must be a **var**. `b` is set once and never changes, so it can be a **let**. Both hold whole numbers, so their type is **Int**. Where the AP code says `DISPLAY`, Swift says `print`: that is how a command-line Swift program shows a value. Swift has a shorter way to write the third line, which the Guessing Game uses: `a += 3`.