1. Variables and Constants

변수와 상수를 선언하는 문법과 여기에 저장된 값을 읽고 쓰는 방법에 대해 공부합니다.

`프로그램에서 데이터를 처리하기 위한 첫 번째 단계는 메모리에 값을 저장하는 거에요 🙂`

여기에 필요한 것이 바로 변수와 상수랍니다.

Variables(변수)

```
// Syntax
var variableName = initialValue
```

변수는 어떤 값을 저장하는 공간이에요.

언제든지 새로운 값을 저장할 수 있어요.

변수에 값을 저장하려면 일단 변수를 만들어야 합니다.

변수를 만든다 이것을 변수를 선언한다 이렇게 표현해요.

변수를 선언할 땐 모두 4개의 토큰(Token)이 필요하답니다.

우리 앞에서 토큰을 공부 했었잖아요 ~ 🙂

토큰이 무엇이였죠?

`'문법적으로 더이상 나눌수 없는 요소'` → Token

그렇다면 어떤 토큰이 4개나 들어갈까요?

가장 먼저 변수 선언 키워드인 ‘var’

그 다음은 변수의 이름 ‘variableName’
여기서 이름은 식별자(Identifiers)이죠?

그 다음은 연산자인 ‘=’
이 연산자는 Equal Sign이라고 부릅니다.

이 Equal Sign은 Assignment Operator(할당 연산자) 입니다.
이것은 ‘오른쪽에 있는 값을 왼쪽에 있는 변수에 저장하는 역할’을 합니다.

우리가 저장은 흔히 save로 번역하죠?
할당이나 저장은 큰 틀에서 보면 똑같은 말 입니다.

하지만 상수나 변수처럼 메모리에 무엇인가를 저장 할 때는
할당이라는 말을 주로 사용한답니다.

파일이나 데이터 베이스에 영구적으로 저장할 때는
저장이라는 용어를 주로 사용하고 있답니다 🙂
참고로 알아두시면 좋아요 🙌

마지막에 있는 ‘initialValue’
이름 그대로 처음에 저장할 값을 뜻합니다.

다시 돌아와서 말해보면 이렇게 총 4개의 토큰이 필요해요.

- var
- 이름

- =
- 초깃값

이번에는 실제로 할당을 해볼게요.

```
var name = "Kobe"
```

이렇게하면 “Kobe” 라는 문자열 값이 name이라는 변수에 저장되는 것 입니다.

Kobe라는 문자열 값을 큰 따옴표가 감싸고 있는게 보이나요?
문자열은 반드시 이렇게 큰 따옴표(” “)로 감싸야 합니다.

이것을 문자열 리터럴(String Literal)이라고 불러요.

그리고 위 코드를 보면 모든 토큰이 하나의 공백으로 분리되어 있는게 보일거에요.

```
varname = "Kobe"
```

위 코드와 같이 var 와 name 키워드 사이에 공백을 지우면 어떻게 될까요?
var는 더이상 키워드가 아니게 됩니다.
그래서 이 코드도 더이상 변수 선언 코드가 아니게 됩니다 😱

이때 부터는 ‘varname’이라는 식별자(identifiers)로 인식하게 됩니다.
이런 변수가 따로 선언되어 있지 않은 이상 아마 컴파일 에러가 발생할거에요.
(Cannot find ‘varname’ in scope)

이번에는 공백이 아닌 탭을 추가해봅시다.

```
var        name = "Kobe"
```

위 코드는 어떻게 될지 짐작이 가시나요?

Tap과 return 도 공백이라고 알고 있을거에요.

그래서 위와 같이 tap을 추가해도 문제가 없어요.

토큰을 명확히 구분할 수 있다면 얼마든지 추가해도 된답니다.

```
var
name = "Kobe"
```

심지어 이렇게 라인을 분리해도 문제가 없어요.

그 이유는 컴파일러는 컴파일 과정에서 모든 공백을 하나의 공백으로 처리하기 때문이죠.

컴파일러 입장에서 보면 라인이 분리된 코드와 일반적인 공백으로 분리된 코드 모두 그냥 같은 코드에요.

```
// 두 코드 모두 컴파일 입장에서는 같은 코드이다.
var name = "Kobe"

var
name = "Kobe"
```

문법적으로 공백을 마음대로 추가해도 되지만 실제로 코드를 그렇게 만들어서는 안됩니다.

`Swift의 표준 코딩 스타일을 지키는것이 매우 중요하고 좋아요.`

이번에는 올해의 연도를 변수에 저장해보는 코드를 보면서 이야기를 해볼게요 🙂

```
var thisYear = 2023
```

이번에는 참과 거짓을 저장하는 코드를 만들어볼까요?

```
var valid = true
```

Swift에는 참과 거짓을 표현하는 리터럴이 하나씩 있답니다.

위 코드에 보이는 true가 Boolean Literal 입니다.

반대로 거짓을 나타내고 싶다면 false를 사용하면 됩니다.

지금까지 3개의 변수를 선언했고 메모리에는 같은 숫자의 공간이 만들어졌으며 우리가 전달한 값이 저장되어 있답니다.

이번에는 저장된 값을 읽어보는 시간을 가져봐요 🙌

선언할 때는 var 키워드를 앞에 사용하지요?

읽을때는 var 키워드를 사용하지 않아요.

자, name 변수에 저장된 값을 읽어 봅시다.

```
name // "Kobe"
```

그냥 위와 같이 name 만 쓰면됩니다.

너무 간단해서 놀랐나요?!

위 코드가 바로 저장된 값을 읽어오는 코드에요.

이 코드를 실행하면, 다시 말해 이 코드를 평가하면 저장된 문자열이 도출되게 됩니다.

그렇다면 이것은 표현식(Expressions)이겠죠?! 👀

위 코드를 실제 프로젝트에서 사용하면 어떻게 될까요?
아마 아무런 현상도 일어나지 않을거에요.
그 이유는 위 코드는 저장된 값을 불러오기만 하고 어떤 처리가 없는 그런 코드거든요.

실제 프로젝트에서 저장된 값을 확인하고 싶을땐 그럼 어떻게 해야할까요?
반드시 print나 dump 함수를 사용해야 한답니다.

```
print(name) // "Kobe\n"
```

print 함수의 괄호 사이에 변수의 이름을 넣으면 됩니다.
그러면 아래쪽 콘솔에서 확인할 수 있을거에요.

그런데 신기한게 있어요 “Kobe\n”
\n이 붙어 나왔네요 과연 이것이 뜻하는 바는 무었일까요?

name과 print(name) 즉, 분명 같은 변수를 출력하는 것인데 결과가 달라요.
뭔가 신기하네요.

자, 이것은 print 함수가 실행될 때 맨 뒤에 자동으로 줄바꿈 문자를 추가해서 그런거랍니다 🙂
\n이 뜻하는 것이 줄바꿈 문자에요.
나중에 문자열 포스트에서 다시 공부할거니까 걱정말아요~

이번에는 변수에 새로운 값을 저장해보는 시간입니다!!

```
// Syntax
variableName = initialValue
```

아까 맨 위에서 본 변수 선언 문법과 다른게 보여요.

이번에는 맨 앞에 var 키워드가 없어요.

변수에 새로운 값을 저장할 때는 var 키워드를 사용하지 않아요.

```
name = "Min Seong"
```

먼저 변수명을 쓰고 =(Equal Sign)을 쓰고 새로 저장할 값을 쓰면 됩니다.

그러면 변수에 저장되어 있는 값이 새로운 값으로 변경이 됩니다.

여기 부분 매우 중요해요!!

`‘값이 누적되는 것이 아니라 바뀌는 거에요. 기존 값은 사라지고 새로운 값으로 완전히 바뀌는거에요.’`

꼭 기억해두세요.

우리 이번에 name 앞에 var 키워드를 붙여볼까요?

```
print(name) // Error: Ambiguous use of 'name'
var name = "Min Seong" // Error : Invalid redeclaration of 'name'
```

위 에러 메시지를 보면 잘못된 재정의(Invalid redeclaration of 'name')와 모호한 사용(Ambiguous use of 'name')이라고 나오네요.

왜 그럴까요?

var는 변수 선언 키워드죠?

지금 name 이름을 다른 곳에서도 사용하고 있어서 그래요

```
var name = "Kobe"
name
print(name)
var name = "Min Seong"
```

위 코드에서 보면 2번 선언하고 있는게 보이나요?

같은 이름을 2번 이상 선언하는 것은 문법적으로 허용되지 않아요.

그리고 위에서 보면 name과 print(name)으로 name을 출력하고 있는게 보일거에요.

이게 에러가 나는 이유는 위에 선언한 var name = “Kobe”인지 아래에 선언한 var name = “Min Seong”인지

정확하게 판단할 수 없어서 모호한 사용(Ambiguous use of 'name') 에러가 나는거에요.

다시 한번 name에 새로운 값을 저장해볼게요.

```
name = "민성"
```

이렇게 하면 최종적으로 변수 name에는 어떤 값이 저장되어 있을까요?

바로 마지막에 저장한 “민성”이에요.

중요한 내용이니 한 번더 반복할게요

`‘변수에 저장된 값은 누적되지 않습니다. 항상 마지막에 저장된 값이 현재 값이 되고 이전 값은 사라지게 됩니다.’`

이번에는 아에 새로운 변수를 선언해볼게요

```
var anotherName = name
```

아까는 문자열 리터럴을 전달했는데 이번에는 name 변수를 전달했어요.

그러면 name에 저장되어있는 문자열이 복사되어서 anotherName에 저장이 됩니다.

이렇게 다른 변수에 저장된 값을 읽어와 저장하는 것도 얼마든지 가능해요.

이 시점에 두 변수는 동일한 값을 저장하고 있어요.

그렇다면 anotherName에 새로운 값을 저장하면 어떻게 될까요?

```
anotherName = "Morgan"
```

일단 anotherName은 “Morgan”으로 바뀔거에요.

그럼 여기서 문제!!

그렇다면 name 변수의 값도 같이 바뀔까요??

```
print(name, anotherName) // 민성 Morgan
```

위 코드의 결과를 보면 알 수 있다시피 변수가 어떤 시점에 같은 값을 저장하고 있더라도 메모리 공간은 다릅니다.

변수를 선언할 때마다 새로운 메모리 공간을 만들기 때문이에요.

그래서 두 변수는 연결되어 있지 않은 상태인거죠.

때문에 한 변수를 바뀌어도 다른 변수는 영향을 받지 않아요.

이것도 중요한 특징입니다.

나중에 값 형식과 참조 형식을 공부하고 나면 정확히 이해할 수 있을겁니다.

자 지금은 변수의 값을 바꿔도 다른 변수의 값은 바뀌지 않는다.
이것만 이해하시면 됩니다.

이번에는 thisYear 변수에 문자열을 저장해볼게요.

```
thisYear = "2023" // Error: Cannot assign value of type 'String' to type 'Int'
```

에러 메시지가 나오는데 String 타입의 값을 Int 타입의 변수에 저장할 수 없다는 뜻이에요.

조금 풀어서 설명해볼게요.
숫자열을 저장하고 있는 메모리 공간에 문자열을 저장할 수 없다는 뜻이에요.

변수를 선언하면 메모리 공간이 만들어진다고 했었죠?! 😆
이 시점에 메모리 공간에 저장할 수 있는 값의 종류도 결정되고 이것은 절대 바뀌지 않아요.

thisYear 변수는 처음 선언 할 때 숫자를 저장한 것으로 기억할거에요 🙂
더 정확히는 정수를 저장했죠 ㅎㅎ
그러면 여기에는 정수만 저장 할 수 있어요 🤩

그렇기 때문에 타입이 다른 것을 저장하는것은 안됩니다.
이런것은 문법적으로 허용되지 않는것이에요 🙂

Constants(상수)

```
/// Syntax
let constantName = initialValue
```

이번에는 상수를 공부해봅시다!!

상수는 값을 저장한 다음 값을 변경할 수 없다는 것과 키워드가 다르다는 것만 빼면 변수와 똑같습니다.

문법을 보면 var 대신 let이 있는 것을 확인할 수 있죠?!

let은 상수 선언 키워드에요 🙂

아래의 코드를 한 번 봐볼까요?

```
let name = "Kobe"
```

위 코드는 상수를 선언하고 name이라는 이름을 만들어 주었어요.

그리고 name 상수에 “Kobe”라는 문자열 값을 넣어줬네요 🙂

상수에 저장된 값을 읽고 싶다면 어떻게 하면 될까요??

```
name // "Kobe"
```

변수와 완전히 똑같아요.

위 코드와 같이 상수의 이름을 써주면 됩니다.

여기까지는 키워드만 빼면 변수와 다른 점을 찾아볼 수 없어요 그쵸?!

이번에는 새로운 값을 저장해볼게요 🙂

```
name = "민성" // Error : Cannot assign to value: 'name' is a 'let' constant
```

위 코드는 동작하지 않아요.

다시 말해 에러를 내보내고 있어요.

상수는 메모리에 딱 한 번만 값을 저장할 수 있고 이후에는 새로운 값을 저장할 수 없어요.

그 때부터는 값을 읽을 수만 있답니다 🙂

에러 메세지를 읽어보면 name은 상수로 선언되어 있어서 새로운 값을 저장할 수 없다고 나와있어요.

name에 새로운 값을 저장하려면 키워드 let을 var로 변경해야합니다.

다시 말해, 상수를 변수로 변경해야해요.

자, 이렇게 변수와 상수가 있는데 둘 중 어떤걸 사용해야 할까요??

값을 저장해야 한다?

그렇다면 일단 상수로 값을 저장하세요.

그 다음에 값을 바꿔야 한다면 그 때 변수로 바꾸세요.

물론 값을 바꿔야 하는게 명확하다면 처음부터 변수로 선언해도 됩니다.

값을 상수로 저장하면 2가지 장점이 있습니다.

- 코드가 안전해집니다.

  실제 프로젝트 코드는 우리가 작성한 예제 코드처럼 단순하지 않아요.

  수십만줄 수백만줄의 코드가 있고 여러 코드에서 변수나 상수에 접근할 수 있습니다.

변수에 값을 저장하면 어디에서 값을 바꾸었는지 그리고 언제 값을 바꿨는지 이런것들을 제대로 파악하고 있어야 합니다.

이게 생각보다 어렵습니다.

이것을 잘못하면 결국 버그가 발생해요.

문제를 찾고 고치는데 많은 시간이 필요하게 됩니다.

반대로 상수로 저장하면 값이 항상 똑같다는 것이 보장됩니다.

값을 바꿀 수 없기 때문이죠.

그래서 그냥 필요할 때 값을 읽기만 하면 된답니다.

나머지는 신경쓸 필요가 없어지는 것이죠 🙂

값을 누가 바꿨는지 언제 바꿨는지를 말이에요.

- 값이 바뀌지 않으면 컴파일러가 최적화를 합니다, 때문에 조금 더 빠른 코드를 만들 수 있습니다.

이러한 이유들 때문에 상수로 일단 값을 저장했다가 필요할 때 변수로 바꾸는게 좋아요.

Summary

이번에는 변수와 상수에 대해 공부했습니다.

- 변수(Variables)
  - 변수는 var 키워드로 선언.
  - 저장된 값을 언제든지 변경 가능.

- 상수(Constants)
  - 상수는 let 키워드로 선언.
  - 값을 저장한 후 변경할 수 없음.

- 변수와 상수를 선언시 메모리에 공간이 만들어지고, 이 공간에 저장할 수 있는 값의 종류는 가장 먼저 저장한 값에 따라서 결정됩니다.
  - 그리고 이 메모리에는 항상 같은 종류의 값만 저장할 수 있습니다.

- 값을 저장할 때는 일단 상수로 선언했다가 나중에 값을 바꿔야 할 때 변수로 바꾸는게 좋습니다.
  - 상수로 선언시 안정성이 높아지고, 조금 더 빠른 코드를 만들 수 있습니다.