변수와 자료형(1)

PDF	비어 있음
비고	변수
숫자	2
실습문제	비어 있음
실습문제답안	비어 있음

01. 변수 (Variable)

01-01. 변수 개요

01-01-01. 변수란

💡 프로그램에서 다루는 어떤 자료(정보)를 담는 공간이다.

01-01-02. 변수 생성

```
변수명 = 자료
```

- 변수 생성은 대입 연산자 = 을 기준으로 좌항에는 변수의 이름, 우항에는 변수에 담을 자료를 작성한다.
- 변수의 생성과 변수의 값 변경 모두 같은 방법으로 작성한다.
- 파이썬은 변수 공간의 자료형 선언이 없다.
  - cf. 자바는 변수 생성(선언) 시 변수명 좌측에 자료형을 지정하므로, 그렇게 생성한 변수에는 특정 자료형의 값만 대입할 수 있다.

```
int num = 1 num = "1" // 에러 발생
```

01-02. 대입 연산자

01-02-01. 대입 연산자 “=”

- 일반적인 수학 기호에서 ‘같다’를 의미하는 등호(=)가 파이썬에서는 대입 연산자로 사용된다.
- 대입 연산자를 기준으로 우항에 있는 값을 좌항의 변수 공간에 대입하는 의미를 가진다.
- 대입 연산자의 우항에도 변수명이 올 수 있는데, 이때 변수명은 공간이 아닌 값으로 인식한다.

```
# 좌항은 변수명, 우항은 원시 문자열로 표현된 자료 team_name = "ohgiraffers" # 좌항은 변수명, 우항은 team_name 변수에 담긴 값 "ohgiraffers" my_team = team_name
```

01-03. 변수 이름 명명규칙

01-03-01. 암묵적 규칙

1. 변수명은 변수에 담긴 값이 무엇인지 직관적으로 알 수 있게끔 짓는다.

```
# 키(cm) 정보를 변수에 담아보세요 height = 207.3 a = 171.8 # 변수명만 보고는 키인지, 몸무게인지, BMI인지, 시력인지... 알 수 없음
```

2. 변수명은 스네이크 케이스(소문자+언더바)로 작성하며, 대소문자를 구별한다.

   (아래 예시에서 만약 대소문자를 구분하지 않는다면 team_name과 Team_name이 같은 것으로 출력될 것이다.)

```
team_name = "오지라퍼스" print(team_name) # 오지라퍼스 Team_name = "Ohgiraffers"
print(team_name) # 오지라퍼스 print(Team_name) # Ohgiraffers
```

3. 한글 변수명을 지정할 수 있다.

   (하지만 인코딩 등의 문제를 야기할 수 있으므로 사용을 지양한다.)

```
우리팀 = "오지라퍼스"
```

01-03-02. 문법 오류를 야기하는 규칙

1. 영문과 숫자를 혼합해 작성할 수 있지만, 숫자를 가장 앞에 작성할 수 없다.

   (숫자로만 구성하는 것도 불가능하다.)

```
team_1_name = "오지라퍼스" 1_team_name = "오지라퍼스" # 컴파일 오류 발생 20251010 = "오지라퍼스" # 컴파일 오류 발생
```

2. 언더바(_)를 제외한 특수 문자는 사용 불가하다.

```
team_@_name = "오지라퍼스" # SyntaxError 발생
```

3. 파이썬 예약어(if, elif, else, for, while, ...)를 사용할 수 없다.

```
else = "오지라퍼스" # 컴파일 오류 발생
```

[참고] 파이썬의 언더스코어 (_)

💡 파이썬에서 언더스코어의 사용은 다음 네 가지와 같은 특별한 기능을 제공한다.

1. 인터프리터(Interpreter)에서 마지막(으로 실행된) 값 저장 시, _(언더스코어)라는 변수로 저장된다.

   (이때, 이 _ 변수를 사용해서 연산을 하는 것도 가능하다.)

```
10
```

```
print(_) # 10 print(_ * 3) # 30
```

2. 값을 무시하고 싶은 경우, 특정 값을 무시하기 위한 용도로 사용할 수 있다.

```
# 특정 값을 무시 x, _, z = (1,2,3) print(x) # 1 print(z) # 3 # 앞에 *을 추가로 붙여 여러 개의 값 무시 x, *_, y = (1, 2, 3, 4, 5) print(x) # 1 print(y) # 5
```

3. 변수나 함수명에 특별한 의미 또는 기능을 부여하고자 할 때 사용할 수 있다.
   a. 변수, 함수, 클래스, 메소드를 선언할 때 _로 시작하는 것들은 import 에서 무시된다.
   b. 파이썬은 private를 지원하지 않고 있지만, 위와 같이 사용함으로써 private 선언 컨벤션처럼 쓸 수 있다. (하지만 private과 동일한 기능을 하는 것은 아니므로, import 에서는 무시되지만 직접 가져다 쓰는 등 호출은 가능하다.)

```
_internal_name = 'one_module' # private 변수처럼 선언 _internal_version = '1.0' # private 변수처럼 선언
```

4. 숫자 리터럴 값의 자릿수 구분을 위한 구분자로 사용할 수 있다.

   (Python 3.6 에서 추가된 문법으로, 언더스코어를 이용해 자릿수를 구분할 수 있다.)

```
million = 1_000_000 print(million) # 1000000
```

5. 파이썬 키워드, 모듈명 충돌을 회피하기 위해 마지막에 _를 붙여줄 수 있다.

```
for_ = 'for blah' print(for_)
```

6. 스페셜 변수, 매직 메소드 등에 사용하기도 한다.

```
class _A: def __init__(self): self.__foo = 'foo' self.bar = 'bar' def foo(self): return self.__foo def _good_morning(self): return self.bar * 10 def hello(self): return self._good_morning()
```

```
a = _A() a.__init__ a.__str__ a.__eq__ # a == b라는 식이 수행될 때 실행되는 스페셜 메서드
```