제1장 변수와 연산자 그리고 문자열

주석

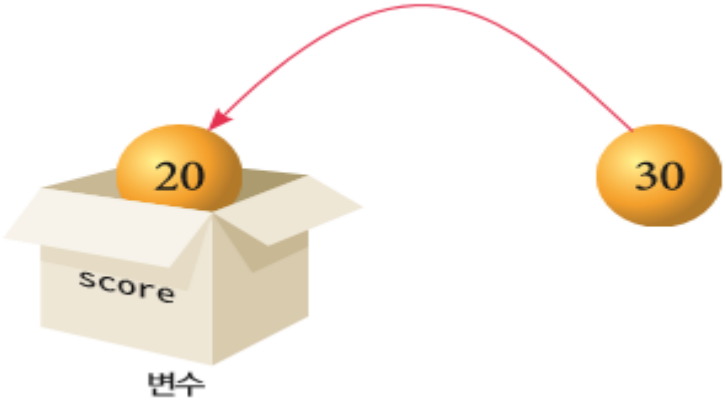
주석(comment)은 소스 코드에 붙이는 설명문

```
# 사각형의 가로 길이
width = 10

# 사각형의 세로 길이
height = 20

# 사각형의 면적 계산
area = width * height
```

변수의 소개

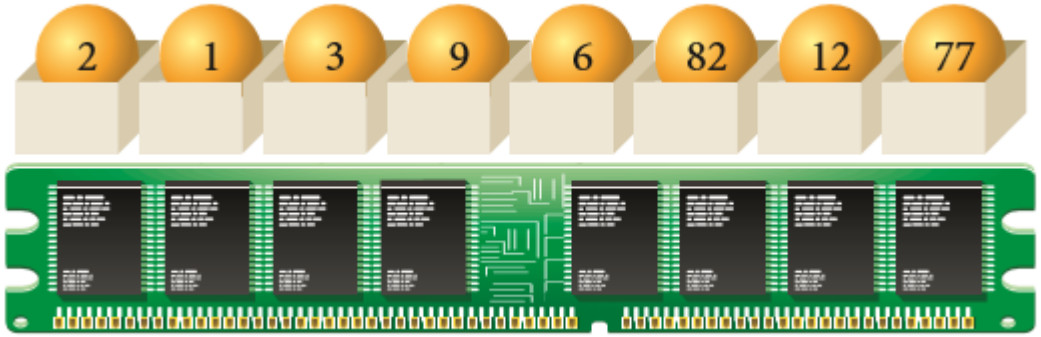
변수(variable)는 값을 저장하는 공간

가장 최근 값으로 변경



변수와 메모리

변수는 메모리(memory)에 만들어진다.



변수 생성

형식) 변수이름 = 값

```
>>> score = 20
>>> score
20
>>> print(score)
20
```

변수의 사용

생성된 변수에는 얼마든지 다른 값을 저장할 수 있다.

```
>>> score = 20
>>> score = 30
>>> score
30
```

변수의 사용

변수에는 다른 변수의 값도 저장할 수 있다.

```
>>> width = 10
>>> height = 20
>>> area = width * height
>>> print(area)
200
```

변수의 사용

파이썬의 변수에는 정수뿐만 아니라 문자열도 저장

```
>>> s = '안녕하세요?'
>>> print(s)
안녕하세요?

>>> pi = 3.141592
>>> print(pi)
3.141592
```

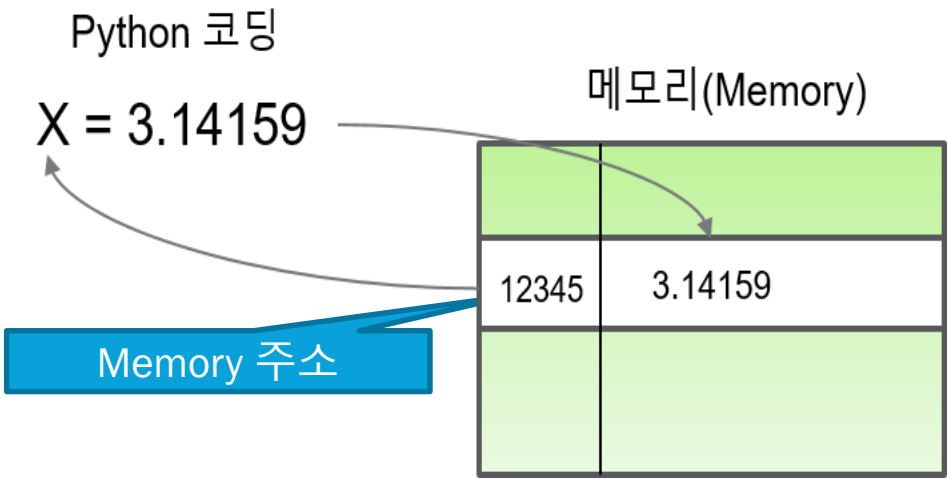
변수가 저장하는 것

파이썬에서 변수는 어떤 데이터든지 저장할 수 있다.

```
value = 3
value = 3.14
value = "hello"
```

PYTHON 변수 저장 방식

- 자료(Data)가 저장된 주소 저장
- 이 주소를 통해서 데이터를 참조 -> 참조변수
- 예) X변수에 값 3.14159가 저장된 메모리 주소가 저장



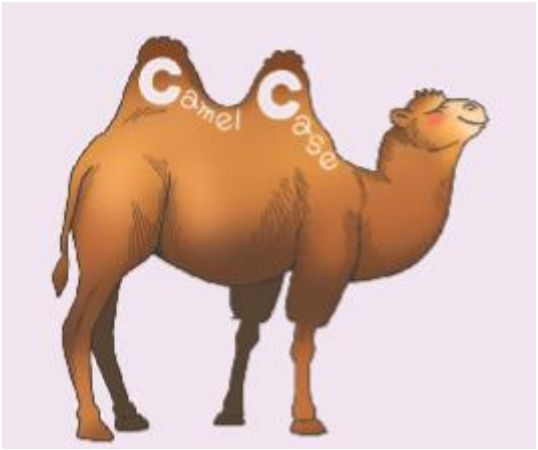
식별자 작성 규칙

- **식별자 : 변수이름, 함수이름, 클래스 이름**
- **의미 있는 이름 사용**
- **소문자와 대문자는 서로 다르게 취급된다.**
- **변수의 이름은 영문자와 숫자, 밑줄(_)로 이루어진다.**
- **변수 이름 중간에 공백이 들어가면 안 된다.**
- **단어를 구분하려면 밑줄(_)을 사용한다.**
- **키워드(예약어) 사용불가, 한글 비권장**

올바른 예	잘못된 예
value	123data
_number	score$$
korScore	Data set
data123	member.name

낙타체

- 낙타체는 변수의 첫 글자는 소문자로, 나머지 단어 의 첫 글자는 대문자로 적는 방법
- 예) myNewCar처럼 첫 'm'은 소문자로, 나머지 단어들의 첫 글 자는 대문자로 표기



상수

상수(constant)는 한번 값이 결정되면 절대로 변경되지 않는 값(대문자 표기)

```
TAX_RATE = 0.35
PI = 3.141592
MAX_SIZE = 100
```

수식과 연산자

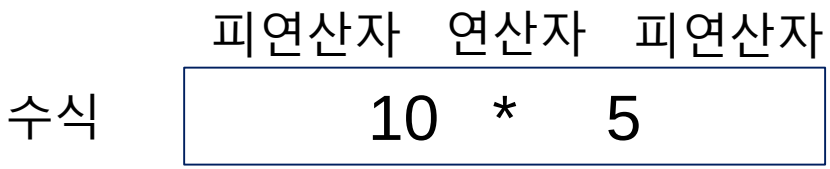
```
>>> 3 + 4
7
>>> 3.14 * 5.0 * 5.0
78.5
```

연산자와 피연산자

연산식, 수식(expression)=피연산자들과 연산자의 조합

연산자(operator)는 연산을 나타내는 기호

피연산자(operand)는 연산의 대상이 되는 것



산술 연산자

덧셈, 뺄셈, 곱셈, 나눗셈, 나머지 연산

연산자	기호	사용예	결과값
덧셈	+	7 + 4	11
뺄셈	-	7 - 4	3
곱셈	*	7 * 4	28
나눗셈	//	7 // 4	1
나눗셈	/	7 / 4	1.75
나머지	%	7 % 4	3

나눗셈

```
>>> 7 / 4                  실수 나눗셈
1.75

>>> 7 // 4                 정수 나눗셈
1
```

지수 계산

지수(power)를 계산하려면 ** 연산자를 사용한다.

```
>>> 2 ** 7
128
```

```
>>> a=1000
>>> r=0.05
>>> n=10
>>> a*(1+r)**n
1628.894626777442
```

나머지 계산

예제로 초 단위의 시간을 받아서 몇 분 몇 초인지를 계산하여 보자.

```
>>> 7 % 4
3
```

```
>>> sec = 1000
>>> min = 1000 // 60
>>> remainder = 1000 % 60
>>> print(min, remainder)
16 40
```

연산자 우선순위

```
>>> 1 + 2 * 3
7
>>> 4 - 40 - 3
-39
```

괄호의 사용

```
>>> 10 + 20 /2
20.0

>>> (10 + 20) /2
15.0
```

PRINT() 함수

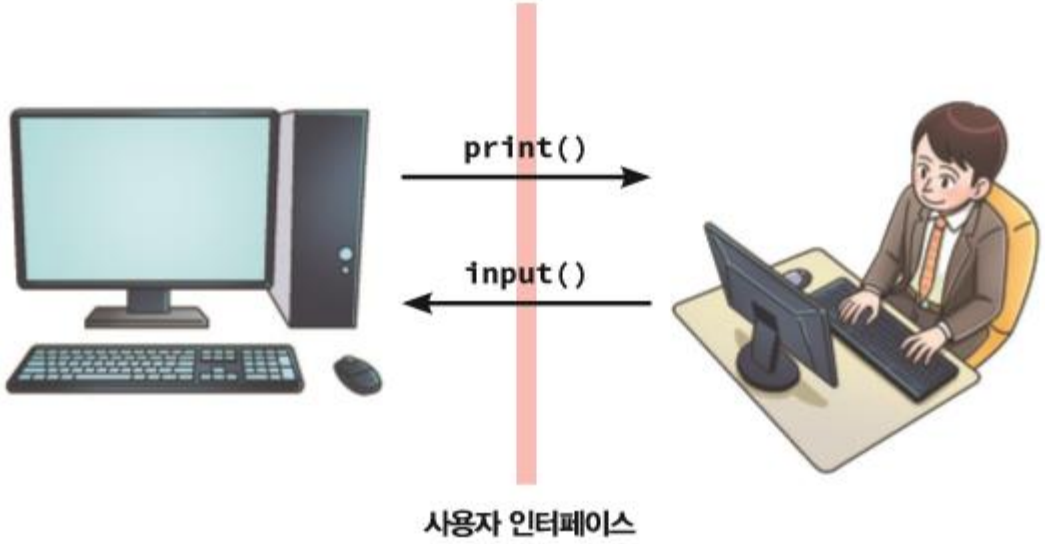
format(값, " 양식문자")

```
print("원주율=", format(3.14159, "8.3f") ) # 원주율= 3.142
print("금액 =", format(10000, "10d")) # 금액 = 10000
print("금액 =", format(125000, "3,d")) # 금액 = 125,000
```

양식문자	출력 형식
%d	10진수 정수
%o	8진수 정수
%x	16진수 정의
%f	실수(%f전체자릿수.소수점자릿)
%s	문자열
%c	단일 문자열

INPUT() 함수

사용자와의 상호작용



INPUT() 함수

변수 = input(“프롬프트 문자열“)

문자열 입력

```
name = input("이름이 무엇인가요? ")
print("만나서 반갑습니다. " + name + "씨!")
age = input("나이는요? ")
print("네, 그러면 당신은 이미 " + age + " 살이시군요, " + name + "씨!")
```

```
이름이 무엇인가요? 홍길동
만나서 반갑습니다. 홍길동씨!
나이는요? 21
네, 그러면 당신은 이미 21 살이시군요, 홍길동씨!
```

숫자 입력

```
x = input("첫 번째 정수: ")
y = input("두 번째 정수: ")
sum = x + y
print("합은 ", sum)
```

```
첫 번째 정수: 10
두 번째 정수: 20
합은  1020
```

문자열로 간주하여 서로 합침!

숫자 입력

```
x = int(input("첫 번째 정수: "))
y = int(input("두 번째 정수: "))
sum = x + y
print("합은 ", sum)
```

```
첫 번째 정수: 10
두 번째 정수: 20
합은  30
```

자료형

정수(integer), 실수(floating-point), 문자열(string)

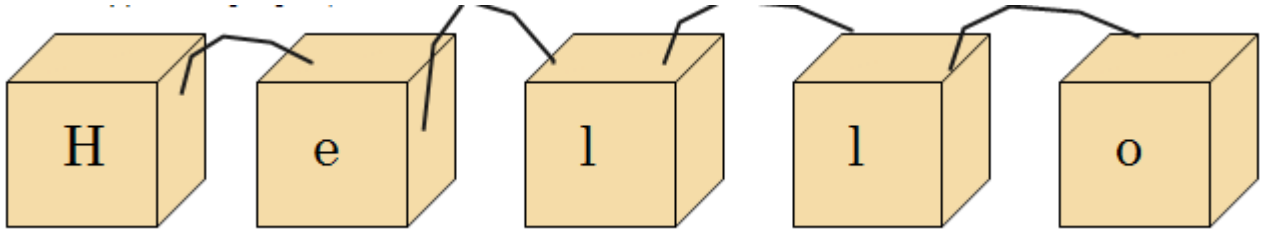
자료형	예
정수	…, −2, −1, 0, 1, 2, …
실수	3.2, 3.14, 0.12
문자열	'Hello World!', "123"

자료형을 알고 싶으면?

```
>>> type("Hello World!")
<class 'str'>
>>> type(3.2)
<class 'float'>
>>> type(17)
<class 'int'>
```

문자열

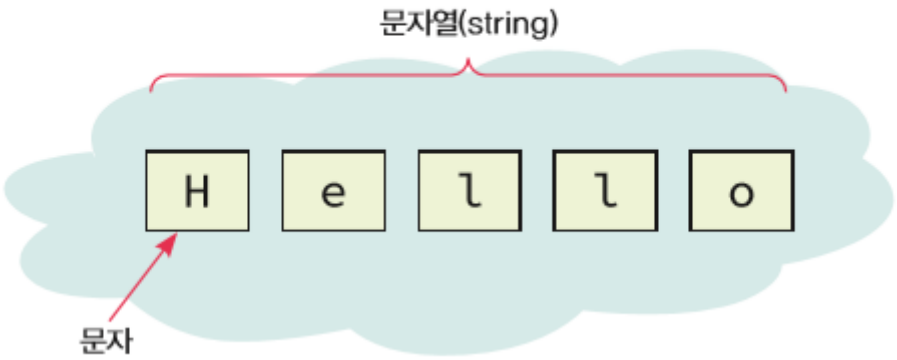
문자열(string) : 큰따옴표("...")나 작은따옴표('...') 안에 들어 있는 텍스트 데이터
반드시 따옴표가 있어야 한다.




```
>>> print(Hello World!)
        SyntaxError: invalid syntax
```

문자열이란?

문자열(string)은 문자들의 순서 있는 집합(sequence of characters)



큰따옴표 사용

```
>>> greeting="Merry Christmas!"
>>> greeting
'Merry Christmas!'

>>> print(greeting)
Merry Christmas!
```

작은 따옴표 사용

```
>>> greeting='Happy Holiday!'
>>> print(greeting)
Happy Holiday!

>>> greeting="Happy Holiday'
SyntaxError: EOL while scanning string literal
>>>
>>> greeting="Happy Holiday
SyntaxError: EOL while scanning string literal
>>>
```

큰 따옴표 안의 작은 따옴표 사용

```
>>> message="철수가 "안녕"이라고 말했습니다."
SyntaxError: invalid syntax
>>>

>>> message="철수가 '안녕'이라고 말했습니다."
>>> print(message)
철수가 '안녕'이라고 말했습니다.
>>>
```

여러 줄의 문자열

```
>>> greeting='''지난 한해 저에게 보여주신 보살핌과
사랑에
깊은 감사를 드립니다.
새해에도 하시고자 하는 일
모두 성취하시기를 바랍니다.'''

>>> print(greeting)
지난 한해 저에게 보여주신 보살핌과 사랑에
깊은 감사를 드립니다.
새해에도 하시고자 하는 일
모두 성취하시기를 바랍니다.
```

특수 문자열

문자 앞에 \가 붙으면 문자의 특수한 의미를 잃어버린다.

```
>>> message= 'doesn\'t'  # \를 사용하여 작은따옴표를 출력한다.
>>> print(message)
doesn't
>>>
>>> message="\"Yes,\" he said."
>>> print(message)
"Yes," he said.
>>>
```

문자열의 연결

```
>>> 'Py' 'thon'
'Python'

>>> 'Harry ' + 'Porter'
'Harry Porter'

>>> first_name="길동"
>>> last_name="홍"
>>> name = last_name+first_name
>>> print(name)
홍길동
```

문자열과 정수 간의 변환

```
>>> "Student"+26
...
TypeError: Can't convert 'int' object to str implicitly

>>> "Student"+str(26)
'Student26'

>>> price = int("259000")
>>> height = float("290.54");
```

문자열의 반복

```
>>> line = "=" * 50
>>> print(line)
==================================================


>>> message = "Congratulations! "
>>> print(message*3)
Congratulations! Congratulations! Congratulations!
```

문자열의 출력

```
>>> price = 10000
>>> print("상품의 가격은 %s 원입니다." % price)
상품의 가격은 10000 원입니다.

>>> message = "현재 시간은 %s 입니다."
>>> time = "12:00pm"
>>> print(message % time)
현재 시간은 12:00pm 입니다.
```

인덱싱

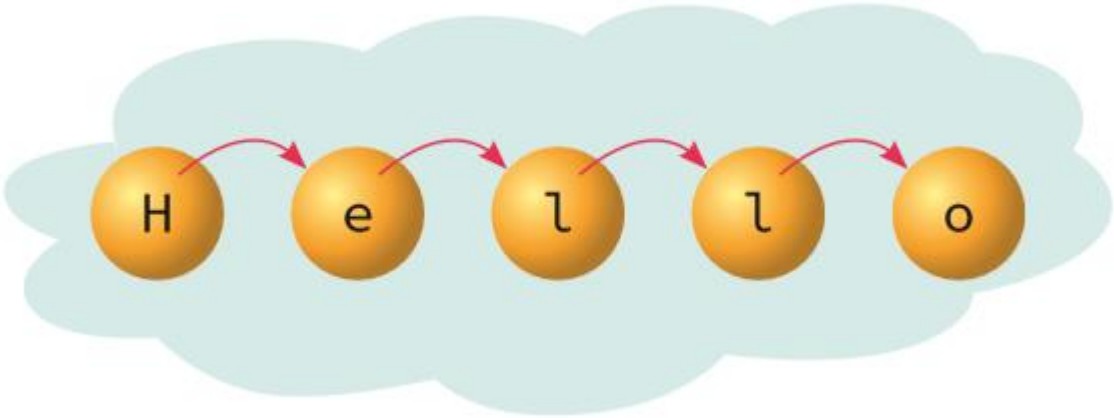
인덱싱(Indexing)이란 문자열에 [과]을 붙여서 문자를 추출하는 것이다.

P	y	t	h	o	n
0	1	2	3	4	5

```
>>> word = 'Python'
>>> word[0]
'P'
>>> word[5]
'n'
```

문자열

- 문자 열은 문자들의 시퀀스로 정의된다. 글자들이 실(string)로 묶여 있는 것이 문자열이라고 생각하면 된다.



문자열의 예

```
s1 = str("Hello")
s2 = "Hello“

s1 = "Hello"
s2 = "World"
s3 = "Hello"+"World"
```

개별 문자 접근하기

```
>>> word = 'abcdef'
>>> word[0]
'a'
>>> word[5]
'f'
```

[-n] R 제외

a	b	c	d	e	f
0	1	2	3	4	5
-6	-5	-4	-3	-2	-1

슬라이싱

```
>>> word = 'Python'
>>> word[0:2]
'Py'
>>> word[2:5]
'tho’
```

슬라이싱의 예

```
>>> word = 'Python'
>>> word[0:2]
'Py'
>>> word[2:5]
'tho’
```

IN 연산자와 NOT IN 연산자

```
>>> s="Love will find a way."
>>> "Love" in s
True
>>> "love" in s
False
```

```
s = input("문자열을 입력하시오")
if 'c' in s:
    print('c가 포함되어 있음')
else:
    print('c가 포함되어 있지 않음')
```

문자열 비교하기

```
a = input("문자열을 입력하시오: ")
b = input("문자열을 입력하시오: ")
if( a < b ):
        print(a, "가 앞에 있음")
else:
        print(b, "가 앞에 있음")
```

```
문자열을 입력하시오: apple
문자열을 입력하시오: orange
apple 가 앞에 있음
```

문자열에서 단어 분리

```
>>> s = 'Never put off till tomorrow what you can do today.'
>>> s.split()
['Never', 'put', 'off', 'till', 'tomorrow', 'what', 'you', 'can', 'do',
'today.']
```

문자열 처리 함수

함수명	기능
s.lower()	문자열 s를 모두 소문자로 변경한다.
s.upper()	문자열 s를 모두 대문자로 변경한다.
s.swapcase()	문자열 s의 대문자는 소문자로, 소문자는 대문자로 변경한다.
s.lstrip()	문자열 s의 왼쪽 공백을 모두 지운다.
s.rstrip()	문자열 s의 오른쪽 공백을 모두 지운다.
s.strip()	문자열 s의 양쪽 공백을 모두 지운다.
s.count(x)	문자열 s에서 x와 일치하는 문자의 갯수를 반환한다.
s.find(x)	문자열 s에서 문자 x가 처음으로 나온 위치를 반환한다. (없으면 -1 반환)

함수명	기능
s.rfind(x)	문자열 s에서 문자 x가 처음으로 나온 위치를 오른쪽 끝에서 부터 찾는다.
s.index(x)	문자열 s에서 문자 x가 처음으로 나온 위치를 반환한다. (없으면 에러 발생)
'구분자'.join(s)	문자열 s를 구성하는 요소 문자를 '구분자'로 결합시킨다.
s.replace(x, r)	문자열 s를 대상으로 x라는 문자를 r이라는 문자로 교체하다.
s.split('구분자')	문자열 s에서 '구분자'를 기준으로 문자열을 분리시킨다.
s.startswith('H')	H라는 문자로 시작되는지 여부를 True 또는 False로 반환한다.