

기본 자료형

수치형

수치형 (Number)

- 숫자 형태로 이루어진 자료형
 - 정수형 (Integer)
 - 정수를 표현하는 자료형
 - 0, 1, 2, 3, ..., -1, -2, -3, ...
 - 실수형 (Floating-point)
 - 실수를 표현하는 자료형
 - 1.5, 3.14, 0.0, -567.89, ...
 - 복소수형 (Complex)
 - 복소수를 표현하는 자료형

정수형 (Integer)

- 정수를 표현하는 자료형
 - $0, 1, 2, 3, \dots, -1, -2, -3, \dots$

1	123
---	-----

123

1	-45
---	-----

-45

1	751642198063095340554216584361574092291840653792
---	--

751642198063095340554216584361574092291840653792

1	100, 2019, 3
---	--------------

(100, 2019, 3)

실수형 (Floating-point)

- 실수를 표현하는 자료형
 - 1.5, 3.14, 0.0, -567.89, ...

1	3.1415926
---	-----------

3.1415926

1	-123.4567890
---	--------------

-123.456789

1	3506518.16035465513960852475
---	------------------------------

3506518.1603546552

1	1.5, 3.14, 0.0, -567.89
---	-------------------------

(1.5, 3.14, 0.0, -567.89)

실수형 (Floating-point)

- 실수의 지수 표현 방식
 - 1.2×10^3 과 같이 10의 지수 형태를 표현할 수 있다.
 - 10의 거듭제곱 부분에 e 또는 E를 기재하고, 그 뒤에 거듭제곱 되는 숫자를 표시한다.
 - 예를 들어, 1.2×10^3 은 1.2e3 으로 표기

1	1.2e3
---	-------

1200.0

1	-3.45E5
---	---------

-345000.0

1	67.89e-2
---	----------

0.6789

기본 자료형

문자열

문자열 (String)

- 문자, 단어 등으로 구성된 문자들의 집합
 - 따옴표로 둘러싸여 있으면 무조건 문자열이다.
 - “hello”, ‘반갑습니다! 잘 부탁드립니다.’, “12345”, ...

1	"hello"
---	---------

'hello'

1	'반갑습니다! 잘 부탁드립니다.'
---	--------------------

'반갑습니다! 잘 부탁드립니다.'

1	"12345"
---	---------

'12345'

문자열 (String)

- 한 줄로 된 문자열 만들기

① 큰 따옴표 “ ” 로 양 쪽을 둘러싼다.

```
1 "hello"
```

```
'hello'
```

```
1 "안녕하세요?   제 이름은 홍길동이라고 합니다.   "
```

```
'안녕하세요?   제 이름은 홍길동이라고 합니다.   '
```

② 작은 따옴표 ‘ ’ 로 양 쪽을 둘러싼다.

```
1 'My birthday is January 29.'
```

```
'My birthday is January 29.'
```

```
1 'print(=^-=)'
```

```
'print(=^-=)'
```


문자열 (String)

- 여러 줄로 된 문자열 만들기

① 연속된 큰 따옴표 3개 “““ ””” 로 양 쪽을 둘러싼다.

```
1  """안녕하세요?  
2     제 이름은 홍길동이라고 합니다.  
3  이렇게 만나게 되어 정말 기쁘네요."""
```

'안녕하세요?␣ 제 이름은 홍길동이라고 합니다.␣이렇게 만나게 되어 정말 기쁘네요.'

※ 줄바꿈(개행)한 부분은 문자열 자료에서 \n 으로 표시된다.

② 연속된 작은 따옴표 3개 “ ” 로 양 쪽을 둘러싼다.

```
1  '''Hi, nice to meet you~  
2  Oh, I do not have time.  
3  Sorry. Bye~!'''
```

'Hi, nice to meet you~␣Oh, I do not have time.␣Sorry. Bye~!'

문자열 (String)

- 여러 줄로 된 문자열 만들기

- ③ 큰 따옴표 또는 작은 따옴표로 문자열 양 쪽을 둘러싼 뒤, 문자열 안에서 줄바꿈(개행) 기호 `\n` 를 사용한다.

```
1 "안녕하세요?\n 제 이름은 홍길동이라고 합니다.\n이렇게 만나게 되어 정말 기쁘네요."
```

```
'안녕하세요?\n 제 이름은 홍길동이라고 합니다.\n이렇게 만나게 되어 정말 기쁘네요.'
```

```
1 'Hi, nice to meet you~\nOh, I do not have time.\nSorry. Bye~!'
```

```
'Hi, nice to meet you~\nOh, I do not have time.\nSorry. Bye~!'
```

문자열 (String)

※ 자료 자체의 실행과 출력 명령어 실행

- 셀에 어떤 자료가 표현되어 있을 때 이 셀을 실행하면 그 자료의 내용 자체가 그대로 화면에 표시된다.

```
1 "안녕하세요?ㄴ 제 이름은 홍길동이라고 합니다.ㄴ이렇게 만나게 되어 정말 기쁘네요."
```

'안녕하세요?ㄴ 제 이름은 홍길동이라고 합니다.ㄴ이렇게 만나게 되어 정말 기쁘네요.'

- 화면 출력 명령어인 **print**를 이용하면 그 자료에 들어 있는 기호가 반영되어 화면에 표시된다.

```
1 print("안녕하세요?ㄴ 제 이름은 홍길동이라고 합니다.ㄴ이렇게 만나게 되어 정말 기쁘네요.")
```

안녕하세요?

제 이름은 홍길동이라고 합니다.

이렇게 만나게 되어 정말 기쁘네요.

문자열 (String)

- 문자열 내용 자체에 따옴표 넣기

- ① 문자열 내용 자체에 넣으려는 따옴표와는 다른 종류의 따옴표를 사용하여 문자열을 둘러싼다.

- 즉, 만약 문자열 안에 작은 따옴표가 들어가야 한다면 문자열을 큰 따옴표로 둘러싸면 된다.

```
1 "안녕하세요? 제 이름은 '홍길동'이라고 합니다."
```

```
"안녕하세요? 제 이름은 '홍길동'이라고 합니다."
```

- 문자열 안에 큰 따옴표가 들어가야 한다면 문자열을 작은 따옴표로 둘러싸면 된다.

```
1 'He said, "Hi, nice to meet you~"'
```

```
'He said, "Hi, nice to meet you~"'
```

문자열 (String)

- 문자열 내용 자체에 따옴표 넣기
 - ② 문자열을 둘러싸는 따옴표의 종류와 상관 없이, 문자열 안에서 작은 따옴표 기호 `'` 및 큰 따옴표 기호 `"` 를 사용한다.

```
1 'He said, "I don't have time now. Sorry"
```

```
'He said, "I don't have time now. Sorry"
```

```
1 "문자열 안에서 작은 따옴표 기호 ' 또는 큰 따옴표 기호 "를 사용합니다."
```

```
'문자열 안에서 작은 따옴표 기호 ' 또는 큰 따옴표 기호 "를 사용합니다.'
```

문자열 (String)

- 탈출 문자 (Escape Character)
 - 문자열 안에서 특수한 표현을 하기 위해서 미리 지정해 둔 문자와 기호의 조합

표기	역할
\n	줄바꿈 (개행)
\t	탭
\"	큰 따옴표
\'	작은 따옴표
\\	문자 \

기본 자료형

논리형

논리형 (Boolean)

- 참과 거짓을 나타내는 자료형
 - 참은 True, 거짓은 False로 표현한다.
 - 문자열이 아니기 때문에 각 단어의 양 쪽에 따옴표로 둘러싸여 있지 않다.
 - 또한 첫 문자는 반드시 대문자로서, true 또는 false처럼 소문자로 기재해서는 안 된다.

1	True
---	------

True

1	False
---	-------

False

논리형 (Boolean)

- 다른 자료를 논리 값으로 변환
 - 모든 자료형은 논리형으로 변환될 수 있다.
 - 즉, 모든 자료형은 논리 값 참 또는 거짓으로 분류할 수 있다.

자료형	참	거짓
정수형	0이 아닌 정수	0
실수형	0.0이 아닌 실수	0.0
문자열	1개 이상의 문자가 들어 있는 문자열	비어 있는 문자열, 즉 “”
논리형	True	False

논리형 (Boolean)

- 논리 값을 수치형으로 변환
 - 논리 값 참과 거짓은 정수 또는 실수로 변환될 수 있다.
 - 참은 정수 1 또는 실수 1.0으로 간주된다.
 - 거짓은 정수 0 또는 실수 0.0으로 간주된다.

논리 값	정수로 변환할 경우	실수로 변환할 경우
True	1	1.0
False	0	0.0

자료형 확인 및 변환

수치형 / 문자열 / 논리형

자료형 확인

- 1개 자료의 자료형 확인하기

type(자료형을 확인하고 싶은 1개의 자료)

```
1 type(123)
```

int

```
1 type(-45)
```

int

```
1 type(3.1415926)
```

float

```
1 type(3506518.16035465513960852475)
```

float

```
1 type(1.2e3)
```

float

```
1 type(-3.45E5)
```

float

```
1 type("hello")
```

str

```
1 type('반갑습니다! 잘 부탁드립니다.')
```

str

```
1 type(True)
```

bool

```
1 type(False)
```

bool

자료형 변환

- 자료를 정수형으로 변환하기

int(정수형으로 변환하고 싶은 1개의 자료)

- 실수형 또는 논리형 자료를 정수형으로 변환할 수 있다.
- 문자열은 정수의 형태인 경우에 변환할 수 있다.

```
1 int(3.1415926)
```

3

```
1 int(True)
```

1

```
1 int("12345")
```

12345

```
1 int("3.1415926")
```

```
ValueError                                Traceback (most recent call last)
<ipython-input-113-24d2d8af0f69> in <module>
--> 1 int("3.1415926")
```

```
ValueError: invalid literal for int() with base 10: '3.1415926'
```

※ 문자열 “3.1415926”은 실수형도 아니고, 정수만으로 이루어진 문자열도 아니기 때문에 명령이 실행되지 않고 오류가 발생

자료형 변환

- 자료를 실수형으로 변환하기

float(실수형으로 변환하고 싶은 1개의 자료)

- 정수형 또는 논리형 자료를 실수형으로 변환할 수 있다.
- 문자열은 정수 또는 실수 형태인 경우에 변환할 수 있다.

```
1 float(3)
```

3.0

```
1 float(False)
```

0.0

```
1 float("12345")
```

12345.0

```
1 float("3.1415926")
```

3.1415926

```
1 float("True")
```

```
-----  
ValueError                                Traceback (most  
<ipython-input-118-652923a8e3f5> in <module>  
--> 1 float("True")
```

ValueError: could not convert string to float: 'True'

※ “True”는 논리형이 아니고 문자열이며,
수치만으로 이루어진 문자열도 아니기 때문에
명령이 실행되지 않고 오류가 발생

자료형 변환

- 자료를 문자열로 변환하기

str(문자열로 변환하고 싶은 1개의 자료)

– 모든 기본 자료형들은 문자열로 변환할 수 있다.

```
1 str(3)
```

'3'

```
1 str(True)
```

'True'

```
1 str(3.1415926)
```

'3.1415926'

```
1 str(False)
```

'False'

```
1 str(1.2e3)
```

'1200.0'

```
1 str("문자열입니다.")
```

'문자열입니다.'

자료형 변환

- 자료를 논리형으로 변환하기

bool(논리형으로 변환하고 싶은 1개의 자료)

– 모든 기본 자료형들은 논리형으로 변환할 수 있다.

```
1 bool(3)
```

True

```
1 bool("문자열입니다.")
```

True

```
1 bool(0)
```

False

```
1 bool("")
```

False

```
1 bool(3.1415926)
```

True

```
1 bool(True)
```

True

기본 연산자

수치형 / 문자열 / 논리형

산술 연산자

- 산술적인 수치 값을 결과로 도출하는 연산자

연산자	기능	참고사항
+	덧셈	문자열끼리의 덧셈도 가능하다.
-	뺄셈	수치형 자료에 대해서만 가능하다.
*	곱셈	문자열과 정수의 곱셈이 가능하다.
**	거듭제곱	수치형 자료에 대해서만 가능하다.
/	나눗셈	
//	나눗셈의 몫 구하기	
%	나눗셈의 나머지 구하기	

산술 연산자

- 산술적인 수치 값을 결과로 도출하는 연산자
 - 덧셈
 - 뺄셈

1	2 + 3
---	-------

5

1	12.345 + 67.89
---	----------------

80.235

1	3.141592 + 7 + True
---	---------------------

11.141592

1	"문자열입니다." + "Hello"
---	---------------------

'문자열입니다.Hello'

1	3 - 2
---	-------

1

1	12.345 - 67.89
---	----------------

-55.545

1	3.141592 - 1
---	--------------

2.141592

1	False - 50 - 100
---	------------------

-150

산술 연산자

- 산술적인 수치 값을 결과로 도출하는 연산자
 - 곱셈 및 거듭제곱
 - 나눗셈

1	12.3 * 4
---	----------

49.2

1	"Hello" * 2
---	-------------

'HelloHello'

1	3 ** 4
---	--------

81

1	9 ** 0.5
---	----------

3.0

1	4 / 2
---	-------

2.0

1	13 / 3
---	--------

4.333333333333333

1	13.0 / 3
---	----------

4.333333333333333

1	-3.141592 / 1.25
---	------------------

-2.5132736

산술 연산자

- 산술적인 수치 값을 결과로 도출하는 연산자
 - 몫 구하기
 - 나머지 구하기

1	4 // 2
---	--------

2

1	13 // 3
---	---------

4

1	13.0 // 3
---	-----------

4.0

1	-3.141592 // 1.25
---	-------------------

-3.0

1	4 % 2
---	-------

0

1	13 % 3
---	--------

1

1	13.5 % 3
---	----------

1.5

1	-3.141592 % 1.25
---	------------------

0.6084079999999998

논리 연산자

- 참 또는 거짓의 논리 값을 결과로 도출하는 연산자

연산자	기능	참고사항
<	작다	수치형 자료끼리 비교하거나 문자열 자료끼리 비교 가능하다.
>	크다	
<=	작거나 같다	
>=	크거나 같다	
==	같다	서로 다른 자료형 간의 비교도 가능하다.
!=	같지 않다	

논리 연산자

- 참 또는 거짓의 논리 값을 결과로 도출하는 연산자
 - 대소 비교
 - 동일 여부 비교

```
1 2 < 3
```

True

```
1 3.141592 > 100
```

False

```
1 "ABCDE" <= "abc"
```

True

```
1 "kim" >= "lee"
```

False

```
1 3 = 1 + 2
```

True

```
1 123 = '123'
```

False

```
1 5 != 5
```

False

```
1 'PYTHON' != 'python'
```

True

논리 연산자

- 참 또는 거짓의 논리 값을 결과로 도출하는 연산자

연산자	기능	설명
not	부정	논리 값을 반대로 만든다.
and	그리고	양 쪽의 논리 값이 모두 참이면 전체가 참이 된다. 둘 중 하나라도 거짓이면 전체가 거짓이 된다.
or	또는	양 쪽의 논리 값 중 하나라도 참이면 전체가 참이 된다. 둘 모두 거짓이면 전체가 거짓이 된다.

논리 연산자

- 참 또는 거짓의 논리 값을 결과로 도출하는 연산자
 - 부정
 - 그리고 / 또는

```
1 not False
```

True

```
1 not 1
```

False

```
1 not ""
```

True

```
1 not "python"
```

False

```
1 3 < 4 and 'a' != 'b'
```

True

```
1 3.14 > 5.5 and 'a' <= 'b'
```

False

```
1 3 < 4 or 'a' > 'b'
```


True

```
1 3.14 > 5.5 or 'a' == 'b'
```

False

연산자 우선 순위

- 연산자들의 우선 순위

순위	연산자 또는 기호
 높음	()
	**
	* / // %
	+ -
	< <= > >= == !=
	not
	and
	or
낮음	

연산자 우선 순위

- 연산자들의 우선 순위
 - 낮은 순위의 연산을 먼저 실행하려면 소괄호로 둘러싼다.

1	$2 + 3 * 4$
---	-------------

14

1	$(2 + 3) * 4$
---	---------------

20

1	$9 - 5 + 2 ** 2 * 1.5$
---	------------------------

10.0

1	$((9 - 5) + 2) ** (2 * 1.5)$
---	------------------------------

216.0

변수

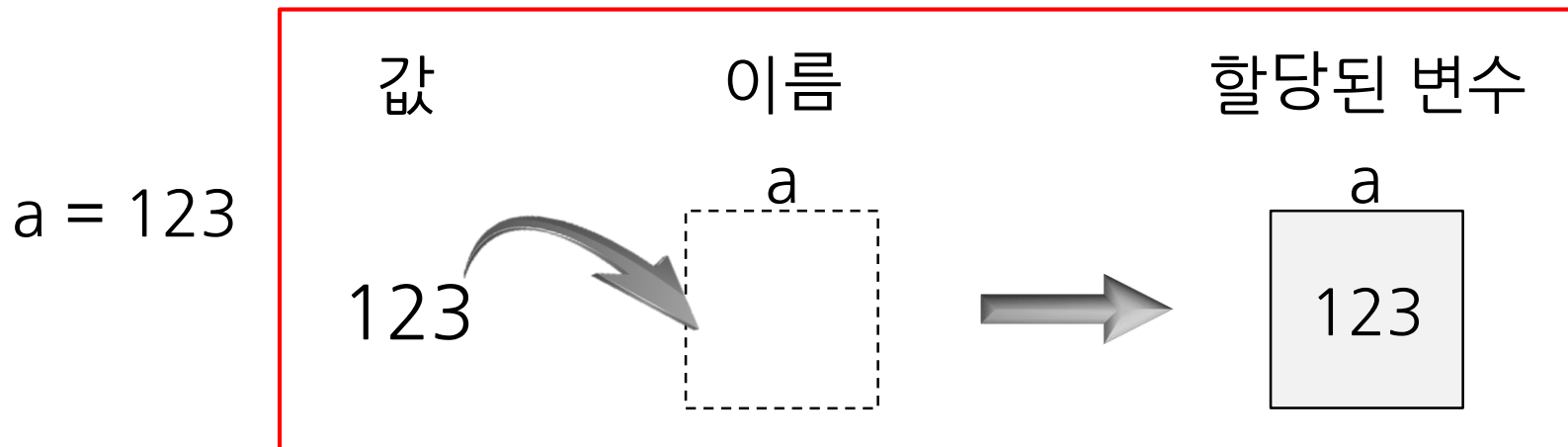
변수 (Variables)

- 변수의 개념
 - 연산의 대상이 되는 대부분의 값(value)들은 상황에 따라 얼마든지 변할 수 있다.
 - 이렇게 “변하는 값”이 바로 **변수**이다.
 - 변하는 값을 일관성 있게 관리하기 위해서, 그 값에 다른 자료와 구별(식별)할 수 있는 **이름(identifier)**을 부여한다.
 - 어떠한 값에 이름을 부여하고 나면, 그 때부터 그 값을 이름으로 식별하여 연산에 사용할 수 있다.

변수 (Variables)

- 변수 할당 (Assignment)
 - 어떠한 값에 이름을 부여하는 것을 **변수 할당**이라고 한다.
 - 변수 할당에는 연산자 **기호 '='**를 사용한다.

연산자	기능
=	연산자의 오른쪽에 위치한 연산 결과 값을 연산자의 왼쪽에 있는 이름(이 부여된 공간)에 저장한 다.

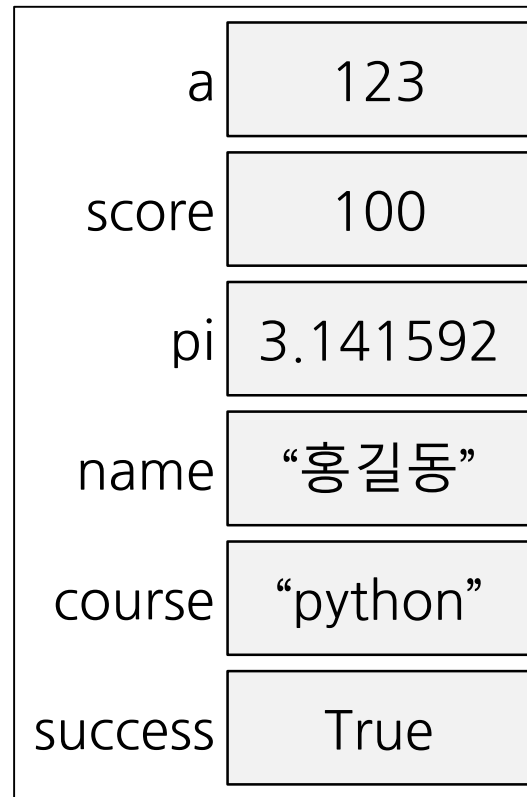


변수 (Variables)

- 변수 할당 (Assignment)

변수의 이름 = 연산 결과 값

```
1 a = 123
2
3 score = 100
4
5 pi = 3.141592
6
7 name = "홍길동"
8
9 course = "python"
10
11 success = True
```



메모리 (RAM)



변수 (Variables)

- 변수 할당 (Assignment)

변수의 이름 = 연산 결과 값

```
1 print(100)
```

100

```
1 score = 100
2 print(score)
```

100

```
1 score = 100
2 print(score)
3
4 score = 59
5 print(score)
```

100

59

```
1 a = 5
2 b = 3
3
4 c = a - b + 1
5 print(c)
```

3

```
1 s1 = "안녕하세요?"
2 s2 = " python"
3
4 s3 = s1 + s2
5 print(s3)
```

안녕하세요? python

변수 (Variables)

- 할당 연산자의 축약된 표현
 - 동일한 이름의 변수가 할당 연산자의 왼쪽과 오른쪽에 중복되어 나오면 축약해서 표현할 수 있다.

결과 값 31이
다시 apple로
할당된다.

```
apple = 30
```

```
apple = apple + 1
```

연산 결과는
31이다.

```
print(apple)
```

따라서 변수 apple의
값을 출력하면
31이 나온다.

31

변수 (Variables)

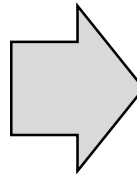
- 할당 연산자의 축약된 표현
 - 동일한 이름의 변수가 할당 연산자의 왼쪽과 오른쪽에 중복되어 나오면 축약해서 표현할 수 있다.

```
apple = 30
```

```
apple = apple + 1
```

```
print(apple)
```

31



```
apple = 30
```

```
apple += 1
```

```
print(apple)
```

31

이름이 apple로 동일한 변수가
할당 연산자의 왼쪽과 오른쪽에
중복되어 나왔다.

이러한 경우, 오른쪽 변수를 생략하고
산술 연산자를 할당 연산자의 왼쪽으로
붙여서 축약하여 쓸 수 있다.

변수 (Variables)

- 할당 연산자의 축약된 표현

축약 표현	원래 표현	의미
$a += v$	$a = a + v$	변수 a 의 값을 다른 변수 또는 값 v 와 더해서 그 값을 다시 변수 a 에 할당한다.
$a -= v$	$a = a - v$	변수 a 의 값에서 다른 변수 또는 값 v 을 뺀 뒤 그 값을 다시 변수 a 에 할당한다.
$a *= v$	$a = a * v$	변수 a 의 값을 다른 변수 또는 값 v 와 곱해서 그 값을 다시 변수 a 에 할당한다.
$a **= v$	$a = a ** v$	변수 a 의 값에 대해서 다른 변수 또는 값 v 의 거듭제곱을 구한 뒤 그 값을 다시 변수 a 에 할당한다.

변수 (Variables)

- 할당 연산자의 축약된 표현

축약 표현	원래 표현	의미
$a /= v$	$a = a / v$	변수 a 의 값을 다른 변수 또는 값 v 로 나눈 뒤 그 값을 다시 변수 a 에 할당한다.
$a //= v$	$a = a // v$	변수 a 의 값을 다른 변수 또는 값 v 로 나눠서 나오는 정수 몫을 다시 변수 a 에 할당한다.
$a \% = v$	$a = a \% v$	변수 a 의 값을 다른 변수 또는 값 v 로 나눠서 나오는 나머지를 다시 변수 a 에 할당한다.

변수 (Variables)

- 할당 연산자의 축약된 표현
 - 동일한 이름의 변수가 할당 연산자의 왼쪽과 오른쪽에 중복되어 나오면 축약해서 표현할 수 있다.

```
1 a = 5
2 b = 3
3 c = 1.5
4
5 a += 1
6 print(a)
7
8 a -= b
9 print(a)
10
11 a *= c
12 print(a)
```

6
3
4.5

```
1 s1 = "안녕하세요?"
2 s2 = " python"
3 d = 2
4
5 s1 += s2
6 print(s1)
7
8 s1 *= d
9 print(s1)
```

안녕하세요? python
안녕하세요? python안녕하세요? python

변수 (Variables)

- 변수 삭제하기

del 삭제하려는 변수의 이름

- 명령어 del을 이용하여 변수를 제거할 수 있다.
- 변수를 삭제하면 그 변수에 들어 있는 값도 더 이상 유효하지 않게 된다.

```
1 a = 123
2 print(a)
3
4 del a
5 print(a)
```

123

```
NameError                                Trac
<i python-input-307-03f268f663b8> in <module>
      3
      4 del a
--> 5 print(a)
```

NameError: name 'a' is not defined

변수 (Variables)

- 변수의 이름을 지을 때 문법 상 주의할 점
 - 첫 문자는 알파벳이거나 밑줄 기호로 시작해야 한다.
 - 두 번째 문자부터는 알파벳 또는 밑줄 기호 또는 0부터 9까지의 숫자를 사용할 수 있다.
 - 그 외의 특수 기호들은 사용할 수 없다.
 - 대문자와 소문자를 구별한다.
- 변수의 이름을 지을 때 의미 상 주의할 점
 - 변수 안에 있는 값이 어떤 의미의 자료인지 사람이 봤을 때 최대한 유추할 수 있도록 작명해야 한다.

주석 (Comment)

- 주석

- 실제로 실행되는 명령 구문이 아니라, 사람이 보고 내용을 쉽게 이해하기 위해서 기재하는 부분이다.
- 기호 '#' 뒤에 기재한 내용은 모두 주석으로 간주된다.

```
1  #문자열 2개
2  s1 = "안녕하세요?"
3  s2 = " python"
4
5  s3 = s1 + s2      #문자열 더하기
6
7  ##### 결과 출력하기
8  print(s3)
```

안녕하세요? python