



컴퓨팅사고와 데이터의 이해

11주차 이론
파이썬의 변수 개념

본 강의 자료는 대한민국 저작권법을 준수합니다. 본 강의 자료는 상명대학교 재학생들의 수업목적으로 제작·배포되는 것이므로, 수업목적으로 내려받은 강의 자료는 수업목적 이외에 다른 용도로 사용할 수 없으며, 다른 장소 및 타인에게 복제, 전송하여 공유할 수 없습니다. 이를 위반해서 발생하는 모든 법적 책임은 행위 주체인 본인에게 있습니다.

CONTENTS

컴퓨팅 사고와 데이터의 이해

학습 목표

1. 파이썬의 변수 개념을 이해 할 수 있다.
2. 변수의 다양한 종류와 활용 용도, 데이터 타입에 대해 이해 할 수 있다.
3. 각종 연산자 (사칙연산, 대입연산, 콤마연산, 복합대입연산 등) 을 이해 할 수 있다.

변수

• 정의

- 변수는 어떤 값을 저장하기 위한 메모리 공간. '그릇'

• 가장 많이 사용하는 변수 형식

- 불형(Boolean, True 또는 False 저장), 정수형, 실수형, 문자열

```
boolVar, intVar, floatVar, strVar=True, 0, 0.0, ""
```



boolVar
불형 변수



intVar
정수형 변수



floatVar
실수형 변수



strVar
문자열 변수

- `type()` 함수는 변수의 종류를 확인하는 함수

```
type(boolVar), type(intVar), type(floatVar), type(strVar)
```

```
(<class 'bool'>, <class 'int'>, <class 'float'>, <class 'str'>)
```

• 변수명 규칙

- 대·소문자를 구분한다(myVar와 MyVar는 다른 변수).
- 문자, 숫자, 언더바(_)를 포함할 수 있다. 하지만 숫자로 시작하면 안 된다(var2(O), _var(O), var_2(O), 2Var(X)).
- 예약어는 변수명으로 쓰면 안 된다. 파이썬의 예약어 (키워드) 는 True, False, None, and, or, not, break, continue, return, if, else, elif, for, while, except, finally, gloval, import, try 등이다.

여기서 잠깐 - 키워드 (Keyword)

- 키워드 (keyword)

특별한 의미가 부여된 단어

파이썬에서 이미 특정 의미로 사용하기로 예약해 놓은 것

프로그래밍 언어에서 이름 정할 때 똑같이 사용할 수 없음 (변수명 사용 금지, 함수와 혼돈 금지)

대소문자 구별

오른쪽 코드로 특정 단어가 파이썬 키워드인지 확인 가능

```
>>> import keyword
>>> print(keyword.kwlist)
```

False	None	True	and	as	assert
break	class	continue	def	del	elif
else	except	finally	for	from	global
if	import	in	is	lambda	nonlocal
not	or	pass	raise	return	try
while	with	yield			

여기서 잠깐 - 주석 (Comment)

- 주석 (comment)

프로그램 진행에 영향 주지 않는 코드

프로그램 설명 위해 사용

한줄 주석은 # 기호를 주석으로 처리하고자 하는 부분 앞에 붙임

문장 주석은 """ 기호를 주석으로 처리하고자 하는 문장 부분 앞 뒤에 붙임

```
7 cursor = conn.cursor()
8 """
9 #쿼리 작성
10 cursor.execute("select top 10 mem_id, mem_name from member_master")
11
12 row = cursor.fetchone()
13 while row:
14     #print("mem_id = %s, mem_name = %s" % (row[0], row[1]))
15     print("mem_id2 = " + str(row[0]) + ", mem_name = " + str(row[1]))
16     row = cursor.fetchone()
17
18 """
19
20 conn.close()
21 |
```

```
8
9 #쿼리 작성
10 cursor.execute("select top 10 mem_id, mem_name from member_master")
11
12 row = cursor.fetchone()
13 while row:
14     #print("mem_id = %s, mem_name = %s" % (row[0], row[1]))
15
16     print("mem_id2 = " + str(row[0]) + ", mem_name = " + str(row[1]))
17     row = cursor.fetchone()
18
19 conn.close()
20
```


변수 (대입연산자)

- 대입 연산자의 대표적 경우 : 변수 or 변수명

값을 저장할 때 사용하는 식별자

파이썬에서는 숫자뿐만 아니라 모든 자료형을 저장할 수 있음 (데이터 타입 구분이 없음)

대입 연산자를 사용

대입 연산자

```
>>> pi = 3.14159265
>>> pi
3.14159265
```

변수 = 값

값을 변수에 할당합니다.

변수 (대입연산자)

- 변수의 활용

변수를 선언하는 방법

변수를 생성

변수에 값을 할당하는 방법

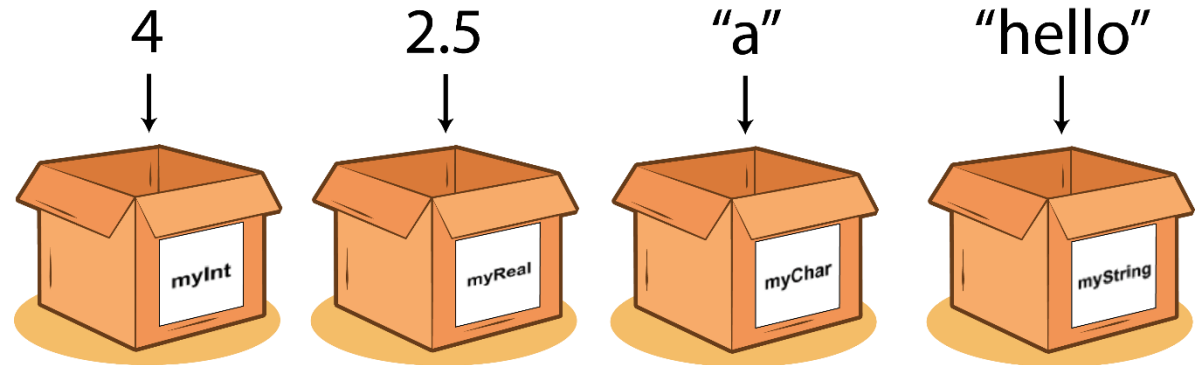
변수에 값을 넣음

= 우변을 값을 좌변에 할당

변수를 참조하는 방법

변수에서 값을 꺼냄

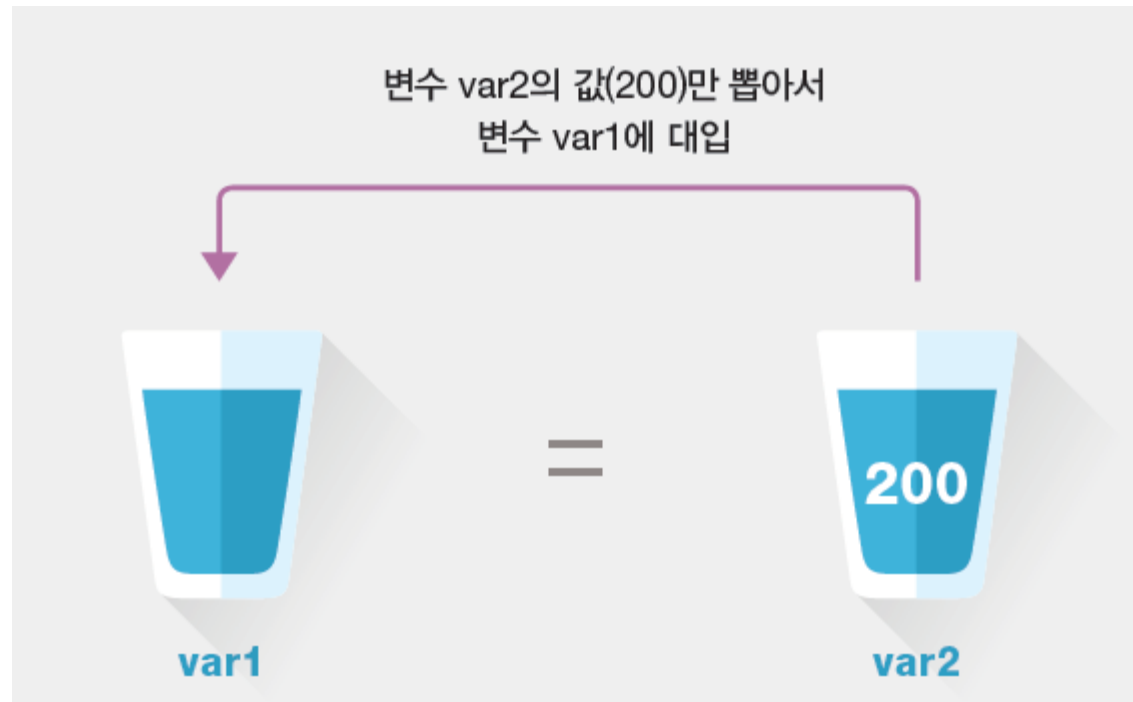
변수 안에 있는 값을 사용



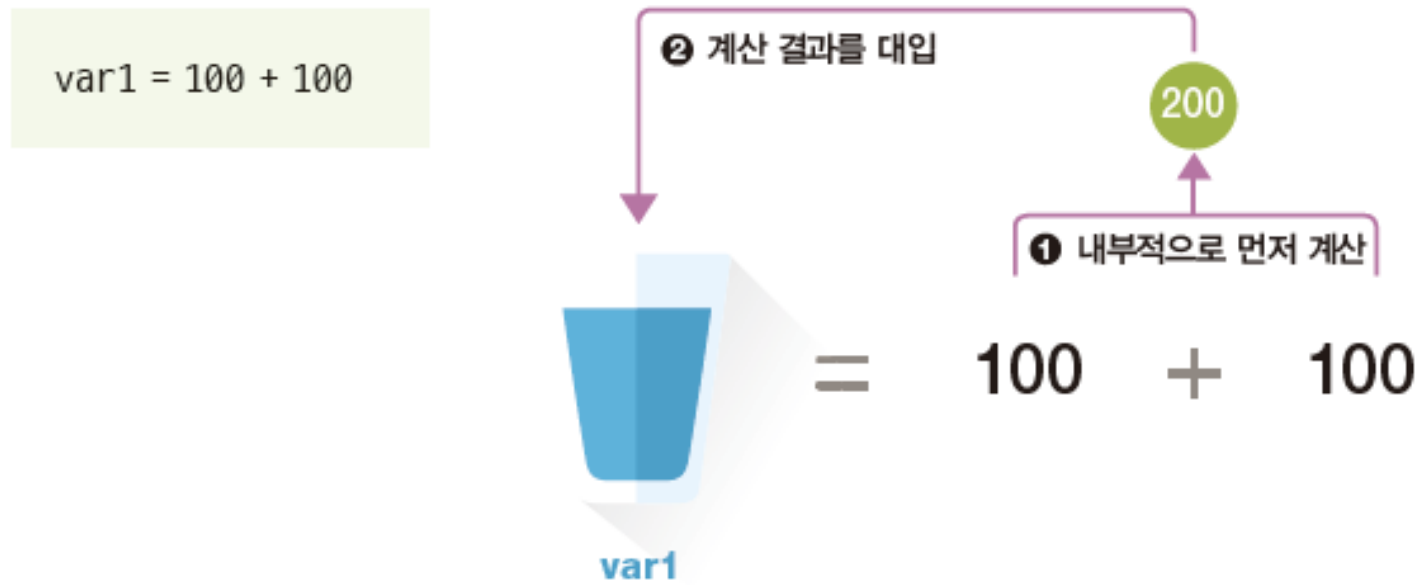
“ 대입연산자(=)와 동등연산자(==)는 혼돈이 될 수 있으므로 항상 인식을 해야한다”

- 변수의 값을 변수에 넣기

```
var2 = 200  
var1 = var2
```



- 변수에는 계산 결과를 넣을 수도 있음

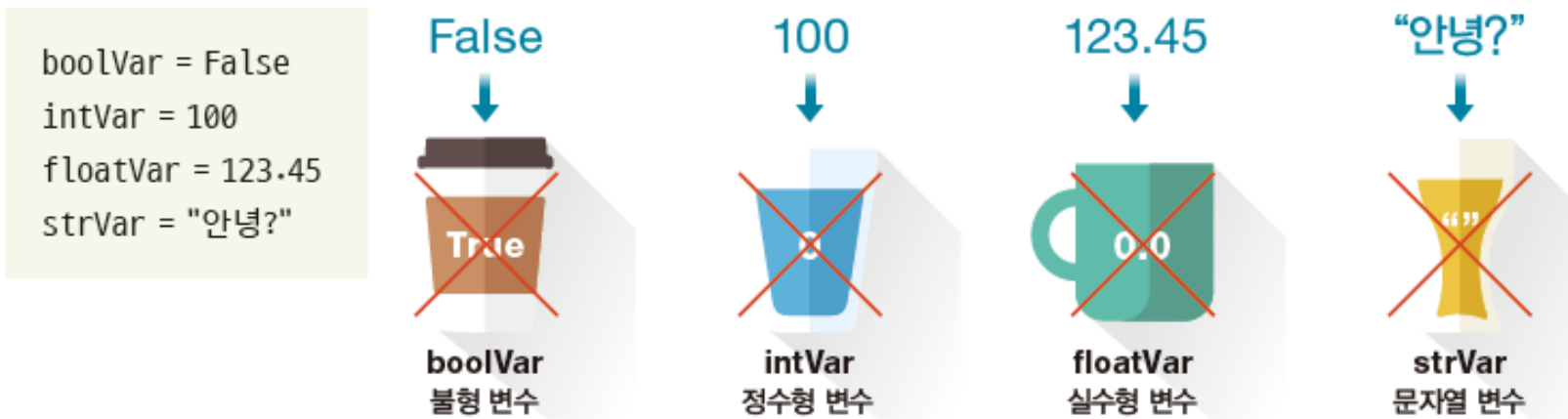


- 변수에는 숫자와 변수의 연산을 넣을 수도 있음

```
var1 = var2 + 100
```

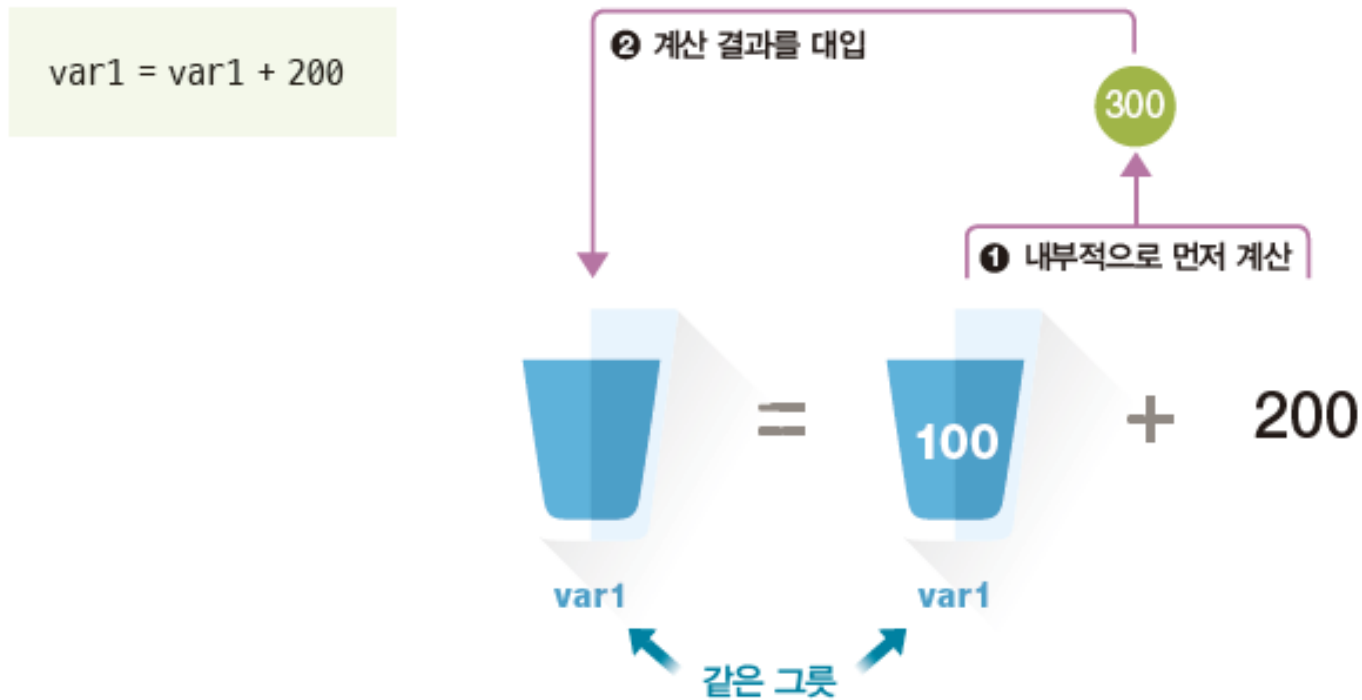


- 변수는 값을 담으면(대입하면) 사용 가능. 변수에 있던 기존 값은 없어지고 새로 입력한 값으로 변경됨



변수 (복합대입연산자)

- 변수에 연산 결과를 자신의 값으로 다시 대입하는 방식



변수 (복합대입연산자)

연산자	동치	연산자	동치
$a += b$	$a = a + b$	$a **= b$	$a = a ** b$
$a -= b$	$a = a - b$	$a \&= b$	$a = a \& b$
$a *= b$	$a = a * b$	$a = b$	$a = a b$
$a /= b$	$a = a / b$	$a \wedge= b$	$a = a \wedge b$
$a \%= b$	$a = a \% b$	$a <<= b$	$a = a << b$
$a //= b$	$a = a // b$	$a >>= b$	$a = a >> b$

- 파이썬에서 변수의 데이터 형식은 값을 넣는 순간마다 변경될 수 있는 유연한 구조

```
myVar = 100      # 정수형 변수를 생성(국 그릇 생성)
type(myVar)      # <class 'int'>가 출력
myVar = 100.0    # 이 순간에 실수형 변수로 변경(밥 그릇으로 변경)
type(myVar)      # <class 'float'>가 출력
```

- 대입 연산자의 왼쪽에는 무조건 변수만 올 수 있고, 오른쪽에는 무엇이든(값, 변수, 수식, 함수 등) 올 수 있음

10 = 100

그릇
없음



대입 안 됨



그림 3-14 왼쪽에 값을 넣을 그릇이 없음

변수(그릇) = 100



대입 가능



coma연산자

- 두개 이상의 데이터 타입을 서로 분산 시켜주거나 연결시켜주는 연산자
- 스크립트 모드로 아래 내용을 타이핑하고 실행 시켜 보자

```
a, b ,c = 10, 20, 30
```

```
print (a)
```

```
print (b)
```

```
print (c)
```

```
d = "안녕하세요"
```

```
e = "Hello"
```

```
print (d, e)
```

```
print ()
```

```
print (d + e)
```

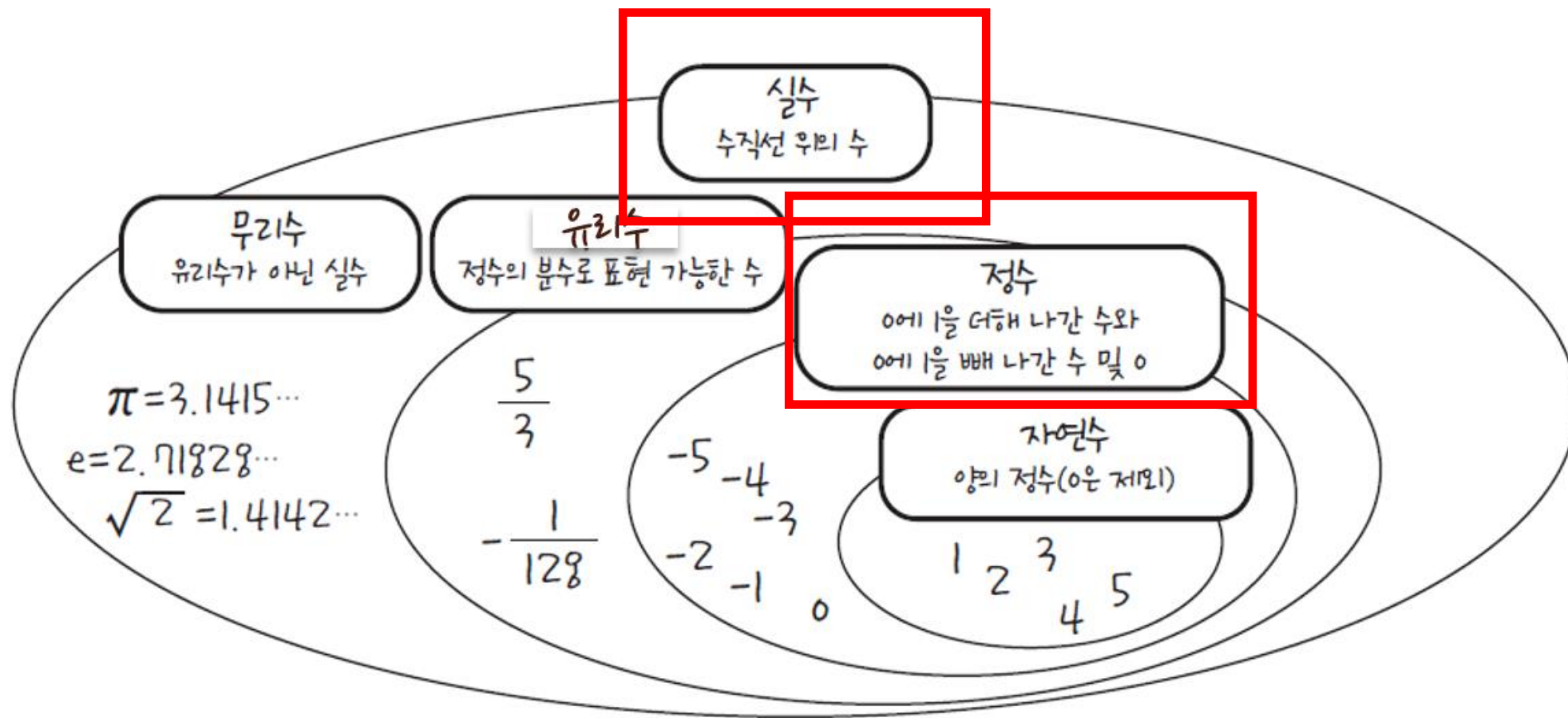
```
10
20
30
안녕하세요 Hello

안녕하세요Hello
>>> |
```

```
>>> print ("여기에서 숫자는", 32 / 6, "입니다.")
여기에서 숫자는 5.333333333333333 입니다.
>>>
```

데이터 타입 (Data Type)

수의 체계



- 데이터 타입은 아니지만 컴퓨팅 사고를 설계하기 위해 생각해 봐야 할 수의 종류

변수 (Variable Number)

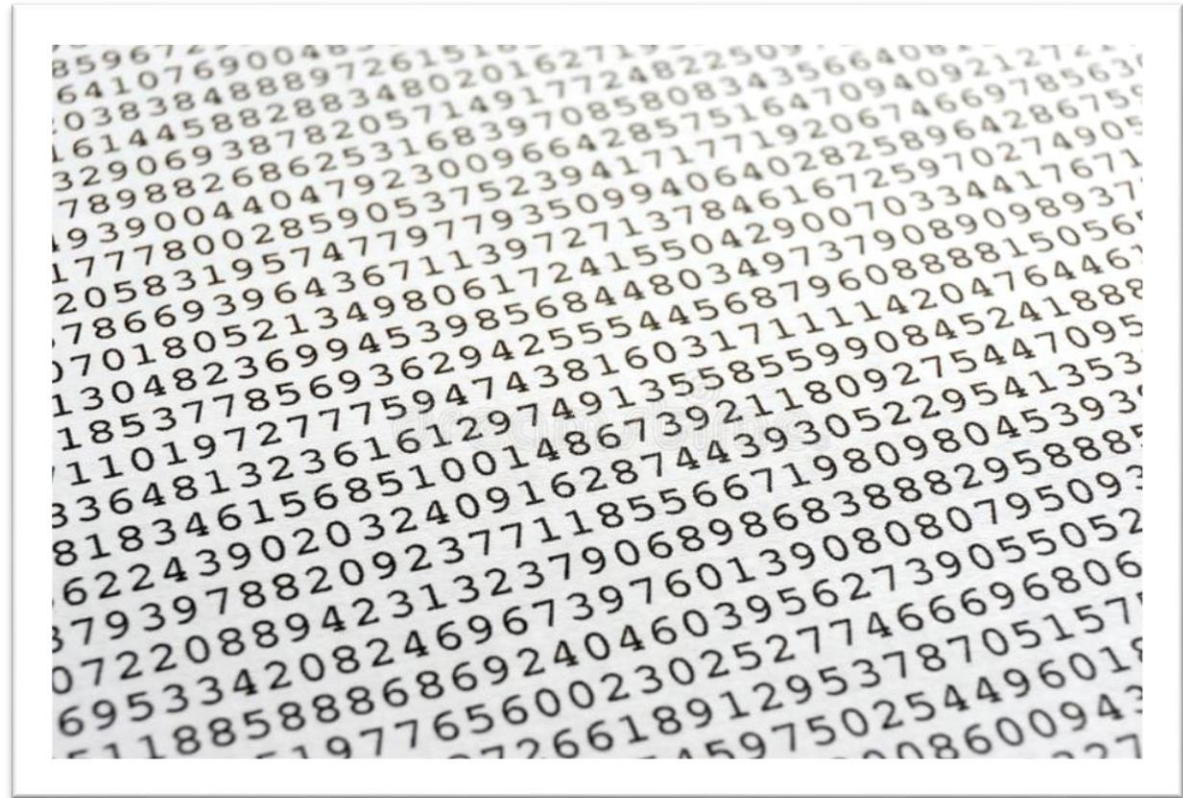
상수 (Constant Number)

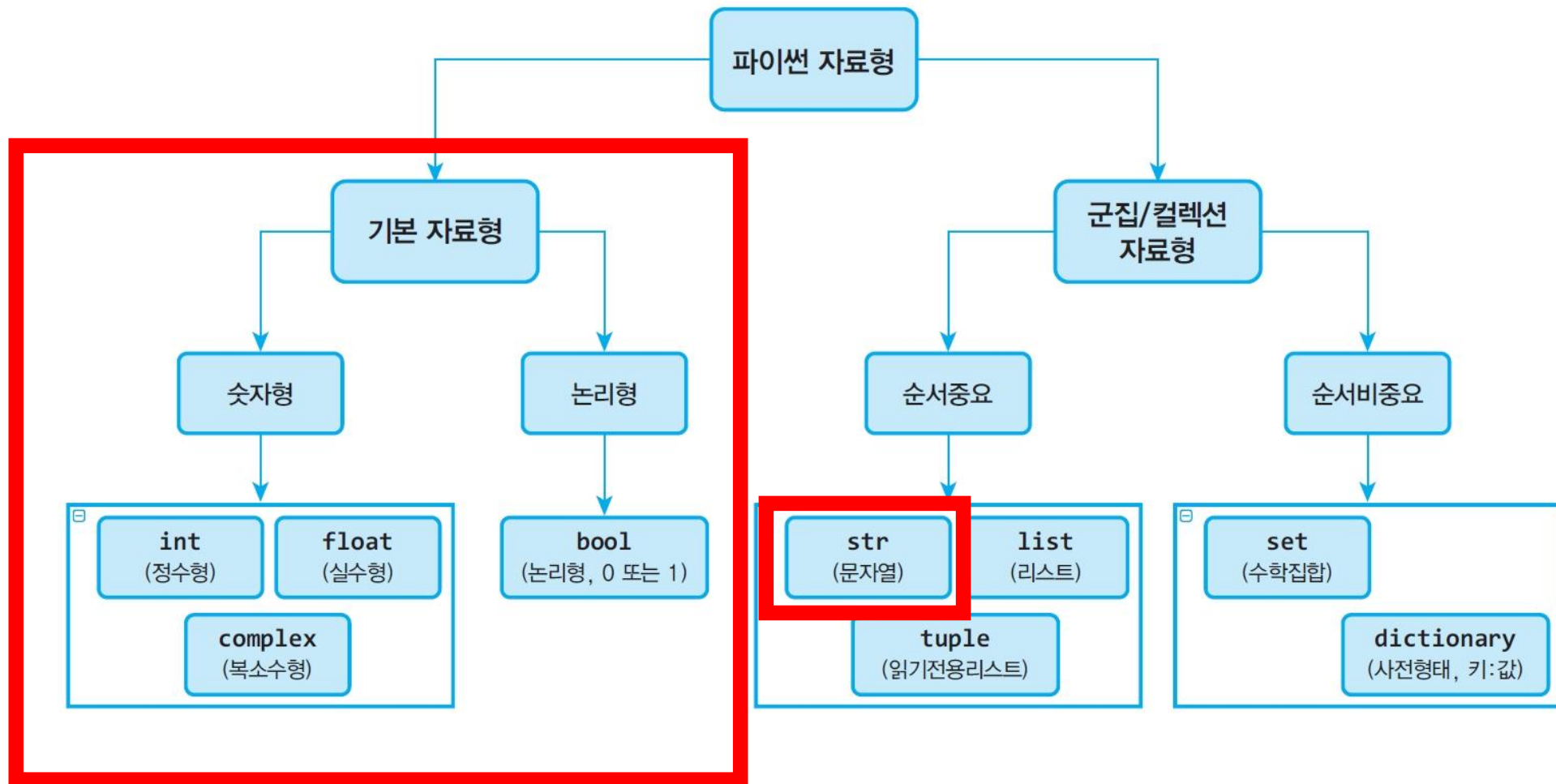
난수 (Random Number)

배수 (Multiple)

약수 (Divisor)

소수 (Prime Number)





- 정수 데이터 형식 (Integer)

- 정수형은 소수점이 없는 데이터임. (기본적인 정수형 100, -123, 0 등)
- 파이썬은 변수의 선언이 없으며 변수에 값을 넣는 순간 변수의 데이터 형식이 결정됨.

```
a = 123  
type(a)
```

```
<class 'int'>
```

- int는 기본적인 정수 데이터 형식이며 int 크기의 제한이 없음

```
a = 100 ** 100  
print(a)
```

```
1000000 ~~~ 00000
```

- 정수 데이터 형식 (Integer)
 - 16진수, 8진수, 2진수도 사용할 수 있다.

```
a = 0xFF  
b = 0o77  
c = 0b1111  
print(a, b, c)
```

출력 결과

255 63 15

- 정수형

자연수를 포함한 소수점이 없는 숫자

0, 1, 273, -52

정수 (integer) : 양수, 0, 음수

숫자를 만들기 위해서는 단순히 숫자 입력하면 됨

int () 함수로 타입 변환

```
>>> print(273)
273
>>> print(52.273)
52.273
```

- 실수 데이터 형식 (Floating Point Number)

- 실수형은 3.14, -2.7처럼 소수점이 있는 데이터

```
a=3.14  
b=3.14e5  
print(a, b)
```

```
3.14 314000.0
```

데이터 타입 - 실수

- 실수형

소수점이 있는 숫자

0.0, 52.273, -1.2

실수 (floating point, 부동 소수점)

숫자를 만들기 위해서는 단순히 숫자 입력하면 됨

float () 함수로 타입 변환

```
>>> print(273)
273
>>> print(52.273)
52.273
```

• 정수형과 실수형

- 사칙연산인 $+$, $-$, $*$, $/$ 를 수행

```
a = 10; b = 20  
print(a + b, a - b, a * b, a / b)
```

```
30 -10 200 0.5
```

- 제곱을 의미하는 $**$, 나머지를 구하는 $\%$,
나눈 후에 소수점을 버리는 $//$ 연산자도 사용할 수 있다.

```
a, b = 9, 2  
print(a ** b, a % b, a // b)
```

출력 결과

```
81 1 4
```

데이터 타입 - 정수, 실수 : 사칙연산

연산 기호	뜻	예시	결과
+	더하기	$7+4$	11
-	빼기	$7-4$	3
*	곱하기	$7*4$	28
/	나누기	$7/4$	1.75
**	제곱 (같은 수를 여러 번 곱함)	$2**3$	8 (2를 세 번 곱함 $2*2*2$)
//	정수로 나누었을 때의 몫	$7//4$	1 (나눗셈의 몫)
%	정수로 나누었을 때의 나머지	$7\%4$	3 (나눗셈의 나머지)
()	다른 계산보다 괄호 안을 먼저 계산	$2*(3+4)$	14

- 불 데이터 형식 (Bool)

- 참(True)이나 거짓(False)만 저장
- 불형은 단독으로 사용하기보다는 if 조건문 이나 while 반복문 등과 함께 주로 사용

```
a=True  
type(a)
```

```
<class 'bool'>
```

```
a=(100==100)  
b=(10>100)  
print(a, b)
```

```
True False
```

- Boolean

불린 / 불리언 / 불

True와 False 값만 가질 수 있음 (참/ 거짓)

```
>>> print(True)
True
>>> print(False)
False
```

관계 (비교) 연산자와 논리 연산자를 통해 만들 수 있음 (연산자에서 실습)

숫자 또는 문자열에 적용

문자열에도 비교 연산자 적용 가능 (A~Z , 한글도 적용 가능)

bool () 함수로 타입 변환

• 문자열 데이터 형식 (String)

- 문자열은 양쪽을 큰따옴표(“)나 작은따옴표(‘)로 감싸야 함

```
a = "파이썬 만세"  
a  
print(a)  
type(a)
```

```
'파이썬 만세'  
파이썬 만세  
<class 'str'>
```

```
"작은따옴표는 ' 모양입니다."  
'큰따옴표는 " 모양입니다.'
```

```
"작은따옴표는 ' 모양입니다."  
'큰따옴표는 " 모양입니다.'
```


• 문자열 데이터 형식 (String)

- 역슬래시(\) 뒤에 큰따옴표나 작은따옴표를 사용해도 글자로 인식한다.

```
a = "이건 큰따옴표 \" 모양."  
b = '이건 작은따옴표 \' 모양.'  
print(a, b)
```

출력 결과

이건 큰따옴표 " 모양. 이진 작은따옴표 ' 모양.

- 문자열을 여러 줄로 넣으려면 중간에 \n을 포함시키면 된다.

```
a = '파이썬 \n만세'  
print(a)
```

출력 결과

파이썬
만세

- 문자열 (string)

따옴표(큰 따옴표 또는 작은 따옴표)로 둘러싸 입력하는, 글자가 나열된 것

파이썬에서는 문자와 문자열을 따로 구분하지 않는다.

변수명 처리 가능

str () 함수로 타입 변환

"Hello"

'String'

'안녕하세요'

"Hello Python Programming"

- 문자열 (string)

큰따옴표로 문자열 만들기

```
>>> print("안녕하세요")  
안녕하세요
```

작은따옴표로 문자열 만들기

```
>>> print('안녕하세요')  
안녕하세요
```

- 문자열 연산자

숫자에는 사칙연산 연산자를, 집합에는 여러 집합 연산자 적용 가능

각 자료는 사용할 수 있는 연산자 정해져 있음

문자열 연결 연산자 : + 또는 , / 더하기와 같은 기호이나 다른 수행임에 주의

"문자열" + "문자열"

문자열 연결 연산자

```
>>> print(52, 273, "Hello")
```

```
52 273 Hello
```

```
>>> print("안녕" + "하세요")
```

```
안녕하세요
```

```
>>> print("안녕하세요" + "!")
```

```
안녕하세요!
```

- 문자열 연산자 오류

문자열과 숫자 사이에는 사용할 수 없음

문자열은 문자끼리, 숫자는 숫자끼리 연결

문자열과 숫자 연결하여 연산하려면 큰따옴표 붙여 문자열로 인식하게 함

```
>>> print("안녕하세요" + 1)
```

 오류

```
TypeError: can only concatenate str (not "int") to str
```

- 문자열 연산자

문자열 반복 연산자 : *

문자열을 숫자와 * 연산자로 연결

```
>>> print("안녕하세요" * 3)  
안녕하세요안녕하세요안녕하세요
```

```
>>> print(3 * "안녕하세요")  
안녕하세요안녕하세요안녕하세요
```

입력 함수 input()의 변환 : cast

입력함수 input()의 변환 : cast

- input() 함수

명령 프롬프트에서 사용자로부터 데이터 입력받을 때 사용

input() 함수로 사용자 입력받기

프롬프트 함수 : input 함수 괄호 안에 입력한 내용

```
>>> input("인사말을 입력하세요> ")
```

블록 (block) : 프로그램이 실행 중 잠시 멈추는 것

인사말을 입력하세요> _ → 입력 대기를 알려주는 커서입니다. 커서는 프로그램에 따라 모양이 다를 수 있습니다.

명령 프롬프트에서 글자 입력 후 [Enter] 클릭

```
인사말을 입력하세요> 안녕하세요 [Enter]
```

```
'안녕하세요'
```


입력함수 input()의 변환 : cast

- input() 함수

input 함수의 결과로 산출 (리턴값)

다른 변수에 대입하여 사용 가능

```
>>> string = input("인사말을 입력하세요> ")
인사말을 입력하세요> 안녕하세요 [Enter]
>>> print(string)
안녕하세요
```

입력함수 input()의 변환 : cast

- input() 함수

input() 함수의 입력 자료형

type() 함수로 자료형 알아보기

```
>>> print(type(string))  
<class 'str'>
```

```
>>> number = input("숫자를 입력하세요> ")  
숫자를 입력하세요> 12345 [Enter]  
>>> print(number)  
12345
```

```
>>> print(type(number))  
<class 'str'>
```

input() 함수는 사용자가 무엇을 입력해도 결과는 무조건 문자열 자료형

입력함수 input()의 변환 : cast

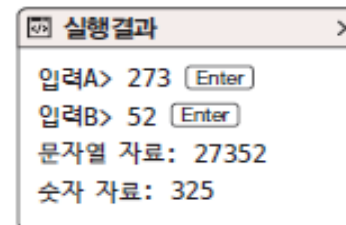
- input() 함수

캐스트 (cast) : input() 함수의 입력 자료형은 항상 문자열이므로 입력받은 문자열을 숫자 연산에 활용하기 위해 숫자로 변환

int() 함수 : 문자열을 int 자료형으로 변환.

float() 함수 : 문자열을 float 자료형으로 변환

```
01 string_a = input("입력A> ")
02 int_a = int(string_a)
03
04 string_b = input("입력B> ")
05 int_b = int(string_b)
06
07 print("문자열 자료:", string_a + string_b)
08 print("숫자 자료:", int_a + int_b)
```



실행결과

입력A> 273 Enter

입력B> 52 Enter

문자열 자료: 27352

숫자 자료: 325

감사합니다.