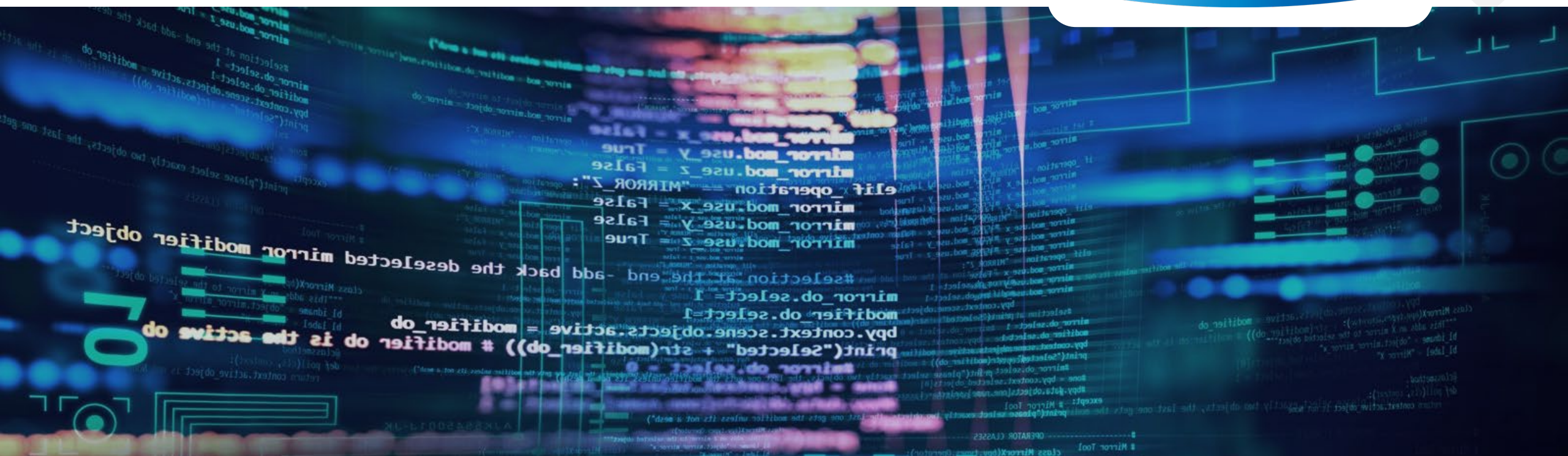


Lecture 2

자료형 및 변수

KAIST





➤ 객체란?

- 여러 속성과 행동을 가지고 있는 데이터
- 데이터의 추상화
- 모든 객체는 identity와 값(value)을 가지고 있고 어느 자료형 (객체 종류, type)에 해당된다
- 자료형에 따라 객체가 가지는 속성과 행동이 달라진다

객체

identity

type

value



- 파이썬은 객체지향적인 언어다 (object-oriented language)
- 객체의 종류는 다양하다
- 우리는 이미 두 자료형을 써봤다

```
1 print("Hello, world!")
```

```
1 print(3 + 3)
```

객체

identity

type

value



- 문자열 타입은 작은 따옴표나 큰 따옴표로 표현된다
- 문자열 안에 들어갈 수 있는 특수 문자가 여러가지가 있다

\n – 줄바꿈

```
1 print("Hello \n world!")
```

```
Hello  
world!
```



- 문자열 타입은 작은 따옴표나 큰 따옴표로 표현된다
- 문자열 안에 들어갈 수 있는 특수 문자가 여러가지가 있다

\n - 줄바꿈

\t - 탭

```
1 print("Hello \t world!")
```

```
Hello world!
```



- 문자열 타입은 작은 따옴표나 큰 따옴표로 표현된다
- 문자열 안에 들어갈 수 있는 특수 문자가 여러가지가 있다

\n - 줄바꿈

\t - 탭

\\ - 백슬래시

```
1 print("Last\next!")  
2 print("Last\\next!")
```

```
Last  
ext!  
Last\next!
```



- 문자열 타입은 작은 따옴표나 큰 따옴표로 표현된다
- 문자열 안에 들어갈 수 있는 특수 문자가 여러가지가 있다

\n - 줄바꿈

\t - 탭

\\ - 백슬래시

' - 작은따옴표

" - 큰따옴표

```
1 print("\"No!\" he said.")
```

```
"No!" he said.
```



➤ 숫자에는 여러 타입이 있다

integer – 정수형

float – 실수형

```
1  # integer 결과
2  print(3 + 3)
3
4  # float 결과
5  print(17 / 6)
```

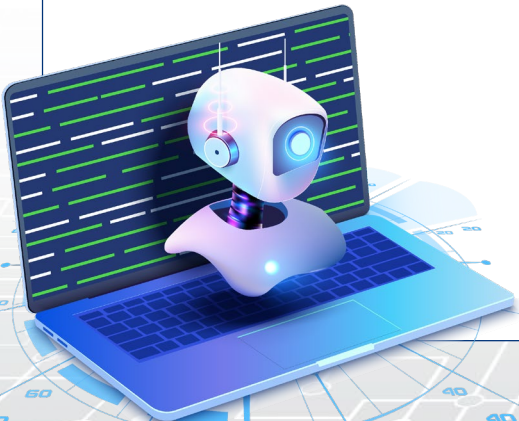
- 출력할 때 **소수점**이 있으면 **float**, 없으면 **integer**이다
- **/ 연산자**는 무조건 **실수형**을 반환한다
- 다른 연산자는 피연산자가 모두 정수형이면 정수형을 반환하고
한 피연산자라도 실수형이면 실수형을 반환한다



Boolean: 참/거짓을 표현하는 자료형

- True – 참
- False – 거짓

```
1  # Boolean
2  print(True)
3  print(False)
```





NoneType - 값이 없다는 것을 의미하는 자료형

- NoneType 자료형의 객체들은 모두 None의 값을 가지고 있다
- 함수가 반환하는 값이 없을 때 파이썬이 반환값을 None으로 설정을 해준다
 - print() 함수의 반환값은 None이다

```
1 print(print("Hello world!"))
```

```
Hello world!  
None
```





➤ type() 함수는 인수가 어느 자료형인지를 반환하는 함수이다

```
1 print(type(1))
2 print(type(1.0))
3 print(type(True))
4 print(type(False))
5 print(type("hello"))
6 print(type(None))
7 print(type("1.0"))
```

```
<class 'int'>
<class 'float'>
<class 'bool'>
<class 'bool'>
<class 'str'>
<class 'NoneType'>
<class 'str'>
```



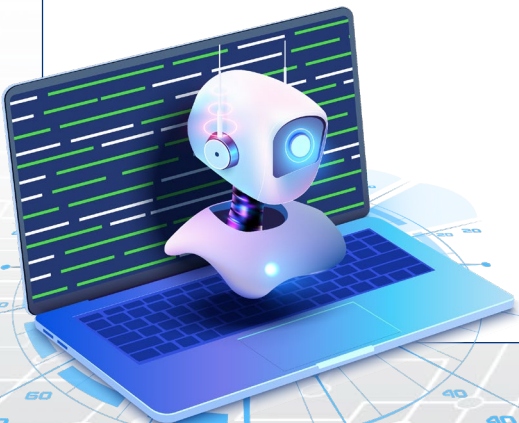
➤ input() 함수는 항상 문자열을 반환한다

```
1 print(type(input()))
```

```
1.5  
<class 'str'>
```

```
False  
<class 'str'>
```

```
None  
<class 'str'>
```





x in str

x 문자열이 str 문자열 안에 있는지 Boolean을 반환한다

x 문자열이 str 문자열 안에 없는지 Boolean을 반환한다

x not in str



str1 + str2

두 문자열을 합쳐 만든 문자열을 반환한다

str 문자열이 number번 합쳐 만든 문자열을 반환한다

str * number / number * str



len(s)

문자열의 길이를 정수형으로 반환한다

인수의 타입을 문자형으로 바꿔 객체 반환

str(x)

* 형변환 (type casting)

:인수의 타입을 바꿔서 다른 자료형의 객체를 반환하는 함수



- 메소드는 각 객체 타입을 위해 기본적으로 제공되는 함수를 뜻한다
- 메소드는 (객체 + . + 함수)의 형태로 부른다

str1.count(str2)

•..... str1에 str2가 몇 번 나오는지 integer 객체를 반환한다





- 메소드는 각 객체 타입을 위해 기본적으로 제공되는 함수를 뜻한다
- 메소드는 (객체 + . + 함수)의 형태로 부른다

str.islower()

..... 문자열에 있는 모든 영문자가 소문자인지 Boolean을 반환한다

모든 영문자가 소문자로 바뀐 문자열을 반환한다

str.lower()



- 메소드는 각 객체 타입을 위해 기본적으로 제공되는 함수를 뜻한다
- 메소드는 (객체 + . + 함수)의 형태로 부른다

str.isupper()

..... 문자열에 있는 모든 영문자가 대문자인지 Boolean을 반환한다

모든 영문자가 대문자로 바뀐 문자열을 반환한다

str.upper()



- 메소드는 각 객체 타입을 위해 기본적으로 제공되는 함수를 뜻한다
- 메소드는 (객체 + . + 함수)의 형태로 부른다

str1.endswith(str2)



str1이 str2로 끝나는지 Boolean을 반환한다

str1이 str2로 시작하는지 Boolean을 반환한다



str1.startswith(str2)



- 메소드는 각 객체 타입을 위해 기본적으로 제공되는 함수를 뜻한다
- 메소드는 (객체 + . + 함수)의 형태로 부른다

str.replace(str2, str3)



str1 안에 있는 모든 str2를 str3으로 대체한 문자열을 반환한다

문자열 맨 왼쪽과 맨 오른쪽에
연속으로 나오는 공백을 없앤 문자열을 반환한다



str.strip()

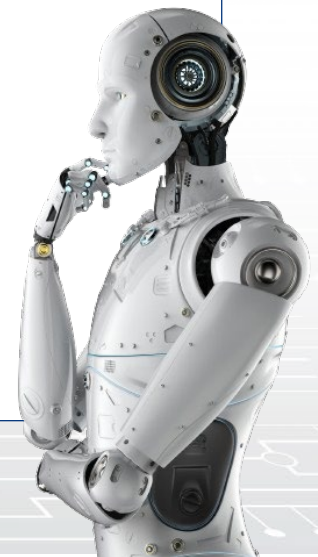


abs(x)

x의 절대값을 반환한다

pow(x, y)

x의 y제곱을 반환한다
(**연산자와 같다)



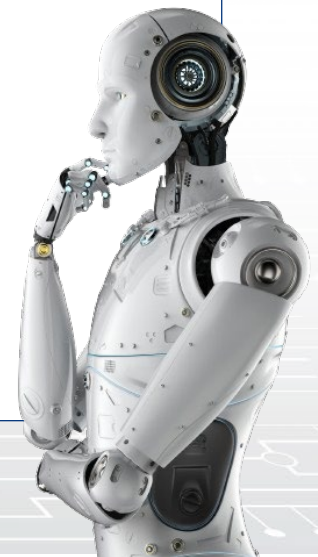


flt.is_integer()

flt 실수형 객체가
정수의 값을 가지고 있는지
Boolean을 반환한다

형변환 (type casting)

int(x)
float(x)





➤ bool 1 or bool 2

- bool 1의 진리값이 True면 bool 2를 확인하지 않고 bool 1을 반환한다
- bool 1의 진리값이 False면 bool 2가 반환된다

bool 1	bool 2	bool 1 or bool 2	bool 1 and bool 2
True	True	True	True
True	False	True	False
False	True	True	False
False	False	False	False



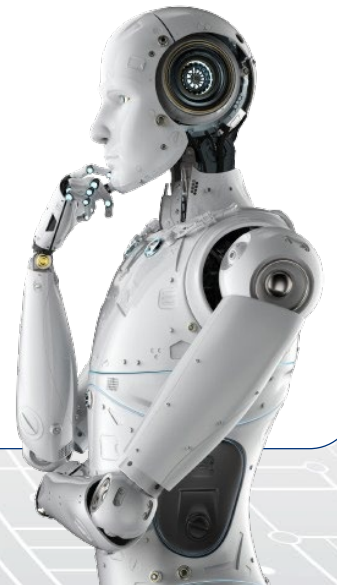
➤ bool 1 and bool 2

- bool 1의 진리값이 False면 bool 2를 확인하지 않고 bool 1을 반환한다
- bool 1의 진리값이 True면 bool 2가 반환된다

bool 1	bool 2	bool 1 or bool 2	bool 1 and bool 2
True	True	True	True
True	False	True	False
False	True	True	False
False	False	False	False

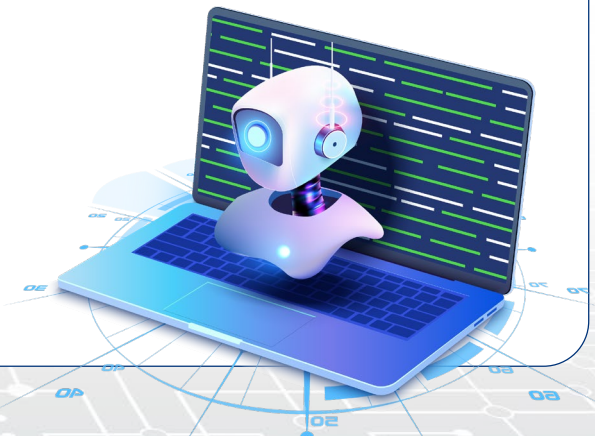


- **not bool** – bool이 **False**면 **True**, bool이 **True**면 **False**를 반환한다
- 우선순위: not, and, or
- 피연산자가 Boolean이 아니더라도 **진리값을 가진다**
 - 0인 int나 float는 False, 0 아니면 True
 - 문자열의 길이가 0이면 False, 아니면 True
 - None은 False
- **bool()** (형변환)





- $<, <=, >, >=, ==, !=$
- Boolean을 반환한다
- Boolean 연산자보다 우선순위가 높다





➤ 객체에 이름(식별자 - identifier)을 줄 수 있다
(이 과정을 대입문(assignment)이라고 부른다)

- 문자, 숫자, _ 사용 가능
- 대소문자 구분
- 이름의 첫 글자로 숫자 사용 불가
- _ 외에 특수 문자 사용 불가
- 예약어(keyword) 사용 불가
True, False, None, and, or, is, not, ...

➤ 이름이 가리키는 객체가 바뀔 수 있어 이름이 있는 객체를 변수라고도 부른다



➤ 증분 대입문 (augmented assignment)

- 자주 쓰이는 연산자와 대입문을 축소한 연산자

$$a += b \Rightarrow a = a + b$$

$$a -= b \Rightarrow a = a - b$$

$$a *= b \Rightarrow a = a * b$$

$$a /= b \Rightarrow a = a / b$$





- f-string: 문자열 안에 중괄호를 넣어 코드의 반환값을 출력할 수 있다
- {출력할 값:출력이 차지할 공간수}로 출력을 더 균일하게 할 수 있다
- <, >, ^로 텍스트 정렬도 정할 수 있다
- float의 경우 몇 자리까지 표현이 되는지도 조절할 수 있음

```
1 my_number = 5.342346
2 print(f"{my_number}")
3 print(f"{my_number:<10}")
4 print(f"{my_number:^10}")
5 print(f"{my_number:>10}")
6 print(f"{my_number:10.3f}")
```

```
5.342346
5.342346
5.342346
5.342346
5.342
```



- 중괄호 안에 중첩된(nested) 중괄호로 문자열이 차지하는 공간 수나 소수점 자리를 변수의 값으로 설정할 수 있다

```
1 my_number = 5.342346
2 width = 10
3 precision = 3
4 print(f"{my_number:{width}.{precision}f}")
```

5.342

