CS101 – 프로그램에서 사용하는 객체와 객체의 형태

Lecture 5

School of Computing
KAIST

학습 목표:

- 객체 (objects)와 형태를(types) 이해할 수 있다.
- 변수 (variables)를 이해할 수 있다.

객체

프로그램은 실행 중에 여러 데이터를 사용합니다.
Python 프로그램에서 사용하는 각각의 데이터는 **객체(Object)**라 부릅니다.

객체의 크기는 아주 작을 수도 있고 (숫자 3), 매우 클 수도 있습니다 (사진 파일).

모든 객체는 **형태(Type)**를 가지고 있습니다.
형태는 객체를 이용해 할 수 있는 일을 결정합니다.

Python 동물원

Python 프로그램이 동물원이라고 생각해 봅시다.
객체를 만드는 일은, 동물을 만드는 일에 빗대어 말할 수 있습니다.
동물이 무엇을 할 수 있는지는 동물의 종류(형태)에 따라 결정됩니다.
(예. 새는 날 수 있고, 물고기는 헤엄칠 수 있고, 코끼리는 무거운 물건을 들 수 있습니다.)
동물이 더 이상 사용되지 않으면, 그 동물은 죽습니다 (사라집니다).

객체 생성

객체는 다음과 같은 방법으로 만들 수 있습니다.

숫자: 숫자 그대로 적습니다.

```
13
3.14159265
-5
3 + 6j
```

문자열: 문자열을 따옴표(", ') 사이에 적습니다.

```
"CS101 is wonderful"
'The instructor said: "Well done!" and smiled'
```

논리값(Boolean): `True` 또는 `False` 로 적습니다.

복잡한 객체 생성

복잡한 객체는 그 객체를 만드는 함수를 부르는 방법으로 만듭니다.

```
from cs1robots import *
Robot()
```

```
from cs1media import *
load_picture("photos/geowi.jpg")
```

튜플(Tuple)은 다른 객체들을 포함하는 객체입니다.
여러 객체들을 쉼표(,)를 사이에 두고 적어서 만들 수 있습니다.

```
(3, 2.5, 7)
("red", "yellow", "green")
(20100001, "Hong Gildong")
```

형태

모든 객체는 **형태(Type)**를 가지고 있습니다.
객체의 형태는 객체가 할 수 있는 일과, 객체를 이용해 할 수 있는 일을 결정합니다.
(예. 두 숫자는 더할 수 있지만, 두 로봇은 더할 수 없습니다)

Python에서는 다음과 같은 방법으로 객체가 어떤 형태를 가지고 있는지 알 수 있습니다.

```
>>> type(3)
<class 'int'>                                  정수: int
>>> type(3.1415)
<class 'float'>                                실수: float
>>> type("CS101 is fantastic")
<class 'str'>                                  문자열: str
>>> type(3 + 7j)
<class 'complex'>                              복소수: complex
>>> type(True)
<class 'bool'>                                 논리값: bool
```

복잡한 형태

복잡한 객체들의 형태는 다음과 같이 표시됩니다.

```
>>> type(Robot())
<class 'cs1robots.Robot'>
>>> type((3, -1.5, 7))
<class 'tuple'>
>>> type(load_picture("geowi.jpg"))
<class 'cs1media.Picture'>
```

이름

객체에는 **이름**을 줄 수 있습니다.

```
message = "CS101 is fantastic"
n = 17
hubo = Robot()
pi = 3.1415926535897931
finished = True
img = load_picture("geowi.jpg")
```

`n = 17`과 같은 문장은 **대입문(Assignment)**이라고 부릅니다.

`n` 이라는 이름이 숫자 17 에 붙여져서, n을 숫자 17처럼 사용할 수 있기 때문입니다.

Python 동물원에서, 이름은 동물 우리 앞의 팻말과 같습니다.

이름 규칙

변수와 함수의 이름을 지을 때는 다음과 같은 규칙을 따라야 합니다.

- 영어 문자, 숫자, 그리고 밑줄 문자(_)로만 이루어져야 합니다.
- 숫자는 이름의 첫 글자로 올 수 없습니다.
- 파이썬에서 등록된 키워드(예약어)와 동일한 이름은 지을 수 없습니다.
  e.g.) **`def`**, **`if`**, **`else`**, **`while`**
- 이름은 대소문자를 구분합니다
  e.g.) `Pi`와 `pi`는 다른 이름입니다.

좋은 예시:

```
my_message = "CS101 is fantastic"
a13 = 13.0
```

나쁜 예시:

```
more@ = "illegal character"
13a = 13.0
def = "Definition 1"
```

변수

이름이 가리키는 객체는 바뀔 수 있기에, 이름은 **변수(Variable)**라고도 불립니다.
다시 말해, 프로그램 실행 중에 변수가 가리키는 객체는 바뀔 수 있습니다.

```
n = 17
n = "Seventeen"
n = 17.0
```

변수에 대입된 객체는 변수의 **값**이라고 부릅니다.
변수의 값은 바뀔 수 있습니다.

None이라는 이름의 특별한 객체는 **비었다**는 것을 의미하기 위해 사용됩니다.

```
n = None
```

```
>>> type(n)
<class ‘NoneType'>
```

멤버 변수

객체가 할 수 있는 일은 객체의 형태에 따라 결정됩니다.
(예. 새는 날 수 있고, 물고기는 헤엄칠 수 있습니다)

객체는 **멤버 함수(Method)**를 통해 이러한 일들을 할 수 있습니다.

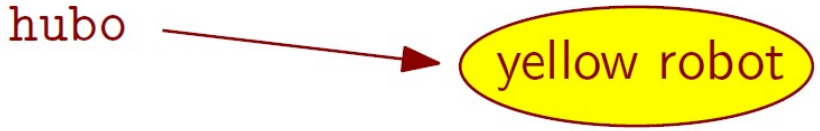
멤버 함수는 **점(.) 연산자**를 통해 실행할 수 있습니다.

```
>>> hubo = Robot()
>>> hubo.move()
>>> hubo.turn_left()

>>> img = load_picture("geowi.jpg")
>>> print(img.size())  # width and height in pixels
(58, 50)
>>> img.show()          # display the image

>>> b = "banana"
>>> print(b.upper())
BANANA
```

여러 이름을 가진 객체

하나의 객체는 여러 이름을 가질 수 있습니다.

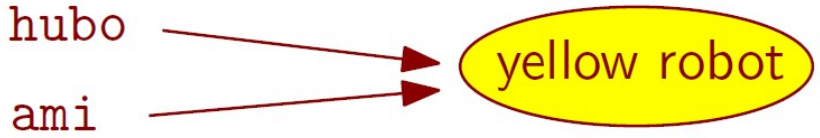
```
hubo = Robot("yellow")
```



여러 이름을 가진 객체

하나의 객체는 여러 이름을 가질 수 있습니다.

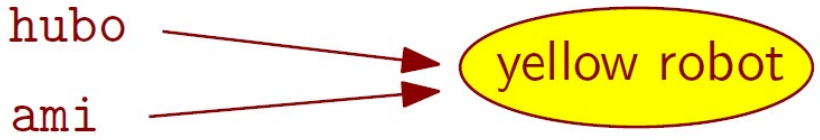
```
hubo = Robot("yellow")
hubo.move()
ami = hubo
```



여러 이름을 가진 객체

하나의 객체는 여러 이름을 가질 수 있습니다.

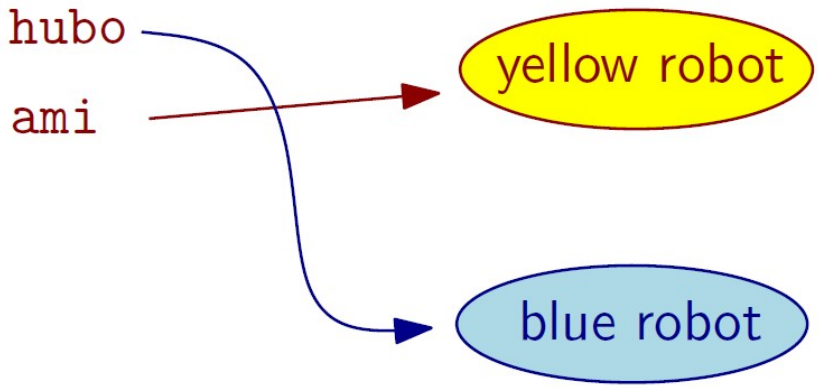
```
hubo = Robot("yellow")
hubo.move()
ami = hubo
ami.turn_left()
hubo.move()
```



여러 이름을 가진 객체

하나의 객체는 여러 이름을 가질 수도 있습니다.

```
hubo = Robot("yellow")
hubo.move()
ami = hubo
ami.turn_left()
hubo.move()

hubo = Robot("blue")
hubo.move()
ami.turn_left()
ami.move()
```



정리 및 예습

본 강의 학습 목표:

- 객체 (objects)와 형태를(types) 이해할 수 있다.
- 변수 (variables)를 이해할 수 있다.

다음 강의 학습 목표:

- 연산자를 통한 식을 이해하고 작성할 수 있다.
- 기초 자료 구조인 튜플을 이해하고 작성할 수 있다.