

2018학년도 2학기 언어와 컴퓨터

제3강 기본 자료형: 수와 문자

박수지

서울대학교 인문대학 언어학과

2018년 9월 10일 월요일

오늘의 목표

- 1 표현식과 문장이 무엇인지 설명할 수 있다.
- 2 값의 이름을 바르게 지을 수 있다.
- 3 값의 자료형이 무엇인지 알고 다른 자료형으로 변환할 수 있다.
- 4 문자열의 메소드를 사용할 수 있다.
- 5 `input()`과 `print()`을 사용하여 프로그램을 작성할 수 있다.

표현식

지난 시간에 한 것

파이썬을 계산기처럼 사용했다. (= 표현식을 작성하고 평가시켰다.)

표현식 (expression)

- 값 (value), 변수 (variable), 연산자 (operator)의 조합
- 하나의 값으로 평가 (evaluate) 된다.

예 표현식 $2 + 2$ 는 4라는 값으로 평가된다.

자료형

주의

값의 자료형에 따라 사용 가능한 연산자가 달라진다.

■ 수치 산술 (사칙연산, 거듭제곱)

■ 문자열 연결, 반복

주요 자료형 (data types)

- 정수, 부동소수점, 문자열
- 복소수, 불

수

수치 자료형

정수 (int), 부동소수점 (float), 복소수 (complex), 불 (bool)

산술 연산자 사용 가능

```
>>> True + 1 + 1.0 + (1+0j)  
(4+0j)
```

부동소수점수

만드는 방법

소수점 입력

```
>>> 1.23
1.23
>>> 4.
4.0
>>> .5
0.5
>>> .
File "<stdin>", line 1
  .
  ^
SyntaxError: invalid syntax
```

정수의 나눗셈

```
>>> 6 / 2
3.0
>>> 6 // 2
3
```

정수 \leftrightarrow 부동소수점수

```
>>> int(1.0)
1
>>> float(1)
1.0
```

부동소수점수

실수 \neq 부동소수점수

수의 체계

자연수 \subset 정수 \subset 유리수 \subset 실수 \subset 복소수

컴퓨터의 주요 수 유형

정수 (int) $\not\subset$ 부동소수점 (float) $\not\subset$ 복소수 (complex)

- 부동소수점 (floating point): 실수를 표현하는 개념

특징

이진수 & 유한한 메모리 공간

\Rightarrow 모든 부동소수점은 이진수 유한소수로 표현된다.

부동소수점수

오차와 한계

정수가 부동소수점보다 더 정확한 예시

```
>>> 0.1 + 0.2
0.30000000000000004
>>> (1 + 2) / 10
0.3
>>> 2.0 ** 1024
[....]
>>> 2 ** 1024
[....]
```


복소수

허수단위 j 사용

예시

```
>>> (-1) ** (1 / 2)
(6.123233995736766e-17+1j)
>>> type((-1) ** (1 / 2))
<class 'complex'>
>>> 1j ** 2
(-1+0j)
```

주의

SyntaxError가 발생한 경우 괄호의 개수를 다시 살펴보자.

불

비교 연산자

불 자료형의 값으로 평가되는 표현식

등식, 부등식 명제

```
>>> 2 == 3
```

```
False
```

```
>>> 2 != 3
```

```
True
```

```
>>> 2 < 3
```

```
True
```

```
>>> 2 >= 3
```

```
False
```

불

논리 연산자

불 자료형의 값으로 할 수 있는 연산

```
>>> True and True
True
>>> True or False
True
>>> not True
False
```

관찰

and, or, not으로 가능한 모든 연산을 만들어 보자.

불

불과 다른 자료형의 관계

사실

불은 정수의 특수한 경우다. True는 1, False는 0이다.

확인

```
>>> isinstance(True, bool)
True
>>> isinstance(True, int)
True
>>> (True == 1) and (False == 0)
True
```

`isinstance(값, 자료형)` 값이 자료형에 속하는지 확인하는 함수

불

불과 다른 자료형의 관계

사실

0은 거짓(False), 이외의 값은 참(True)으로 해석된다.

확인

```
>>> bool(0)
False
>>> bool(5)
True
>>> bool(3.14)
True
```

관찰

자료형의 이름을 함수처럼 사용하여 자료형을 변환할 수 있다.

문장

문장(statement)

영향을 주는 코드의 단위

예시

- 1 할당문: 변수 name에 값을 지정해 준다.
예 `name = input('이름을 입력하세요: ')`
- 2 print 문: 변수 name의 값을 표시한다.
예 `print(name, '님, 반갑습니다.')`

할당문

문법

할당문 (assignment statement)의 구성

이름, 할당 연산자(=), 표현식

예시

```
>>> height = 170
>>> height / 2.5
68.0
>>> height = height + 1
>>> height
171
```

할당문

이름의 규약

필수

- 1 키워드를 쓸 수 없다.
- 2 특수 문자는 _(밑줄)만 허용된다.
- 3 숫자로 시작하면 안 된다.
- 4 공백을 포함할 수 없다.

추가

- 1 영문자와 숫자와 밑줄만 사용한다.
- 2 영어 소문자와 밑줄로 단어 경계를 표시한다.
 - my_number (o)
 - myNumber (x)

키워드 ['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']

문자열

문자란 무엇인가?

問 문자열 (string)이란 무엇인가?

답 문자들의 열 (sequence of characters)이다.

問 문자란 무엇인가?

컴퓨터의 기준

단어 (word) 공백 문자를 경계로 하는 문자들의 연쇄

- 공백 문자: 빈 칸, 탭, 줄바꿈 문자, ...

행 (line) 줄바꿈 문자로 끝나는 문자들의 연쇄

- 줄바꿈 문자 = 개행 (開行) 문자 = line break = EOL(end-of-line)

문자열

문자열 연산

연결 (concatenation), 반복 (repetition), 비교

```
>>> '안녕하세요' + '...!'  
'안녕하세요...!'
```

```
>>> 3 * '안녕하세요'  
'안녕하세요안녕하세요안녕하세요'
```

```
>>> '안녕하세요' < '안녕하십니까'  
True
```

문자열

문자열 연산

TypeError: 자료형과 연산자가 맞지 않는 경우

```
>>> '안녕하세요' * '...!'
```

```
Traceback (most recent call last):
```

```
File "<stdin>", line 1, in <module>
```

```
TypeError: can't multiply sequence by non-int of type 'str'
```

```
>>> 3 + '안녕하세요'
```

```
Traceback (most recent call last):
```

```
File "<stdin>", line 1, in <module>
```

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

```
>>> '안녕하세요' < 333
```

```
Traceback (most recent call last):
```

```
File "<stdin>", line 1, in <module>
```

```
TypeError: '<' not supported between instances of 'str' and 'int'
```

문자열

문자열과 수 사이의 형변환

문자열로

```
>>> str(1)
'1'
>>> str(1.0)
'1.0'
>>> str(1j)
'1j'
```

정수로

```
>>> int('11')
11
>>> int('11', base=2)
3
>>> int('11', base=3)
4
>>> int('1.0') # ValueError
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10
```

문자열

문자의 열 — 지표 (index)

앞에서부터 찾기

```
>>> "안녕하세요" [0]
'안'
>>> "안녕하세요" [1]
'녕'
>>> "안녕하세요" [2]
'하'
>>> "안녕하세요" [3]
'세'
>>> "안녕하세요" [4]
'요'
```

뒤에서부터 찾기

```
>>> "안녕하세요" [-1]
'요'
>>> "안녕하세요" [-2]
'세'
>>> "안녕하세요" [-3]
'하'
>>> "안녕하세요" [-4]
'녕'
>>> "안녕하세요" [-5]
'안'
```

문자열

문자의 열 — 썰기(slicing)

m번째부터 n번째 직전까지

```
>>> "안녕하세요"[0:2]
'안녕'
>>> "안녕하세요"[1:3]
'녕하'
>>> "안녕하세요"[2:4]
'하세'
>>> "안녕하세요"[3:]
'세요'
>>> "안녕하세요"[:3]
'안녕하'
```

여러 칸씩 썰기

```
>>> "안녕하세요"[0:4:2]
'안하'
>>> "안녕하세요"[1::3]
'녕요'
>>> "안녕하세요"[:, -1]
'요세하녕안'
```

문자열

메소드

대문자로 바꾸기

```
>>> 'Python'.upper()  
'PYTHON'  
>>> str.upper('Python')  
'PYTHON'
```

빈칸 채우기

```
>>> student = '이름: {}, 나이: {}'  
>>> student.format('강은수', 21)  
'이름: 강은수, 나이: 21'  
>>> student.format('조재영', 20)  
'이름: 조재영, 나이: 20'
```

문자열

메소드

문자의 유형 확인하기

```
>>> 'Python'.isalpha()
True
>>> 'abc'.islower()
True
>>> '123'.isdigit()
True
>>> 'ABC'.isupper()
True
>>> '\n'.isspace()
True
```


문자열

메소드

메소드 찾기

```
>>> dir(str)
[....]
>>> dir('아무말')
[....]
>>> dir(student)
[....]
```

메소드 도움말 보기

```
>>> help(str.upper)
[....]
>>> help('아무말'.upper)
[....]
>>> help(student.upper)
[....]
```

프로그램의 구성 요소

입력 키보드, 파일, 네트워크, 또는 다른 기기에서 데이터를 가져온다.

출력 스크린에 데이터를 표시하기, 파일에 데이터를 저장하기, 데이터를 네트워크로 전송하기 등

수학 덧셈, 곱셈 같은 기본 산술 연산을 수행한다.

조건 실행 어떤 조건을 검사해서 적절한 코드를 실행한다.

반복 어떤 동작을 반복해서 수행하며, 보통은 몇 가지 변형도 있다.

오늘 할 일

키보드로 입력받은 데이터로 적절한 연산을 수행하여 스크린에 표시한다.

count_a.py

```
1 # -*- coding: utf-8 -*-  
2 # 입력된 영어 문장에서 a의 개수를 세는 프로그램  
3 sentence = input('영어 문장을 입력하세요: ')  
4 print('a가 {} 개 들었습니다.'  
      .format(sentence.lower().count('a')))
```

주의

- 1 #로 시작하는 내용은 컴퓨터에 영향을 미치지 않는다.
- 2 화면에서 다음 줄로 넘어가도 행 번호가 바뀌지 않으면 줄을 바꾸지 않는다.

sqrt.py

```
1 # -*- coding: utf-8 -*-  
2 # 입력된 수의 제곱근을 구하는 프로그램  
3 number = input('값을 입력하세요: ')  
4 print('제곱근은 {}입니  
    다.'.format(complex(number) ** (1 / 2)))
```

주의

- 1 `complex(number)` 대신 `number`를 사용하면 어떻게 되는가?

요약

개념

- 표현식
- 자료형
- 연산자
- 문장

프로그램 작성

- 입력
- 연산
- 출력

더 생각해 볼 문제

- 1 `int(3.14)`의 결과를 예측하고 확인하라.
- 2 정수나 부동소수점이나 불 자료형의 객체에 `len()` 함수, indexing, slicing을 적용하면 무슨 일이 일어나는지 설명하라.
- 3 복소수의 켄레복소수를 찾는 메소드를 찾으라.
- 4 `type(len('string'))`의 결과를 예측하고 확인하라.