2021학년도 2학기 언어와 컴퓨터

제3강 반복 가능한 자료형

박수지

서울대학교 인문대학 언어학과

2021년 9월 8일 수요일

오늘의 목표

1. 표현식과 문장이 무엇인지 설명할 수 있다.
2. 값의 이름을 바르게 지을 수 있다.
3. 문자열의 메소드를 사용할 수 있다.
4. 객체의 구성 요소를 설명할 수 있다.
5. 동일성과 동등성을 구별할 수 있다.
6. 반복 가능한 자료형으로서 문자열, 리스트, 튜플의 공통점과 차이점을 말할 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?

答 문자들의 열(sequence of characters)이다.

問 문자란 무엇인가?

컴퓨터의 기준

단어(word) 공백 **문자**를 경계로 하는 문자들의 연쇄

- 공백 문자: 빈 칸, 탭, 줄바꿈 문자, ...

행(line) 줄바꿈 **문자**로 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: '1.0'
```

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)
[....]
```

문자열: 문자의 열

문자열과 문자의 관계

`len()`	길이
`str.find()`	위치
`str.count()`	출현 횟수
`in` 연산자	소속 여부

예시

```
>>> len('Python')
6
>>> 'Python'.find('P')
0
>>> 'Python'.find('p')
-1
>>> 'Python'.count('p')
0
>>> 'p' in 'Python'
False
```

의문

0과 0.0은 같은가?

관찰

- 값은 같다.
- 그런데 뭔가 다르다.

확인하기

```
>>> zero_int = 0
>>> zero_float = 0.0
>>> zero_int == zero_float
True
>>> zero_int is zero_float
False
```

`==` 동등성 (equality) 비교 연산자

`is` 동일성 (identity) 비교 연산자

같은 값을 가진 다른 객체

```
>>> a = '파이썬'
>>> b = '파이썬'
>>> a == b
True
>>> a is b
False
>>> id(a)
4339513456
>>> id(b)
4339513552
```

`id()` 객체의 식별성을 돌려주는 함수

파이썬에서의 객체

데이터를 추상적으로 나타낸 것

⇒ 거의 모든 것이 객체다.

객체가 가지는 것

- 식별성 (identity)
- 유형 (type)
- 값 (value)

문자열: 단어의 목록

예시

```
>>> words = '달아 달아 밝은 달아'.split()
>>> words_list = words.split()
>>> words_list
['달아', '달아', '밝은', '달아']
>>> words_list[2]
'밝은'
>>> words_list.index('밝은')
2
>>> words_list.count('달아')
3
>>> '달' not in words
True
```

리스트

내장된 가변열 (Built-in mutable sequence)

https://docs.python.org/ko/3/library/stdtypes.html#list

형식

```
[항목1, 항목2, ....]
```

형변환 예시

```
>>> list()
[]
>>> list('Python')
['P', 'y', 't', 'h', 'o', 'n']
>>> list(0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'int' object is not iterable
```

열(sequence) 유형

공통 연산

- 소속(in), 연결(+), 반복(*), 비교(<)
- 인덱싱, 슬라이싱
- `len()`, `min()`, `max()` 함수
- `s.index()`, `s.count()` 메소드

해당 자료형

- `str`
- `list`
- `tuple`
- `range`

주의

비교 연산자와 `min()`, `max()` 함수는 특정 조건하에서만 사용할 수 있다.

가변열

리스트 연산 확인하기

```
>>> dir(list)
[...., 'append', 'clear', 'copy', 'count', 'extend', 'index',
'insert', 'pop', 'remove', 'reverse', 'sort']
```

리스트 메소드

늘리기

추가

```
>>> basket = ['egg', 'milk']
>>> basket.append('cheese')
>>> basket
['egg', 'milk', 'cheese']
```

확장

```
>>> basket = ['egg', 'milk']
>>> basket.extend(['cheese'])
>>> basket
['egg', 'milk', 'cheese']
```

`list.append()` 리스트 뒤에 항목을 추가하는 메소드

`list.extend()` 리스트 뒤에 다른 열을 붙이는 메소드

연결 및 할당 연산자를 사용한 확장

```
>>> basket = ['egg']
>>> basket = basket + ['milk', 'cheese']
>>> basket
['egg', 'milk', 'cheese']
```

복합 할당 연산자

`basket = basket + ['milk', 'cheese']`를 줄여서
`basket += ['milk', 'cheese']`로 쓸 수 있다.

리스트 메소드

줄이기

끝에서 제거

```
>>> fi = [1, 1, 2, 3, 5]
>>> fi.pop()
5
>>> fi
[1, 1, 2, 3]
```

값으로 제거

```
>>> fi = [1, 1, 2, 3, 5]
>>> fi.remove(1)
>>> fi
[1, 2, 3, 5]
```

del과 인덱스로 항목 제거

```
>>> fi = [1, 1, 2, 3, 5]
>>> del fi[3]
>>> fi
[1, 1, 2, 5]
```

del과 슬라이스로 항목 제거

```
>>> fi = [1, 1, 2, 3, 5]
>>> del fi[1::2]
>>> fi
[1, 2, 5]
```

리스트 메소드

순서 바꾸기

정렬

```
>>> py = list('Python')
>>> py.sort()
>>> py
['P', 'h', 'n', 'o', 't', 'y']
```

뒤집기

```
>>> py = list('Python')
>>> py.reverse()
>>> py
['n', 'o', 'h', 't', 'y', 'P']
```

파괴적 함수 對 비파괴적 함수

원본에 영향을 주는가?

list.sort() 메소드: 파괴적

```
>>> pi = [3, 1, 4, 1, 5]
>>> pi.sort()
>>> pi
[1, 1, 3, 4, 5]
```

sorted() 함수: 비파괴적

```
>>> pi = [3, 1, 4, 1, 5]
>>> sorted(pi)
[1, 1, 3, 4, 5]
>>> pi
[3, 1, 4, 1, 5]
```

튜플

내장된 불변열(Built-in immutable sequence)

https://docs.python.org/ko/3/library/stdtypes.html#tuple

형식

```
(항목1, 항목2, ....)
```

괄호 생략하기

```
>>> a = 1, 2
>>> a
(1, 2)
>>> type(a)
<class 'tuple'>
```

메소드 탐색

```
>>> dir(tuple)
[...., 'count', 'index']
```

가변 vs. 불변

리스트: 가변열

```
>>> basket = ['bacon', 'egg', 'milk']
>>> basket[0] = 'ham'
>>> basket
['ham', 'egg', 'milk']
```

가변 vs. 불변

튜플: 불변열

```
>>> food = ('milk', 200, 800)
>>> food[1] = 500
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

list vs. tuple: 용도

- 리스트: 동질적인 자료 모음
  - 예시: [학생1의 키, 학생2의 키, 학생3의 키, ...]
- 튜플: 이질적인 자료 모음
  - 예시: (학생1의 키, 학생1의 몸무게)

범위

형식

`range(start, end, step)`

- 슬라이스의 `[start:end:step]`과 사용법이 거의 같음

예시

```
>>> list(range(5)) # 기본적으로 0부터 시작함
[0, 1, 2, 3, 4]
>>> list(range(-5, 5))
[-5, -4, -3, -2, -1, 0, 1, 2, 3, 4]
>>> list(range(10, 0, -2))
[10, 8, 6, 4, 2]
```

딕셔너리

형식

{키1: 값1, 키2: 값2, ...}

- 인덱스 대신 키를 사용하여 값을 찾는다.

예시

```
>>> numbers = {'one': 1, 'two': 2, 'three': 3}
>>> numbers['two']
2
>>> numbers[3]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 3
```

딕셔너리 연산

늘리기

예시

```
>>> hansol = {'HP': 80}
>>> hansol['MP'] = 90 # 새로운 키와 값 할당
>>> hansol
{'HP': 80, 'MP': 90}
>>> hansol.update({'class': 'mage'})
>>> hansol
{'HP': 80, 'MP': 90, 'class': 'mage'}
```

`>>> dir(dict)`로 메소드를 더 확인해 봅시다.

```
1 # 문장 내에서 특정 문자의 개수를 세는 프로그램
2 sentence = input('문장을 입력하세요: ')
3 char = input('찾고 싶은 문자를 입력하세요: ')
4 n = sentence.count(char)
5 print('{}의 출현 횟수: {}'.format(char, n))
```

프로그램의 구성

- 1행: 주석
- 2-3행: 입력
- 4행: 연산
- 5행: 출력

```python
# 문장 내 단어를 가나다순으로 정렬하는 프로그램
sentence = input('문장을 입력하세요: ')
words = sentence.split()
words.sort()
print(words)
```

프로그램의 구성

- 1행: 주석
- 2행: 입력
- 3-4행: 연산
- 5행: 출력

요약

객체

1. 데이터의 추상적 표현
2. 식별성, 유형, 값을 가짐

객체의 같음

동등성 값이 같음
- == 연산자

동일성 식별성이 같음
- is 연산자

반복 가능한 자료형

1. 열
   - 가변: `list`
   - 불변: `str`, `range`
2. 매핑(mapping)
   - `dict`

다음 시간 예고

반복문, 조건문

더 생각할 문제

1. 비교, `min()`, `max()`는 어떤 조건에서 사용 가능한가?
   예시 `>>> min([1, '2'])`
2. 문자열은 변경 가능한 자료형인가?
3. 딕셔너리는 변경 가능한 자료형인가?
   - 확인하는 방법
     1. 요소의 값을 새로 할당할 수 있는가?
     2. 메소드 중에서 파괴적 함수가 있는가?