테크 지식 & 파이썬 프로그래밍 기초

3주차
리스트와 문자열 다루기

20 . .

지은이: 이은재

Ⅰ. 리스트 다루기

(1) 리스트에 값을 추가하기

- 메소드(method) 사용하기

- 메소드란?

메소드는 함수의 일종입니다. 특정한 기능을 수행하고 결과를 반환한다는 점에서는 동일합니다. 일반함수는 독립적으로 선언하고 호출할 수 있지만 메소드는 특정한 클래스를 선언할 때 그 구성요소로 포함된다는 차이가 있습니다. 리스트와 문자열 자료형은 기본 내장형 클래스이기 때문에 별도의 선언 없이 아래와 같이 메소드를 사용하면 됩니다.

ex)
변수명.메소드명(매개변수)

▲ append 메소드

: 리스트의 맨 마지막 인덱스에 이어서 새로운 값을 추가합니다.

```
ex)
week = [‘월’, ‘화’, ‘수’, ‘목’, ‘금’]
week.append(‘토’)
print(week)
week.append(‘일’)
print(week)

>>> ['월', '화', '수', '목', '금', '토']
['월', '화', '수', '목', '금', '토', ‘일’]
```

▲ insert 메소드

: 리스트의 원하는 인덱스에 새로운 값을 추가합니다.

```
ex)
fruit = [‘딸기’, ‘당근’, ‘참외’, ‘메론’]
fruit.insert(2, ‘수박’)
print(fruit)

>>> [‘딸기’, ‘당근’, ‘수박’, ‘참외’, ‘메론’]
```

▲ extend 메소드

: 리스트에 다른 리스트를 더해서 이어 붙입니다.

```
ex)
greek = [‘탈레스’, ‘파르메니데스’, ‘헤라클레이토스’]
athen = [‘소크라테스’, ‘플라톤’]
athen.extend([‘프로타고라스’, ‘아리스토텔레스’])
greek.extend(athen)
print(athen)
print(greek)

>>> ['소크라테스', '플라톤', '프로타고라스', '아리스토텔레스']
['탈레스', '파르메니데스', '헤라클레이토스', '소크라테스', '플라톤', '프로타고라스',
'아리스토텔레스']
```

- 리스트 더하기

▲ 리스트 + 연산하기

: 리스트끼리는 숫자 자료형처럼 사칙연산을 수행할 수 있습니다.

```
ex)
forest = [‘여우’, ‘곰’, ‘토끼’]
sea = [‘고래’, ‘펭귄’]
zoo = forest + sea
print(zoo)

>>> ['여우', '곰', '토끼', '고래', '펭귄']
```

(2) 리스트에서 값을 출력하기

▲ [ ] 슬라이싱 사용하기

: 슬라이싱은 리스트에서 원하는 요소를 골라내어 출력할 수 있는 방법 중 하나입니다. 0부터 시작하는 요소번호를 사용해서 다양한 방식으로 출력할 리스트 요소를 고를 수 있습니다. 슬라이싱은 아래처럼 리스트명과 대괄호를 입력해 사용할 수 있습니다.

```
리스트명[시작점:끝점+1]
: 시작점 <= x < 끝점
```

```
bigmac = [‘참깨빵’, ‘패티’, ‘패티’, ‘소스’, ‘양상추’, ‘치즈’, ‘피클’, ‘양파’]
vegan = bigmac[0:1] + bigmac[3:8]
meat_lover = bigmac[0:4] + bigmac[5:6]
no_onion = bigmac[:-1]
```

▲ 하나, 둘, 셋, count 메소드

: count 메소드는 리스트에 해당 요소가 몇 개나 들어있는지 세어줍니다.

```
word = ‘패티’
print(bigmac.count(word))
print(‘쇠고기’, bigmac[1], bigmac.count(word), ‘장’, sep=‘’)
```

▲ 찾아줘요, index 메소드

: index 메소드는 리스트에서 해당 요소가 몇 번째에 들어있는지 찾아줍니다. 중복으로 들어있을 경우 가장 앞에 있는 것을 찾습니다.

```
idx = '소스'
print(bigmac.index(idx))
print(bigmac[bigmac.index(idx)])
```

▲ 헤쳐모여! sort 메소드

: sort 메소드는 리스트 내의 요소들을 오름차순, 혹은 내림차순으로 정렬해줍니다.

```
bigmac.sort()
print(bigmac)
bigmac.sort(reverse=True)
print(bigmac)
```

▲ 뒤집어져요, reverse 메소드

: reverse 메소드는 리스트 내의 요소들의 순서를 역순으로 바꾸어 줍니다.

```
bigmac.reverse()
print(bigmac)
```

▲ 튀어나와요, pop 메소드

: pop 메소드는 리스트의 맨 마지막 요소를 반환한 후 삭제합니다. 출력이 아니라 반환이므로 출력함수를 사용해야만 결과를 눈으로 확인할 수 있습니다.

```
print(bigmac)
print(bigmac.pop())
print(bigmac)
print(bigmac.pop())
print(bigmac)
```

(3) 리스트에서 값을 제거하기

▲ [ ] 슬라이싱을 응용한 삭제

```
ex)
philosophy = [‘형이상학’, ‘심리학’, ‘경제학’, ‘인식론’, ‘윤리학’]
philosophy [1:3] = [ ]
print(philosophy)

>>> ['형이상학', '인식론', '윤리학']
```

▲ del 예약어 사용하기

: del은 함수가 아니라 데이터의 삭제를 위해 파이썬 내부에 예약된 예약어입니다. 아래처럼 사용하면 해당 요소를 제거할 수 있습니다.

```
ex)
asia = [‘한국’, ‘일본’, ‘중국’, ‘대만’, ‘캐나다’]
del asia[-1]
print(asia)

>>> [‘한국’, ‘일본’, ‘중국’, ‘대만’]
```

▲ remove 메소드

: .remove(x)는 리스트에 존재하는 x가 두 개 이상일 때, 첫 번째 x를 제거합니다.

```
ex)
coffee = [‘아메’, ‘아메’, ‘아메리카노’, ‘좋아’, ‘좋아’, ‘좋아’]
coffee.remove(‘아메’)
print(coffee)

>>> [‘아메’, ‘아메리카노’, ‘좋아’, ‘좋아’, ‘좋아’]
```

Ⅱ. 문자열 다루기

(1) 문자열 이어 붙이기

조금 이상하게 들릴 수도 있겠지만 문자열 자료형은 서로 연산이 가능하고, 숫자 자료형과도 연산이 가능합니다. 물론 문자열은 숫자가 아닌 만큼 빼기나 나누기는 불가능하고 더하기와 곱하기 연산만을 실행할 수 있습니다. 또한 문자열을 한 글자씩 골라서 출력하거나 부분을 잘라내어 출력하는 것도 가능합니다.

▲ 문자열 + 연산하기

: 문자열 덧셈 연산은 문자열끼리의 연산입니다. 문자열끼리의 덧셈은 두 문자열을 하나로 연결해줍니다.

ex1)

```
print('르네' + '데카르트')

>>> 르네데카르트
```

ex2)

```
print('P'+'YT'+'HON')

>>> PYTHON
```

▲ 문자열 * 연산하기

: 문자열 곱셈 연산은 문자열과 양의 정수 사이의 연산입니다. 문자열과 숫자의 곱셈은 문자열을 곱한 수만큼 반복 출력합니다.

ex1)

```
print('주먹은 가위를 이기고 가위는 보를 이기고 보는 주먹을 이기고' * 10)
```

>>> 주먹은 가위를 이기고 가위는 보를 이기고 보는 주먹을 이기고 주먹은 가위를 이기고 가위는 보를 이기고 보는 주먹을 이기고 주먹은 가위를 이기고 가위는 보를 이기고 보는 주먹을 이기고 주먹은 가위를 이기고 가위는 보를 이기고 보는 주먹을 이기고 주먹은 가위를 이기고 가위는 보를 이기고 보는 주먹을 이기고 주먹은 가위를 이기고 가위는 보를 이기고 보는 주먹을 이기고 주먹은 가위를 이기고 가위는 보를 이기고 보는 주먹을 이기고 주먹은 가위를 이기고 가위는 보를 이기고 보는 주먹을 이기고

이때 덧셈과 곱셈의 혼합연산도 가능합니다. 일반 사칙연산과 마찬가지로 곱셈의 우선순위가 덧셈보다 높다는 사실도 확인할 수 있습니다.

ex2)
```
print('코딩은 하나도 어렵지' + ' 아니하지' * 4 + ' 않다.')
>>> 코딩은 하나도 어렵지 아니하지 아니하지 아니하지 아니하지 않다.
```

ex3)
```
noise1 = '쿵'
noise2 = '짝'
print((noise1 + ' ' + noise2 + ' ') * 5)
print((noise1 + ' ' + (noise2 + ' ') * 2) * 2)

>>> 쿵 짝 쿵 짝 쿵 짝 쿵 짝 쿵 짝
쿵 짝 짝 쿵 짝 짝
```

(2) 문자열 인덱싱

: 문자열에 직접 표시가 되지는 않지만, 문자열에도 리스트의 요소처럼 각 문자마다 위치에 따른 번호가 매겨져 있습니다. 이 번호를 주소처럼 사용해서 해당 문자열 번호의 글자를 골라낼 수 있는데 이것을 '문자열 인덱싱'이라고 합니다. 간단한 예를 들어 살펴보겠습니다.

ex1)
다음과 같은 문자열 Rene가 있다면,

```
Rene = 'I think, therefore I am.'
```

Rene 문자열의 인덱스는 아래와 같이 매겨져 있습니다.

```
Rene = 'I think, therefore I am.'
        0         1         2
        01234567890123456789012
```

따라서, 만약 therefore의 마지막 글자 e를 출력하고자 한다면 Rene[17]을 출력하면 됩니다.

```
print(Rene[17])
>>> e
```

ex2)
그러나 내가 출력할 문자열의 길이가 길다면 어떨까요? 게다가 뒤쪽에서 더 가까운 글자를 출력하고자 한다면 어떻게 할까요? 문자열의 시작점에서부터 해당 글자의 번호를 세는 것은 비효율적일뿐더러 매우 짜증나는 일이 될 겁니다. 그래서 파이썬에서는 문자열의 마지막 글자에서부터 역으로 음수를 세어나가는 방법도 제공합니다. 직접 해보겠습니다.

```
Rene = ‘I think, therefore I am.’
print(Rene[-1])
>>> .
print(Rene[-2])
>>> m
```

※ 주의할 점! :
- 띄어쓰기 공백도 당연히 하나의 글자로 저장된다.
- 컴퓨터는 숫자를 0부터 센다!
- 문자열 인덱싱으로는 문자열을 수정할 수 없다.

(3) 문자열 슬라이싱

:문자열 슬라이싱은 문자열의 일부분을 골라내어 출력한다는 점에서는 인덱싱과 같습니다. 하지만 시작점과 끝점을 지정해서 그 사이의 문자들을 한 번에 골라낼 수 있기 때문에 문장에서 단어를 뽑아야 하거나 여러 글자를 한 번에 뽑아야 할 때 훨씬 편리하게 표현할 수 있습니다.

슬라이싱을 하는 방법!
변수명[시작점:끝점+1]

```
Socrates = ‘The unexamined life is not worth living.’
           0         1         2         3
           01234567890123456789012345678901234567890
```

‘life’를 골라내고 싶을 때는?

ex1) 슬라이싱을 사용하지 않는 경우

```
print(Socrates[15]+Socrates[16]+Socrates[17]+Socrates[18])
>>> life
```

ex2) 슬라이싱을 사용하는 경우

```
print(Socrates[15:19])
>>> life
```

```
Socrates = 'The unexamined life is not worth living.'
           0         1         2         3
           01234567890123456789012345678901234567890
```

ex3) 끝점을 생략하는 경우 – 지정한 지점부터 문자열의 끝까지를 출력한다

```
print(Socrates[15:])
>>> life is not worth living.
```

ex4) 시작점을 생략하는 경우 – 문자열의 처음부터 지정한 지점까지를 출력한다

```
print(Socrates[:19])
>>> The unexamined life
```

ex5) 둘다 생략하는 경우 – 문자열을 모두 출력한다

```
print(Socrates[:])
>>> The unexamined life is not worth living.
```

ex6) 음수를 사용할 수 있다

```
print(Socrates[-7:])  #-7 지점부터 문자열의 끝까지 출력.
>>> living.

>>>print(Socrates[:-8])  #문자열의 처음부터 -8앞 까지 출력.
The unexamined life is not worth
```

응용 ex) [::n] 도 가능할까? - 첫글자로 시작해서 n번째 마다 글자를 출력하는 것을 알 수 있다.

```
thanks_Sejong = '가나다라마바사아자차카타파하'
print(thanks_Sejong[::3])
>>> 가라사차파

print(thanks_Sejong[::2])
>>> 가다마사자카파
```

(4) 기타 문자열 처리

▲ index 메소드

: 지정한 값이 문자열에 들어있다면, 들어있는 위치를 반환합니다. 단, 찾는 문자가 문자열에 없을 시에는 오류가 발생합니다.

```
ex)
V = "WWWWWWWWWWWWWWVWWWWWWWW"
print(V.index("V"))
>> 14
```

▲ find 메소드

: 문자열에서 지정한 문자가 있는 위치를 숫자로 반환합니다. 만약 찾는 문자가 없다면, -1을 반환합니다.

```
ex)
funny = "ㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅎㅋㅋㅋㅋㅋㅋ"
funny.find("ㅎ")
12

funny.find("ㅊ")
-1
```

▲ replace 메소드

: .replace(“교체전”, “교체후”) 형태로 문자열의 일부분을 교체해서 출력합니다. (실제로 변수에 저장된 문자열이 바뀌지는 않습니다!)

```
hobbang = “받으렴 호빵맨, 헌 얼굴이란다!”
print(hobbang.replace("헌", "새"))
print(hobbang)

>>>
받으렴 호빵맨, 새 얼굴이란다!
받으렴 호빵맨, 헌 얼굴이란다!
```

▲ split 메소드

: .split() 괄호 안의 값을 기준으로 문자열을 끊어서 리스트로 반환합니다. 괄호 안에 아무것도 쓰지 않으면 공백을 기준으로 끊어줍니다.

```
ex)
cold = “제콜록가콜록감콜록기콜록에콜록걸려서콜록목소리가콜록잘콜록안나와요”
cold.split(“콜록”)
['제', '가', '감', '기', '에', '걸려서', '목소리가', '잘', '안나와요']
```

▲ join 메소드

: .join() 괄호 안의 문자열에 각 문자 사이마다 변수의 값을 끼워 넣어 줍니다.

```
ex)
egg = “달걀”
egg.join(“닭닭닭닭닭”)
'닭달걀닭달걀닭달걀닭달걀닭'
```