python

#12

>>> 6월 4일 배웠던 내용

리스트 복제의 특징

```
list1 = ['이정주', '황동진', '윤혜원', '지승훈']
list2 = list1

list2.append('문승기')

'문승기' in list1 # True
```

왜 그럴까?

list2 = list1

list1을 대입해 만든 list2에 항목을 추가하면 list1에도 항목이 추가된다. 결국 두 리스트는 같은 리스트로 유지된다.

list2 = list1.copy()

list2는 list1과는 별개의 리스트가 된다.

리스트+조건문+반복문

```
list3 = [n**2 for n in range(1,20)]

list4 = [n for n in range(1,21) if n % 2 == 0]

list5 = [3, 5, 6, 7, 8, 66, 296]

list6 = [n*5 for n in list5]
```

TODAY

- 리스트 더 알아보기
- 조건제시법

>>> 리스트의 다양한 기능

리스트의 다양한 기능

```
>>> list1 = [135, 462, 27, 2753, 234]
>>> list1.index(27)
2
```

리스트의 다양한 기능

```
>>> list2 = [1, 2, 3] + [4, 5, 6]
>>> list2
[1, 2, 3, 4, 5, 6]
# 기존에 있는 리스트에 새로운 리스트 합치기?
```

리스트의 다양한 기능

```
>>> list1
[135, 462, 27, 2753, 234]
>>> list1.extend([9, 10, 11])
>>> list1
[135, 462, 27, 2753, 234, 9, 10, 11]
```

리스트의 다양한 기능

```
>>> list1
[135, 462, 27, 2753, 234, 9, 10, 11]
>>> list1.insert(2, 999)
[135, 462, 999, 27, 2753, 234, 9, 10, 11]
# index==2 자리에 999 넣고, 뒤로 한 칸씩 밈
```

리스트의 다양한 기능

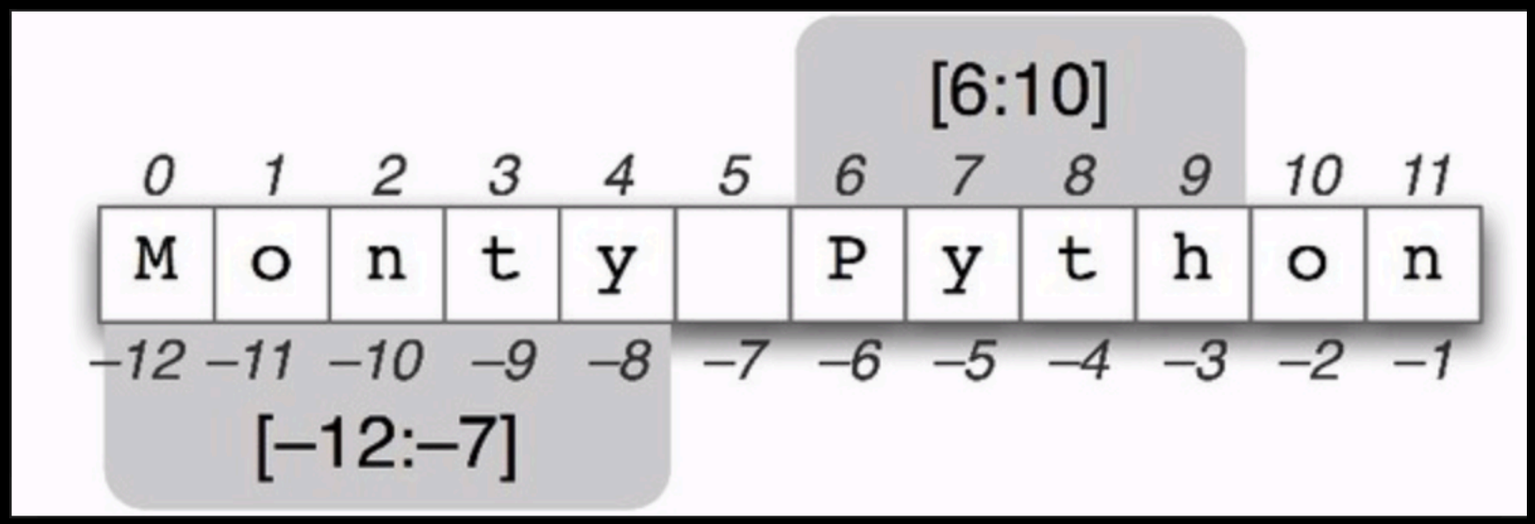
```
>>> list1

[135, 462, 999, 27, 2753, 234, 9, 10, 11]

>>> list1.insert(-1, 9999)

[135, 462, 999, 27, 2753, 234, 9, 10, 9999, 11]

# 마지막 자리에 9999를 넣고, 11은 한 칸 밀림
```

인덱스가 음수일 때

인덱스가 음수면 뒤에서부터 하나씩 값을 가져옴

리스트가 처음과 끝이 연결되어 있는 하나의 '띠'

라면 맨 처음 인덱스(0)보다 하나 적은 인덱스(-1)는 마지막 인덱스가 될 것



리스트의 다양한 기능

```
>>> list1

[135, 462, 999, 27, 2753, 234, 9, 10, 9999, 11]

>>> list1.insert(10000, 555)

[135, 462, 999, 27, 2753, 234, 9, 10, 9999, 11, 555]

# 유효하지 않은 index라면 가장 마지막에 넣음
```

리스트의 다양한 기능

```
>>> list1
[135, 462, 999, 27, 2753, 234, 9, 10, 9999, 11, 555]
>>> list1.sort() # 오름차순 정렬
>>> list1
[9, 10, 11, 27, 135, 234, 462, 555, 999, 2753, 9999]
```

리스트의 다양한 기능

```
>>> list1
[9, 10, 11, 27, 135, 234, 462, 555, 999, 2753, 9999]
>>> list1.reverse() # 내림차순 정렬
>>> list1
[9999, 2753, 999, 555, 462, 234, 135, 27, 11, 10, 9]
```

>>> 리스트와 문자열

문자열 다루기

```
>>> a = 'python'
>>> b = 'fun'
>>> a+b
'pythonfun'
>>> a*3
'pythonpythonpython'
```

문자열 다루기

```
>>> a = 'python'
>>> a.find('th')
2
>>> len(a)
6
```

문자열 다루기

```
>>> a = 'python'
>>> a.upper()
'PYTHON' # a 자체가 바뀌는 것은 아님
>>> a.lower()
'python' # a 자체가 바뀌는 것은 아님
```

문자열 다루기

```
>>> a = 'python is fun'
>>> a.replace('fun', 'boring')
'python is boring'
# a 자체가 바뀌는 것은 아님
```

문자열 다루기

```
>>> a = ' koreauniv '
>>> a.strip()
'koreauniv' # a 자체가 바뀌는 것은 아님
>>> a.lstrip()
'koreauniv ' # a 자체가 바뀌는 것은 아님
```

리스트와 문자열의 공통점

기본적으로 문자열은 개별 문자들을 값들로 가지는 리스트이다. 'hello' = ['h', 'e', 'l', 'l', 'o']

리스트와 문자열의 활용은 유사한 점이 많고, 상호 변환이 가능하다.

앞으로 다룰 언어 데이터를 분석하기 위해서 문자열의 특성을 이해할 필요가 있다.

리스트와 문자열

```
>>> str1 = 'Hello World'
>>> str1[0]
'H'
>>> str1[1]
'e'
```

리스트와 문자열

```
>>> 'H' in str1
True
>>> 'z' in str1
False
>>> str1.index('r')
8
```

문자열을 리스트로 바꾸기

```
>>> characters = list('abcdef')
>>> characters
['a', 'b', 'c', 'd', 'e', 'f']
>>> words = '파이썬 그리고 데 이 터 과학'
>>> words_list = words.split()
>>> words_list
['파이썬', '그리고', '데', '이', '터', '과학']
```

리스트를 문자열로 바꾸기

```
>>> time_str = '10:35:27'
>>> time_list = time_str.split(':')
>>> time_list
['10', '35', '27']
>>> ':'.join(time_list)
'10:35:27'
```

리스트를 문자열로 바꾸기

```
>>> words_list
['파이썬', '그리고', '데', '이', '터', '과학']
>>> ''.join(words_list)
'파이썬그리고데이터과학'
```

>>> 슬라이스

슬라이스(slice)

리스트, 문자열의 일부분을 '썰어서 가져오기'

list[first_index:last_index]

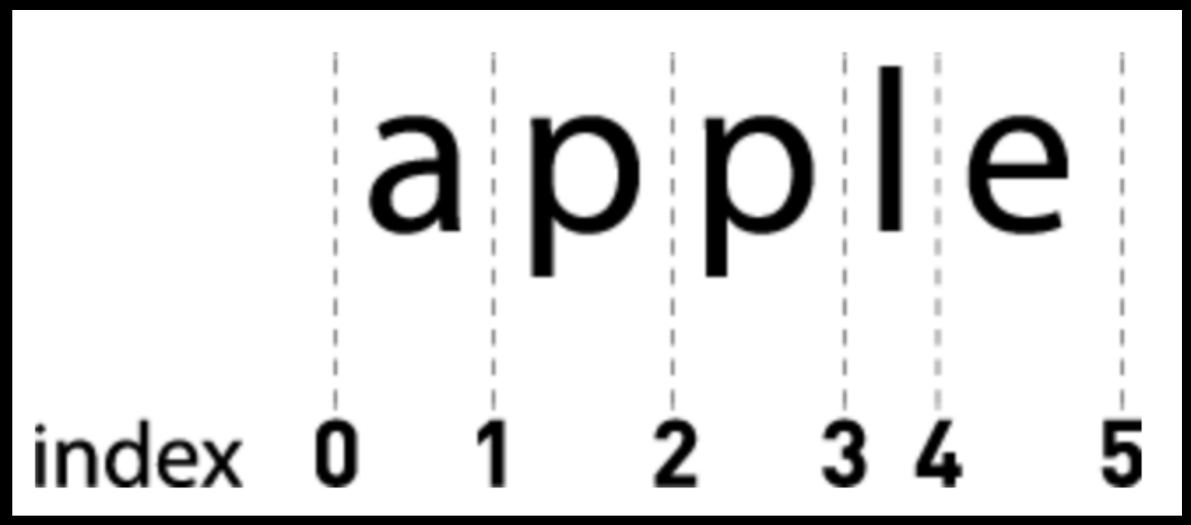
str[first_index:last_index]

리스트에서의 슬라이스

```
>>> list1 = ['영', '일', '이', '삼', '사', '오']
>>> list1[1:3]
['일', '이'] # 리스트를 슬라이스하면 리스트
>>> list1[3]
'삼' # 뭔가 이상하지 않은가?
```

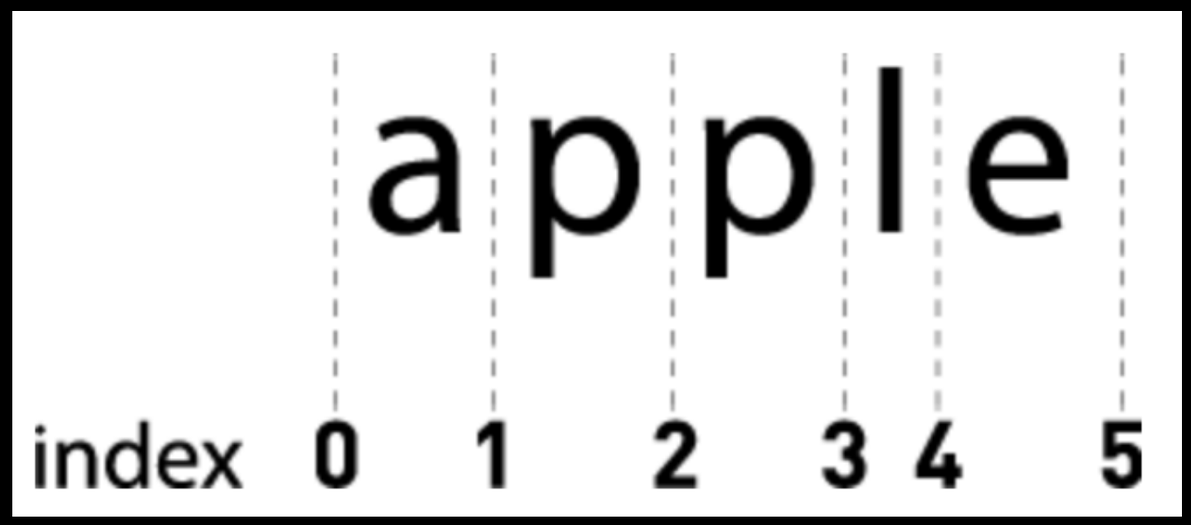
혼동하기 쉬운 슬라이스

```
>>> str1 = 'apple'
>>> str1[3]
'l'
>>> str1[0:3]
'app' # 문자열을 슬라이스하면 문자열
```



처음부터? 끝까지?

```
>>> str1 = 'apple'
>>> str1[3:]
'le'
>>> str1[:2]
'ap'
```



처음부터 끝까지?

```
>>> list1 = ['영', '일', '이', '삼', '사', '오']
>>> str1 = 'apple'
>>> list1[:]
['영', '일', '이', '삼', '사', '오']
>>> str1[:]
'apple'
```

=, [:], copy()

```
>>> list1 = ['영', '일', '이', '삼', '사', '오']
>>> list2 = list1
>>> list3 = list1[:]
>>> list4 = list1.copy()
>>> list1.append('육') # list1~4 중 바뀌는건?
```

>>> 듬성듬성한 슬라이스

듬성듬성한 슬라이스

```
>>> list1 = list(range(20)) # 0부터 19까지

>>> list1

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
15, 16, 17, 18, 19]

>>> list1[5:15]

[5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

듬성듬성한 슬라이스

```
>>> list1[5:15:2] # 5부터 15까지, 2씩 건너뜀
[5, 7, 9, 11, 13]
```

>>> 슬라이스로 리스트 수정하기

슬라이스로 리스트 수정

리스트의 추가, 수정, 삭제 등 배웠던 내용!

전부 슬라이스로 수행할 수 있음

슬라이스로 리스트 수정

```
>>> numbers = list(range(10))
>>> numbers
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> del numbers[0]
>>> numbers[:5]
[1, 2, 3, 4, 5]
```

슬라이스로 리스트 수정

```
>>> numbers[:5]
[1, 2, 3, 4, 5]
>>> del numbers[:5]
>>> numbers
[6, 7, 8, 9]
```

슬라이스로 리스트 수정

```
>>> numbers[1:3]
[7, 8]
>>> numbers[1:3] = [77, 88]
>>> numbers
[6, 77, 88, 9]
```

슬라이스로 리스트 수정

```
>>> numbers[1:3] = [77, 88, 99]
>>> numbers
[6, 77, 88, 99, 9]
```

슬라이스로 리스트 수정

```
>>> numbers[1:4] = [77, 88, 99]
>>> numbers[1:4] = [8]
[6, 8, 9]
```

>>> 조건제시법

if-elif-else

- 첫 조건문에 해당될 때...
- 첫 조건문에 해당되지 않으면서 이번 조건문에 해당될 때...
- 어떤 조건문에도 해당되지 않을 때...

for in list, for in range

- for in list: 리스트의 각 항목에 대해서...
- for in range: 숫자 하나씩 올라가면서 n번...

while, break, continue

- while: 특정한 조건을 만족시키는 한 계속 반복
- break: 반복문 탈출
- continue: 다음 반복으로

리스트의 조건제시법

```
>>> areas = []
>>> for i in range(1, 11):
    areas = areas + [i*i]
>>> areas
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

리스트의 조건제시법

```
>>> areas = [i*i for i in range(1, 11)]
>>> areas
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

리스트의 조건제시법

```
>>> areas = []
>>> for i in range(1, 11):
    if i % 2 == 0:
        areas = areas + [i*i]
>>> areas
[4, 16, 36, 64, 100]
```

리스트의 조건제시법

```
>>> areas = [i*i for i in range(1, 11) if i % 2
== 0]

>>> areas

[4, 16, 36, 64, 100]
```

리스트의 조건제시법

```
>>> areas = [i*i for i in range(1, 11) if i % 2]
>>> areas
[1, 9, 25, 49, 81]
```

딕셔너리의 조건제시법

```
>>> students = ['정주', '원호', '찬희', '나경', '승훈']
>>> for index, value in enumerate(students):
    print('{}번의 이름은 {}입니다.'.format(index, value))
```

딕셔너리의 조건제시법

```
>>> students = ['정주', '원호', '찬희', '나경', '승훈']

>>> students_dict = {'{}번'.format(index+1): value
for index, value in enumerate(students)}

>>> students_dict

{'1번': '정주', '2번': '원호', '3번': '찬희', '4번': '나경', '5번':
'승훈'}
```

딕셔너리의 조건제시법

```
>>> students = ['정주', '원호', '찬희', '나경', '승훈']

>>> scores = [20, 80, 75, 99, 97]

>>> scores_dict = {student : score for student, score in zip(students, scores)}

>>> scores_dict

{'정주': 20, '원호': 80, '찬희': 75, '나경': 99, '승훈': 97}
```

THANK YOU