시퀀스 자료형

시퀀스 자료형 이해하기

❖ 시퀀스 자료형

- 어떤 객체가 순서를 가지고 나열되는 것
- 문자열
  - 문자나 기호들이 순서대로 나열되는 시퀀스
- 리스트
  - 임의의 객체가 순서대로 나열되는 시퀀스
- 튜플
  - 리스트와 동일하나 수정은 불가

```
strdata = 'abcde'
listdata = [1, [2,3],'안녕']
tupledata = (100, 200, 300)
```

❖ 문자열 지정

- '', ""로 문자열 지정
- 여러 라인은 """ """ (삼중 따움표)로 지정

```
strdata1 = "나는 파이썬 프로그래머다"
strdata2 = "You are a programmer"
strdata3 = """I love
    python. You love
python too!
"""
strdata4 = "My son's name is John"
strdata5 = '문자열 "abc"의 길이는 3입니다.'
```

❖ 문자열 포맷팅

- 변하는 값을 포함사는 문자열을 표현하기 위해 하나의 양식으로 문자열을 만드는 것
- 포맷 문자열
  - 문자열 포맷팅에서 변하는 값을 나타내기 위해 사용하는 기호

포맷 문자열	설명
%s	문자열에 대응됨
%c	문자나 기호 한 개에 대응됨
%f	실수에 대응됨
%d	정수에 대응됨
%%	'%'라는 기호 자체를 표시함

❖ 문자열 포맷팅

- '포맷 문자열 '%(인자…)

```
txt1 = '자바'; txt2 = '파이썬'
num1 = 5; num2 = 10

print('나는 %s보다 %s에 더 익숙합니다.'%(txt1, txt2))
print('%s은 %s보다 %d배 더 쉽습니다.'%(txt2, txt1, num1))
print('%d + %d = %d'%(num1, num2, num1 + num2))
print('작년 세계 경제 성장률은 전년에 비해 %d%% 포인트 증가했습니다'%num1)
```

  - 인자 개수가 맞지 않는 경우 예외 발생

```
from time import sleep
for i in range(100):
    msg = '\r진행률 %d%%'%(i+1)
    print(' '*len(msg), end='')     // 출력할 문자열 길이만큼 지움
    print(msg, end='')
    sleep(0.1)
```

이스케이프 문자

❖ 이스케이프 문자

이스케이프 문자	설명
\n	줄 바꾸기
\t	탭
\Enter	줄 계속 (다음 줄도 계속되는 줄이라는 표시)
\\	'\' 기호 자체
\' 또는 \"	' 기호 또는 " 기호 자체

```
print('나는 파이썬을 사랑합니다.\n파이썬은 자바보다 훨씬 쉽습니다')

print('Name:John Smith\tSex:male\tAge:22')

print('이 문장은 화면폭에 비해 너무 길어 보기가 힘듭니다.\
그래서 \\Enter 키를 이용해 문장을 다음줄과 연속되도록 했습니다.')

print('작은따움표(\')와 큰 따움표(")은 문자열을 정의할 때 사용합니다.')
```

리스트 [ ]

❖ 리스트

- [ ]로 표시하며 요소를 콤마로 구분하여 순서있게 나열
- 요소로 임의의 객체가 모두 가능

```
list1 = [1, 2, 3, 4, 5]
list2 = ['a', 'b', 'c']
lsit3 = [1, 'a', 'abc', [1,2,3,4,5], ['a','b', 'c']]

list1[0] = 6
print(list1)
def myfunc():
    print('안녕하세요')

list4 = [1, 2, myfunc]
list4[2]()
```

튜플 ( )

❖ 튜플

- [ ]로 표시하며 요소를 콤마로 구분하여 순서있게 나열
- 요소로 임의의 객체가 모두 가능
- 수정은 불가(리스트와 차이점)

```
tuple1 = (1, 2, 3, 4, 5)
tuple2 = ('a', 'b', 'c')
tuple3 = (1, 'a', 'abc', [1,2,3,4,5], ['a','b','c'])
# tupe1[0] = 6

def myfunc():
    print('안녕하세요')

tuple4 = (1, 2, myfunc)
tuple4[2]()
```

사전 { }

❖ 사전

- 키와 값을 하나의 요소로 하는 순서가 없는 집합
  - 키와 값은 임의의 객체

  → 해시맵
- { }으로 요소를 나열

```
dict1 = { 'a' : 1, 'b':2, 'c':3}
print(dict1)
print(dict1['a'])
dict1['b'] = 7;
print(dict1)
print(len(dict1))
```

❖ 시퀀스 자료형의 연산

특성	설명
인덱싱	인덱스를 통해 해당 값에 접근할 수 있습니다. 인덱스는 0부터 시작합니다.
슬라이싱	특정 구간의 값을 취할 수 있습니다. 구간은 시작 인덱스와 끝 인덱스로 정의합니다.
연결	'+' 연산자를 이용해 두 시퀀스 자료를 연결하여 새로운 시퀀스 자료로 생성합니다.
반복	'*' 연산자를 이용해 시퀀스 자료를 여러 번 반복하여 새로운 시퀀스 자료로 생성합니다.
멤버체크	'in' 키워드를 사용하여 특정 값이 시퀀스 자료의 요소로 속해 있는지 확인할 수 있습니다.
크기정보	len( )을 이용해 시퀀스 자료의 크기를 알 수 있습니다. 시퀀스 자료의 크기는 문자열의 경우 문자의 개수, 리스트와 튜플인 경우 멤버의 개수가 됩니다.

시퀀스 자료 인덱싱

❖ 인덱싱(indexing)

- 시퀀스 자료형에서 인덱스를 통해 해당하는 값을 얻는 방법
- 인덱스는 0부터 시작
- 음수는 뒤에서부터 해석
- 잘못된 인덱스 지정시 예외 발생

```
strdata = 'Time is money!!'
listdata = [1, 2, [1, 2, 3]]
print(strdata[5])
print(strdata[-2])
print(listdata[0])
print(listdata[-1])
print(listdata[2][1])
```

strdata	T	i	m	e		i	s		m	o	n	e	y	!	!
인덱스	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	–15	–14	–13	–12	–11	–10	–9	–8	–7	–6	–5	–4	–3	–2	–1

시퀀스 자료 슬라이싱

❖ **시퀀스[시작 인덱스:끝 인덱스:스텝]**

- 시작 인덱스 : 슬라이싱 범위의 시작
- 끝 인덱스 : 슬라이싱 범위의 끝(미포함)
- 스텝 : 자료를 취하는 간격, 생략시 1

슬라이싱 범위	의미
[m:n]	시퀀스 자료의 인덱스가 m 이상 n 미만인 요소를 슬라이싱합니다.
[:n]	시퀀스 자료의 처음부터 인덱스가 n 미만인 요소까지 슬라이싱합니다.
[m:]	시퀀스 자료의 인덱스가 m인 요소부터 시퀀스 자료의 끝까지 슬라이싱합니다.
[:-n]	시퀀스 자료의 처음부터 끝에서 n번째 미만인 요소까지 슬라이싱합니다.
[-m:]	시퀀스 자료의 끝에서 m번째 요소부터 시퀀스 자료의 끝까지 슬라이싱합니다.

```
strdata = 'Time is money!!'

print(strdata[1:5])
print(strdata[:7])
print(strdata[9:])
print(strdata[:-3])
print(strdata[-3:])
print(strdata[:])
print(strdata[::2])

# 출력 결과
ime
Time is
oney!!
Time is mone
y!!
Time is money!!
Tm smny!
```

strdata	T	i	m	e		i	s		m	o	n	e	y	!	!
인덱스	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	−15	−14	−13	−12	−11	−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

시퀀스 자료 연결(+)

❖ **자료 형이 같은 시퀀스 자료는 +연산자로 연결가능**

- 새로운 시퀀스 자료가 생성
- 문자열 + 문자열
- 리스트 + 리스트
- 튜플 + 튜플

```
strdata1 = 'I love '; strdata2 = 'Phython'; strdata3 = 'you'
listdata1 = [1, 2, 3]; listdata2 = [4, 5, 6]

print(strdata1 + strdata2)
print(strdata1 + strdata3)
print(listdata1 + listdata2)
```

- 자료형이 다른 경우 +연산시 예외 발생

시퀀스 자료 반복(*)

❖ 시퀀스 *연산자

- 동일한 시퀀스 자료를 반복하여 새로운 시퀀스 자료를 생성
- 시퀀스자료*n
  - n: 반복 회수

```
artist ='빅뱅'
sing = '뱅~'
dispdata = artist + '이 부르는 ' + sing*3
print(dispdata)
```

시퀀스 자료 크기(len)

❖ len(시퀀스 자료)

- 시퀀스 자료의 크기 리턴
- 크기 : 시퀀스 요소의 개수

```
strdata1 = 'I love python'
strdata2 = '나는 파이썬을 사랑합니다'
listdata = ['a', 'b', 'c', strdata1, strdata2]

print(len(strdata1))
print(len(strdata2))
print(len(listdata))
print(len(listdata[3]))
```

멤버 체크 (in)

❖ in 연산자

- 자료에 어떤 값이 있는지 없는지 확인

```
<값> in <자료>
```

```
listdata  = [1, 2, 3, 4]
ret1 = 5 in listdata
ret2 = 4 in listdata
print(ret1); print(ret2)

strdata = 'abcde'
ret3 = 'c' in strdata
ret4 = 'l' in strdata
print(ret3); print(ret4)
```