02.파이썬 기본 문법-1

3.1 리스트(list)

순서를 가지는 객체들의 집합, 파이썬 자료형들 중에서 가장 많이 사용한다.

1) 리스트의 생성과 연산

[예제 list.py]

- 순서(index)가 있는 시퀀스 자료형이다.
- 시퀀스의 연산(인덱싱, 슬라이싱, 연결(+), 반복(*), len(), in, not in 등의 연산이 가능하다
- del() 를 이용하여 삭제도 가능하다.

```
testList = [1, 2, 'python']  #testList =[]   #testList=list()


print(testList[0], testList[1], testList[2], testList[-2], testList[-1], )
print(testList[1:3])
print(testList * 2)
print(testList + [3, 4, 5])
print(len(testList))
print(2 in testList)


del(testList[0])
print(testList)
```

리스트는 변경 가능한(Mutable) 자료형이다.

```
a = ['apple', 'banana', 10, 20]
a[2] = a[2] + 90
print(a)
```

리스트는 변경 가능한(Mutable) 자료형이다.

```
a = [1, 12, 123, 1234]

a[0:2] = [10, 20]
print(a)

a[0:2] = [10]
print(a)

a[1:2] = [20]
print(a)

a[2:3] = [30]
print(a)
```

슬라이스를 통한 삭제

```
a = [1, 12, 123, 1234]
a[1:2] = [ ]
print(a)
a[0:] = [ ]
print(a)
```

슬라이스를 통한 삽입

```
a = [1, 12, 123, 1234]

a[1:1] = ['a']
print(a)

a[5:] = [12345]
print(a)

a[:0] = [-12, -1, 0]
print(a)
```

2) 리스트의 메소드

```
a = [1, 12, 123, "hello", 3.14, 1000]
print(a)

#추가
a.append(5)
print(a)

a.insert(3, 1000)
print(a)

a.extend([6, 7, 8])
print(a)

#제거
a.remove(1000)
print(a)

a.pop(2)
print(a)

a.pop()
#del(a[4])
```

2) 리스트의 메소드

```
b = [1, 123, 1000, 12, 1000]
print(b)

# 카운트
print(b.count(1000))

# 뒤집기
b.reverse();
print(b)

# 정렬
b.sort()
print(b)

# index 찾기
print(b.index(1000))
```

3.2 튜플(tuple)

순서를 가지는 객체들의 집합이라는 것은 list와 같지만 **수정 불가(immutable)** 자료형이다.

1) 튜플 생성과 연산

[예제 tuple.py]

- 순서(index)가 있는 시퀀스 자료형이다.
- 시퀀스의 연산(인덱싱, 슬라이싱, 연결(+), 반복(*), len(), in, not in 등의 연산이 가능하다

```
t = (1, 2, 3)    # t = 1, 2, 3   # t = tuple([1,2,3])
print(t, type(t))

t = 1, 2, 'python'
print(t, type(t))

print(t[-2], t[-1], t[0], t[1], t[2])
print(t[1:3])
print(t[:])

print(t * 2)
print(t + (3, 4, 5))   # t = t + (3, 4, 5)  이렇게 해야 합쳐짐
print(t)
print(len(t))
print(5 in t)
```

튜플은 변경 불가능(immutable)한 시퀀스 형이다.

```
t = ('apple', 'banana', 10, 20)
t[0] = 'apple'
t[2] = t[2] + 90
```

X

따라서 슬라이스를 통한 치환, 삭제, 추가등은 지원하지 않는다.

```
t = ('apple', 'banana', 10, 20)
t[0] = 'apple'
t[2] = t[2] + 90
```

X

3.3 사전(dict) :딕션어리

순서를 가지지 않는 객체의 집합으로 key 와 value를 갖는다.

1) 딕션어리의 생성과 연산

[예제 dict.py]

시퀀스 자료형이 아니기 때문에 시퀀스의 연산 중 len(), in, not in 정도만 가능하다.

```
a={} # 생성방법

#추가
a['r38'] = "빅데이타"
a['r32'] = 'c언어'
print(a)

# 생성방법
d = {'basketball': 5, 'soccer': 11, 'baseball': 9}
d['volleyball'] = 6
print(d)

# 값출력
print(d['basketball'])
print(d.get('basketball'))
```

값 삭제 방법

```
d = {'basketball': 5, 'soccer': 11, 'baseball': 9}
del(d['basketball'])
print(d)

d = {'basketball': 5, 'soccer': 11, 'baseball': 9}
d ={}
print(d)

d = {'basketball': 5, 'soccer': 11, 'baseball': 9}
d.clear()
print(d)
```

기타

```
print(type(d))

print(len(d))

print('soccer' in d)
print('volleyball' not in d)
```

2) 사전의 메서드

딕션어리의 키들을 리스트로 반환

```
d = {'basketball': 5, 'soccer': 11, 'baseball': 9}
keys = d.keys()
print(keys)
print(type(keys))

for key in keys:
    print('{0}:{1}'.format(key, d[key]))
```

딕션어리의 값들을 리스트로 반환

```
values = d.values()
print(values)
print(type(values))
```

(key, values) 튜플의 리스트를 반환

```
items = d.items()
print(items)
```

3.4 집합(set)

순서가 없는 객체들의 집합이다. 수정이 가능한 자료형이다.

1) 세트의 생성과 연산

[예제 set.py]

시퀀스 자료형이 아니기 때문에 시퀀스의 연산 중 len(), in, not in 정도만 가능하다.

```
a = {1, 2, 3}
print(a, type(a))

print(len(a))
print(2 in a)
print(2 not in a)
```

2) 세트의 메서드

수정이 가능한 자료형이나 순서가 없기 때문에 메서드가 list에 비해 제한적이다.

```
s.add(4)
s.add(1)
s.discard(2)
s.remove(3)
s.update({2, 3})
s.clear()
```

수학의 집합 관련 메서드

```
s1 = set([1, 2, 3, 4, 5, 6, 7, 9, 10])
s2 = set([10, 20, 30])
print(type(s1), type(s2))

s3 = s1.union(s2) #합집합
#s3 = s1 | s2
print(s3)

s4 = s1.intersection(s2) #교집합
#s4 = s1 & s2
print(s4)

s5 = s1.difference(s2)  #차집합
#s5 = s1-s2
print(s5)
```

3.5 패킹과 언패킹

packing은 나열된 객체를 tuple로 저장하는 것을 말한다. unpacking은 반대의 작업으로 tuple, list 안의 객체를 개개의 변수로 할당할 수 있다.

[예제 unpacking.py]

```
t = 10, 20, 30, 'python'
print(t)
print(type(t))

# unpacking tuple
a, b, c, d = t
print(a, b, c, d)

# unpacking list
a, b, c, d = [10, 20, 30, 'python']
print(a, b, c, d)
```

unpacking에서는 왼쪽 변수가 부족한 경우 에러가 발생한다.

```
a, b = (10, 20, 30, 40)
```

3.6 형변환

내장 함수 list(), tuple(), set()를 사용해서 서로서로 변환이 가능하다.

[예제 unpacking.py]

```
t = (1, 2, 3)

s = set(t)
print(s, type(s))

l = list(s)
print(l, type(l))

t = tuple(l)
print(t, type(t))
```