파이썬 스터디 #9

오늘의 목표

- 딕셔너리
- 딕셔너리와 반복문
- 딕셔너리와 리스트
- 튜플
- 패킹과 언패킹
- 튜플을 이용한 함수 값 반환

딕셔너리

딕셔너리

- 리스트와 비슷해 보이나 다르다.
- 중괄호 { } 사용
- '이름표' : '값'형식으로 이름표에 따른 값을 반환하는 구조

IN NOTEPAD++

```
wintable = {

    'scissor': 'paper',

    'rock': 'scissor',

    'paper': 'rock',

}

print(wintable['scissor']) # 'paper'
```

IN NOTEPAD++

```
def rsp(mine, yours):

    if mine == yours: # 내가 낸 패와 상대방이 낸 패가 같음

        return 'draw' # 비겼다

elif wintable[mine] == yours: # 나==이름표, 상대방==값

    return 'win' # 내가 이겼다

else:

    return 'lose' # 그렇지 않으면 내가 졌다
```

IN NOTEPAD++

```
messages = {

    'win': '이겼다!',

    'draw': '비겼네.',

    'lose': '졌어...',

}

result = rsp('scissor', 'rock')

print(messages[result])
```

딕셔너리와 반복문

딕셔너리와 반복문

- for key in dict.keys()
- for value in dict.values()
- 이를 이용하여 이름들만, 혹은 값들만 출력할 수 있다.
- 둘 다 출력하고 싶다면 어떻게 해야 할까?

IN NOTEPAD++

```
ages = {'JJ':24, 'EB':22, 'SJ':20}

for key in ages.keys():

    print('{}의 나이는 {}입니다'.format(key, ages[key])

for key in ages:

    print('{}의 나이는 {}입니다'.format(key, ages[key])

for key, value in ages.items():

    print('{}의 나이는 {}입니다'.format(key, value)
```

두 개 이상의 값?

- list.enumerate() 에서는 두 개의 값을 동시 반환
- dict.items() 에서도 두 개의 값을 동시 반환
- #3(함수)에서 근의 공식 함수도 두 개의 값을 동시 반환
- 이를 튜플(tuple)이라고 한다.

딕셔너리와 리스트

값 추가, 수정

- 리스트에서 값을 추가할 때는 list.append 함수를 사용하며, 값을 수정할 때는 list[index]=value 로 대입한다.

- 딕셔너리에서는 값을 추가하거나 수정할 때 모두 dict[key]=value로 대입한다.

IN REPL

```
>>> list = [1, 2, 3, 4, 5]

>>> list[2] = 33

>>> list
>>> list

[1, 2, 33, 4, 5]

>>> list.append(6)

>>> list

[1, 2, 33, 4, 5, 6]
```

IN REPL

```
>>> dict = {'one':1, 'two':2}

>>> dict['one'] = 11

>>> dict

{'one':11, 'two':2}

>>> dict['three'] = 3

>>> dict

{'three':3, 'one':11, 'two':2}
```

값 삭제

- 리스트에서 특정 index의 값을 삭제할 때는 del 구문을 사용한다.

- 딕셔너리에서 특정 key의 값을 삭제할 때는 역시 del 구문을 이용한다.

- list.pop 또는 dict.pop 함수는 해당되는 값을 출력하면서 동시에 값을 삭제한다. 이는 원래 stack의 기능이다.

IN REPL

```
>>> list

[1, 2, 33, 4, 5, 6]

>>> del(list[0])

>>> list

[2, 33, 4, 5, 6]
```

IN REPL

```
>>> dict

{'three':3, 'one':11, 'two':2}

>>> del(dict['one'])

>>> dict

{'three':3, 'two':2}
```

딕셔너리와 리스트의 공통점

- 호출 # list[0], dict['one']
- 개수 확인 # len(list), len(dict)
- 값 확인 # 2 in list, 'two' in dict.keys(), 2 in dict.values()
- 모두 삭제 # list.clear(), dict.clear()

딕셔너리와 리스트의 차이점

- 삭제할 때 list의 index에 따른 value가 달라지나, dict의 key에 따른 value는 달라지지 않는다.

- 결합할 때 두 list를 + 연산자로 더할 수 있으나, 딕셔너리는 dict1.update(dict2)와 같이 사용한다.

튜플

리스트와 튜플

- 리스트: 순서가 정해진 값의 집합 (추가, 수정, 삭제 가능)
- [ ] 사용
- 튜플: 순서가 정해진 값의 집합 (순서와 값이 고정)
- ( ) 사용
- 튜플의 값들은 쉼표로 구분하기만 해도 됨

IN REPL

```
>>> tuple1 = (1, 2, 3)

>>> tuple1

(1, 2, 3)

>>> tuple2 = 1, 2, 3

>>> tuple2

(1, 2, 3)
```

IN REPL

```
>>> list1 = [1, 2, 3]

>>> tuple3 = tuple(list1)

>>> tuple3

(1, 2, 3)

>>> tuple3[0]

1
```

튜플을 사용하는 이유

- 두 변수의 값을 맞바꿀 때 - 패킹과 언패킹
- 여러 개의 값을 한 번에 전달하고 싶을 때
- 딕셔너리의 키에 값을 여러 개 넣고 싶을 때

FYI: 언어학에서의 튜플

- 형식의미론: 타동사(intransitive verb)의 논항들이 튜플로 묶여 있다고 본다.

- [**LOVE(John, Mary)**]^{M} is true if and only if "John loves Mary." is true in our model M.

- [**LOVE(Mary, John)**]^{M} is true if and only if "Mary loves John." is true in our model M.

- In LOVE(x,y), there is an **ordered** pair (x,y).

패킹과 언패킹

변수 하나에 여러 값 대입하기

- 패킹: 하나의 변수에 여러 개의 값을 넣는 것
- 언패킹: 패킹된 변수에서 여러 개의 값을 꺼내오는 것
- 튜플을 사용해서 할 수 있음

IN REPL

```
>>> a, b = 1, 2

>>> a
1

>>> b
2
```

IN REPL

```
>>> c = 3, 4
```

```
>>> c # 튜플
```

```
(3, 4)
```

IN REPL

```
>>> d, e = c # 대입

>>> d

3

>>> e

4
```

IN REPL

```
>>> f = d, e # 대입
```

```
>>> f
```

```
(3, 4)
```

IN REPL

```
>>> x = 5

>>> y = 10

>>> x, y = y, x

>>> x

10

>>> y

5
```

튜플을 이용한 함수 값 반환

여러 개의 값을 한 번에 전달

- list.enumerate() 에서는 두 개의 값을 동시 반환
- dict.items() 에서도 두 개의 값을 동시 반환
- #3(함수)에서 근의 공식 함수도 두 개의 값을 동시 반환
- 튜플(tuple)의 특성 중 하나이다.

IN REPL

```
list = [1, 2, 3, 4, 5]

for i, v in enumerate(list):

    print('{}번 값: {}'.format(i, v))
```

IN REPL

```
list = [1, 2, 3, 4, 5]

for a in enumerate(list):

    print('{}번 값: {}'.format(a[0], a[1]))

for a in enumerate(list):

    print('{}번 값: {}'.format(*a)
```

IN REPL

```
ages = {'JJ':24, 'EB':22, 'SJ':20}

for key, val in ages.items():

    print('{}의 나이는 {}'.format(key,val))

for a in ages.items():

    print('{}의 나이는 {}'.format(a[0],a[1]))

for a in ages.items():

    print('{}의 나이는 {}'.format(*a))
```

감사합니다.