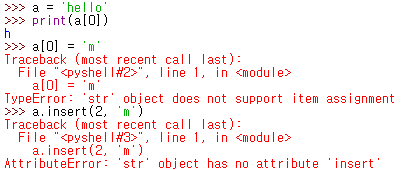
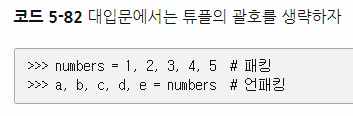
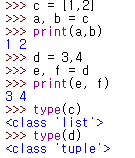
**< 튜플, 딕셔너리 >**

1. **튜플**

* 리스트와의 차이점 : 튜플은 인덱스를 통한(참조연산을 통한) 값 **변경** 불가능!
* 또한, sort(), pop(), insert() 등도 불가능 (추가, 삽입, 삭제 x)
* 리스트 vs 튜플&문자열 (문자열의 일부 변경도 불가능!)
* 
* **튜플**은 **“변경 불가능한 자료형”** 이다!!
* **값 변경은 x,** 그러나 **값 참조까지는 가능** (인덱스로 값 가져오는 것 까지는 가능!) 🡺 **슬라이싱**, **+ 연산**(이어붙이기)

1. **패킹과 언패킹**

* 
* 튜플은 좌우 괄호를 생략해도 튜플 타입으로 생성됨
* 따라서 numbers의 type은 tuple!
* 튜플 언패킹 : 튜플의 각 요소를 여러 개의 변수에 할당하는 것

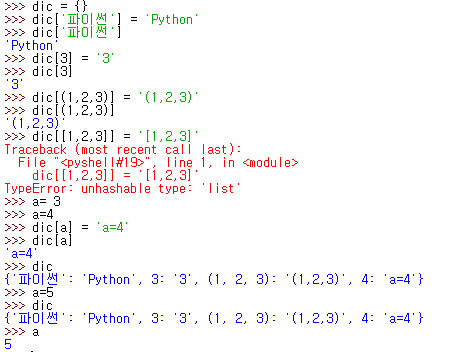


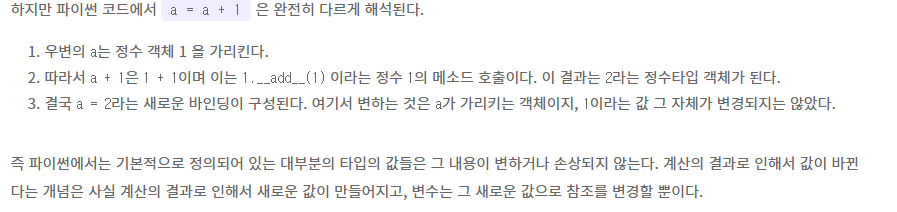
* 이런 식으로 패킹(c와 d)과 언패킹(a, b, e, f) 가능!

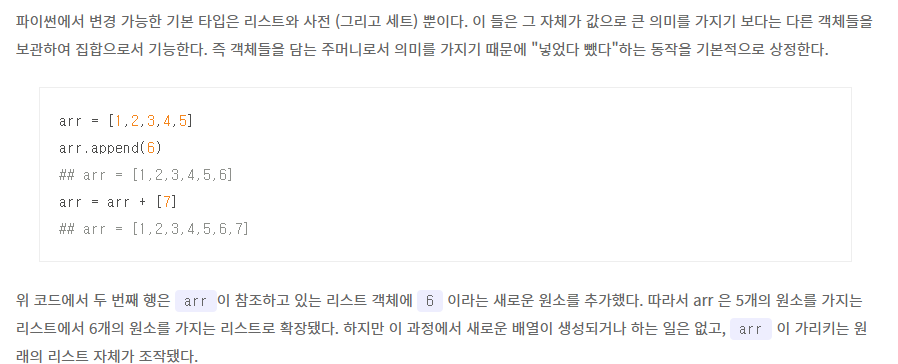
1. **튜플 메소드**

* 튜플은 변형 불가능한 자료형이므로, 메소드가 **2개뿐**!!
* **index()** : 찾고자하는 데이터의 인덱스, **count()** : 일치하는 데이터 개수 세기

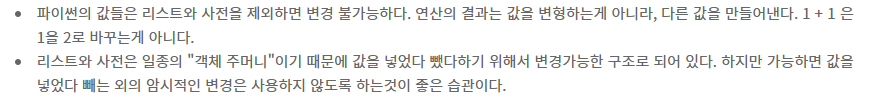
1. **딕셔너리**

* 튜플보다는 **리스트와 비슷**. 값 접근 & **변경 가능**!!
* 단, 리스트와 달리 인덱스~~[0]~~ 이렇게 순서대로 접근하는 것이 아니라,
* **dic[key값]**으로 접근!! 🡺 키(key) & 값(value) 쌍으로 구성
* **‘해싱’(Hashing)** : 키(key)를 이용해서 단번에 데이터가 저장된 위치의 주소를 계산하는 방식
* *딕셔너리 : ‘변경 가능한 자료형’ (ex.* ***리스트, 딕셔너리****)*
* *딕셔너리의* ***key(인덱스)*** *: ‘변경 불가능한 자료형’ (ex.* ***튜플, 문자열, 숫자형****… 등 기본 모든 객체!!)*
* **
* 여기서 알 수 있듯이, **list형은 변경 가능한 자료형**(**객체!**)이므로 딕셔너리의 key값으로 사용될 수 없다!
* <https://soooprmx.tistory.com/entry/%ED%8C%8C%EC%9D%B4%EC%8D%AC-%EA%B0%92%EB%93%A4%EC%9D%98-%EB%B3%80%EA%B2%BD%EA%B0%80%EB%8A%A5%EB%B3%80%EA%B2%BD%EB%B6%88%EA%B0%80%EB%8A%A5>

*🡺* ***파이썬 값들의 변경 가능/불가능****에 대한 글*

**

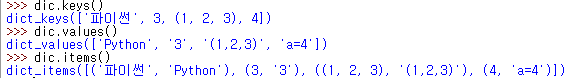
* 참고로, a=4에서 걍 4만 의미있는 것임! 다시 말해, a라는 변수에 4라는 변경 불가능한 객체(숫자형 객체)가 들어갔고, 그 값을 dic의 key값으로 사용한 것일 뿐, a를 그 다음에 5로 바꾼다 한들, key값이 바뀌진 않음!!

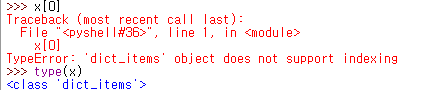
**

* **(추가) : dic = {} 후, dic[‘one’] = ‘1’ 이렇게 안하고, 한번에 { } 안에 정의와 값 할당을 한번에 할 수 있음!!!**
* 

1. **딕셔너리 메소드**

* **keys()** : 키의 리스트 추려냄 / **values()** : 값들의 리스트 추려냄
* ****items()** : 키&값 쌍(*튜플!*)으로 이루어진 리스트 반환





* 이런 식으로, 아예 ‘dict\_items’라는 type! 따라서 리스트라고 착각해서 인덱스로 접근할 수 없다.