Week 8

File I/O

기본적으로 open('path', 'mode')형식으로 파일을 읽을 수 있다.

모듈을 불러와 사용하는 것과는 다른 개념인데

오픈을 할때 path에 경로를 입력해야하는데 그냥 파일이름만 써서 열려면 현재 디렉터리에 같은 위치에 있어야한다.

경로에는 절대 경로가 있고 상대 경로가 있는데

절대경로는 /~/~ 이런식으로 파일의 경로를 모두 적는 것이고

상대경로는 이 파일이 위치한 위치를 기준으로 주소를 적는 것이다.

../  : 상위 디렉터리로 이동

/ : 하위 디렉터리로 이동

. (아님 생략) :  현재 디렉터리

상대경로를 쓰는 것이 파일이 이동해도 디렉터리 구조만 유지하고 있으면 에러가 나지 않으므로 상대경로를 쓰는게 더 좋다.

close() : 열었던 파일을 닫는 함수

사실 파이썬은 프로그램이 종료되면 열었던 파일은 자동으로 닫혀서 close()를 생략해도 되지만 close()를 써서 직접 닫아줘야지 오류가 발생할 일이 없다.

(쓰기모드로 열은 파일을 다시 열려고 하면 오류가 발생한다고 한다.)

with open('path', 'mode') as 변수 :

를 쓰면 변수에 파일 객체에 저장되고 블록문이 수행되고 끝나면 자동으로 close를 해준다

파일 열 때 모드가 있는데 종류는 세 가지 이다. r, w, a

'r' : 읽기 모드 - 파일을 읽을 때 사용

'w' : 쓰기 모드 -파일을 쓸 때 사용 (쓰기모드로 열 때는 해당 파일이 없다면 새로 생성 , 하지만 파일이 있고 안에 내용이 있다면 원래 있던 내용이 모두 사라진다, 조심해서 쓰기)

'a' : 추가 모드 - 파일의 마지막에 새로운 내용을 추가할 때 사용

파일을 읽는 방법

file=open(path, 'r')

file.read() : 파일의 내용 전체를 문자열로 리턴한다. read(숫자)를하면 숫자크기만큼의 텍스트를 읽는다(숫자는 공백도 포함 즉 (10)을 하면 10글자를 읽는 것)

file.readline() : 파일의 가장 첫 줄을 문자열로 리턴한다.

(파일을 닫지 않은 상태에서 readline으로 첫 줄을 읽고 다시 readline을 하면 첫 줄은 읽은 것이므로 그 다음 줄이 출력된다. 따라서 반복문(readline)을 통해 readlines처럼 기능을 할 수 있다.)

file.readlines() : 파일의 모든 줄을 읽어 각각의 줄을 요소로하는 리스틀 리턴한다.

Tip : .strip()으로 공백을 지우자! (week6 string method)

* set ( 순서 상관 x, 중복요소 없음)

set method

* + s.add(value) : set에 value값 추가한다.
  + s1.update(iterable) : dict의 update, list의 extend와 같다 여러개의 값을 추가할 때 사용
  + s.remove(value) : set에서 value값을 지운다
  + s.clear() : set에 있는 모든요소를 제거한다.
  + s1.union(s2) : s1과 s2의 합집합 (Operators  : s1|s2)
  + s1.symmetric\_difference(s2) : s1과 s2의 교집합 을 제외한 합집합 (Operators  : s1^s2)
  + s1.differecne(s2) : s1-s2 (s1의 차집합) (Operators  : s1-s2)
  + s1.intersection(s2):  s1,s2교집합 (Operators  : s1&s2)
  + s1.issubset(s2) : s1이 s2의 부분집합이냐 라고 물어보는 형식 (리턴값 boolean True or False) (s1<=s2)
  + s1.issuperset(s2) : s2가 s1의 부분집합이냐라고 물어보는 형식 (리턴값 boolean) (s1>=s2)

빈 set은 {}이 아니다 그냥  {}을 쓰면 빈 딕셔너리를 만든것이라고 한다.

빈 set은 set()으로 인자를 넣지 않고 typecast함수를 쓰자.

Tuple(순서는 있지만 값은 변할 수 없다.)

* + 하나의 요소만 갖는 튜플은 (a,)처럼 뒤에 ,를 붙여줘야한다.
  + 튜플의 요소는 변할수 없다 근데 튜플의 요소가 리스트인경우 리스트의 값은 변할 수 있다. (즉 리스트인것은 변하지 않지만 리스트안의 요소는 변하게 할 수 있다.)
  + 튜플은 한번에 여러가지 변수를 assign 할 때 쓰인다.

빈 튜플은 ()이라고 표현한다.

Dictionary (map, (순서는 없고, key는 변하지 않고 value는 변할 수 있다.), key에 따라 value를 꺼내오는 형식)

* + key값은 딕셔너리 내 중복이 없어야하고 , 변하지 않는 값이어야 한다.
  + Value값은 중복되어도 상관 없고, 리스트형식이어도 상관없다(즉 value값은 하나지만 하나안에 여러가지 데이터를 담는 리스트가 가능하다.)

dict method

생성

* + D[key]=value : key:value 값이 들어간다.
  + del D[key] :  key값에 해당하는 value와 함께  key:value 쌍이 지워진다.
  + D.clear() : 딕셔너리 안에 있는 모든 항목이 지워진다
  + D.get(key) : key값에 해당하는 value값을 리턴한다.
  + D.get(key,value) : 왜 사용하는지는 모르겠는데 기존에 key값이 있으면 기존 value값을 돌려주고 없으면 새로입력한 value를 리턴한다. 대신 새로운 key값을 넣어도

      기존 딕셔너리에 추가되지는 않는다.

* + D.setdefault(key) : D.get(key)와 같다 key값에 해당하는 value값을 리턴한다.
  + D.setdefault(key,value) : get 과는 다른데 기존에 key값이 있으면 기존 value값을 돌려주고 없으면 새로입력한 value를 리턴한다. 대신 새로운 key값을 넣으면 기존 딕셔너리에 추가된다.
  + D1.update(D2) : D1에 D2에 있는 key: value를 모두 추가시킨다.( list 의 extend)
  + D.pop(key) : key값을 안쓰면 에러가 나고(순서가 없으니 마지막이라는 것도 없다.) key값에 해당하는 value를 리턴하고 기존 딕셔너리에서 제거한다.
  + D.pop(key,value) : 기존에 key값이 있으면 기존 value값을 돌려주고 제거한다. 없으면 위의 경우와 달리 에러가 나지 않고 입력한 value값을 리턴한다.
  + D.keys() : 딕셔너리에 있는 key값들을 dict\_keys 리스트형태인 객체로  리턴한다.
  + D.values() : 딕셔너리에 있는 value값들을 dict\_values 리스트형태인 객체로 리턴한다.
  + D.items() : (key,value) 튜플 형태로 모든 key,value를 dict\_items 리스트형태인 객체로 리턴한다.[(key,value),(,)....]

~객체로 리턴한다고 하면 형태는 리스트지만 리스트로 타입을 변경하지않으면 리스트의 매서드를 적용할 수 없다.

빈 딕셔너리는 {}이다.

from collections import defaultdict를 이용해 defaultdict함수를 이용해

a=defaultdict(int) 를 하면 a의 딕셔너리는 key값이 없으면 자동으로 int의 초기값인 0을 배정해준다.

int대신 list도 가능한 것 같다.

in operator(반복가능한 자료형에 찾는 값이 있는지 알려주는 연산자(리턴값 boolean))

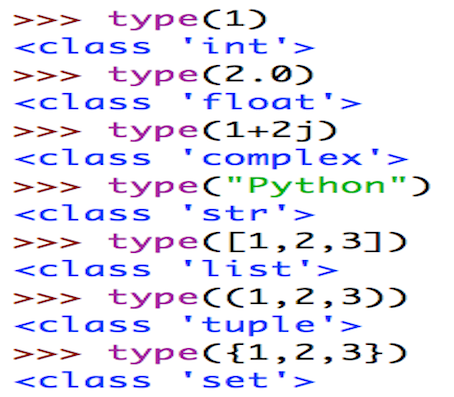
* + set, tuple가능
  + dict에서는 key값만 가능 value값은 안된다.

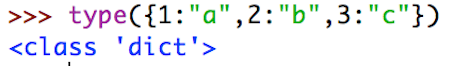
* 박정진#38452

2018.04.30. 18:16:19 (126일 전)

[이동](https://khlug.org/group/35951) [수정](https://khlug.org/group/35951) [삭제](https://khlug.org/group/35951) [보관](https://khlug.org/group/35951)

Week 2에서 자료의 형태 type 을 배웠는데 지금까지 안  type을 정리해보면

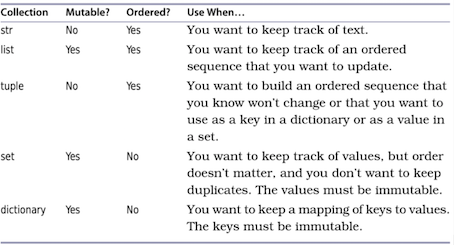




type함수를 써서 각각의 객체의 타입을 확인해보면 출력값에 <class  'type'> 이라고 나온다.

class는 파이썬이 type을 보여주는 방법이다.

각각의 기본으로 파이썬이 갖고있는 class들은 그 클래스가 갖고있는 고유의 method(매서드)가 있다.



반복 가능한 자료형(시퀸스)를 변할수 있는지 와 순서가 정해져있는지 에 대한 특징으로 구별 해놓은 표이다.(Week 8 Data collection type)

정리를 하면

|  |  |  |  |
| --- | --- | --- | --- |
| 타입 | 변할 수 있는지 ? | 순서가 정해져 있는지? | 언제 쓰는지? |
| str(문자열) | No | Yes | 텍스트를 조사(추적)하고 싶을 때 |
| List(리스트) | Yes | Yes | 업데이트(변경)이 가능한 순서가 있는 자료를 원할 때 |
| tuple(튜플) | No | Yes | 변경이 불가능한 순서가 있는 자료를 원할 때 혹은 딕셔너리에서 key로 사용하려거나 혹은 set의 value(값)으로 사용하고자 할 때 |
| set(집합) | Yes | No | 순서는 상관없는 값(value)만 있고 복제가 불가능한 자료를 원할 때 (값은 바뀌지 않는다) |
| Dictionary(딕셔너리) | Yes | No | 키(key)에 대응하는 값(valule)을 찾고 싶을 때 (value값은 바뀌지 않는다.) |