Week 3

* Text (문자열 자료형)

문자열들은 자료형이 str인데  string은 따옴표 안에만 들어있으면 문자열이라고 생각하면 된다.

 ' ', " " 의 따옴표를 쓰는데 이 안에 있는 내용은 전부 문자열 자료형이 되는것이다.

그후 string에 관한 operation 이 설명되는데 지금 배운 것 말고 후에 string method에서 더 다양한 str관련 내장함수를 배우니 여기서는 설명되는 것만 공부를 하면 될 것 같다.

len() : 반복 가능한 자료형의 요소의 개수를 리턴해주는 함수이다(여기서는 문자열의 개수를 리턴해준다고 생각하자) -> 리턴값의 자료형은 정수형이다

int() : 객체의 자료형을 정수형으로 바꿔준다

float() : 객체의 자료형을 실수형으로 바꿔준다

* 문자열 연산

문자열 연산에는 +와 \*가 있는데 그냥 생각하는 대로

+는 문자열을 더해서 연결하기(Concatenation)

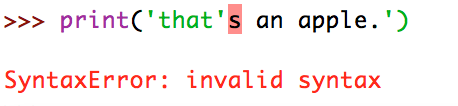
\*는  문자열 곱하기 인데  문자열을 곱한 만큼 반복하는 것이다.

Tip : 문자열을 +, \* 을 할 때 계산하는 자료가 모두 문자형인지 확인해야한다 숫자형+문자형을 하면 에러가 나므로 확인을 하자!

* 이스케이프 코드

우리가 문자열을 작성할 때 That's an apple. 이라는 문자열을 출력해본다고 가정해보자

아는대로 print('that's an apple.')이라고 입력해보자



이렇게 문법오류가 난다.

또 he said, "Life is too short, you need a python." 라고 출력해본다고 하자.

그래서 print("he said, "Life is too short, you need a python."")라고 입력해보자

https://khlug.org/image/4745abbaa35aac13.png

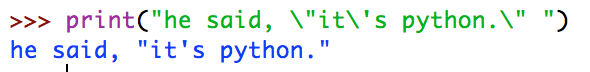
똑같이 문법오류가 난다.

해결하는 방법은 '를 쓸거면 ""로 열고닫고 "를 쓸거면 '로 열고닫으면 되는것이다.

하지만 ', "둘다 써야되는 경우가 있다 그럴 때 이스케이프 코드를 쓰는데 교재에 escape sequence라고 나와있는 것에 대해 알아보자



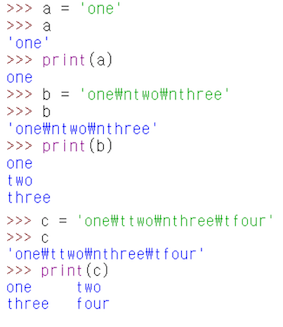
여기서  필요한건 \' 와 \" 이다. 즉 쓰는 방법이



이런 식으로 필요한 곳에 \", \'을 써넣으면 ", ' 이 출력된다.

\n, \t도 많이 쓰이는 이스케이프 코드다.

\n은 줄을 바꿔주는 코드이고 \t는 탭간격만큼 단어를 띄워준다(정렬해준다).



예시는 교재에 있는 그림이다.  맨 밑에 c를 보면 \t를 써서 한 열에 딱 맞게 끔 정렬해준 모습이다.

\n 과 함께  줄을 바꿀 수 있는 방법이 있는데 Week 2에서 docstring  할 때 """, ''' 따옴표 3개를 썼었다.

""", '''를 쓰면  여러 줄을 쓸 수있게 파이썬에서 만들었다.

이스케이프 코드에서는 \n, \t, \\, \", \' 만 알면 어느정도 해결 되는 것 같다.

¶ Tip : len()으로 문자열의 개수를 셀 때 이스케이프 코드는 하나로 센 다. 즉 \"을 썼으면 \" 자체가 두 개가 아닌 하나로 계산된다.

* print() 의 기능

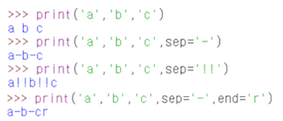
print() 기능에서 출력 기능 외에도 sep="(내용)" end="(내용)" 이 있다. 순서는

print("출력 내용 1","출력 내용 2","출력 내용 3", ..."출력 내용 n",  sep="(내용)", end="(내용)")순이다.

sep="(내용)"은  출력 내용 사이사이에 들어갈 내용을 정한다.

end="(내용)"은  출력할 내용의 끝에 들어갈 내용을 정한다.

예시를 보며 알아보자.

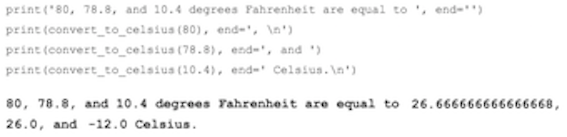


print()에서 출력내용 사이에 ,(쉼표)를 넣으면 출력 될 때 띄어쓰기가 되고 출력이 된다.

두번째 print()를 보면 sep="-"가 되있어서 출력할 때 a,b,c 사이에 -이 들어간걸 볼 수 있다.

세번째 도 마찬가지 !!이 내용 사이사이에 들어간다.

마지막은 end='r' 이 들어갔는데 출력내용을 보면 끝 내용에 붙여서 r이 출력된 것을 볼 수 있다.



예시를 보면 첫 문장에 ~equal to', end=" "이 되어있는데 이 것은 마지막에 띄어쓰기가 붙어 다음 출력내용을 전 문장에 이어서 하게 된다.

확인해보면 equal to 다음에 나오는 출력내용이 같은 문장에 붙어서 나오는 것을 확인할 수 있다.

두번째 문장을 보면 end=",  \n"가 되어있는데 끝에 ,를 붙이고 다음출력은 줄을 바꿔서 하라는 내용이다.

확인해보면  다음 출력 내용인  26.0 이 행이 바뀌어서 출력되는 것을 볼 수 있다.

세 번째 네 번째도 첫 ,두 번째 와 마찬가지로 이어서 붙이게 하는 end=" ", 와 다음문장으로 넘어가서 출력하라는 end="\n" 이 모두 적용된 것을 확인 할 수 있다.

sep=" " 와 end=" " 를 써서 출력되어지는 내용의 형태를 조종할 수 있다.

* input()

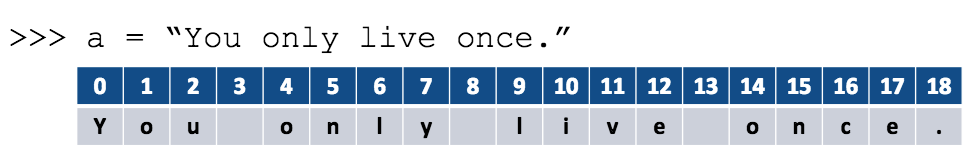
input()은 사용자의 입력을 받는 함수이다.

input()으로 입력을 받은 값의 type은 str이므로 수식이나 계산을 할거면 int, float등 숫자형태의 자료형으로 typecast를 해야한다.

* 문자열 인덱싱 및 슬라이싱

인덱싱은 가르키는 것이고 슬라이싱은 잘라낸다는 의미이다

일단 **파이썬은 처음을 0부터 세는 것을 꼭 기억해야한다.**



이 표를 보면서 생각해보자.

파이썬은 0부터 세므로 0부터 시작하는 것을 볼 수 있고 띄어쓰기 한 공백도 세는 것, 마지막 .도 세는 것을 볼 수 있다.

인덱싱, 슬라이싱 모두 기본형태는        변수(문자열)[문자번호]             이런 형태이다

즉 내가 Y를 가르키려면 a[0]이라고 입력하면 되는 것이다.

슬라이싱의 기본 형태는

변수(문자열)[start:stop:step]형태이다.

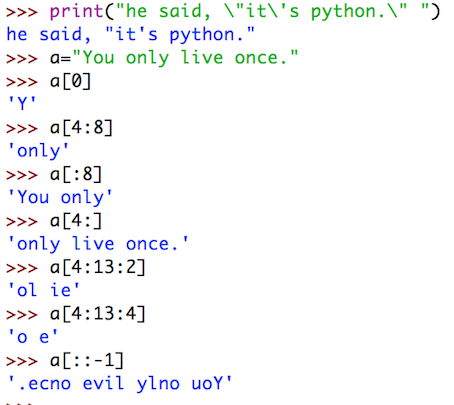
step은 안 써도 괜찮다.

그래서 start한 숫자부터 stop숫자 하나 전까지 출력이 된다. 즉 start<=슬라이싱 되는 번호<stop까지 슬라이싱 되는 것이다

예제로 내가 only를 슬라이싱 할려면 a[4:8]을 입력해야 된다는 것이다.

start 의 숫자를 안쓰면 처음부터 슬라이싱이 되고

stop의 숫자를 안쓰면 끝까지 슬라이싱 된다.



step은 얼마나 점프할 것이냐 라는 개념인데

a[4:13]은 only live가 출력 되는데 step을 2라 설정하니까 두 칸 씩 건너뛰어서 ol ie가 출력되는 것을 볼 수 있다

a[::-1] 은 a전체를 출력하는데 -1만큼 건너뛰라는 것이다. -1만큼 건너뛰는 것이니까 문자가 역순으로 출력된다.

* Boolean

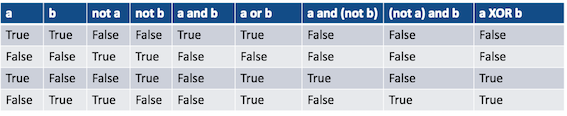
bool자료형은 참과 거짓 즉,  True와 False 두가지 형태이다.

파이썬에 입력할 때 첫번째 글자 T, F가 보는 것과 같이 꼭 대문자로 써야 인식을 한다.

* 불리언자료에 쓰이는 연산자가 있다

일단 우리가 흔히 아는 and, or, not이 있다 .이를 논리연산자라고 한다

논리 연산자 관련해서는 교재에 있는 표를 보면서 하면 이해가 더 쉬울 듯 하다.



and : a,b 모두 True이어야 True를 출력하는 연산자 -> 하나라도 False이면 False을 출력

or : a,b 하나라도 True이면 True인 연산자  -> 둘다 False이어야 False을 출력

not  : 해당 진위의 반대를 출력 ->True면 False, False면  True

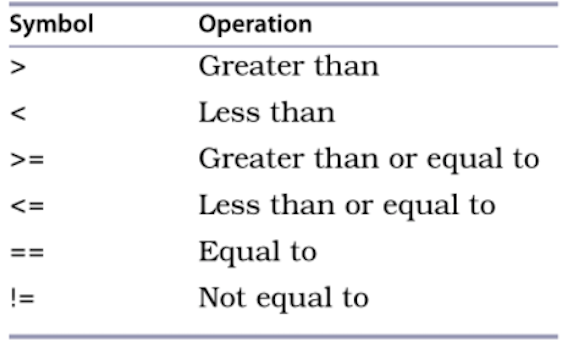
표를 보다보면 XOR이 있는데 Exclusive OR이라는데 표현을

XOR -> (a and not b) or (not a and b)

라고 표현한다.

* 다음엔 비교 연산자가 있다 숫자의 크고 작음 같음을 비교해준다

비교연산자 관련해서 교과서에 나온 표를 보자



x < y : x가 y보다 작다

x > y : x가 y보다 크다

x >= y : x가 y보다 크거나 같다

x <= y : x가 y보다 작거나 같다

x == y : x와 y가 같다

x != y : x와 y가 같지 않다

이런 비교연산자를 입력하면 출력되는 값은 True , False인 bool자료형이 나온다

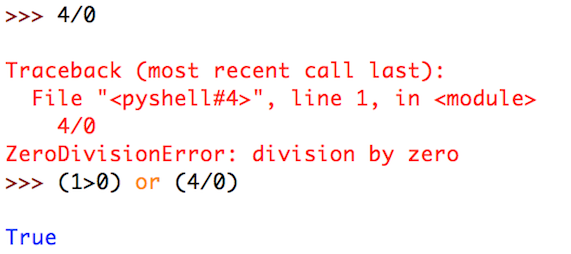
* 문자열, 숫자 등 불리언이 아닌 자료형에도 참과 거짓이 있는데 조건문에서 쓰이므로 알아두면 좋다.

|  |  |  |
| --- | --- | --- |
| 자료형 | 참 | 거짓 |
| 숫자 | 0이 아닌 숫자 | 0 |
| 문자열 | "어떤 것이든 써있으면 됨" | ""(empty string) |
| 리스트 | [요소가 하나 이상 있으면 됨] | [](empty list) |
| 튜플 | (요소가 하나 이상 있으면 됨) | ()(empty tuple) |
| 딕셔너리 | {key : value가 하나 이상 있으면 됨} | {}(empty dictionary} |

* short-circuit evaluation

파이썬은 계산을 효율적으로 하기위해서 이런 short-circuit evaluation이 있다.

1. a or b 를 연산할 때  먼저 나온 a를 연산했을 때 True면 뒤에 있는 b는 연산하지 않는다 -> b의 결과가 어떻든 a 가 True이므로 a or b는 True 가 된다
2. a and b를 연산할 때 먼저 나온 a를 연산했을 때 False면 뒤에 있는 b는 연산하지 않는다 ->  b의 결과가 어떻든 a가 False이므로 a and b는 False가 된다



예제로 ZeroDivisionError를 썼는데 기존에 4/0을 하면 에러가 난다

하지만 (1>0)(True) or (4/0)(Error)를 하면 4/0가 Error코드이므로 Error가 난다고 생각하지만

출력값은 True로 나온다. 즉 파이썬은 4/0을 계산을 하지도 않은 것이라는 것을 알 수있다.

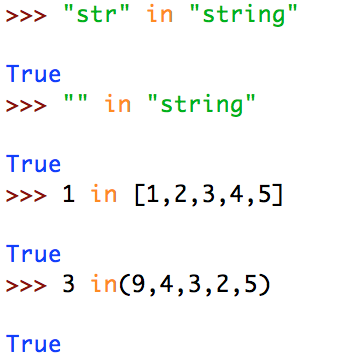
* 숫자말고도 문자열에도 비교연산자를 적용할 수 있다.

우위에 있는 기준은 사전식 배열로 봤을 때 뒤에 있으면 있을 수록 더 크게 연산한다.

즉 가<나<다<라 .... 이런 순이다.

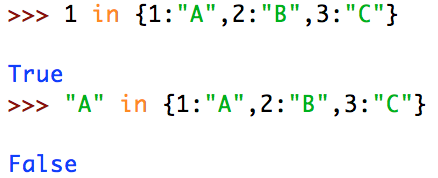
이런 코드를 입력하면 출력값은 bool자료형이다.

* 문자열, 리스트, 튜플, 딕셔너리 등 반복가능한 자료형에서는 요소가 해당 자료 안에 있는지 확인하는 in 함수로도 bool자료형을 출력 할 수 있다.



empty string 은 공집합이라고 생각하면 될 것같다 집합의 부분집합에 항상 공집합이 들어가는 것 처럼 empty string도 마찬가지로 어떤 문자열에도 속해있다.

위에 보는 것처럼 리스트, 튜플도 요소가 있는지 없는지 로 bool자료형을 출력할 수 있다



딕셔너리는 조금 다른데 딕셔너리에서는  key : value 값으로 구성되는데 in 은 key만을 찾는다

위의 예에서 딕셔너리 안에 있는 key 값이 있는지 물어보면 True가 나오는데 value값인 "A"가 있는지 물어보면 있는데도 False라고 나오는 것을 볼 수 있다.

결론 -> 딕셔너리에서 in은 해당 key값이 있는지 물어본다