정규식과 람다 완전 이해를 목표로 작성한 글

이었지만 그 후로 여러가지가 짬뽕되어 그냥 파이썬 정리 글이 되어버린

**1. 정규식**

re모듈을 통해 사용 할 수 있다. (import re). 참고([링크1](https://shoark7.github.io/programming/python/about-list-comprehension-python))

**정규식= [(**변수를 활용한 값**) for (**사용할 변수 이름**) in (**순회할 수 있는 값**)]**

여기서 순회할 수 있는 값이란 다른 리스트, set, dict, tuple 등을 이야기 한다.

**첫번째 예제. (정규식과 메타 문자를 모두 사용함.)**

정규표현식을 쓰지 않은 스크립트↓

data = """

park 800905-1049118

kim  700905-1059119

"""

result = []

for line in data.split("\n"):

    word\_result = []

    for word in line.split(" "):

        if len(word) == 14 and word[:6].isdigit() and word[7:].isdigit():

            word = word[:6] + "-" + "\*\*\*\*\*\*\*"

        word\_result.append(word)

    result.append(" ".join(word\_result))

print("\n".join(result))

정규표현식을 사용한 스크립트↓

import re

data = """

park 800905-1049118

kim  700905-1059119

"""

pat = re.compile("(\d{6})[-]\d{7}")

print(pat.sub("\g<1>-\*\*\*\*\*\*\*", data))

쓰인 수식들을 분석한다.

\d= 모든 문자

{6})= 앞의 여섯자리

[-]= 가운데에 –가 있다.

{7}= 그(-) 뒤의 숫자 7자리

들을 pat이라고 정의한다. 그리고 그 par들을 data들의 값과 매치하여 앞의 \g<1> 은 그루핑 하여 남기고 뒤의 7자리는 \*표시로 가린다. 이를 프린트 한다. 라는 뜻이다.

정규식(comprehension)에 메타 문자를 사용하면 특별한 의미를 갖게 된다.

메타문자 설명: (<https://wikidocs.net/4308>) <-솔직히 정규식 자체는 이해하였으나 메타문자를 이해하지 못했다. 외우지 못해서 아는게 없다는 것이 맞는듯... ㅋㅋ.

**두번째 예제. (정규식만 사용함)**

정규표현식을 쓰지 않은 스크립트↓

size = 10

arr = [0] \* size

for i in range(len(size)):

    arr[i] = i \* 2

정규표현식을 사용한 스크립트↓

size = 10

arr = [i \* 2 for i in range(size)]

*+) 추가 예제 1*

word = '가나다'

print([c \* 2 for c in word])

['가가', '나나', '다다']

문자 ‘가나다’가 word 리스트 안에 들어있고, 그 가, 나, 다 문자들을 순서대로 돌린다. 그리고 x2를 해주니 가가 두개가 되고, 나, 다 또한 두개가 된다. 중간에 추가된 c는 for뒤에 오는 임시 변수다.

*+) 추가 예제 2*

tuple\_boy = tuple(n for n in range(1, 10))

print(tuple\_boy)

(1, 2, 3, 4, 5, 6, 7, 8, 9)

튜플로 정규식을 만들고 싶다면 앞에 tuple이라는 것을 명시 해줘야만 한다.

**1. 람다 (lambda)**

함수를 한 줄로 줄여주는 형식. (정규식은 문법, 람다는 형식.)

참고([링크1](https://wikidocs.net/64), [링크2](https://velog.io/@euisuk-chung/%ED%8C%8C%EC%9D%B4%EC%8D%AC-%EC%8B%9C%EA%B0%81%ED%99%94-%EB%A7%88%EC%8A%A4%ED%84%B0%ED%95%98%EA%B8%B0-%EB%9E%8C%EB%8B%A4Lambda-%ED%95%A8%EC%88%98))

**lambda 매개변수 : 표현식**

람다를 쓰지 않은 스크립트↓

def hap(x, y):

    return x + y

hap(10, 20)

람다를 사용한 스크립트↓

(lambda x,y: x + y)(10, 20)

*+)추가 예제1*

filter(lambda x: x < 5, range(10))

괄호의 위치가 안과 밖으로 다르다. 사용한 함수가 다르니까 당연함. 이건 5보다 작은 x를 range(10)에서 찾아다 바치라는 뜻이다. (≒ x<5가 참이면 리스트에 넣어라.)

*+)추가 예제2*

mylist = sorted(mylist, key=lambda x: len(x))

print(mylist)

mylist = ['applpie', 'bananaice', 'cherry']

mylist를 길이 순으로 정렬하고자 sorted() 함수와 람다 함수를 사용함. 그러면 'cherry', 'applpie', 'cherry' 순으로 정렬된다.

**1. 파이썬 추가 설명(Python)**

쓰지 않고 보고만 있으니 계속 졸기만 해서 쓰면서 공부하기로 결정!

+(파이썬 내장함수: <https://docs.python.org/3/library/functions.html> )

**1. 조건식= 식 중에서 평가 또는 치환한 결과 참 또는 거짓의 결과로 귀결되는 식을 총칭.**

예) -,+,=,>,< 등등 값이 참과 거짓으로 판별 될 수 있는 것들.

+ 객체나 값 자체를 조건식으로 활용할 수 있다. 다시 말하면 논리 연산자를 사용하지 않고, if문에 객체나 값 자신을 넣는 것이 가능하다. (파이썬만)

+ 파이썬에서는 모든 객체 자신을 조건식처럼 평가할 수 있는데, 자료구조의 경우는 공집합이 아닌 자료구조는 모두 True, 공집합, 다시 말해 원소가 전혀 없는 자료구조는 False로 평가된다. 그외 정수, 실수, 복소수 등에서 0은(0, 0.0, 0 + 0j) False, 그 외에는 True가 된다.

+조건식에는 if문이 있다. For문이나 while문은 반복문. 이것도 이제 처음 알았음.

**2. 문자열 (string)과 리스트(list)는 다른 녀석이었다!!!** 이걸 이제서야 알다니... tuple이 str과 다른놈이라는 것은 알고있었지만. str, tuple, set을 따로따로 해놓은 이유는 잘 모르겠음. 미묘하게 활용법이나 특성이 다르긴 하다. 중복허용여부나 순서의 유무 등등.



**3. 클래스** = 전사, 마법사, 궁수 등/ 속성= 힘, 민첩, 지력, 운, 물리 공격력, 마법 공격력/ 메서드= 찌르기, 파이어볼, 활 쏘기 등의 스킬? 이라고 비유하면 될 듯 하다. 내가 겜 충이라 이 비유가 제일 잘 와 닿는다. [인스턴스=객체 ≒메서드]??

- **\_\_init\_\_** =클래스에서 제일 처음 들어가야 하는 메서드(초기화) initialize의 준말. (설명: [링크1](https://writingstudio.tistory.com/entry/%ED%8C%8C%EC%9D%B4%EC%8D%ACPython-%ED%81%B4%EB%9E%98%EC%8A%A4class-%EC%95%88-def-initself-%EC%99%80-self-%EB%93%B1%EC%9D%84-%EC%A0%9C%EB%8C%80%EB%A1%9C-%EC%9D%B4%ED%95%B4%ED%95%98%EA%B8%B0))

+) ui에서 계속 self 인수를 넣어줬던 건 클래스를 사용했기 때문인 듯... 이걸 이제서야 알다니 아니 이제라도 알아서 다행이다. self라는 이름의 인수가 아니어도 잘 작동함. 하지만 헷갈리니 도전하지 말자. ㅋ...

self.init\_ui()

요로코롬 메서드 앞에 붙는self.(온점 포함) 를 다른 것으로 바꿔도 된다는 소리. 맨 앞에 오기만 하면 된다. 그 자신을 가장 먼저 전달해야하기 때문.

**-\_\_repr\_\_ =**메서드(프린팅) '문자열'을 'return' 한다.

+) **객체와 인스턴스의 차이**

- 클래스에 의해서 만들어진 객체를 인스턴스라고도 한다.

- a를 단독으로 지칭할 때: 'a는 객체'

- 클래스와 연관지어서 지칭할때: 'a는 a를 포함한 클래스의 인스턴스.

+) 클래스에선 \_\_init\_\_를 두 개 이상 쓰지 않기. 충돌이 일어날 수도 있음.

4. **Try: (엔터 탭) except:** 예외처리. 저 사이에 쓰인 구문이라면 오류가 뜨더라도 무시한다.

**5. 파이썬의 지수표현**

a = 4.24E10 / a = 4.24e-10

여기서 4.24E10은 4.24\*1010, 4.24e-10은 4.24\*10-10을 의미한다.(e와 E 둘 중 어느 것을 사용해도 됨.)

1. f'{"hi":<10}' # 왼쪽 정렬 => 'hi '

2. f'{"hi":>10}' # 오른쪽 정렬 => ' hi'

3. f'{"hi":^10}' # 가운데 정렬 => ' hi '

4. f'{"hi":=^10}' # 가운데 정렬하고 '=' 문자로 공백 채우기 => '====hi===='

5. f'{"hi":!<10}' # 왼쪽 정렬하고 '!' 문자로 공백 채우기 => 'hi!!!!!!!!'