Tuple()[/list[] 도 가능] 로 입력 이후, 대괄호 안에 숫자를 쓰면

e.g. a = (1,2,3)

print(a[0])

➔ "a list 안의 [몇 번째] 숫자를 실행시켜"

Out: 1

a = (1,2,3,4); print(type(a)); print(a[1])

out: <class: tuple>

2

c = {'a' : 'love', 'b' : 'banana'}; print(type(c)); print(c['b'])

(페어 셋 (pair set)에서 앞부분 'a', 'b', 를 index의 수단으로 쓴다.)

out: <class: dict>

banana

float(함수) -> 소수점 있는 수로 전환

float(1)

out: 1.0

int(함수) -> 정수로 전환

float(2.3)

out: 2

S='abcdef'가 있을 때, 0번째는 제일 첫 번째이므로 a인 것이며, -1번째는 제일 마지막 번째이므로 f가 되는 것이다. 그래서 첫 번째를 지칭할 때는 0번째, 마지막 문자는 -1번째라고 지칭하면 되는 것이다.

그리고 print(s[1:3])이라고 하면 첫 번째에서 3번째 직전인 즉 2번째까지만 들고 오라는 듯이다. 그래서 첫 번째와 두 번째가 가져오게 된다. [:3]은 처음에서 세 번째 직전까지 오는 것이다. 그래서 S='abcdef'이 있을 때는 [:3]이라고 하면 abc까지인 것이다. 그냥 [:]만 찍으면 전부 다를 들고 오라는 뜻이 된다. S[1:]은 1부터 마지막까지를 실행하라는 뜻이 되는 것이다.

- Len 이라는 함수는 variable 내에 있는 정보의 개수 (정보의 길이 length)를 준다고 생각하면 된다.

result(함수) -> 제일 첫번째 나오는 index 찾기 (같은 index가 2개 있으면 앞에 있는 거 사용!)

s = 'love you'

result=s. find("you")

result

5

문법 구조

a = [1,2,3,4]

for i in a:

➔ in에 뒤에 있는 것을 하나씩 돌려서 한번 한번 할때마다 i가 받아서 ~(for 밑의 문장)를 하라

print(i)

➔ "a 속에 있는 것을 하나하나 씩 불러서 i에다가 할당하고 계속 돌려라!"

i <- a의 1 할당, 그 다음에 2, 그 다음에 3, 그 다음에 4

a = [1, 2, 3, 4]

for i in range(4):

➔ range함수: 뒤의 숫자가 나오면 list를 만들어준다 / index 형성

e.g range(4) 0~3까지 list를 만들어준다

i가 맨 처음에 0을 받는다/ a[i] a의 i번째 를 루프

```
    print(a[i])
```

```
a = [1, 2, 3, 4]
for i in range(4):
    print(i)
```

➔ 0~3의 함수값이 제출됨

```
a = [1, 2, 3, 4]
for i in range(len(a)):
    print(a[i])
```

```
a = ['red', 'green', 'blue', 'purple']
print(a[0])
print(a[1])
print(a[2])
print(a[3])
```

➔ For loop 안하고 모두 출력하는 방법

근데 안의 변수가 너무 많음면?

For loop 이용!

```
a = ['red', 'green', 'blue','purple']
for s in a:
    print(s)
```

➔ s 의 변수는 지속적으로 바뀐다 loop로

```
a = ['red', 'green', 'blue','purple']
for s in range(4):
    print(a[s])
```

```
a = ['red', 'green', 'blue','purple']
for s in range(len(a)): a의 길이만큼을~
    print(a[s])
```

```
a = ['red', 'green', 'blue', 'purple']    str
b = [0.2, 0.3, 0.1, 0.4]        num
```

```
a = ['red', 'green', 'blue','purple']
b = [0.2, 0.3, 0.1, 0.4]
```

for i,s in enumerate(a): / enumerate(함수) => 번호를 (추가적으로) 매긴다

a의 list도 되지만 그걸 번호로도 받는다

e.g. i – 0 / s-red / 앞이 번호, 뒤가 element

```
a = ['red', 'green', 'blue','purple']
b = [0.2, 0.3, 0.1, 0.4]
```

```
for i,s in enumerate(a):
    print(a[i]) ~a의 i번째를 print하라
```

```
a = ['red', 'green', 'blue','purple']

b = [0.2, 0.3, 0.1, 0.4]


for i,s in enumerate(a): "a list를 뿌려주되/ 1번째는 index값, 2번째 각각의 element

                                1번째 i-0, s-red

                                2번째 i-1, s=green

                                Emd emd

    print("{}: {}%".format(s, b[i]*100))
```

➔ 이런식의 format을 적고 싶을 때(앞의 점찍기 전의 식), format 다음의 조건으로 바뀌어라

```
a = ['red', 'green', 'blue','purple']

b = [0.2, 0.3, 0.1, 0.4]


for s,i in zip(a,b): / zip -> 두개를 합치는 함수 / pair 로 4개짜리로 바뀜

    print("{}: {}%".format(s, i*100))
```

결과값은 똑같

If 관련

```
a = 0

if a == 0:

    print("yay!") / 진짜 equal sign? == =등호 2개, 등호 1개 변수로 assign한다라는 의미
```

```
a = 0
if a != 0: / a는 0이 아니라면~
    print("yay!")
```

근데 a 는 0으로 assign되어있으므로 print 안됨

그 밑에다가 print 많이 해도 x

```
a = 0
if a >= 0: / a가 0보다 크거나 같으면~
    print("yay!")
    print("let's go")

a = 0
if a != 0:
    print("yay!")
    print("let's go")
else: / 만약 그렇지 않는다면 밑 내용을 출력하라
    print("no")


for i in range(1,3):
    print(i) / 1부터 3 직전까지 간다 -> 1,2 값

for i in range(1,3): / 1부터 [크게는 총 2번 돈다]
    for j in range(3,5): / 2번 for loop 되는 경우 [작게도 2번 돈다, 3,4 로]
        print(i*j) -> 총 4번 실시
```

1*3, 1*4, 2*3, 2*4 이렇게 loop

```
for i in range(1,3):
    print(i) -> 총 2번 (1,2)
    for j in range(3,5):
        print(i*j) -> 총 4번 (3,4,6,8)
```

```
for i in range(1,3):
    for j in range(3,5):
        if j >=4: 3일때는 기각, 4일때만 된다
        print(i*j) -> 총 2번 (1*4, 2*4)
```

```
for i in range(1,3):
    if i >=3:
    for j in range(3,5):
            print(i*j)
```

➔ Error

```
for i in range(1,3):
    if i >=3:
        for j in range(3,5):
            print(i*j)
```

➔ 답 없음