Python

함수명(매개변수)
print라는 독립적인 함수
ex) print(“Hi~”)
format
“문자열” .format()
format값은 숫자이든지 문자이든지 구분 하지 않음. format안에 있는 값을 대입함.

인덱스
str = “apple”
01234 ← Index값

{} {}안 숫자 몇 번째 인덱스인지 나타냄
ex) "I eat {2} apples.".format("one","two","three")
'I eat three apples.’

변수값은 format안에 써줘야함.
포맷 형식
→ {인덱스[:][채울문자][정렬방식][전체자리수]}

f형식 사용(포맷 – python 3.6버전 이상 사용 가능)
name = ’홍길동’
age = 30
print (f'나의 이름은 {name}입니다. 나이는 {age}입니다.')
→ 나의 이름은 홍길동입니다. 나이는 30입니다 출력됨.
본래 포맷 형식
’나의 이름은 {0}입니다. 나이는 {1}입니다.' .format(“홍길동”,30)
’나의 이름은 {0}입니다. 나이는 {1}입니다.' .format(name,age)
’나의 이름은 {name1}입니다. 나이는 {age1}입니다.' .format(name1=”홍길동”, age1=30)

문자 개수 세기 count
len() : 문자열의 갯수(길이)를 구하는 함수 ->정수형태에는 사용 불가
a=hobby에서 b가 몇 개인지 나타냄 → print(a.count('b')) a
count는 문자의 개수가 몇 개인지 나타냄
문자열.count()형태로 사용
포함하는 문자의 개수를 구해줌.

문자의 위치를 알려주는 것
find
a = "Python is the best choice"

print(a.find('b'))
b가 14번째 위치하기 때문에 14로 표시됨
a = "Python is the best choice"
print(a.find('i'))
7번째 표시함
없는 글자는 –1로 표시됨

a = "python is the best choice"
a.find('best')
문자열 중 문자 b가 처음으로 나온 위치를 반환한다. 만약 찾는 문자나 문자열이 존재하지 않는다면 -1을 반환한다.

※ 파이썬은 숫자를 0부터 세기 때문에 b의 위치는 15가 아닌 14가 된다.

a = "Python is the best choice"
print(a.index('i'))
7이 나옴
index에서는 없는 문자 찾을 때는 에러가 뜸.

문자열 삽입(join)
",".join('abcd')
'a,b,c,d'
각각의 문자 사이마다 ,표시를 함
upper(소문자 → 대문자)와 lower(대문자 → 소문자)
a='hi'
a.upper()
'HI'

a='hi'
a.lower()
'hi'

왼쪽 공백 지우기 & 오른쪽 공백지우기, 양쪽공백 지우기 (lstrip&rstrip, strip)
a="abc@naver.com"
b="abc@naver.com "
공백도 문자로 취급함

a=" abc@naver.com"
b="abc@naver.com "
print(a.lstrip())
print(b.rstrip())

abc@naver.com으로 출력이 됨
그냥 print를 하면 공백까지 출력되어 나옴

양쪽 공백을 지울때는 strip하면 모두 없어짐.

문자열 변경(replace)
a=" abc@naver.com"
print(a.replace("abc", "123"))
→ 123@naver.com으로 변경이 됨

문자열 나누기(split)
공백을 기준으로 문자열을 나눔
b="Life is too short"
print(b.split( ))
'Life', 'is', 'too', 'short'으로 나뉨

■ list (※사용하는 빈도수가 많기 때문에 숙지를 해야함)
1에서 10까지의 숫자값이 필요
number = [1,2,3,4,5,6,7,8,9,10]
odd = [1,3,5,7,9]
even = [2,4,6,8,10]

print("number=", number)
print("홀수=", odd)
print("짝수=", even)

number= [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
홀수= [1, 3, 5, 7, 9]
짝수= [2, 4, 6, 8, 10]
출력이 됨

여러개의 값(요소)를 넣은 것 []로 감싸놓은 것을 리스트라고 함.
집합(딕셔너리)은 {} 큐플(자료형)은 ()표시함.
요소는 문자 숫자 섞어서 써도 됨
리스트를 찾을 때는 요소값으로 찾음

a=list() = []

del : 요소값에 있는 것을 삭제할 때 사용
append : 요소값 추가(추가 될 때는 마지막에 추가)
리스트 정렬(Sort)

a=[1,4,3,2]
a.sort()하면 출력이 [1,2,3,4]가 됨
Sort는 오름차순
Reverse는 내림차순(뒤집어줌) sort후 reverse해야 내림차순으로 됨
※ 그냥 reverse 하면 순서만 역순으로 뒤집어짐

list에서는 find 명령어 적용이 안됨.

insert 명령어
a=[1,2,3]
a.insert(0,5)
[5, 1, 2, 3, 4]
a.insert(3,6)
[5, 1, 2, 6, 3, 4]

remove 명령어
값을 가지고 지움
a=[1,1,2,2,3,3,4,5]
[1, 1, 2, 2, 3, 3, 4, 5]
a.remove(1)
[1, 2, 2, 3, 3, 4, 5]

remove(x)는 리스트에서 첫 번째로 나오는 x를 삭제하는 함수이다.

```
>>> a = [1, 2, 3, 1, 2, 3]
>>> a.remove(3)
>>> a
[1, 2, 1, 2, 3]
```
a가 3이라는 값을 2개 가지고 있을 경우 첫 번째 3만 제거되는 것을 알 수 있다.

```
>>> a.remove(3)
>>> a
[1, 2, 1, 2]
```
remove(3)을 한 번 더 실행하면 다시 3이 삭제된다.

<del과 remove의 차이점>
del은 특정한 값(인덱스)을 지울 때
remove는 왼쪽의 값을 기준으로 값 자체를 지울 때 사용

● 자료구조(Data Structure)

자료 구조 중 스택, 리스트, 큐가 있음

스택(stack)이란 쌓아 올린다는 것을 의미한다.
따라서 스택 자료구조라는 것은 책을 쌓는 것처럼 차곡차곡 쌓아 올린 형태의 자료구조를 말한다.

데이터베이스는 내용을 가지고 값을 찾음
스택은 후입선출(LIFO : Last In First Out)형태를 가짐.

큐(queue)는 컴퓨터의 기본적인 자료 구조의 한가지로, 먼저 집어 넣은 데이터가 먼저 나오는 선입선출(FIFO(First In First Out))구조로 저장하는 형식을 말한다.

■ 리스트 요소를 끄집어 내기
pop 리스트의 맨마지막 요소를 돌려주고 그 요소는 삭제
```
a=[1,2,3]
a.pop()
3
print(a)
[1, 2]
```

del은 그냥 삭제 pop은 뽑아내고 없앰

count
```
a=[1,2,3,1]
print(a.count(1))
2
a=['a','p','p','p','l','e']
print(a.count('p'))
3
```

append : 1개 추가(여러개 동시에는 추가 불가)
```
a=[1,2,3]
b=[4,5,6]
c=a+b
[1, 2, 3, 4, 5, 6]
print(c.append(7))
[1, 2, 3, 4, 5, 6, 7]
print(c.extend([8,9,10]))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

extend : 리스트 형태로만 들어감(여러개 추가 가능)

리스트+리스트와 같음

■튜플 자료형
a=list()
b=tuple()
type(a)
<class 'list'>
type(b)
<class 'tuple'>
a
[]
b
()
a=[1,2,3]
b=(1,2,3)
a
[1, 2, 3]
b
(1, 2, 3)

b[0]=4 : 에러 걸림
튜플은 값을 입력하면 수정이 불가.
튜플(tuple)은 몇 가지 점을 제외하곤 리스트와 거의 비슷하며 리스트와 다른 점은 다음과 같다.

리스트는 [ ]으로 둘러싸지만 튜플은 ( )으로 둘러싼다.
리스트는 그 값의 생성, 삭제, 수정이 가능하지만 튜플은 그 값을 바꿀 수 없다.

요소가 하나일때는 tuple은 ,표시를 해야함
a=("apple",)

※ list vs tuple
: 수정 불가(list에서 사용이 가능한 delete, remove, insert, append 불가), 1개짜리 튜플을 만들때는 ,를써야함

참조하는 데이터는 tuple로 만들고 변동이 되는 데이터는 list로 만듬.

■ 딕셔너리
사용하는 대부분의 언어도 이러한 대응 관계를 나타내는 자료형을 갖고 있는데, 이를 연관 배열(Associative array) 또는 해시(Hash)라고 한다.

파이썬에서는 이러한 자료형을 딕셔너리(Dictionary)라고 함.

dic = {'name':'pey', 'phone':'0119993323', 'birth': '1118'}
위에서 Key는 각각 'name', 'phone', 'birth'이고, 각각의 Key에 해당하는 Value는 'pey', '0119993323', '1118'이 된다.

<딕셔너리의 특징>
딕셔너리는 {}로 묶음 리스트는 []로 묶고 튜플은 ()로 묶음

딕셔너리는 요소의 키를 적어주어야 인덱싱이 됨
dic['name']
'pey'
dic['phone']
'0119993323'
dic['birth']
'1118'

key중에 하나를 적으면 key에 해당되는 value를 나타냄
key와 value가 한쌍으로 이루어지고 key를 이용해서 값을 찾음.

<만들어 놓은 딕셔너리에 추가>
dic = {'a': [1,2,3], 'b':(4,5,6), 'c' : 'apple'}
dic['a']
[1, 2, 3]
dic['b']
(4, 5, 6)
dic['c']
'apple'
dic['d'] = 123
dic
{'a': [1, 2, 3], 'b': (4, 5, 6), 'c': 'apple', 'd': 123}

<딕셔너리 값 삭제>
del [key] 하면 딕셔너리가 삭제됨.

```
name_list = ["김연아", "류현진", "박지성", "귀도"]
specialty = ["피겨스케이팅", "야구", "축구", "파이썬"]
dic = {"김연아":"피겨스케이팅", "류현진":"야구", "박지성":"축구", "귀도":"파이썬"}

print(name_list[0], "=", specialty[0])
print(name_list[1], "=", specialty[1])
print(name_list[2], "=", specialty[2])
```

print(name_list[3], "=", specialty[3])
※ 리스트로 하면 번거롭게 해야함.

<딕셔너리로 찾는 과정>
name = "김연아"
idx = name_list.index(name)
spe = specialty[idx]
print(f"{name}의 특기는 {spe}입니다.")
-> 김연아의 특기는 피겨스케이팅입니다.
name = input("찾는 이름은?")
idx = name_list.index(name)
spe = specialty[idx]
print(f"{name}의 특기는 {spe}입니다.")

▶ 찾는이름은? 귀도
▶ 귀도의 특기는 파이썬입니다. 라고 표시됨

dic = {"김연아":"피겨스케이팅", "류현진":"야구", "박지성":"축구", "귀도":"파이썬"}
dic["김연아"]
'피겨스케이팅'
key와 값(value)이 있으면 value값이 나옴.

name = input("찾는 이름은? ")
spe = dic[name]
print(f"{name}의 특기는 {spe}입니다.")
key를 입력하면 key에 해당되는 value가 들어감.
index를 찾을 필요 없이 key에 해당되는 value를 찾아서 나타냄.

● Virtual Box로 우분투 설치하기
Cent OS vs Ubuntu
Cent OS(RedHat 계열) : 서버용
Ubuntu : 프로그램용(요즘은 서버용으로도 쓰기도 함)