########20191210

Python IDLE사용

2장. 프로그래밍의 기초, 자료형

I. 숫자형(numeric)

II. 문자열(string) 자료형

=> 문자열은 1차열 배열이다.

III. 리스트(list) 자료형

=> Python에서 리스트는 숫자나 문자를 모아 놓은 것이다. 수정이 가능하다.

=> R에서 리스트는 2차원 배열이다.

IV. 튜플(tuple) 자료형

=> 튜플은 리스트하고 형태 같은데 수정이 불가하다.

V. 딕셔너리(dictionary) 자료형

=> 딕셔너리는 사전으로 생각하면 된다. key value

VI. 집합(set) 자료형

=> 집합은 특징은 중복이 되면 안되고 순서가 없다. 합집합 교집합 차집합 구할 경우

VII. 불(bool) 자료형

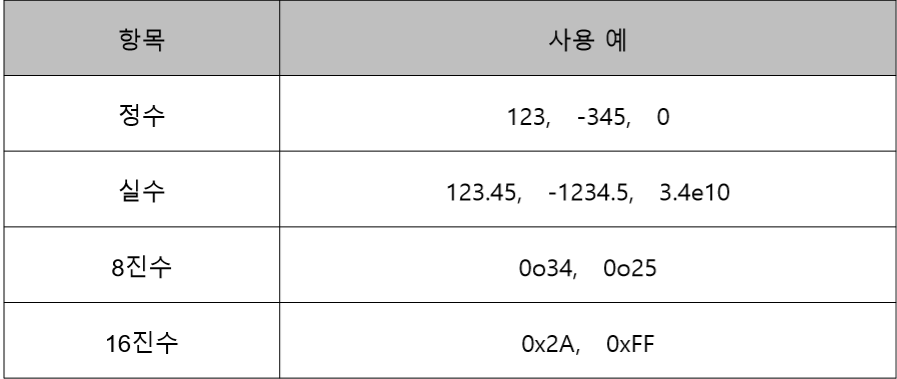
=> 불 while문에서 자주 사용 참이나 거짓 이나 이런 것 등

VIII.자료형의 값을 저장하는 공간, 변수

=>자료형의 값을 저장하는 공간, 변수 변수를 대입하게 되면 메모리에 저장하게 된다.

## I. 숫자형

\*\*숫자형(Numeric)이란 숫자 형태로 이루어진 자료형



=>정수형(Integer)이란 정수를 뜻하는 자료형을 말함

>>> a = 123

>>> a = -178

>>> a = 0

=>실수형(Floating-point)은 소수점이 포함된 숫자를 말함

>>> a = 1.2

>>> a = -3.45

실수형의 소수점 표현 방식

>>> a = 4.24E10

>>> a = 4.24e-10#뛰여쓰기 없어야 함 아님 오류

컴퓨터식 지수 표현 방식

e와 E 둘 중 어느 것을 사용해도 무방함

3.4e10->지수를 표현한다.

10의 10승등으로 이해하기 어려워서 e를 사용했다.

=>8진수와 16진수

>>> a = 0o177

8진수(Octal)를 만들기 위해서는 숫자가 0o 또는 0O로 시작함

>>> 0o34 #8진수 34

>>> b = 0x5ff

>>> b = 0xABC

>>> 0x2A #16진수

16진수(Hexadecimal)를 만들기 위해서는 0x로 시작함

=> 숫자형을 활용하기 위한 연산자

==> 사칙연산(+, -, \*, /)

>>> a = 3

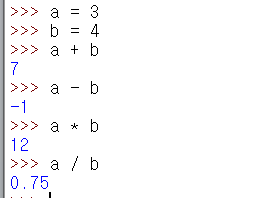
>>> b = 4

>>> a + b

>>> a - b

>>> a \* b

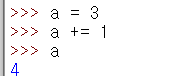
>>> a / b



==> 대입연산(+=, -=, \*=, /=)

>>> a += 1

print(a)

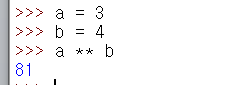


=> x의 y제곱을 나타내는 연산자

>>> a = 3

>>> b = 4

>>> a \*\* b



=>나눗셈 후 나머지를 반환하는 % 연산자

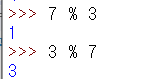
아래 두 개는 페이지 처리 시 사용한다.

%는 나머지

//은 몫

>>> 7 % 3 #1

>>> 3 % 7 #3

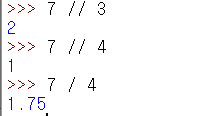


나눗셈 후 몫을 반환하는 // 연산자

>>> 7 // 3

>>> 7 // 4

>>> 7/4 #1.75



## II. 문자열 자료형

문자열(String)이란 문자, 단어 등으로 구성된 문자들의 집합을 의미

=>문자열 자료형 만드는 4가지 방법

1. 큰따옴표(“)로 양쪽 둘러싸기

>>> "Hello World"

2. 작은따옴표(‘)로 양쪽 둘러싸기

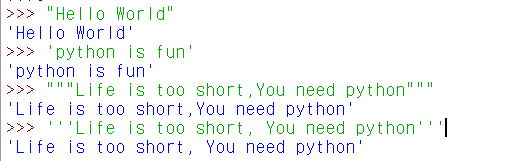
>>> 'python is fun'

3. 큰따옴표 3개를 연속(“””)으로 써서 양쪽 둘러싸기

>>> """Life is too short,You need python"""

4. 작은따옴표 3개를 연속(‘’’)으로 써서 양쪽 둘러싸기

>>> '''Life is too short, You need python'''



=>문자열에 따옴표 포함시키기

1. 문자열에 작은따옴표(‘) 포함시키기

>>> food = "Python's favorite food is perl"

>>> food



큰따옴표(“)가 아닌 작은따옴표(‘)로 문자열을 둘러싼 후 다시 실행함

‘Python’이 문자열로 인식되어 구문 오류(Syntax Error)가 발생함

>>> food = 'Python's favorite food is perl'#오류



2. 문자열에 큰따옴표(“) 포함시키기

>>> say = '"Python is very easy." he says.'

>>> say



3. 백슬래시(\)를 이용해서 작은따옴표(‘)와 큰따옴표(“)를 문자열에 포함시키기 

백슬래시(\)를 작은따옴표(‘)나 큰따옴표(“) 앞에 삽입하면 백슬래시(\) 뒤의 작은따옴표(‘)나 큰따옴표(“)는 문자열을 둘러싸는 기호의 의미가 아니라 (‘), (“) 그 자체를 의미함

>>> food = 'Python\'s favorite food is perl'

>>> say = "\"Python is very easy.\" he says."



=> 여러 줄인 문자열을 변수에 대입하고 싶을 때

1. 줄을 바꾸는 이스케이프 코드 ‘\n’ 삽입하기

>>> multiline = "Life is too short\nYou need python"

>>> multiline



[이스케이프 코드]

프로그래밍할 때 사용할 수 있도록 미리 정의해 둔 ‘문자 조합‘이며, 출력물을 보기 좋게 정렬해 줌

|  |  |
| --- | --- |
| 코드 | 설명 |
| **\n** | **문자열 안에서 줄을 바꿀 때 사용** |
| **\t** | **문자열 사이에 탭 간격을 줄 때 사용** |
| **\\** | **문자 \를 그대로 표현할 때 사용** |
| **\’** | **작은따옴표(‘)를 그대로 표현할 때 사용** |
| **\”** | **큰따옴표(“)를 그대로 표현할 때 사용** |
| \r | 캐리지 리턴(줄 바꿈 문자,현재 커서를 다음 줄로 이동) |
| \a | 벨 소리(출력할 때 pc 스피커에서 '삑' 소리가 난다) |
| \b | 백스페이스 |
| \000 | 널 문자 |

2. 연속된 작은따옴표 3개(‘’’) 또는 큰따옴표 3개(“””) 이용

>>> multiline = """

Life is too short

You need python

"""

multiline



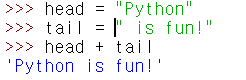
=> 문자열 연산하기

1. 문자열 더해서 연결하기

>>> head = "Python"

>>> tail = " is fun!"

>>> head + tail



2. 문자열 곱하기

>>> a = "python"

>>> a\*2



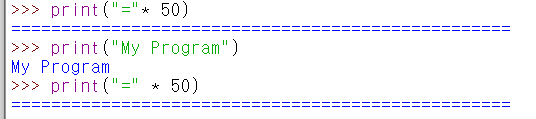
3. 문자열 곱하기 응용

#multistring.py

>>> print("="\* 50)

>>> print("My Program")

>>> print("=" \* 50)



4. 문자열 길이 구하기

>>> a = "Life is too short"

>>> len(a)



=> 문자열 인덱싱과 슬라이싱

인덱싱(Indexing)이란 무엇인가를 가리킨다는 의미이고, 슬라이싱(Slicing)은 무엇인가를 잘라낸다는 의미임

==> 문자열 인덱싱이란?

\*\*\*파이썬은 0부터 숫자를 셈

>>> a = "Life is too short, You need Python"

>>> a[3]

>>> a[0]

>>> a[12]

\*\*\* a[-1]은 문자열을 뒤에서부터 읽기 위해서 마이너스(-)를 붙임

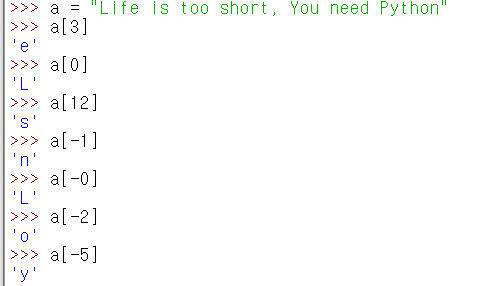
>>> a[-1]

\*\*\* a[-0]은 a[0]과 똑같은 값을 보여줌

>>> a[-0] #0과 0은 같다.

>>> a[-2]

>>> a[-5]

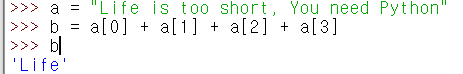


==>문자열 슬라이싱이란?

>>> a = "Life is too short, You need Python"

>>> b = a[0] + a[1] + a[2] + a[3]

>>> b



>>> a[0:4]

a[:]

a[시작 번호:끝 번호]에서 시작 번호와 끝 번호를 생략하면 처음부터 끝까지 뽑아냄

0 <= a < 4

==> 문자열을 슬라이싱하는 방법

>>> a[0:5]

>>> a[0:4]

>>> a[0:5]

>>> a[0:2]

>>> a[5:7]

>>> a[12:17]

>>> a[19:]

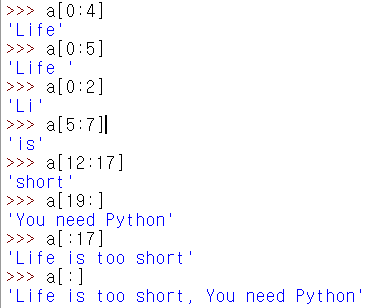
a[시작 번호:끝 번호]에서 끝 번호 부분을 생략하면 시작 번호부터 그 문자열의 끝까지 뽑아냄

>>> a[:17]

a[시작 번호:끝 번호]에서 시작 번호 부분을 생략하면 처음부터 끝 번호까지 뽑아냄

>>> a[:]

a[시작 번호:끝 번호]에서 시작 번호와 끝 번호를 생략하면 처음부터 끝까지 뽑아냄



a[19:-7]

a[-7]은 포함하지 않음



=> 슬라이싱으로 문자열 나누기

>>> a = "20010331Rainy"

>>> date = a[:8]

>>> weather = a[8:]

>>> date

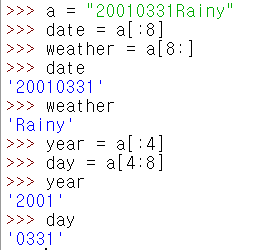
>>> weather

>>> year = a[:4]

>>> day = a[4:8]

>>> year

>>> day

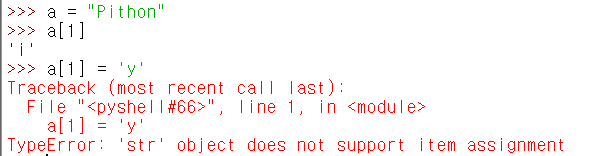


=> 문자열 바꾸기

>>> a = "Pithon"

>>> a[1]

>>> a[1] = 'y'

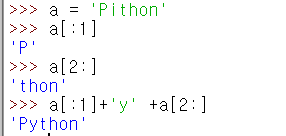


>>> a = 'Pithon'

>>> a[:1]

>>> a[2:]

>>> a[:1]+'y' +a[2:]



=> 문자열 포매팅

==> 문자열 포매팅 따라하기

1. 숫자 바로 대입

>>> "I eat %d apples." % 3



2. 문자열 바로 대입

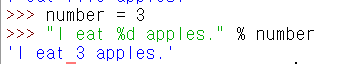
>>> "I eat %s apples." % "five"



3. 숫자 값을 나타내는 변수로 대입

>>> number = 3

>>> "I eat %d apples." % number



4. 2개 이상의 값 넣기

변수를 대입한다.

>>> number = 10

>>> day = "three"

>>> "I ate %d apples. so I was sick for %s days." % (number,day)

포맷한다 .대입한다.

숫자 %d



마지막 % 다음 괄호 안에 콤마(,)로 구분하여 각각의 값을 넣어 주면 됨

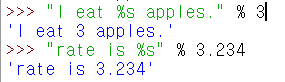
==>문자열 포맷 코드

|  |  |
| --- | --- |
| 코드 | 내용 |
| %s | 문자열(String) |
| %c | 문자 1개(Character) |
| %d | 정수(Integer) |
| %f | 부동 소수(Floating-point) |
| %o | 8진수 |
| %x | 16진수 |
| %% | Literal %(문자 ‘%’ 자체) |

%s 포맷 코드는 어떤 형태의 값이든 변환해서 넣을 수 있음

>>> "I eat %s apples." % 3

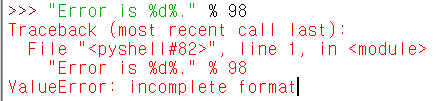
>>> "rate is %s" % 3.234



문자 %s

==> 포맷팅 연산자 %d와 %를 같이 쓸 때는 %%를 씀

>>> "Error is %d%." % 98



문자열 포맷팅 코드인 %d와 %가 문자열 내에 존재하는 경우, %를 나타내려면 반드시 %%로 써야함

>>> "Error is %d%%." % 98



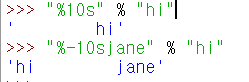
==> 포맷 코드와 숫자 함께 사용하기

1. 정렬과 공백

>>> "%10s" % "hi" #오른쪽

>>> "%-10s" % "hi" #왼쪽

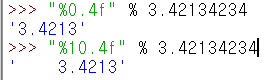
>>> "%-10sjane" % "hi"



2. 소수점 표현하기

>>> "%0.4f" % 3.42134234

>>> "%10.4f" % 3.42134234



=> format 함수를 사용한 포매팅

좀 더 발전된 스타일로 문자열 포맷을 지정할 수 있음

==>숫자 바로 대입하기

>>> "I eat {0} apples".format(3)



==> 문자열 바로 대입하기

>>> "I eat {0} apples".format("five")



==>숫자 값을 가진 변수로 대입하기

>>> number = 3

>>> "I eat {0} apples".format(number)



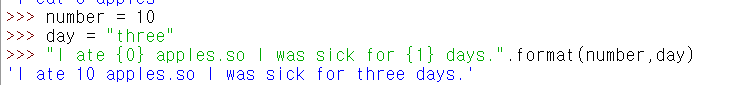
문자열의 {0} 항목이 number 변수 값인 3으로 바뀜

==>2개 이상의 값 넣기

>>> number = 10

>>> day = "three"

>>> "I ate {0} apples.so I was sick for {1} days.".format(number,day)



==>이름으로 넣기

>>> "I ate {number} apples. so I was sick for {day} days.".format(number=10, day=3)

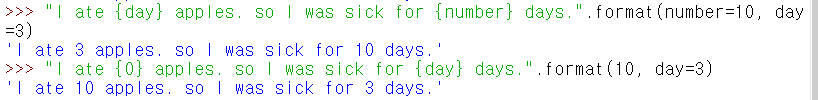


{0}, {1}과 같은 인덱스 항목 대신 더 편리한 {name} 형태를 사용함

==>인덱스와 이름을 혼용해서 넣기

>>> "I ate {day} apples. so I was sick for {number} days.".format(number=10, day=3)

>>> "I ate {0} apples. so I was sick for {day} days.".format(10, day=3)



===>왼쪽 정렬

>>> "{0:<10}".format("hi")

:< 표현식을 사용하면 치환되는 문자열을 왼쪽으로 정렬하고, 문자열의 총 자릿수를 10으로 맞춤

===>오른쪽 정렬

>>> "{0:>10}".format("hi")

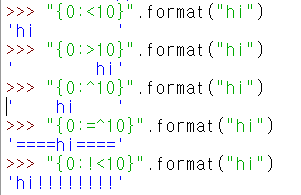
===>가운데 정렬

>>> "{0:^10}".format("hi")

===>공백 채우기

>>> "{0:=^10}".format("hi")

>>> "{0:!<10}".format("hi")

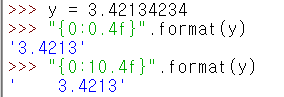


===>소수점 표현하기

>>> y = 3.42134234

>>> "{0:0.4f}".format(y)

>>> "{0:10.4f}".format(y)



===>{ 또는 } 문자 표현하기

>>> "{{ and }}".format()



=>f 문자열 포매팅

파이썬 3.6 버전부터는 f 문자열 포매팅 기능을 사용할 수 있음

==>문자열 앞에 f 접두사를 붙여서 사용

>>> name = '홍길동'

>>> age = 30

>>> f'나의 이름은 {name}입니다.나이는 {age}입니다.'



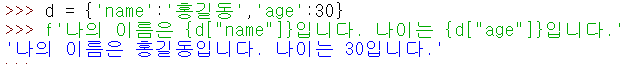
==>표현식 지원



==>딕셔너리 사용

>>> d = {'name':'홍길동','age':30}

>>> f'나의 이름은 {d["name"]}입니다. 나이는 {d["age"]}입니다.'



==> 정렬

>>> f'{"hi":<10}' #왼쪽 정렬

>>> f'{"hi":>10}' #오른쪽 정렬

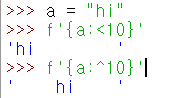
>>> f'{"hi":^10}' #가운데 정렬



>>> a = "hi"

>>> f'{a:<10}'

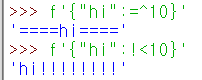
>>> f'{a:^10}'



==>공백 채우기

>>> f'{"hi":=^10}'

>>> f'{"hi":!<10}'

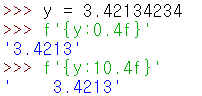


==>소수점 표현하기

>>> y = 3.42134234

>>> f'{y:0.4f}'

>>> f'{y:10.4f}'



==> { 또는 } 문자 표현하기

>>> f'{{ and }}'



=> 문자열 관련 함수

문자열 자료형은 자체적으로 문자열 내장 함수를 가지고 있으며, 그것들을 사용하려면 문자열 변수 이름 뒤에 ‘.’를 붙인 다음에 함수 이름을 써주면 됨

==> 문자 개수 세기(count)

>>> a = "hobby"

>>> a.count('b')

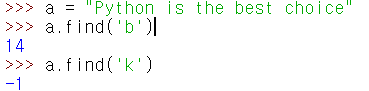


==>위치 알려주기 1(find)

>>> a = "Python is the best choice"

>>> a.find('b')

>>> a.find('k')



find는 -1

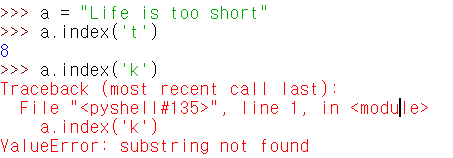
index는 error난다. indexOfbound

==> 위치 알려주기 2(index)

>>> a = "Life is too short"

>>> a.index('t')

>>> a.index('k')



==>문자열 삽입(join)

>>> a = ","

>>> a.join('abcd')



>>> a.join(['a','b','c','d'])



Python은 대소문자 구분한다.

==>소문자를 대문자로 바꾸기(upper)

>>> a = "hi"

>>> a.upper()



==>대문자를 소문자로 바꾸기(lower)

>>> a = "HI"

>>> a.lower()



==> 왼쪽 공백 지우기(lstrip)

>>> a = " hi "

>>> a.lstrip()



==> 오른쪽 공백 지우기(rstrip)

>>> a = " hi "

>>> a.rstrip()



==> 양쪽 공백 지우기(strip)

문자 주위만 없애 준다.

>>> a = " hi "

>>> a.strip()



==>문자열 바꾸기(replace)

>>> a = "Life is too short"

>>> a.replace("Life","Your leg")



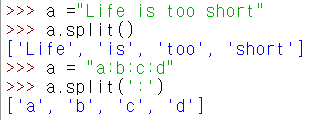
==>문자열 나누기(split)

>>> a ="Life is too short"

>>> a.split()

>>> a= "a:b:c:d"

>>> a.split(":")



## III. 리스트 자료형

리스트(List)란 숫자나 문자 모음으로 이루어진 자료형

>>> odd = [1, 3, 5, 7, 9]

리스트명 = [요소1, 요소2, 요소3, ···]

>>> a =[]

>>> b = [1,2,3]

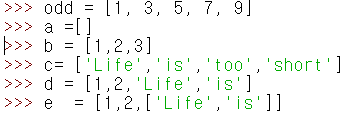
>>> c= ['Life','is','too','short']

>>> d = [1,2,'Life','is']

>>> e = [1,2,['Life','is']]#중첩리스트

리스트가 3개 겹쳐있으면 3중리스트

e처럼 리스트 자체를 요소 값으로 가질 수 있음



=>리스트의 인덱싱과 슬라이싱

==>리스트의 인덱싱

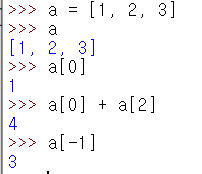
>>> a = [1, 2, 3]

>>> a

>>> a[0]

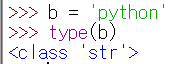
>>> a[0] + a[2]

>>> a[-1]



>>> b = 'python'

>>> type(b)



>>> a = [1,2,3,['a','b','c']]

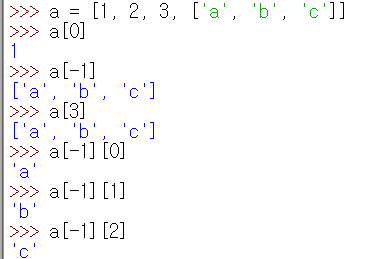
>>> a[0]

>>> a[3]

>>> a[-1][0]

>>> a[-1][1]

>>> a[-1][2]



>>> a= [1,2,['a','b',['Life','is']]]

>>> a[2][2][0]



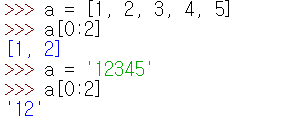
==>리스트의 슬라이싱

>>> a = [1, 2, 3, 4, 5]

>>> a[0:2]

>>> a = '12345'

>>> a[0:2]



>>> a = [1,2,3,4,5]

>>> a[2:]

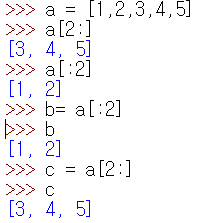
>>> a[:2]

>>> b= a[:2]

>>> b

>>> c = a[2:]

>>> c



==>중첩된 리스트에서 슬라이싱하기

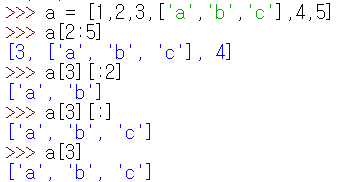
>>> a = [1,2,3,['a','b','c'],4,5]

>>> a[2:5]

>>> a[3][:2]

>>> a[3][:]

>>> a[3]



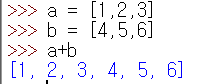
=>리스트 연산하기

1. 리스트 더하기(+)

>>> a = [1,2,3]

>>> b = [4,5,6]

>>> a+b

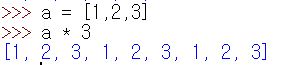


리스트 사이에서 + 기호는 2개의 리스트를 합치는 기능을 함

2. 리스트 반복하기(\*)

>>> a = [1,2,3]

>>> a \* 3



[1, 2, 3] 리스트가 세 번 반복되어 새로운 리스트를 만들어 냄

3. 리스트 길이 구하기

>>> a= [1,2,3]

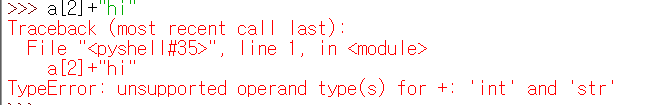
>>> len(a)



==>리스트 연산 오류

>>> a = [1,2,3]

>>> a[2]+"hi"



정수와 문자열은 서로 더할 수 없기 때문에 형(type) 오류가 발생함

>>> str(a[2])+"hi"



str 함수는 정수나 실수를 문자열의 형태로 바꾸어 주는 내장 함수임

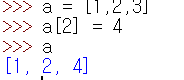
=>리스트의 수정과 삭제

==>리스트에서 값 수정하기

>>> a = [1,2,3]

>>> a[2] = 4

>>> a



각각의 요소이기때문에 가능하다. 하지만 string은 안된다.

==>del 함수를 사용해 리스트 요소 삭제하기

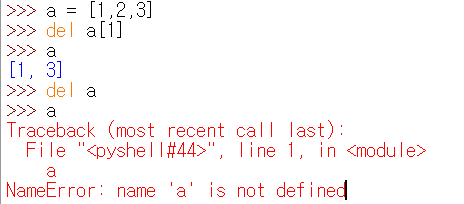
>>> a = [1,2,3]

>>> del a[1]

>>> a

>>> del a

>>> a



del a[x]는 x번째 요소값을 삭제함. del 객체는 변수를 삭제함

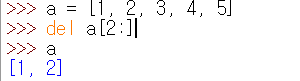
del a 는 메모리에서 데이터가 삭제된다.

del(a)

>>> a = [1, 2, 3, 4, 5]

>>> del a[2:]

>>> a



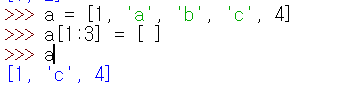
슬라이싱 기법을 사용하여 리스트의 요소 여러 개를 삭제함

del[x:y]는 x번째부터 y번째 요소 사이의 값을 삭제함

>>> a = [1, 'a', 'b', 'c', 4]

>>> a[1:3] = [ ]

>>> a



[ ] 사용해 리스트 요소 삭제함

a[1:3]을 [ ]으로 바꿔 주었기 때문에 [‘a’, ‘b’]가 삭제된 [1, ‘c’, 4]가 됨

id(a)를 메모리에 있는 주소를 의미한다.

########20191211

20191210 정리

\백스럴시 escape code

문자열%%

\\

문자열은 곱한다는 것은 문자열을 반복한다는 것이다.

slicing은 짜르기 0 <= a <4

%f 실수

문자열 포매팅

r이라는 것이있다.

find 못찾을 경우 -1

sort원자료도 정렬된다.

sorted정렬되지만 원자료는 안바꿔진다.

extend는 list형태로 값을 줘야 한다.

리스트자료형

=>리스트 관련 함수

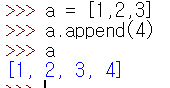
리스트 변수 이름 뒤에 ‘.’를 붙여서 리스트 관련 함수를 사용할 수 있음

==>리스트에 요소추가

>>> a = [1,2,3]

>>> a.append(4)

>>> a

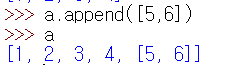


append(x)는 리스트의 맨 마지막에 x를 추가하는 함수임

리스트 안에는 어떤 자료형도 추가할 수 있음

>>> a.append([5,6])

>>> a



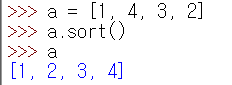
리스트에 다시 리스트를 추가한 결과임

==>리스트 정렬(sort)

>>> a = [1,4,3,2]

>>> a.sort()

>>> a

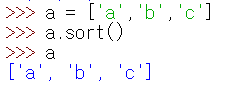


sort 함수는 리스트의 요소를 순서대로 정렬해 줌

>>> a = ['a','c','b']

>>> a.sort()

>>> a

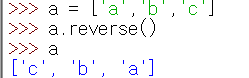


==>리스트 뒤집기(reverse)

>>> a = ['a','b','c']

>>> a.reverse()

>>> a



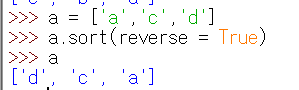
===>desc 정렬하기

방안 1 :

>>> a = ['a','c','d']

>>> a.sort(reverse = True)

>>> a

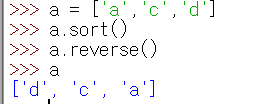


방안 2:

>>> a = ['a','c','d']

>>> a.sort()

>>> a.reverse()



==>위치 반환(index)

>>> a= [1,2,3]

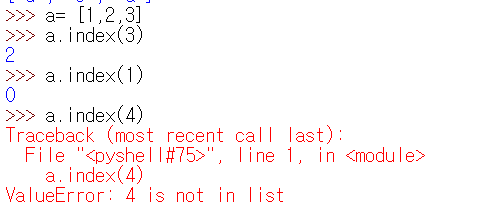
>>> a.index(3)

>>> a.index(1)

>>> a.index(4)

index(x) 함수는 리스트에 x라는 값이 있으면 x의 위치 값을 돌려줌

0이라는 값은 a 리스트에 존재하지 않기 때문에 값 오류가 발생함



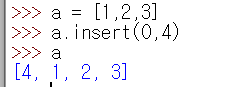
==>리스트에 요소 삽입(insert)

>>> a = [1,2,3]

>>> a.insert(0,4)

>>> a

0번째 자리, 첫 번째 요소(a[0]) 위치에 값 4를 삽입하라는 뜻임



>>> a.insert(3,5)

>>> a



3번째 자리, 네 번째 요소(a[3]) 위치에 값 5를 삽입하라는 뜻임

>>> a.append(6)

>>> a



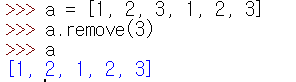
==>리스트에 요소 제거(remove)

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

>>> a = [1, 2, 3, 1, 2, 3]

>>> a.remove(3)

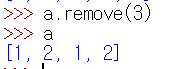
>>> a



a가 3이라는 값을 2개 가지고 있을 경우 첫 번째 3만 제거되는 것을 알 수 있다.

>>> a.remove(3)

>>> a



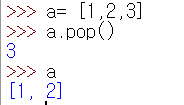
remove(3)을 한 번 더 실행하면 다시 3이 삭제된다.

==>리스트 요소 끄집어 내기(pop)s

>>> a= [1,2,3]

>>> a.pop()

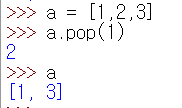
>>> a



>>> a = [1,2,3]

>>> a.pop(1)

>>> a



a.pop(1)은 a[1]의 값을 끄집어 내고 값이 삭제된 것이 확인됨

==>리스트에 포함된 요소 x의 개수 세기(count)

>>> a= [1,2,3,1]

>>> a.count(1)



count(x)는 리스트 내에 x개가 몇 개 있는지 조사하여 그 개수를 돌려주는 함수

=>리스트 확장(extend)

>>> a = [1,2,3]

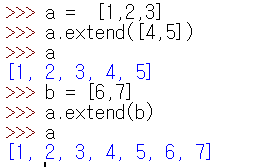
>>> a.extend([4,5])

>>> a

>>> b = [6,7]

>>> a.extend(b)

>>> a



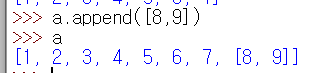
extend(x)에서 x에는 리스트만 올 수 있으며, a리스트에 x리스트를 더하게 됨

튜플로 입력하고 더하기 한다

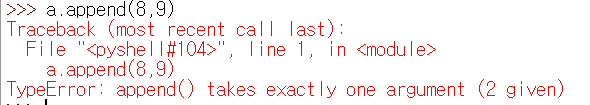
append(),insert(),extend()의 차이점

>>> a.append([8,9])

>>> a

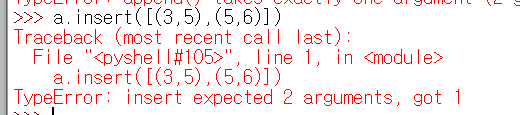


>>> a.append(8,9)



append는 하나에 하나씩 가능하다.

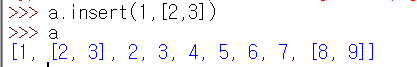
>>> a.insert([(3,5),(5,6)])



insert는 하나에 하나씩 가능하다.

>>> a.insert(1,[2,3])

>>> a



IV. 튜플 자료형

튜플(tuple)은 리스트와 거의 비슷하며 다른 점은 다음과 같음 

리스트는 [ ]으로 둘러싸지만, 튜플은 ( )으로 둘러쌈 

리스트는 그 값의 생성, 삭제, 수정이 가능하지만, 튜플은 그 값을 바꿀 수 없음

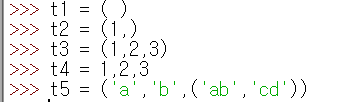
>>> t1 = ()

>>> t2 = (1,)

>>> t3 = (1,2,3)

>>> t4 = 1,2,3

>>> t5 = ('a','b',('ab','cd'))



t2 = (1,)처럼 1개의 요소만을 가질 때는 요소 뒤에 콤마(,)를 반드시 붙여야 함

t4 = 1, 2, 3처럼 괄호( )를 생략해도 무방함

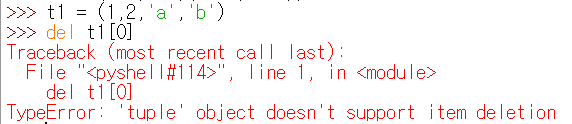
프로그램 실행되는 동안 그 값이 변하지 않기를 바란다면 튜플을 사용하고,수시로 그 값을 변화시켜야 한다면 리스트를 사용함

=>튜플의 요소값을 지우거나 변경하려고 하면 어떻게 될까?

==>튜플 요소 값을 삭제하려 할 때

>>> t1 = (1,2,'a','b')

>>> del t1[0]

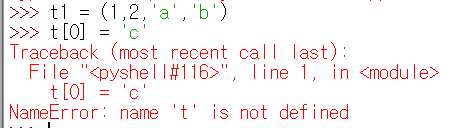


튜플은 요소를 지우는 행위가 지원되지 않는다는 메시지를 확인할 수 있음

==>튜플 요소 값을 변경하려 할 때

>>> t1 = (1,2,'a','b')

>>> t[0] = 'c'



튜플의 요소 값을 변경하려고 해도 오류가 발생하는 것을 확인할 수 있음

튜플은 삭제오 변경이 불가능하다.

=>튜플 다루기

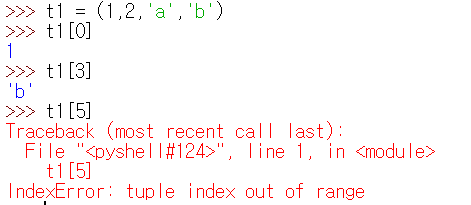
==>인덱싱하기

>>> t1 = (1,2,'a','b')

>>> t1[0]

>>> t1[3]

>>> t1[5]



문자열, 리스트와 같이 t1[0], t1[3]처럼 인덱싱이 가능함

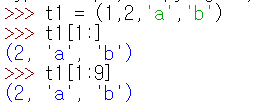
index의 반환한 값이 없을 경우 오류난다.

==>슬라이싱하기

>>> t1 = (1,2,'a','b')

>>> t1[1:]

>>> t1[1:9]



t1[1]부터 튜플의 마지막 요소까지 슬라이싱 함

==>튜플 더하기

>>> t2 = (3,4)

>>> t1+t2



==>튜플 곱하기

>>> t2 \* 3



==>튜플의 길이 구하기

>>> t1= (1,2,'a','b')

>>> len(t1)



==>대입문

>>> a = 4

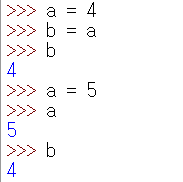
>>> b = a

>>> b

>>> a = 5

>>> a

>>> b

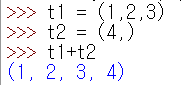


#연습문제 87

t1 = (1,2,3)

t2 = (4,)

t1+t2



## V. 딕셔너리 자료형

딕셔너리는 단어 그대로 해석하면 사전이라는 뜻이며, 연관 배열(Associative array) 또는 해시(Hash)라고 함

딕셔너리는 리스트나 튜플처럼 순차적으로 해당 요소값을 구하지 않고, Key를 통해 Value를 얻으며 이것이 딕셔너리의 가장 큰 특징임

딕셔너리는 어떻게 만들까?

{Key1:Value1, Key2:Value2, Key3:Value3 ···}

Key와 Value의 쌍 여러 개가 { }로 둘러싸여 있음

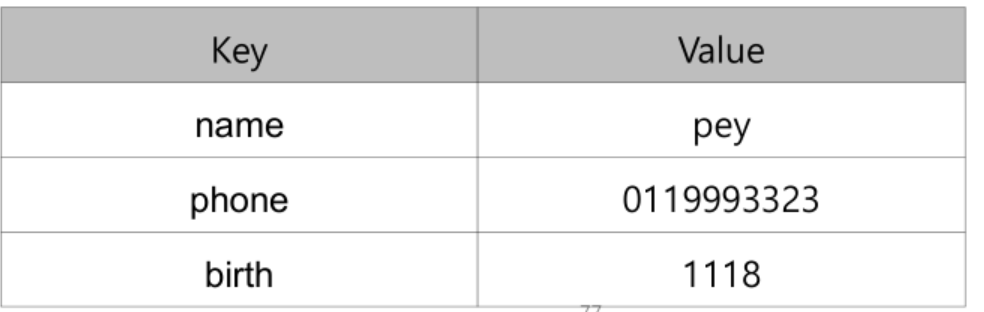
각각의 요소는 Key : Value 형태로 이루어져 있고 쉼표(,)로 구분되어 있음

>>> dic = {'name':'pey','phone':'0119993323','birth':'1118'}

>>> dic



딕셔너리 dic의 정보



>>> a = {1:'hi'}



Key로 정수값 1, Value로 ‘hi’라는 문자열을 사용함

>>> a = {'a':[1,2,3]}



Value에 리스트도 넣을 수 있음

=>딕셔너리 쌍 추가하기

1. 딕셔너리 쌍 추가하기

>>> a = {1:'a'}

>>> a[2] = 'b'

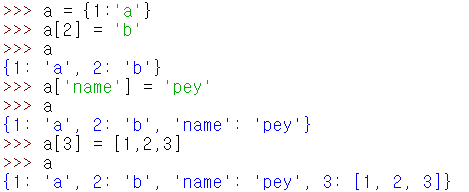
>>> a

>>> a['name'] ='pey'

>>> a

>>> a[3] = [1,2,3]

>>> a



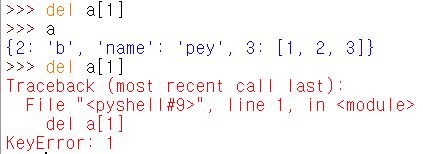
Key는 3, Value는 [1, 2, 3]을 가지는 한 쌍이 추가됨

2. 딕셔너리 요소 삭제하기

>>> del a[1]

>>> a

>>> del a[1] #다시 실행 할 경우 오류 난다.



=>딕셔너리를 사용하는 방법

{“김연아”:”피겨스케이팅“, ”류현진”:”야구“, “박지성“:”축구“, “귀도”:”파이썬”}

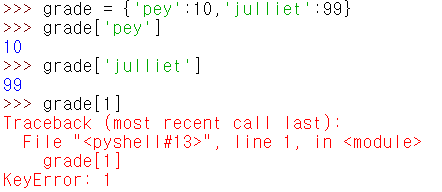
==>딕셔너리에서 Key를 사용해 Value 얻기

>>> grade = {'pey':10,'julliet':99}

>>> grade['pey']

>>> grade['julliet']

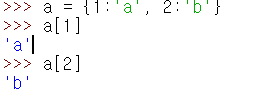
>>> grade[1] #다시 실행 할 경우 오류 난다.



>>> a = {1:'a', 2:'b'}

>>> a[1]

>>> a[2]

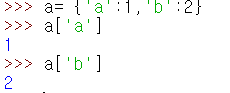


[ ]안의 숫자 1은 두 번째 요소를 뜻하는 것이 아니라, Key에 해당하는 1을 나타냄

>>> a= {'a':1,'b':2}

>>> a['a']

>>> a['b']



딕셔너리 a는 a[Key]로 입력해서 Key에 해당하는 Value를 얻음

>>> dic = {'name':'pey','phone':'0119993323','birth':'1118'}

>>> dic['name']

>>> dic['phone']

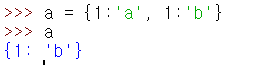
>>> dic['birth']



==>딕셔너리를 만들 때 주의할 사항

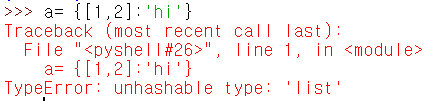
>>> a = {1:'a', 1:'b'}

>>> a



딕셔너리에서 Key는 고유한 값이므로 중복되는 Key 값을 설정해 놓으면 하나를 제외한 나머지 것들이 모두 무시됨. 마지막 Value가 등록됨

>>> a= {[1,2]:'hi'}



Key에 리스트를 쓸 수 없고, 튜플은 Key로 쓸 수 있음

리스트는 그 값이 변할 수 있기 때문에 Key로 쓸 수 없음

딕셔너리의 Key 값으로 딕셔너리를 사용할 수 없음

Value에는 변하는 값이든 변하지 않는 값이든 상관없이 아무 값이나 넣을 수 있음

=>딕셔너리 관련 함수

==>Key 리스트 만들기(keys)

>>> a = {'name':'pey','phone':'0119993323','birth':'1118'}

>>> a.keys()

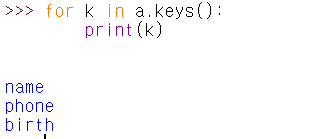


a.keys( )는 딕셔너리 a의 Key만을 모아서 dict\_keys라는 객체를 돌려줌

>>> for k in a.keys():

... print(k)

...

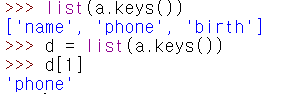


dict\_keys 객체는 리스트를 사용하는 것과 차이가 없지만, 리스트 고유의 append, insert, pop, remove, sort 함수는 수행할 수 없음

>>> list(a.keys())

>>> d = list(a.keys())

>>> d[1]



==>Value 리스트 만들기(values)

>>> a.values()



values 함수는 딕셔너리 a의 Value만을 모아서 dict\_values라는 객체를 돌려줌

a[]이런식으로 하면 오류난다.그래서 get()으로 하면 된다.

==>Key, Value 쌍 얻기(items)

>>> a.items()



==>Key : Value 쌍 모두 지우기(clear)

>>> a.clear()

>>> a

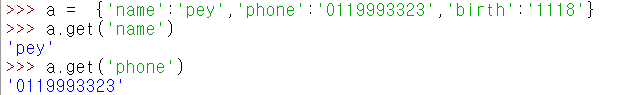


==>Key로 Value 얻기(get)

>>> a = {'name':'pey','phone':'0119993323','birth':'1118'}

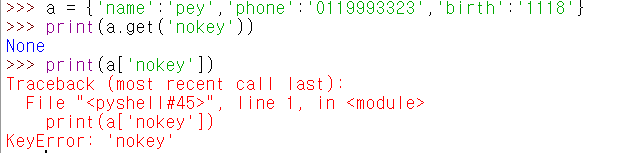
>>> a.get('name')

>>> a.get('phone')



>>> print(a.get('nokey'))

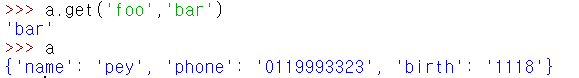
>>> print(a['nokey'])



a[‘nokey’]처럼 존재하지 않는 키(nokey)로 값을 가져오려고 할 때 a[‘nokey’]는 Key 오류를 발생시키고, a.get(‘nokey’)는 None을 리턴한다는 차이가 있음

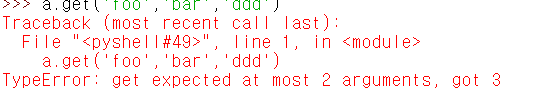
>>> a.get('foo','bar')

>>> a



딕셔너리 안에 찾으려는 Key 값이 없을 경우 미리 정해 둔 디폴트 값을 대신 가져오게 하고 싶을 때에는 get(x, ‘디폴트 값‘)을 사용하면 편리함

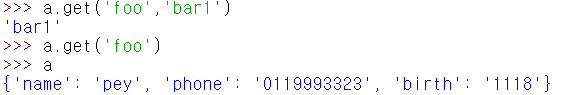
>>> a.get('foo','bar','ddd')



>>> a.get('foo','bar1')

>>> a.get('foo')

>>> a



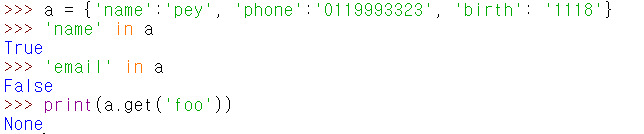
==>해당 Key가 딕셔너리 안에 있는지 조사하기(in)

>>> a = {'name':'pey', 'phone':'0119993323', 'birth': '1118'}

>>> 'name' in a

>>> 'email' in a

>>> print(a.get('foo'))



## VI. 집합 자료형

집합에 관련된 것들을 쉽게 처리하기 위해 만들어진 자료형

집합 자료형은 어떻게 만들까?

집합 자료형은 set 키워드를 이용해 만들 수 있음

중복제거하고 순서없이 해주는 것

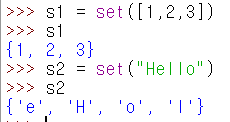
중복된것을 없에고 집합을 만든다.

>>> s1 = set([1,2,3])

>>> s1

>>> s2 = set("Hello")

>>> s2



>>> s1 = set([1,2,3,3])

>>> s1



집합 자료형의 특징

– 중복을 허용하지 않음

– 순서가 없음(Unordered) 

리스트나 튜플은 순서가 있기 때문에 인덱싱을 통해 자료형의 값을 얻을 수 있지만, set 자료형은 순서가 없기 때문에 인덱싱으로 값을 얻을 수 없음 

set 자료형에 저장된 값을 인덱싱으로 접근하려면 리스트나 튜플로 변환 후 해야 함

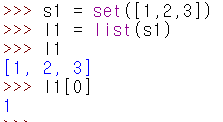
>>> s1 = set([1,2,3])

-------list로 꾸기

>>> l1 = list(s1)

>>> l1

>>> l1[0]

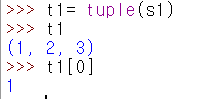


-------tuple로 꾸기

>>> t1= tuple(s1)

>>> t1

>>> t1[0]



=>교집합, 합집합, 차집합 구하기 

set 자료형을 유용하게 사용하는 경우는 교집합, 합집합, 차집합을 구할 때임 

2개의 set 자료형, s1은 1부터 6까지의 값을 가지게 되었고 s2는 4부터 9까지의 값을 가지게 되었음

>>> s1 = set([1,2,3,4,5,6])

>>> s2 = set([4,5,6,7,8,9])



==>교집합(&)

>>> s1 & s2



‘&’ 기호를 사용하면 교집합을 구할 수 있음

>>> s1.intersection(s2)



==>합집합( | )

>>> s1 | s2



>>> s1.union(s2)

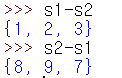


‘|’ 기호나 union 함수를 사용하면 합집합을 구할 수 있음

==>차집합( - )

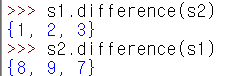
>>> s1-s2

>>> s2-s1



>>> s1.difference(s2)

>>> s2.difference(s1)



빼기(-) 기호나 difference 함수를 사용하면 차집합을 구할 수 있음

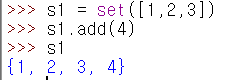
=>집합 자료형 관련 함수 

==>값 1개 추가하기(add)

>>> s1 = set([1,2,3])

>>> s1.add(4)

>>> s1



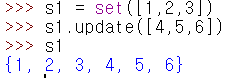
이미 만들어진 set 자료형에 값을 추가할 수 있음 

==>값 여러 개 추가하기(update)

>>> s1 = set([1,2,3])

>>> s1.update([4,5,6])

>>> s1

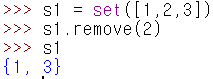


==>특정 값 제거하기(remove)

>>> s1 = set([1,2,3])

>>> s1.remove(2)

>>> s1



## VII. 불 자료형

불(bool) 자료형이란?

참(True)과 거짓(False)을 나타내는 자료형

• True : 참

• False : 거짓

>>> a = True

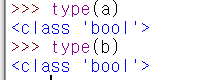
>>> b = False



따옴표로 감싸지 않은 True나 False 예약어를 변수에 지정해서 사용함

>>> type(a)

>>> type(b)

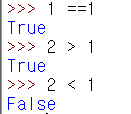


=> 비교연산자

>>> 1 ==1

>>> 2 >1

>>> 2 <1

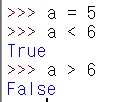


조건문의 반환 값으로도 사용됨

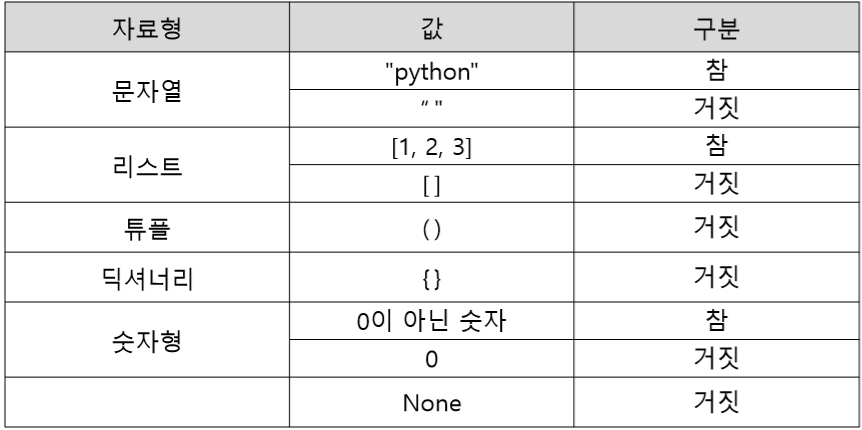
>>> a = 5

>>> a < 6

>>> a > 6



자료형의 참과 거짓



문자열, 리스트, 튜플, 딕셔너리 등의 값이 비어 있으면(““, [], (), {}) 거짓이 됨 

당연히 비어 있지 않으면 참이 됨 

숫자는 그 값이 0일 때 거짓이 됨

FIFO->QUEUE

LIFO->STACK

=>참과 거짓이 프로그램에서 쓰이는 사례

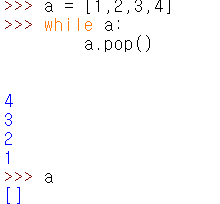
>>> a = [1,2,3,4]

>>> while a:

... a.pop()

...

>>> a



a.pop() 함수는 리스트 a의 마지막 요소를 끄집어 내고, 그 요소는 삭제함

[]는 비어있는 리스트로 거짓이므로 False란 문자열이 출력됨

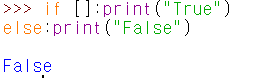
>>> if []:

... print("True")

... else:

... print("False")

...



[1, 2, 3]은 요소값이 있는 리스트로 참이므로 True를 출력함

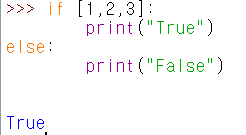
>>> if [1,2,3]:

... print("True")

... else:

... print("False")

...



=> 불(bool) 연산

>>> bool('python')

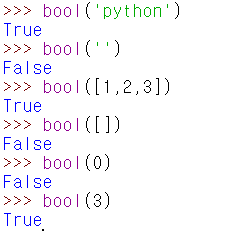
>>> bool('')

>>> bool([1,2,3])

>>> bool([])

>>> bool(0)

>>> bool(3)



문제

1.

>>> a = [1,1,1,2,2,3,3,3,4,4,5]

>>> s = set(a)

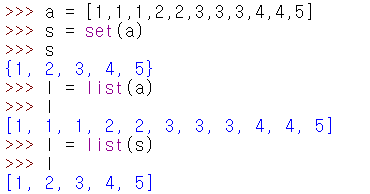
>>> s

>>> l = list(a)

>>> l

>>> l = list(s)

>>> l

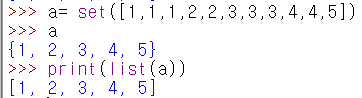


2.

>>> a= set([1,1,1,2,2,3,3,3,4,4,5])

>>> a

>>> print(list(a))

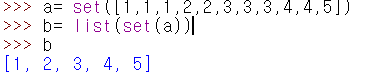


3.

>>> a= set([1,1,1,2,2,3,3,3,4,4,5])

>>> b= list(set(a))

>>> b



## VIII. 자료형의 값을 저장하는 공간, 변수

변수를 만들 때는 =(assignment) 기호를 사용

변수명 = 변수에 저장할 값

 변수를 만들 때 자료형을 직접 지정할 필요가 없음 

변수에 저장된 값을 스스로 판단하여 자료형을 지정하기 때문에 더 편리함

>>> a = 1

>>> b = "python"

>>> c = [1,2,3]



파이썬에서 사용하는 변수는 객체를 가리키는 것

>>> a = [1,2,3]



변수 a는 [1, 2, 3] 리스트가 저장된 메모리의 주소를 가리키게 됨

>>> a = [1,2,3]

>>> id(a)



=> 리스트를 복사할 때

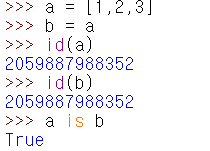
>>> a = [1,2,3]

>>> b = a

>>> id(a)

>>> id(b)

>>> a is b



b라는 변수에 a가 가리키는 리스트를 대입하였음

b는 a와 완전히 동일하다고 할 수 있음

다만 [1, 2, 3] 리스트를 참조하는 변수가 a변수 1개에서 b변수가 추가됨

>>> a[1] = 4

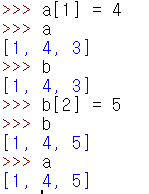
>>> a

>>> b

>>> b[2] = 5

>>> b

>>> a



>>> a = 4

>>> b = a

>>> a is b

>>> a = 5

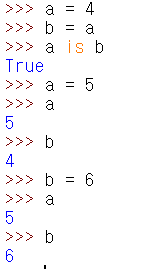
>>> a

>>> b

>>> b = 6

>>> a

>>> b



=>변수를 생성할 때 a 변수의 값을 가져오면서 다른 주소를 가리키도록 만드는 방법

1. [:] 이용

리스트 전체를 가리키는 [:]을 이용해서 복사함

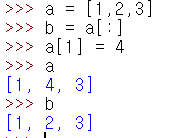
>>> a = [1,2,3]

>>> b = a[:]

>>> a[1] = 4

>>> a

>>> b



a 리스트 값을 바꾸더라도 b 리스트에는 영향을 끼치지 않음

2. copy 모듈 이용

>>> from copy import copy

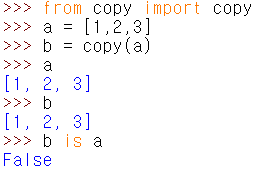
>>> a = [1,2,3]

>>> b = copy(a)

>>> a

>>> b

>>> b is a



b와 a가 가리키는 객체는 서로 다르다는 것을 알 수 있음

=>변수를 만드는 여러 가지 방법

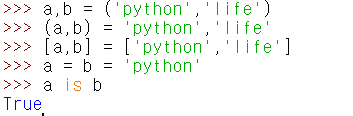
>>> a,b = ('python','life')

>>> (a,b) = 'python','life'

>>> [a,b] = ['python','life']

>>> a = b = 'python'

>>> a is b



튜플로 a, b에 값을 대입할 수 있음

튜플은 괄호를 생략해도 됨

리스트로 변수를 만들 수 있음

여러 개의 변수에 같은 값을 대입할 수 있음

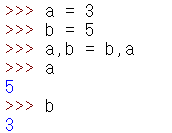
>>> a = 3

>>> b = 5

>>> a, b = b,a

>>> a

>>> b

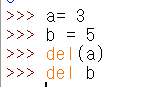


>>> a= 3

>>> b = 5

>>> del(a)

>>> del b



연습문제: 112페이지

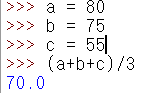
#1.평균점수

>>> a = 80 #국어

>>> b = 75 #영어

>>> c = 55 #수학

>>> (a+b+c)/3



#2.13이 홀수 인지 짝수

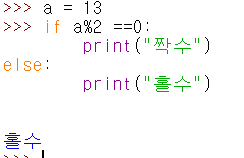
>>> a = 13

>>> if a%2 ==0:

print("짝수")

else:

print("홀수")



#3.

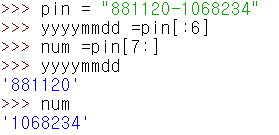
>>> pin = "881120-1068234"

>>> yyyymmdd =pin[:6]

>>> num =pin[7:]

>>> yyyymmdd #연월일 부분 출력

>>> num #숫자 부분 출력



#4.

>>> pin = "881120-1068234"

>>> pin[7]

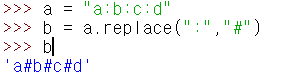


#5.

>>> a = "a:b:c:d"

>>> b = a.replace(":","#")

>>> b



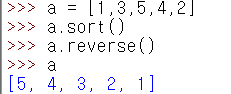
#6.

>>> a = [1,3,5,4,2]

>>> a.sort()

>>> a.reverse()

>>> a

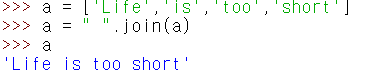


#7.

>>> a = ['Life','is','too','short']

>>> a = " ".join(a)

>>> a

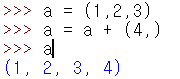


#8.

>>> a = (1,2,3)

>>> a = a + (4,)

>>> a



#9.

>>> a = dict()

>>> a['name'] = 'python'

>>> a

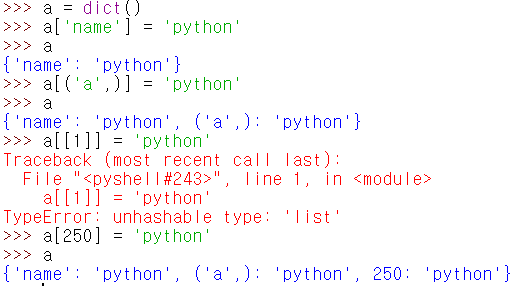
>>> a[('a',)] = 'python'

>>> a

>>> a[[1]] = 'python'

>>> a[250] = 'python'

>>> a



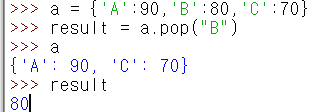
#10.

>>> a = {'A':90,'B':80,'C':70}

>>> result = a.pop("B")

>>> a

>>> result



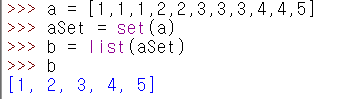
#11.

>>> a = [1,1,1,2,2,3,3,3,4,4,5]

>>> aSet = set(a)

>>> b = list(aSet)

>>> b



#12.

>>> a = b = [1,2,3]

>>> a[1] = 4

>>> b

>>> a is b

