

2. 데이터 타입의 종류

식별자 및 데이터 타입

식별자

- 식별자(identifier)는 언어에서 **변수명, 함수명** 등을 지정하는데 사용
- Python 에서는 식별자로 **한글, 영문자(letter), 숫자(digit), 밑줄(_)** 을 사용가능
- 식별자의 시작은 **숫자외 다른 문자**로 시작하며, 영문자는 **대소문자를 구분함**

데이터 타입 종류

- 파이썬은 많은 내장 데이터 타입(built-in data type)들을 가지고 있음

자료형	타입	설명	예
숫자	int	정수형 데이터(integer)	<u>1</u> , 10, -50, -101
	float	실수형 데이터(floating-point number))	1.25, -23.1546
	complex	<u>복소수형</u> 데이터(complex number)	3+3j
불린	<u>bool</u>	참/거짓(boolean)	True
문자열	<u>str</u>	문자열(string)	' <u>Big_data</u> '
리스트	list	순서가 있는 배열, 수정/추가/삭제가 가능한 구조	[1,2,3,'a','b']
튜플	<u>tuple</u>	순서가 있는 배열, 수정/추가/삭제가 불가능한 구조	(1,2,3,'a','b')
<u>딕셔너리</u>	dict	{key, value}키와 값의 쌍으로 데이터를 저장	{'Math':99,'English':88}
집합	set	키로 <u>구성되어</u> 있는 구조, 순서가 없고 <u>중복되지</u> 않는 값들의 집합	{'a','b','c'}

□ 숫자형 타입 : int, float, complex 확인

- `type()` 함수: 변수의 타입 확인 가능

```
In [1]: a = 1
```

```
In [2]: b = 1.2345
```

```
In [3]: c = 3+4j
```

```
In [4]: type(a)
```

```
Out[4]: int
```

```
In [5]: type(b)
```

```
Out[5]: float
```

```
In [6]: type(c)
```

```
Out[6]: complex
```

```
In [7]: d = a + b + c
```

```
In [8]: d
```

```
Out[8]: (5.2345+4j)
```

□ 문자열(str) 타입

- 큰따옴표(" "), 작은따옴표(' '), 큰따옴표세개(""" """), 작은따옴표세개('' ''')로 문자열 표현 가능
- 따옴표 안에 숫자를 써도 문자로 인식
- `len()`: 문자열 길이

```
In [37]: sa = 'Hello Bigdata'
```

```
In [38]: print(sa)
```

```
Hello Bigdata
```

```
In [39]: sb = "Hello Bigdata"
```

```
In [40]: print(sb)
```

```
Hello Bigdata
```

```
In [41]: sc = """Hello Bigdata"""
```

```
In [42]: print(sc)
```

```
Hello Bigdata
```

```
In [44]: sd = '''Hello Bigdata'''
```

```
In [45]: print(sd)
```

```
Hello Bigdata
```

```
In [47]: se = 'and Cheer up!'
```

```
In [48]: print(se)
```

```
and Cheer up!
```

```
In [49]: print(sd + se)
```

```
Hello Bigdataand Cheer up!
```

```
In [50]: a = '='
```

```
In [51]: print(a)
```

```
=
```

```
In [52]: print(a*50)
```

```
=====
```

```
In [56]: len(a)
```

```
Out[56]: 1
```

```
In [57]: len(se)
```

```
Out[57]: 13
```

```
In [58]: len(a*50)
```

```
Out[58]: 50
```

- 문자열 인덱싱(Indexing)

- Indexing : 특정한 위치의 데이터만 골라내는 것 ([] 대괄호 사용)
- 각각의 문자열은 위치에 따른 고유 번호를 가짐
- a의 2를 인덱싱 할 때 세 번째 문자를 나타냄
why) 파이썬에서는 0,1,2,3...으로 0부터 숫자를 셈
- -1은 뒤에서 부터 숫자를 셈
- 'this is a string' 이라는 문자열 자료형의 인덱스 번호

문자		t	h	i	s		i	s		a		s	t	r	i	n	g
번호	앞	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	뒤	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- 인덱싱 방법 : 변수이름[번호]
 - 변수 a 에 문자 'this is a string' 입력
 - a[2] 는 3번째 있는 'i' 출력

```
In [59]: a = 'this is a string'
```

```
In [60]: print(a)
this is a string
```

```
In [61]: print(a[2])
i
```

```
In [62]: print(a[0])
t
```

```
In [63]: print(a[-1])
g
```

```
In [64]: print(a[5])
i
```

```
In [65]: print(a[4])
```

- 문자열 슬라이싱(Slicing)

- Slicing : 자료 구조 내에서 여러 개의 항목에 접근

- Slicing 방법 : 변수이름[시작번호:끝번호]

- 슬라이싱에서는 '끝번호' 값은 제외하고 출력

- 앞자리가 시작번호 뒷자리가 끝번호인데 끝번호는 포함되지 않음

```
In [13]: 1 b = '20200722Wed'
```

```
In [14]: 1 print(b)
```

```
20200722Wed
```

```
In [15]: 1 print(b[0:4]) #print([0:8])
```

```
2020
```

```
In [16]: 1 print(b[4:6])
```

```
07
```

```
In [17]: 1 print(b[:8]) #print([0:8])
```

```
20200722
```

```
In [18]: 1 print(b[8:]) #8번째 부터 끝까지
```

```
Wed
```

```
In [19]: 1 print(b[:]) #전체
```

```
20200722Wed
```

```
In [20]: 1 print(b[:-5]) #-6번째까지
```

```
202007
```

- 문자열 수정

- `replace()` 함수 이용하여 해당 문자 변경 가능

```
In [80]: a = 'this is a string'
```

```
In [81]: a[10:] = 'rope'
```

```
TypeError                                Traceback (most recent call last)
<ipython-input-81-110c0139d231> in <module>
----> 1 a[10:] = 'rope'

TypeError: 'str' object does not support item assignment
```

```
In [82]: b = 'rope'
```

```
In [84]: a[:10] + b
```

```
Out[84]: 'this is a rope'
```

```
In [86]: c = a.replace('string', 'rope')
```

```
In [87]: print(c)
```

```
this is a rope
```

- 문자열 format

- %d: 숫자(정수)
- %s: 문자 (뒤에 반드시 %"문자"와 같이 ""넣어줘야 함)
 - 어떤 형식이든지 다 받아줌
 - %d, %f 대신 %s 사용 가능
- %f: 소수 자리수 출력,
 - %.숫자f: 표시할 소수점 아래 자릿수
- 여러 개의 문자열을 입력할 때 %(3, "all")와 같이 ,로 구분하여 추가
- %10s: %와 s 사이에 숫자를 넣는 경우는 그 숫자 자릿수 공간에서 오른쪽 정렬을 해라는 뜻, 숫자앞에 마이너스(-)를 붙이면 왼쪽 정렬을 하라는 뜻임
- 소수점 자릿수 표현
 - %0.5f -> 소수점 뒤에 표현될 숫자 개수를 지정(소수점 이하 5자리)
 - %20.5f -> 전체 20자리의 문자열 공간에서 오른쪽 정렬로 소수점 이하 5 자리

```
In [91]: a=11
In [98]: "a equals %d." % 3
Out[98]: 'a equals 3.'
In [99]: "a equals %d." % a
Out[99]: 'a equals 11.'
In [100]: b=-22
In [101]: "vars are %d and %d." %(a,b)
Out[101]: 'vars are 11 and -22.'
In [103]: "I have %s children." % "three"
Out[103]: 'I have three children.'
```

```
In [104]: "vars are %s and %s." %(a,b)
Out[104]: 'vars are 11 and -22.'
In [105]: "vars are %s and %s." %(3.14,5)
Out[105]: 'vars are 3.14 and 5.'
In [106]: "%10s"% "Good"
Out[106]: '      Good'
In [107]: "%-10s"% "Good"
Out[107]: 'Good      '
In [108]: "%0.5f"% 3.141592
Out[108]: '3.14159'
In [109]: "%20.5f"% 3.141592
Out[109]: '          3.14159'
```

- 문자열에 자주 사용하는 함수들

- 문자열.count('문자'): 한문자에 대한 개수를 셈
- 문자열.find('문자'): 찾고자 하는 문자의 위치 파악(같은 문자가 있을 땐 가장 앞에 있는 문자 위치, 대소문자 구분함, -1이 결과값으로 나올 땐 해당 문자가 없음)
- 문자열.index('문자'): find 함수와 같이 찾고자 하는 문자의 위치를 알려주지만, 문자가 없을 땐 에러가 발생
- ", ".join('문자'): 문자 각각 사이를 ,로 구분지음
- 문자열.upper(): 소문자를 모두 대문자로 표현
- 문자열.lower(): 대문자를 모두 소문자로 표현
- 문자열.strip(): 앞뒤에 양쪽의 공백을 모두 없애기
- 문자열.lstrip(): 왼쪽의 공백만 없애기
- 문자열.rstrip(): 오른쪽의 공백만 없애기
- 문자열.replace("old문자", "new문자"): old문자를 new문자로 변경
- 문자열.split(): 문자열을 나누어 주는 함수. space, tab, enter등을 기준으로 문자열을 나누어줌. 특정 값을 입력하면 해당 값 기준으로 나누어줌

```
In [139]: #문자열 내장함수
In [140]: a = "Korean culture"
In [141]: a.count('u')
Out[141]: 2
In [142]: a.find('u')
Out[142]: 8
In [143]: a.find('K')
Out[143]: 0
In [144]: a.find('k')
Out[144]: -1
In [145]: a.index('u')
Out[145]: 8
In [146]: a.index('k')
ValueError                                Traceback (most recent call last)
<python-input-146-cb6c8ec35229> in <module>
----> 1 a.index('k')
ValueError: substring not found
In [148]: ", ".join("Korean culture")
Out[148]: 'K,o,r,e,a,n, ,c,u,l,t,u,r,e'
In [152]: a.upper()
Out[152]: 'KOREAN CULTURE'
```

```
In [153]: a.lower()
Out[153]: 'korean culture'
In [156]: b = ' Korean culture '
In [157]: b.strip()
Out[157]: 'Korean culture'
In [158]: b.lstrip()
Out[158]: 'Korean culture '
In [159]: b.rstrip()
Out[159]: ' Korean culture'
In [160]: a.replace("Korean", "American")
Out[160]: 'American culture'
In [161]: a.split()
Out[161]: ['Korean', 'culture']
In [162]: c = "Korean & American"
In [163]: c.split("&")
Out[163]: ['Korean ', ' American']
```


□ 불린(Boolean) 타입

- `bool()` 함수 : 참/거짓을 식별
- `bool()` 함수 이용시 문자열, 리스트, 튜플, 딕셔너리가 비어져있으면 `False`가 되고 그렇지 않으면 `True`가 됨
- `bool()` 함수 이용시 숫자일 때 0이면 `False`이고 그렇지 않으면 `True`가 됨

```
In [25]: d = True
```

```
In [26]: type(d)
```

```
Out[26]: bool
```

```
In [27]: e = False
```

```
In [28]: d and e
```

```
Out[28]: False
```

```
In [29]: d or e
```

```
Out[29]: True
```

```
In [30]: 6 == 6
```

```
Out[30]: True
```

```
In [31]: 6 > 4
```

```
Out[31]: True
```

```
In [46]: not d
```

```
Out[46]: False
```

```
In [32]: bool("ABC")
```

```
Out[32]: True
```

```
In [33]: bool("")
```

```
Out[33]: False
```

```
In [34]: bool([1,2,3])
```

```
Out[34]: True
```

```
In [35]: bool(0)
```

```
Out[35]: False
```

```
In [36]: bool(1)
```

```
Out[36]: True
```

□ 리스트(LIST) 타입

- 리스트는 대괄호[] 안에 각 요소(숫자, 문자, 숫자+문자, 리스트 안에 리스트 등) 값들은 쉼표(,)로 구분

```
In [164]: #리스트 생성
```

```
In [165]: a = [] #a=list()
```

```
In [166]: b = [1,2,3,4,5]
```

```
In [167]: c = ['red','blue','yellow']
```

```
In [168]: d = [1,2,'red','blue']
```

```
In [169]: e=[1,2,['red','blue']]
```

```
In [170]: print(a)
```

```
[]
```

```
In [171]: print(b)
```

```
[1, 2, 3, 4, 5]
```

```
In [173]: print(c)
```

```
['red', 'blue', 'yellow']
```

```
In [174]: print(d)
```

```
[1, 2, 'red', 'blue']
```

```
In [175]: print(e)
```

```
[1, 2, ['red', 'blue']]
```

- 리스트 인덱싱

- 리스트 인덱싱도 문자열 인덱싱과 마찬가지로 위치를 기반으로 인덱싱함(0부터 시작)
- 인덱싱 방법 : 변수이름[번호]

```
In [164]: #리스트 생성
In [165]: a = [] #a=list()
In [166]: b = [1,2,3,4,5]
In [167]: c = ['red','blue','yellow']
In [168]: d = [1,2,'red','blue']
In [169]: e=[1,2,['red','blue']]
In [170]: print(a)
[]
In [171]: print(b)
[1, 2, 3, 4, 5]
In [173]: print(c)
['red', 'blue', 'yellow']
In [174]: print(d)
[1, 2, 'red', 'blue']
In [175]: print(e)
[1, 2, ['red', 'blue']]
In [176]: b[0] #b의 첫번째 값
Out[176]: 1
```

```
In [177]: b[0] + b[2] #첫번째 요소와 세번째 요소 합
Out[177]: 4
In [178]: b[-1] #뒤에서 첫번째
Out[178]: 5
In [179]: a = [1,2,3,4,5,['a','b','c','d'],7,8,9]
In [180]: print(a)
[1, 2, 3, 4, 5, ['a', 'b', 'c', 'd'], 7, 8, 9]
In [181]: a[0]
Out[181]: 1
In [182]: a[-1]
Out[182]: 9
In [185]: a[5] #list 안에 6번째
Out[185]: ['a', 'b', 'c', 'd']
In [186]: a[5][2] #list 안에 6번째 인덱싱 해서 다시 한번 세번째 요소 인덱싱
Out[186]: 'c'
```

● 리스트 슬라이싱

○ Slicing 방법 : 변수이름[시작번호:끝번호]

- 슬라이싱에서는'끝번호' 값은 제외하고 출력
- 앞자리가 시작번호 뒷자리가 끝번호인데 끝번호는 포함되지 않음

```
In [187]: #슬라이싱
In [188]: b = [1,2,3,4,5]
In [190]: b[0:4] #첫번째 부터 네번째 까지 실행
Out[190]: [1, 2, 3, 4]
In [191]: c = "12345"
In [192]: c[0:4]
Out[192]: '1234'
In [193]: b[:3] #첫번째 부터 세번째 까지 실행
Out[193]: [1, 2, 3]
In [194]: b[3:] #네번째 부터 끝까지
Out[194]: [4, 5]
In [195]: a = [1,2,3,4,5,['a','b','c','d'],7,8,9]
In [196]: a[0:6]
Out[196]: [1, 2, 3, 4, 5, ['a', 'b', 'c', 'd']]
```

```
In [197]: a[0:6:1] #0:6의 안에서 한자리 수마다 출력해라,a[0:6] 동일한 결과
Out[197]: [1, 2, 3, 4, 5, ['a', 'b', 'c', 'd']]
In [198]: a[0:6:2] #0:6의 안에서 두자리 수마다 출력해라
Out[198]: [1, 3, 5]
In [199]: a[0:6:3] #0:6의 안에서 세자리 수마다 출력해라
Out[199]: [1, 4]
In [200]: a[5]
Out[200]: ['a', 'b', 'c', 'd']
In [201]: a[5][:2]
Out[201]: ['a', 'b']
```

● 리스트 연산

```

In [202]: #연산
In [203]: a=[1,2,3,4]
In [204]: b=[6,7,8,9]
In [205]: a+b
Out[205]: [1, 2, 3, 4, 6, 7, 8, 9]
In [206]: a*2
Out[206]: [1, 2, 3, 4, 1, 2, 3, 4]
In [208]: len(a)
Out[208]: 4
In [209]: k=[12345678]
In [210]: len(k)
Out[210]: 1
In [211]: m='12345678'
In [212]: len(m)
Out[212]: 8
In [213]: c='good' #문자열
In [214]: b+c #리스트 + 문자열
-----
TypeError                                Traceback (most recent call last)
<ipython-input-214-5862b4a280fe> in <module>
----> 1 b+c #리스트 + 문자열

TypeError: can only concatenate list (not "str") to list

```

```

In [215]: c = ['good'] #리스트로 변경
In [216]: b+c
Out[216]: [6, 7, 8, 9, 'good']
In [217]: #수정
In [218]: a[3]
Out[218]: 4
In [219]: a[3]=9
In [220]: print(a)
[1, 2, 3, 9]
In [221]: n='1234'
In [222]: n[3]
Out[222]: '4'
In [223]: n[3]='9' #문자열 수정 불가
-----
TypeError                                Traceback (most recent call last)
<ipython-input-223-587d84da8a39> in <module>
----> 1 n[3]='9' #문자열 수정 불가

TypeError: 'str' object does not support item assignment

```

- 리스트 기초 연산 함수

- sum:합, max:최대값, min:최소값, del:삭제, str:문자형으로 변경

```
In [234]: a = [1,2,3,4,[a,b,c]]
In [235]: b = [6,7,8,9]
In [236]: len(a)
Out[236]: 5
In [237]: sum(a)
TypeError                                 Traceback (most recent call last)
<ipython-input-237-bed23830df6e> in <module>
----> 1 sum(a)
TypeError: unsupported operand type(s) for +: 'int' and 'list'

In [238]: sum(b)
Out[238]: 30
In [239]: max(b)
Out[239]: 9

In [240]: min(b)
Out[240]: 6
In [241]: del(b[0]) #b[0]삭제
In [242]: print(b)
[7, 8, 9]
In [244]: b = [6,7,8,9]
In [245]: del b[:3] #b의 첫번째 요소에서 세번째 요소까지 삭제
In [246]: print(b)
[9]
In [247]: str(b[0]) #숫자형을 문자형으로 변경
Out[247]: '9'
```

- append: 리스트 추가, sort: 올림차순으로 정렬(숫자, 문자 정렬가능)
- reverse: 현재 리스트의 순서를 반대로 출력
- index: 해당 값의 위치를 출력, 없는 값 입력하면 에러발생
- insert: 특정 위치에 값을 추가

```
In [250]: b = ['6','7','8','9']
In [251]: b.append('10') #기존의 리스트에 요소 추가
In [252]: print(b)
['6', '7', '8', '9', '10']
In [253]: b.append(['11','12'])
In [254]: print(b)
['6', '7', '8', '9', '10', ['11', '12']]
In [255]: c=['6','9','8','7']
In [256]: print(c)
['6', '9', '8', '7']
In [257]: c.sort() #크기순서대로 정렬
In [258]: print(c)
['6', '7', '8', '9']
In [266]: d = ['x','z','y']
In [267]: d.sort()
In [268]: print(d)
['x', 'y', 'z']

In [269]: c=['6','9','8','7']
In [270]: c.reverse() #반대순서로 변환
In [271]: print(c)
['7', '8', '9', '6']
In [273]: c.index('9') #해당값의 위치, 3번째 요소를 뜻하는 2가 출력
Out[273]: 2
In [274]: c.index('5') #리스트에 없는 값 출력
ValueError                                 Traceback (most recent call last)
<ipython-input-274-ab8c6e192a19> in <module>
----> 1 c.index('5') #리스트에 없는 값 출력
ValueError: '5' is not in list

In [275]: c.insert(4,'10') #4번 위치(실제로는 5번째)에 '10'을 추가
In [276]: print(c)
['7', '8', '9', '6', '10']
In [277]: c.insert(0,'5') #0번 위치(실제로는 1번째)에 '5'를 추가
In [278]: print(c)
['5', '7', '8', '9', '6', '10']
```

- remove: 특정 값을 삭제(list안에 동일한 값이 여러개 있을 때 맨 앞의 값만 삭제)
- pop: 빈칸 입력시 리스트의 마지막 값을 보여주고 삭제시킴. 입력시 삭제 요소의 위치 지정

```

In [279]: e=['3','4','3','4','5']

In [280]: e.remove('4') # '4'를 제거(여러개 있으면 맨 앞의값 제거)

In [281]: print(e)
['3', '3', '4', '5']

In [282]: e.remove('4') # '4'를 제거

In [283]: print(e)
['3', '3', '5']

In [284]: e.pop() #리스트에서 마지막 요소 보여주고 삭제
Out[284]: '5'

In [285]: print(e)
['3', '3']

In [286]: e.pop(1) #삭제시킬 요소 입력, 1번삭제(2번째요소)
Out[286]: '3'

In [287]: print(e)
['3']

In [288]: e.pop(0)
Out[288]: '3'

count: 동일한 값 개수 세기

extend: 리스트의 요소 추가 (괄호 안에 리스트만 올 수 있음, append와의 차이구분)

In [290]: c = [6,6,6,6,6,7,8,9]

In [292]: c.count(6) #6의 갯수 세기
Out[292]: 5

In [293]: c.extend([4,5]) #괄호안에 리스트만 올 수 있음.리스트의 요소만 추가됨

In [294]: print(c)
[6, 6, 6, 6, 6, 7, 8, 9, 4, 5]

In [295]: c.append([4,5])

In [296]: print(c)
[6, 6, 6, 6, 6, 7, 8, 9, 4, 5, [4, 5]]

```

□ 튜플(TUPLE) 타입

- 튜플 타입은 리스트 타입과 비슷
- 차이점은 리스트는 대괄호 []를 사용하고 튜플은 소괄호 ()를 사용(괄호생략가능)하며, 리스트는 수정이 가능하지만 튜플은 수정이 불가능함

- 튜플에서는 괄호안에 한 개의 요소를 가질 때는 (1,)와 같이 마지막에 콤마 필요
- 콤마(,)를 붙이지 않으면 튜플이 아닌 정수 타입이 생성
- 프로그램을 실행되는 동안 **그 값이 항상 변화지 않아야 한다면**, 튜플을 사용하고 값이 자주 바뀐다면 리스트를 사용

```
In [297]: #튜플의 생성
```

```
In [298]: a = tuple()
```

```
In [299]: a=()
```

```
In [300]: type(a)
```

```
Out[300]: tuple
```

```
In [303]: b = (1,) #하나만 입력시 반드시 콤마 추가
```

```
In [306]: type(b)
```

```
Out[306]: tuple
```

```
In [307]: c = (1,2,3)
```

```
In [308]: type(c)
```

```
Out[308]: tuple
```

```
In [309]: d = 1,2,3 #괄호 생략 가능
```

```
In [310]: type(d)
```

```
Out[310]: tuple
```

```
In [311]: e = 1,2,('ab','cd'),3,4
```

```
In [313]: type(e)
```

```
Out[313]: tuple
```

- 튜플은 수정 및 변경 불가

```
In [311]: e = 1,2,('ab','cd'),3,4
```

```
In [313]: type(e)
```

```
Out[313]: tuple
```

```
In [314]: e[0]
```

```
Out[314]: 1
```

```
In [315]: del e[0]
```

```
-----
TypeError                                 Traceback (most recent call last)
<ipython-input-315-88f9a3ba7b63> in <module>
----> 1 del e[0]

TypeError: 'tuple' object doesn't support item deletion
```

```
In [316]: e.remove[1]
```

```
-----
AttributeError                             Traceback (most recent call last)
<ipython-input-316-92eade322d4d> in <module>
----> 1 e.remove[1]

AttributeError: 'tuple' object has no attribute 'remove'
```

```
In [318]: e[:] #요소 전체 출력
```

```
Out[318]: (1, 2, ('ab', 'cd'), 3, 4)
```

```
In [319]: c+e
```

```
Out[319]: (1, 2, 3, 1, 2, ('ab', 'cd'), 3, 4)
```

```
In [320]: c*2
```

```
Out[320]: (1, 2, 3, 1, 2, 3)
```

```
In [321]: len(c)
```

```
Out[321]: 3
```

```
In [322]: x,y,z=(1,2,3)
```

```
In [324]: print(x,y,z)
```

```
1 2 3
```

```
In [325]: print(x)
```

```
1
```

□ 사전(DICT)타입

- key와 value를 한 쌍으로 가지고 있는 자료형
- {key1:value1,key2:value2,key3:value3...}

- 위치자리에 key값을 입력하면 value 출력

```
In [326]: #사전의 생성
```

```
In [328]: #{key1,:value1,key2:value2,...}
```

```
In [329]: a ={'gender':'Female', 'age':30, 'name':'Kristina'}
```

```
In [330]: type(a)
```

```
Out[330]: dict
```

```
In [332]: a['gender'] #위치자리에 key를 입력하면, key해당되는 value가 반환됨
```

```
Out[332]: 'Female'
```

```
In [333]: a['age']
```

```
Out[333]: 30
```

```
In [334]: a[0] #기존 방법처럼 위치를 지정하면 예러남
```

```
-----  
KeyError                                Traceback (most recent call last)  
<ipython-input-334-6a1284577a36> in <module>  
----> 1 a[0]  
  
KeyError: 0
```

```
In [1]: a['Krestina']
```

```
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-1-24c09df5fdd1> in <module>  
----> 1 a['Krestina']  
  
NameError: name 'a' is not defined
```

- 사전에서 추가 및 삭제하기

```
In [12]: b = {'good': 1} #key자리에 정수값 1을 넣고 dictionary 만들. 실행가능

In [13]: print(b)
{'good': 1}

In [17]: c = {'a': [1, 3, 5]} #key자리에 문자열, value자리에 리스트 [1,3,5]입력

In [18]: print(c)
{'a': [1, 3, 5]}

In [21]: d = {'a': (1, 3, 5)} #튜플도 딕셔너리 지정 가능

In [22]: print(d)
{'a': (1, 3, 5)}

In [23]: c = {[1, 3, 5]: 'a'} #리스트(수정가능)는 key자리에는 사용 불가
File "<ipython-input-23-fd9f75c50466>", line 1
      c = {[1, 3, 5]: 'a'} #리스트(수정가능)는 key자리에는 사용 불가
            ^
SyntaxError: invalid syntax
```

```
In [25]: d = {(1, 3, 5): 'a'} #튜플(바꿀수없음)은 key자리에 사용가능

In [26]: a = {'gender': 'Female'}

In [27]: a['age'] = 30 #age = 30 key value 추가

In [28]: print(a)
{'gender': 'Female', 'age': 30}

In [29]: a['name'] = 'Kristina' #key value 추가

In [30]: print(a)
{'gender': 'Female', 'age': 30, 'name': 'Kristina'}

In [32]: a['a'] = [1, 3, 5] #[1,3,5] 리스트 추가

In [33]: print(a)
{'gender': 'Female', 'age': 30, 'name': 'Kristina', 'a': [1, 3, 5]}

In [34]: del a['a'] #사전의 삭제, 해당key와 value 한쌍이 삭제

In [35]: print(a)
{'gender': 'Female', 'age': 30, 'name': 'Kristina'}

In [36]: del a['name']

In [37]: print(a)
{'gender': 'Female', 'age': 30}
```

• 사전 함수

- keys: 전체 key출력
- values: key값 출력
- items: 전체 key와 value를 쌍으로 출력
- clear: 딕셔너리안에 요소 삭제
- get: key값(value) 출력

```
In [51]: #사전 함수

In [61]: a = {'gender': 'Female', 'age': 30, 'name': 'Kristina'}

In [62]: a.keys() #key 출력, 파이썬3.00이상부터 적용
Out [62]: dict_keys(['gender', 'age', 'name'])

In [63]: list(a.keys())
Out [63]: ['gender', 'age', 'name']

In [64]: a.values() #전체 value값 출력
Out [64]: dict_values(['Female', 30, 'Kristina'])

In [65]: a.items() #key, value(key값)를 묶은 한쌍을 출력
Out [65]: dict_items([('gender', 'Female'), ('age', 30), ('name', 'Kristina')])
```

```
In [66]: a.clear() #딕셔너리안에 모든 요소들을 삭제

In [67]: print(a)

NameError                                Traceback (most recent call last)
<ipython-input-67-5255153360e2> in <module>
----> 1 print(a)

NameError: name 'printer' is not defined

In [68]: a = {'gender': 'Female', 'age': 30, 'name': 'Kristina'}

In [69]: a.get('gender') #a['gender']
Out [69]: 'Female'

In [70]: a.get('height', '?') #key가 딕셔너리 안에 아무값 없을때, ?로 표현
Out [70]: '?'

In [71]: 'gender' in a # 값이 있으면 True
Out [71]: True

In [72]: 'height' in a # 값이 없으면 False
Out [72]: False
```

