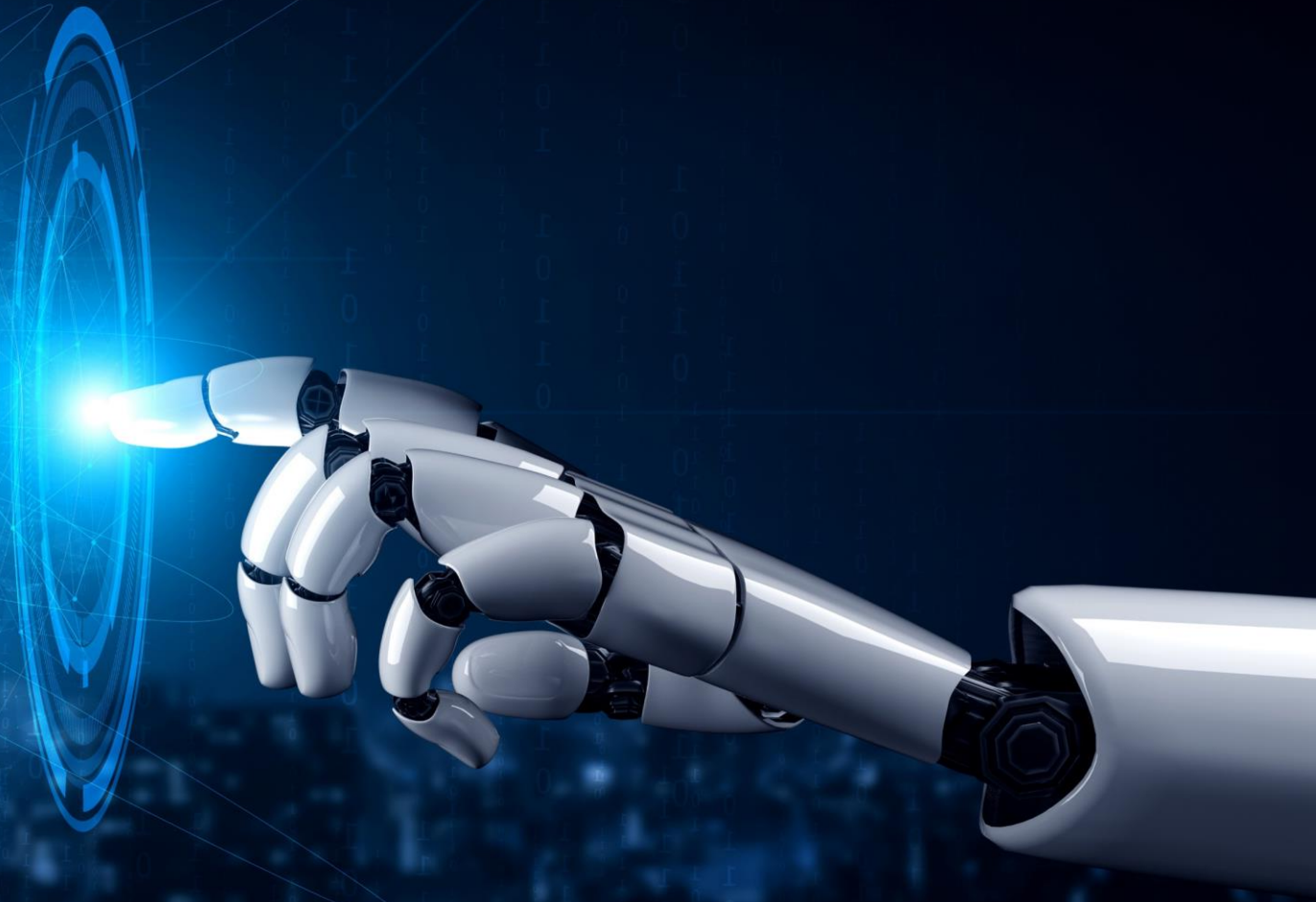


파이선의 기초 II

충북대학교 소프트웨어학과
류관희



목 차

❖ Part 1 주피터를 이용한 파이선 프로그래밍 하기

- Jupyter를 통한 프로그램 생성
- Python 언어의 기본기능

❖ Part 2 파이선 자료구조 I

- 리스트
- 복합리스트
- 튜플

❖ Part 3 파이선 자료구조 II

- 딕셔너리
- 자료유형변환



목 차

❖ Part 1

- Jupyter를 통한 프로그램 생성
- Python 언어의 기본기능

❖ Part 2

- 리스트
- 복합리스트
- 튜플

❖ Part 3

- 딕셔너리
- 자료유형변환



Jupyter를 통한 python 프로그램

```
관리자: 명령 프롬프트
Microsoft Windows [Version 10.0.18362.1016]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd %users%
C:\Users>cd khyoo
C:\Users\khyoo>python
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated. Libraries may fail to load. To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>>
C:\Users\khyoo>jupyter notebook
```

Jupyter를 통한 프로그램 생성

jupyter

Quit Logout

Files Running Clusters

Select items to perform actions on them.

Upload New

Python3 선택

Upload New

Notebook:

PyTorch

Python 3

Other:

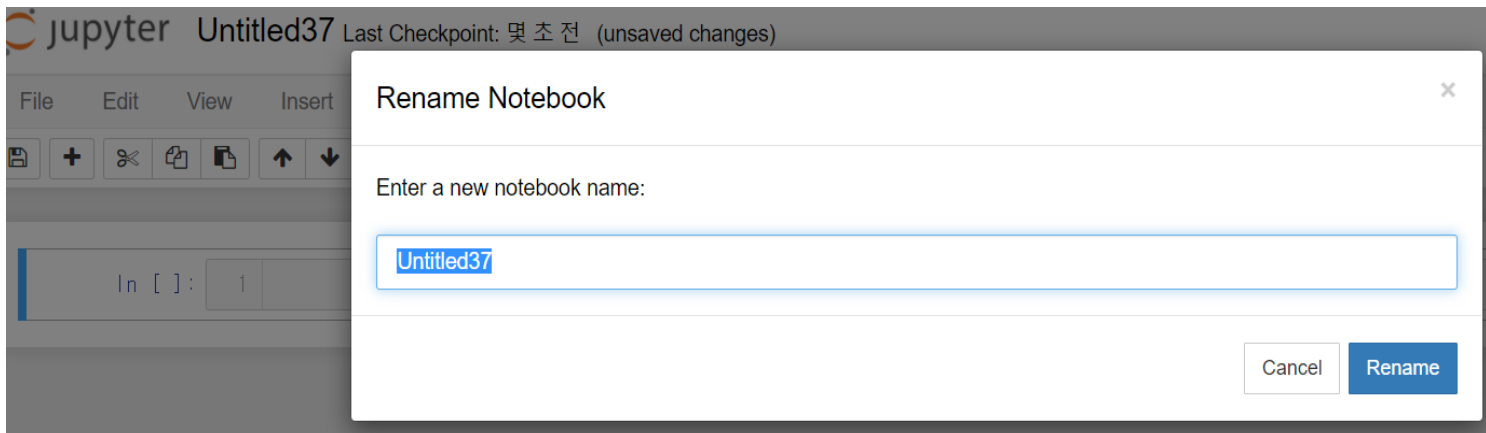
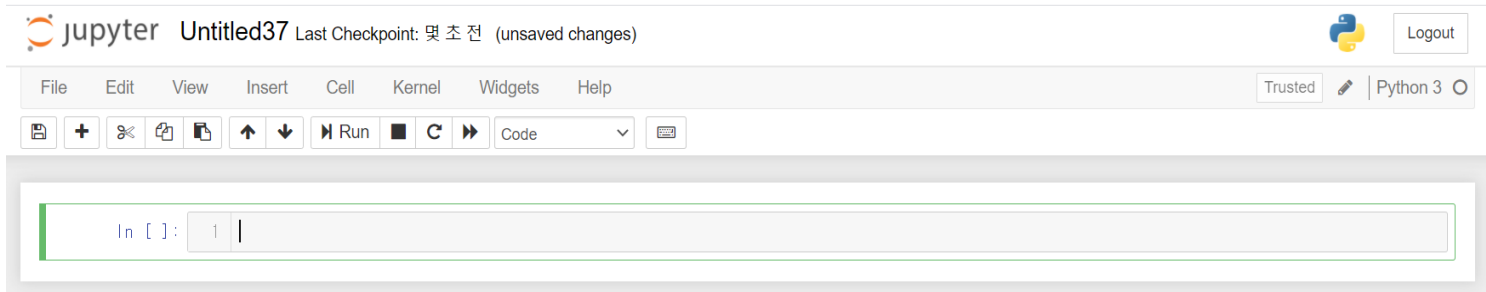
Text File

Folder

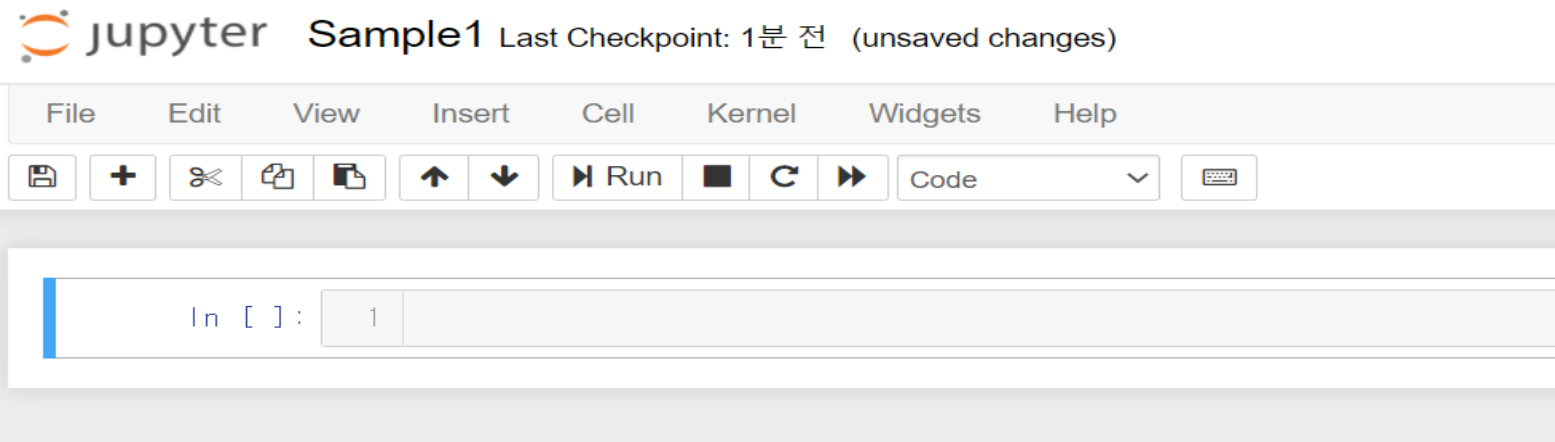
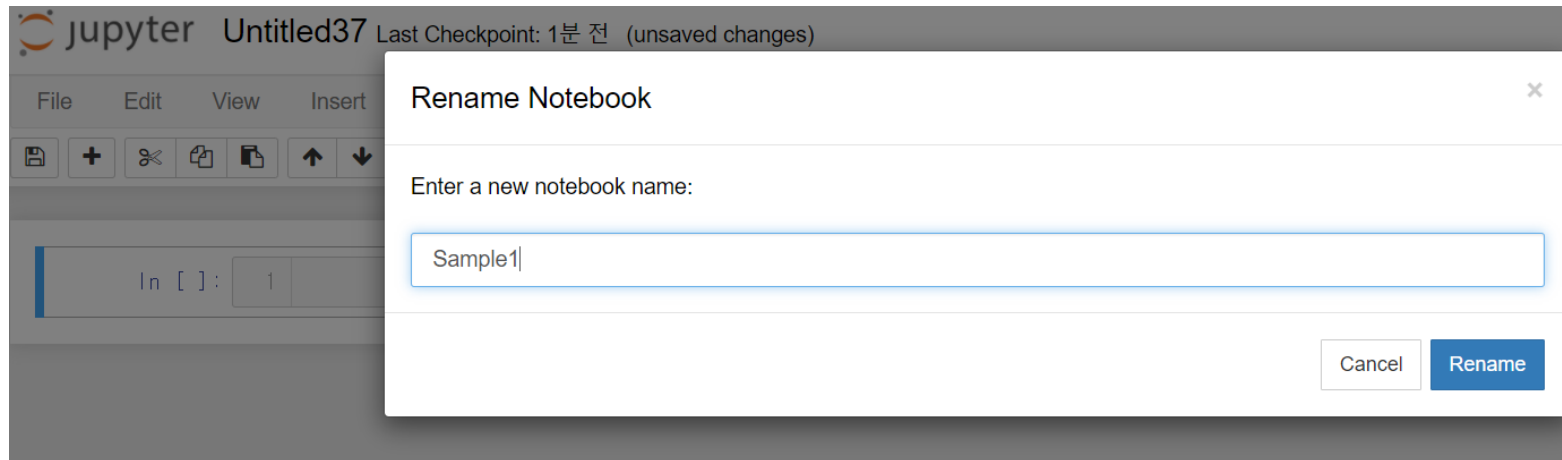
Terminal

6달 전

Python 프로그램 이름 변경



Python 프로그램 이름 변경



Number

항목	사용 예
정수	123, -345, 0
실수	123.45, -1234.5, 3.4e10
복소수	$1 + 2j, -3j$ <복소수 내장함수> .real(실수) .imag(허수) .conjugate(켈레복소수)
8진수	0o34, 0o25
16진수	0x2A, 0xFF
2진수	0b1010

Number 연산자

- 사칙연산
 - $+$ $-$ $*$ $/$
 - `print(7/3)` # 2.3333333333333335
- 거듭제곱
 - $**$
 - `print(3**4)` # 81
- 나머지
 - $%$
 - `print(7%3)` # 1
- 소수점 버리기(몫) – 정수 나누기
 - $//$
 - `print(7//3)` # 2

Number

항목	사용 예
정수	123, -345, 0
실수	123.45, -1234.5, 3.4e10
복소수	$1 + 2j$, $-3j$ <복소수 내장함수> .real(실수) .imag(허수) .conjugate(켈레복소수)
8진수	0o34, 0o25
16진수	0x2A, 0xFF
2진수	0b1010

Number 연산자(Operators)

- 산술연산자
 - + - * / % **(누승) //(몫)
- 비교연산자
 - == != < > <= >=
- 대입연산자
 - = += -= *= /= %= **= //=
- 비트연산자
 - & | ^ ~ << >>
- 논리연산자
 - and or not
- 멤버연산자
 - in not in
- 식별연산자
 - is is not
- ++, -- : 없음

Number 연산자 우선순위

Operator	Description
**	지수
~ + -	비트not, 부호(+), 부호(-)
* / % //	곱하기, 나누기, 나머지, 몫
+ -	덧셈과 뺄셈
>> <<	좌우 비트 시프트
&	비트 'AND'
^	비트 전용 'OR'와 정기적 인 'OR'
<= < > >=	비교 연산자
<> == !=	등가 연산자
= %= /= //= -= += *= **=	할당 연산자
is is not	식별 연산자
in not in	멤버 연산자
not or and	논리 연산자

자료 유형 - 숫자형

산술연산자

+ - * / % **(누승) //(몫)

비교연산자

== != < > <= >=

대입연산자

= += -= *= /= %= **= //=

비트연산자

& | ^ ~ << >>

논리연산자

and or not

멤버연산자

in not in

식별연산자

is is not

++, -- : 없음

자료 유형 - 숫자형

- 산술연산자
 - + - * / % **(누승) //(몫)
- 비교연산자
 - == != < > <= >=
- 대입연산자
 - = += -= *= /= %= **= //=
- 비트연산자
 - & | ^ ~ << >>
- 논리연산자
 - and or not
- 멤버연산자
 - in not in
- 식별연산자
 - is is not
- ++, -- : 없음

정수형 숫자

X = 15

```
print("출력 결과: {0}".format(X))
```

```
print("출력 결과: {0}".format(3/5))
```

```
print("출력 결과: {0}".format(X**3))
```

```
print("출력 결과: {0}".format(int(10.5)))
```

```
print("출력 결과: {0}".format(int(10.5)/int(8.5)))
```

```
출력 결과: 15  
출력 결과: 0.6  
출력 결과: 3375  
출력 결과: 10  
출력 결과: 1.25
```


실수형 숫자

```
X = 10.5*4.7  
print("출력 결과: {0}".format(X))  
print("출력 결과: {0:.1f}".format(X))  
print("출력 결과: {0:.3f}".format(X))
```

```
출력 결과: 49.35  
출력 결과: 49.4  
출력 결과: 49.350
```

Number 데이터 입력

```
x = int(input("정수 x = "))  
y = int(input("정수 y = "))
```

```
fx = float(input("실수 fx = "))  
fy = float(input("실수 fy = "))
```

```
정수 x = 7  
정수 y = 5  
실수 fx = 5.7  
실수 fy = 4.5
```

예제

```
x = int(input("정수 x = "))  
y = int(input("정수 y = "))  
  
print(x, "+", y, "=", x + y)  
print(x, "-", y, "=", x - y)  
print(x, "*", y, "=", x * y)  
print(x, "/", y, "=", x / y)
```

```
정수 x = 7  
정수 y = 5  
7 + 5 = 12  
7 - 5 = 2  
7 * 5 = 35  
7 / 5 = 1.4
```

주의

예제

```
x = int(input("정수 x = "))  
y = int(input("정수 y = "))  
  
print("%d + %d = %d" % (x, y, x+y))  
print("%d - %d = %d" % (x, y, x-y))  
print("%d * %d = %d" % (x, y, x*y))  
print("%d / %d = %d" % (x, y, x/y))
```

```
정수 x = 7  
정수 y = 5  
7 + 5 = 12  
7 - 5 = 2  
7 * 5 = 35  
7 / 5 = 1
```

주의

문제풀이

- 파이썬에서 Number 종류는?
- 파이썬 Number에서 제공하는 연산자는?

요약

- 파이썬에서 Number 종류를 공부하였다
- 파이썬 Number에서 제공하는 연산자를 공부하였다.

목 차

❖ Part 1

- Jupyter를 통한 프로그램 생성
- Python 언어의 기본기능

❖ Part 2

- **리스트**
- **복합리스트**
- **튜플**

❖ Part 3

- 딕셔너리
- 자료유형변환



날짜 date

```
from math import exp, log, sqrt
import re
from datetime import date, time,
datetime, timedelta
```

```
today = date.today()
print("today: {0!s}.format(today))
print("today: {0!s}.format(today.year))
print("today: {0!s}.format(today.month))
print("today: {0!s}.format(today.day))
```

today: 2020-09-13

예제

```
current_datetime = datetime.today()
print("current_datetime : {0!s}".format(current_datetime))
print("current_datetime : {0!s}".format(current_datetime.year))
print("current_datetime : {0!s}".format(current_datetime.month))
print("current_datetime : {0!s}".format(current_datetime.day))
print("current_datetime : {0!s}".format(current_datetime.hour))
print("current_datetime : {0!s}".format(current_datetime.minute))
print("current_datetime : {0!s}".format(current_datetime.second))
```

current_time : 2020-09-13 10:11:50.750583

```
print("current_datetime : {0!s}".format(current_datetime.strftime("%m/%d/%Y")))
print("current_datetime : {0!s}".format(current_datetime.strftime("%b %d, %Y")))
print("current_datetime : {0!s}".format(current_datetime.strftime("%B %d, %Y")))
```

리스트 (list)

- 리스트 만들기
- `a = []`
`b = [1, 2, 3]`
`c = ['Life', 'is', 'too', 'short']`
`d = [1, 2, 'Life', 'is']`
`e = [1, 2, ['Life', 'is']]`

리스트 예제

```
sum = 0
for data in [23, 45, 67, 43, 12]:
    sum += data
print("자료의 합 =", sum)
```

```
score = [23, 45, 67, 43, 12]
sum = 0
for data in reversed(score):
    sum += data
print("자료의 합 =", sum)
```

자료의 합 = 190

복합 리스트

- 복합 List 접근의 예
 - `e = [1, 2, ['Life', 'is']]`
`print(e[2][0])`

Life

2차원 리스트

```
mat1 = [  
    [1, 2, 3],  
    [4, 5, 6],  
    [7, 8, 9]  
]
```

```
mat2 = [ [0]*4 for i in range(3) ]
```

```
mat3 = []  
for i in range(3):  
    mat3.append([1] * 4)
```

```
print(mat1)  
print(mat2)  
print(mat3)
```

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]  
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]  
[[1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1]]
```

리스트 연산자

indexing

slicing

+ : 더하기

* : 곱하기

수정 & 삭제

(예)

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

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

```
print(a+b)
```

```
print(a*3)
```

```
a[2] = 7
```

```
print(a)
```

```
b[0:2] = []
```

```
print(b)
```

```
[1, 2, 3, 4, 5, 6]  
[1, 2, 3, 1, 2, 3, 1, 2, 3]  
[1, 2, 7]  
[6]
```

```
# 더하기
```

```
# 곱하기
```

```
# 수정
```

```
# 삭제 del b[0:2]
```


리스트 연산자

- `a = [1, 3, 2]`
`a.append(4)` # 단일요소 추가
`print(a)` # (1)
`a.extend([4, 8])` # 복수요소 추가
`a.sort()` # 정렬(순방향)
`print(a)` # (2)
`a.reverse()` # 정렬(역방향)
`print(a)` # (3)
`a.insert(2, 7)` # index 2에 7을 삽입
`print(a)` # (4)
`print(a.index(3))` # (5) 3의 index 알아내기
`print(a.count(4))` # (6) 4의 개수 알아내기

<code>[1, 3, 2, 4]</code>	(1)
<code>[1, 2, 3, 4, 4, 8]</code>	(2)
<code>[8, 4, 4, 3, 2, 1]</code>	(3)
<code>[8, 4, 7, 4, 3, 2, 1]</code>	(4)
<code>4</code>	(5)
<code>2</code>	(6)

리스트 연산자

- List 복사

- a = [1, 2, 3]

```
[1, 4, 3]  
[1, 4, 3]
```

- b = a #reference 복사

- a[1] = 4

- print(a)

- print(b)

- a = [1, 2, 3]

- b = a[:] #list 복사

- a[1] = 4

- print(a)

- print(b)

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

리스트 연산자

- List에서 “in”, “not in” 연산자
 - `a = [1, 2, 3]`
 - `if 2 in a:`
 - `print("2는 리스트 a에 있습니다.")`
 - `if 6 in a:`
 - `print("6은 리스트 a에 없습니다.")`

튜플(Tuple)

- List와 비슷함
- ()로 둘러싸고, 수정, 삭제가 불가능
- (예)
 - `t1 = ()`
 - `t2 = (1,)`
 - `t3 = (1,2,3)`
 - `t4 = 1,2,3`
 - `t5 = ('a', 'b', ('ab', 'cd'))`
- 연산
 - indexing
 - slicing
 - +(더하기) *(곱하기)

문제풀이

- 파이선에서 제공하는 날짜 유형은 ?
- 파이선에서 제공하는 리스트 유형은 언제 사용하는가 ?

요약

- 파이선에서 제공하는 다양한 자료 유형을 공부하였다.
 - 날짜
 - 리스트
 - 복합리스트

목 차

❖ Part 1

- Jupyter를 통한 프로그램 생성
- Python 언어의 기본기능

❖ Part 2

- 리스트
- 복합리스트
- 튜플

❖ Part 3

- **딕셔너리**
- **자료유형변환**



딕셔너리(Dictionary)

- Associative array, Hash의 개념
- key와 value의 쌍으로 이루어진 데이터
 - { }안에 데이터를 ,로 구분하여 열거
 - `dic = {'name': 'pey', 'birth': '1118'}`
`print(dic)`
`print(dic['name'])` # pey
 - `print(dic.get('name'))` # pey
- 주의사항
 - key의 중복 (X)
 - List를 key로 사용 (X)

딕션너리 연산자

- keys()
 - dict_keys 객체를 리턴
 - `dic = {'name': 'pey', 'birth': '1118'}`
`for i in dic.keys():`
 `print(i)` # key를 출력
 - 출력의 순서는 임의
 - dict_keys 객체를 List로 변환
 - `dic = {'name': 'pey', 'birth': '1118'}`
`lst = list(dic.keys())`
`for i in lst:`
 `print(i)`
- values()
 - dict_values 객체를 리턴

딕션너리 연산자

items()
dict_items 객체를 리턴
dic = {'name': 'pey', 'birth': '1118'}
for i in dic.items():
 print(i) # 각 item(키,값)을 출력

clear()
모두 지우기

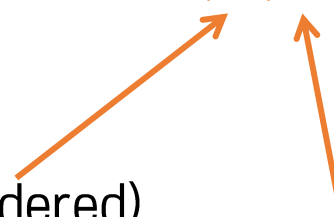
get()
key로 value 얻기
dic = {'name': 'pey', 'birth': '1118'}
print(dic.get('name'))

in
해당 key의 존재유무 확인
dic = {'name': 'pey', 'birth': '1118'}
print('name' in dic)

집합(Sets)

- 집합을 쉽게 처리하기 위한 것
- ```
s1 = set([1,2,3])
print(s1)
s2 = set("Hello")
print(s2)
```

# {'e', 'l', 'o', 'H'}


- 주의사항
  - 중복 불가
  - 순서 무시(unordered)
    - indexing 불가 (List나 Tuple로 변환 후에 가능)

## 집합(Sets)

- 교집합 : `&`, `intersection()`
- 합집합 : `|`, `union()`
- 차집합 : `-`, `difference()`
- (예)

```
s1 = set([1,2,3,4,5,6])
s2 = set([4,5,6,7,8,9])
print(s1 & s2)
print(s1.intersection(s2))
print(s1 | s2)
print(s1.union(s2))
print(s1 - s2)
print(s1.difference(s2))
```

```
{4, 5, 6}
{4, 5, 6}
{1, 2, 3, 4, 5, 6, 7, 8, 9}
{1, 2, 3, 4, 5, 6, 7, 8, 9}
{1, 2, 3}
{1, 2, 3}
```

- 추가 & 삭제 : `add()`, `update()`, `remove()`
  - `add()` : 한 개의 값만 추가
  - `update()` : 여러 개의 값을 추가

# True & False

| 값        | 참 or 거짓          |
|----------|------------------|
| "Python" | True             |
| " "      | False            |
| [1,2,3]  | True List        |
| []       | False List       |
| ()       | False Tuple      |
| {}       | False Dictionary |
| 1        | True             |
| 0        | False            |
| None     | False            |

## 자료 유형 변환

| Function                      | Description                                                        |
|-------------------------------|--------------------------------------------------------------------|
| <b>int</b> (x [,base])        | 정수로 변환                                                             |
| <b>long</b> (x [,base] )      | Long정수로 변환                                                         |
| <b>float</b> (x)              | 실수로 변환                                                             |
| <b>complex</b> (real [,imag]) | 복소수 생성                                                             |
| <b>str</b> (x)                | 객체 x를 문자열로 변환                                                      |
| <b>tuple</b> (s)              | Converts s to a tuple.                                             |
| <b>list</b> (s)               | Converts s to a list.                                              |
| <b>set</b> (s)                | Converts s to a set.                                               |
| <b>dict</b> (d)               | Creates a dictionary. d must be a sequence of (key ,value) tuples. |
| <b>chr</b> (x)                | Converts an integer to a character.                                |
| <b>unichr</b> (x)             | Converts an integer to a Unicode character.                        |
| <b>ord</b> (x)                | Converts a single character to its integer value( ascii).          |
| <b>hex</b> (x)                | Converts an integer to a hexadecimal string.                       |
| <b>oct</b> (x)                | Converts an integer to an octal string.                            |

# 변수

- 객체를 가리키는 것(reference)
- 변수 만들기
  - `(a, b) = ('python', 'life')`  
`print(a, b)`  
`[a,b] = ['python', 'life']`  
`print(a, b)`  
`a = b = 'python'`  
`print(a, b)`
- 변수값 교환
  - `a = 3`  
`b = 5`  
`a, b = b, a`      `# 교환`  
`print(a, b)`
- 변수 제거
  - `a = 3`  
`del(a)`



## 문제풀이

- 파이선에서 제공하는 딕셔너리 자료 유형을 언제 사용하나요?
- 파이선에서 자료 유형 변환은 언제 사용하는가?

## 요약

- 파이선에서 제공하는 다양한 자료 유형을 공부하였다.
  - 딕셔너리
  - 집합
- 파이선에서 제공하는 자료 유형의 변환 방법을 공부했다.