딕셔너리 자료형
(Dictionary)

파이썬 딕셔너리(Dictionary)

- 사람의 정보를 저장

각각 변수를 사용

```
name = "Chulsoo Kim"
age = 20
height = 182.5
weight =  75.9
nationality = "Korea"
mbti = "INTJ"
is_smoke = True
is_married = False
```

변수의 개수가
너무 많아 질 수 있음

리스트를 사용

```
person = ["Chulsoo Kim", 20, 182.5, 75.9,
"Korea", "INTJ", True, False]
```

리스트 내의 데이터가
어떤 의미인지 파악하기 어려움

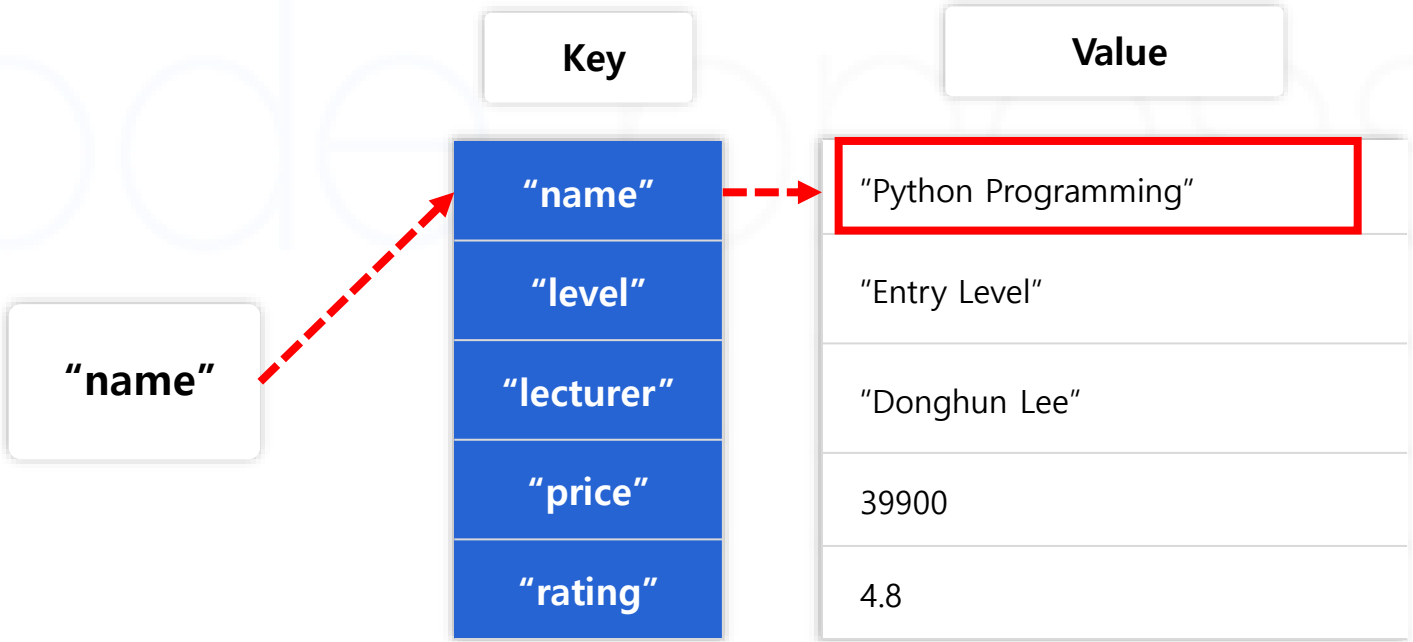
파이썬 딕셔너리(Dictionary)

- **사람의 정보를 딕셔너리로 저장**
  - 유사한 정보들을 그룹화 하여 저장/관리 가능
  - 각각의 데이터가 Key/Value 형태로 저장되어 있어서 어떤 종류의 데이터인지 파악 가능

```
person = {
  "name": "Chulsoo Kim",
  "age": 20,
  "height": 182.5,
  "weight":  75.9,
  "nationality": "Korea",
  "mbti": "INTJ",
  "is_smoke": True,
  "is_married": False
}
```

파이썬 딕셔너리(Dictionary)

- **딕셔너리는 Key와 Value를 같이 저장하고, Key로 Value를 가져오는 자료구조**
  - 영어 단어(Key)로 단어의 뜻(Value)을 찾을 수 있는 영어 사전과 유사
  - 파이썬 과정 정보를 딕셔너리 형태로 저장 한 예



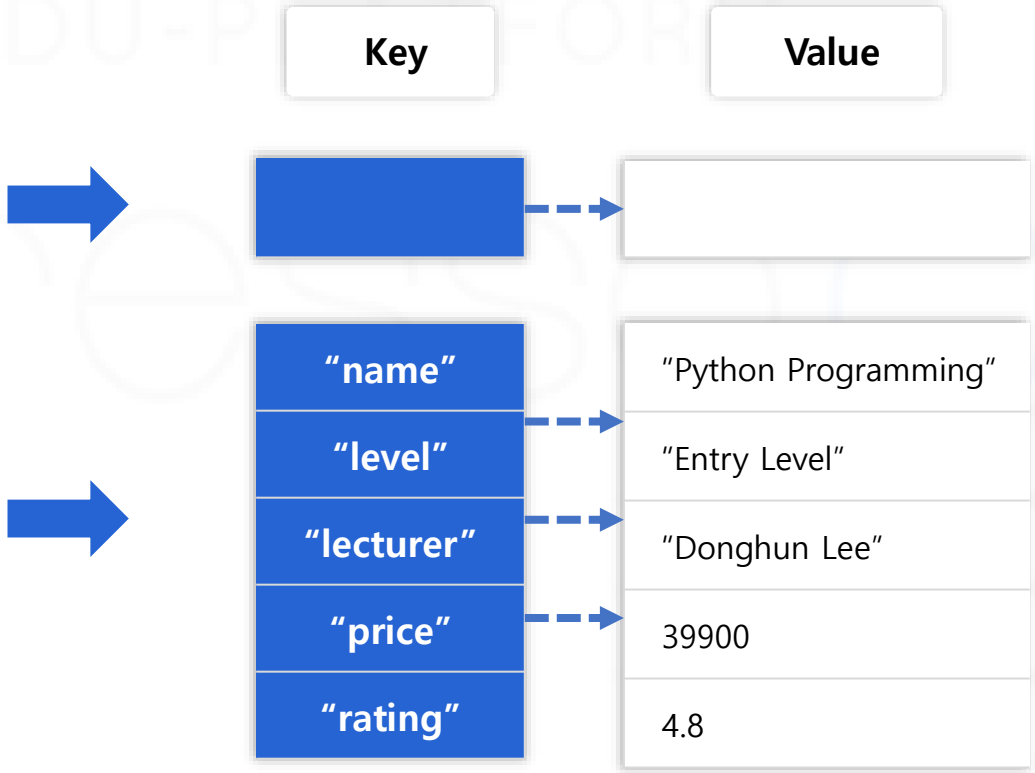
파이썬 딕셔너리의 생성

- **딕셔너리는 중괄호`{}`를 이용하여 생성**
  - 표기법 : { Key1:Value1, Key2:Value2}

Pair1 Pair2

```
course_dict = {}
```

```
python_course = {
  "name": "Python Programming",
  "level": "Entry Level",
  "lecturer": "Donghun Lee",
  "price": 39900,
  "rating": 4.8
}
```



파이썬 딕셔너리 자료형 확인

- **print(type())을 활용하여 자료형 확인**

```
python_course = {
   "name": "Python Programming",
   "level": "Entry Level",
   "lecturer": "Donghun Lee",
   "price": 39900,
   "rating": 4.8,
}

print(type(python_course))
```

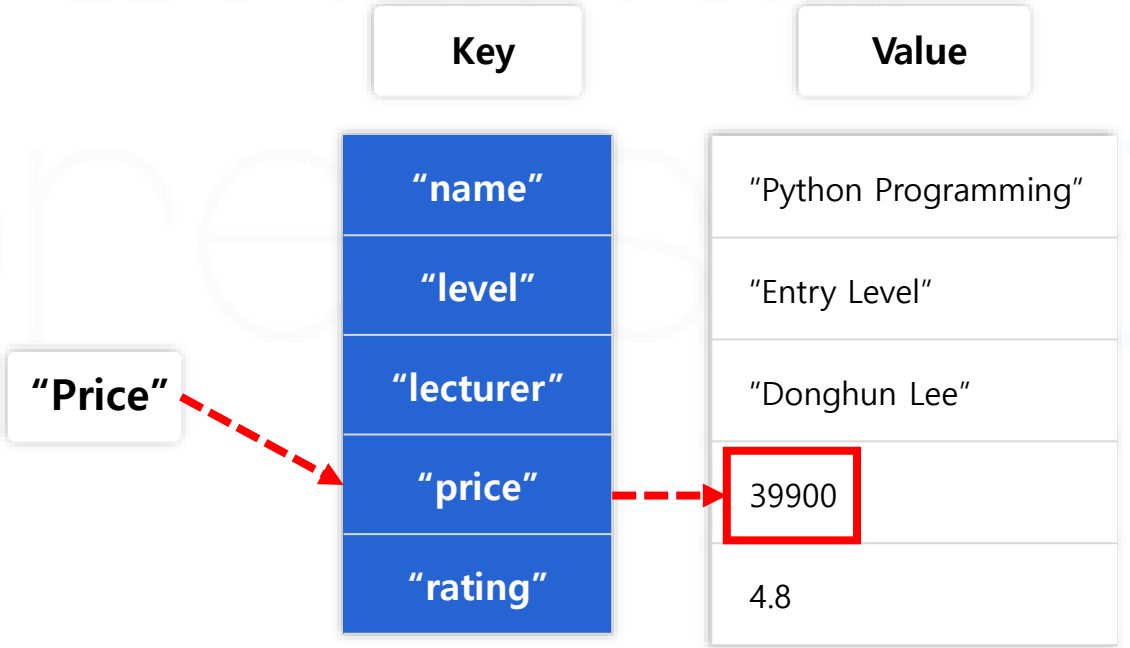
```
<class 'dict'>
```

딕셔너리 데이터 가져오기

- **딕셔너리의 Key를 이용하여 Value를 가져올 수 있음**
  - 문자열, 리스트은 인덱스를 활용, 딕셔너리는 Key 값을 활용
  - **딕셔너리변수명["Key"]**

```
python_course = {
  "name": "Python Programming",
  "level": "Entry Level",
  "lecturer": "Donghun Lee",
  "price": 39900,
  "rating": 4.8,
}

print(python_course["price"])
```



KeyError의 발생

- 딕셔너리에 존재하지 않는 Key를 이용해 접근 시 error 발생
  - key 가 존재 하지 않는다는 **KeyError** 발생

```
python_course = {
   "name": "Python Programming",
   "level": "Entry Level",
   "lecturer": "Donghun Lee",
   "price": 39900,
   "rating": 4.8,
}

print(python_course["image"])
```

```
KeyError: 'image'
```

"image"

Key 존재하지 않음

Key	Value
"name"	"Python Programming"
"level"	"Entry Level"
"lecturer"	"Donghun Lee"
"price"	39900
"rating"	4.8

파이썬 딕셔너리(Dictionary)

- 딕셔너리의 Value에 다양한 종류의 자료형 저장 가능

```
python_course = {
  "name": "Python Programming",
  "level": ["Entry Level", "Basic Level"],
  "lecturer": "Donghun Lee",
  "price": 39900,
  "rating": 4.8,
  "is_open": True,
  "including": {"video": 50, "exercise": 20}
}
```

Key	Value	
"name"	"Python Programming"	문자열
"level"	["Entry Level", "Basic Level"]	리스트
"lecturer"	"Donghun Lee"	문자열
"price"	39900	정수
"rating"	4.8	실수
"is_open"	True	불리안
"including"	{"video": 50, "exercise": 20}	딕셔너리

딕셔너리 Key의 중복

- 중복된 Key 입력하여 딕셔너리 생성 후 중복 된 Key 사용 하여 Value 접근

```
python_course = {
    "name": "Python Programming",
    "level": "Entry Level",
    "lecturer": "Donghun Lee",
    "name": "Python Introduction"
    "price": 39900,
    "rating": 4.8,
    "name": "Python Basic"
}

print(python_course["name"])
```

```
Python Basic
```

가장 마지막 Value를 가져옴

딕셔너리 관련 함수

딕셔너리의 전체 Key 값 가져오기 - keys() 함수

- **dict.keys() :**
  - key 값들을 dict_keys 자료형으로 반환

```
python_course = {
   "name": "Python Programming",
   "level": "Entry Level",
   "lecturer": "Donghun Lee",
   "price": 39900,
   "rating": 4.8,
}
dict_keys = python_course.keys()
print(type(dict_keys))
print(dict_keys)
```

Key	Value
"name"	"Python Programming"
"level"	"Entry Level"
"lecturer"	"Donghun Lee"
"price"	39900
"rating"	4.8

```
<class 'dict_keys'>
dict_keys(['name', 'level', 'lecturer', 'price', 'rating'])
```

딕셔너리의 전체 Key 값 가져오기 - keys() 함수

- **형 변환: 특정 자료형을 다른 자료형으로 변환하는 것**
- **list() 함수**
  - list(dict_kyes) 사용 시 dict_kyes 자료형의 데이터를 파이썬 리스트 자료형으로 형 변환

```
python_course = {
  "name": "Python Programming",
  "level": "Entry Level",
  "lecturer": "Donghun Lee",
  "price": 39900,
  "rating": 4.8
}

dict_keys = python_course.keys()
print(type(dict_keys))
print(dict_keys)
```

```
<class 'dict_keys'>
dict_keys(['name', 'level', 'lecturer', 'price', 'rating'])
```

```
key_list = list(dict_keys)
print(type(key_list))
print(key_list)
print(key_list[0])
```

```
<class 'list'>
['name', 'level', 'lecturer', 'price', 'rating']
name
```

딕셔너리의 전체 Value 값 가져오기 - values() 함수

- **dict.values() :**
  - value 값들을 dict_values 자료형으로 반환

```
python_course = {
  "name": "Python Programming",
  "level": "Entry Level",
  "lecturer": "Donghun Lee",
  "price": 39900,
  "rating": 4.8,
}
dict_values = python_course.values()
print(type(dict_values))
print(dict_values)
```

Key	Value
"name"	"Python Programming"
"level"	"Entry Level"
"lecturer"	"Donghun Lee"
"price"	39900
"rating"	4.8

```
<class 'dict_values'>
dict_values(['Python Programming', 'Entry Level', 'Donghun Lee', 39900, 4.8])
```

딕셔너리의 전체 Value 값 가져오기 - values() 함수

```
python_course = {
  "name": "Python Programming",
  "level": "Entry Level",
  "lecturer": "Donghun Lee",
  "price": 39900,
  "rating": 4.8,
}
dict_values = python_course.values()
print(type(dict_values))
print(dict_values)


value_list = list(dict_values)
print(type(value_list))
print(value_list[0])
```

```
<class 'dict_values'>
dict_values(['Python Programming', 'Entry Level',
```

```
<class 'list'>
Python Programming
```

딕셔너리의 전체 Key/Value 쌍 가져오기 - items() 함수

- **dict.items() :**
  - 전체 key/value의 쌍을 저장하고 있는 dict_items 자료형으로 반환

```
python_course = {
    "name": "Python Programming",
    "level": "Entry Level",
    "lecturer": "Donghun Lee",
    "price": 39900,
    "rating": 4.8,
}
dict_items = python_course.items()
print(type(dict_items))
print(dict_items)
```

Key	Value
"name"	"Python Programming"
"level"	"Entry Level"
"lecturer"	"Donghun Lee"
"price"	39900
"rating"	4.8

```
<class 'dict_items'>
dict_items([('name', 'Python Programming'), ('level', 'Entry Level'),
('lecturer', 'Donghun Lee'), ('price', 39900), ('rating', 4.8)])
```

딕셔너리의 전체 Key/Value 쌍 가져오기 - items() 함수

```
python_course = {
  "name": "Python Programming",
  "level": "Entry Level",
  "lecturer": "Donghun Lee",
  "price": 39900,
  "rating": 4.8,
}
dict_items = python_course.items()
print(type(dict_items))
print(dict_items)


item_list = list(dict_items)
print(type(item_list))
print(item_list[0])
```

```
<class 'dict_items'>
dict_items([('name', 'Python Programming'), ('level', 'Entry Level'),
('lecturer', 'Donghun Lee'), ('price', 39900), ('rating', 4.8)])
```

```
<class 'list'>
('name', 'Python Programming')
```

Key로 Value를 가져오기 - get() 함수

- **dict.get(key) :**
  - 딕셔너리 내 key 에 해당하는 value 반환

```
python_course = {
   "name": "Python Programming",
   "level": "Entry Level",
   "lecturer": "Donghun Lee",
   "price": 39900,
   "rating": 4.8,
}

print(python_course["name"])
print(python_course.get("name"))
```

```
Python Programming
Python Programming
```

Key로 Value를 가져오기 - get() 함수

```
python_course = {
   "name": "Python Programming",
   "level": "Entry Level",
   "lecturer": "Donghun Lee",
   "price": 39900,
   "rating": 4.8,
}

print(python_course["image"])
```

```
KeyError: 'image'
```

```
python_course = {
   "name": "Python Programming",
   "level": "Entry Level",
   "lecturer": "Donghun Lee",
   "price": 39900,
   "rating": 4.8,
}

print(python_course.get("image"))
```

```
None
```

None은 비어 있음, 존재하지 않음
을 의미하는 값
자료형은 **NoneType**

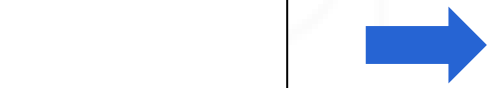
딕셔너리와 반복문

for문으로 딕셔너리의 Key 값 접근

- for문을 이용하여 딕셔너리 내 데이터에 순차적으로 접근
- for문에 딕셔너리의 이름을 사용하면 기본적으로 Key 값에 접근

```
python_course = {
   "name": "Python Programming",
   "level": "Entry Level",
   "lecturer": "Donghun Lee",
   "price": 39900,
   "rating": 4.8,
}

for key in python_course:
   print(key)
```



```
name
level
lecturer
price
rating
```

for문으로 딕셔너리의 Value 값 접근

- values() 함수를 이용하여 딕셔너리의 value에 반복적으로 접근

```
python_course = {
   "name": "Python Programming",
   "level": "Entry Level",
   "lecturer": "Donghun Lee",
   "price": 39900,
   "rating": 4.8,
}

for value in python_course.values():
   print(value)
```

```
Python Programming
Entry Level
Donghun Lee
39900
4.8
```

for문으로 딕셔너리의 Key/Value 쌍 접근

- **items() 함수를 이용하여 딕셔너리의 key/value 쌍에 반복적으로 접근**
  - in 기호 앞에 2개의 변수를 지정, 각각 Key와 Value를 저장

```
python_course = {
  "name": "Python Programming",
  "level": "Entry Level",
  "lecturer": "Donghun Lee",
  "price": 39900,
  "rating": 4.8,
}

for key, value in python_course.items():
  print(key, ":", value)
```

```
name : Python Programming
level : Entry Level
lecturer : Donghun Lee
price : 39900
rating : 4.8
```

딕셔너리와 리스트의 중첩

자료구조의 중첩

- **다양한 자료구조들이 서로 중첩 될 수 있음**
  - 자료구조 내에 다른 자료구조가 하나의 데이터로 저장 됨
  - 다양한 패턴으로 사용됨
    - 리스트 안의 딕셔너리
    - 딕셔너리 안의 리스트
    - 딕셔너리 안의 딕셔너리
    - 딕셔너리 안의 리스트 안의 딕셔너리

리스트와 딕셔너리의 중첩

- 리스트 자료구조 안에 딕셔너리가 저장 되어 있는 경우

```
course_list = [
   {"name": "Python", "level": "Entry"},
   {"name": "AI", "level": "Basic"},
   {"name": "Deep Learning", "level": "Intermediate"}
]

print(course_list[1])
print(course_list[1]["name"])
print(course_list[2]["level"])
```

```
{'name': 'AI', 'level': 'Basic'}
AI
Intermediate
```

딕셔너리와 다양한 자료구조의 중첩

- 딕셔너리 안에 또 다른 딕셔너리나 리스트가 저장되어 있는 경우

```
course = {
    "name": "Python",
    "level": ["Entry", "Basic"],
    "rating": 4.9,
    "including": {"video": 50, "exercise": 20}
}

print(course["level"])
print(course["level"][1])
print(course["including"])
print(course["including"]["video"])
```

```
['Entry', 'Basic']
Basic
{'video': 50, 'exercise': 20}
50
```