

파이썬 프로그래밍

7-1: 파이썬의 데이터형 (3)

2022.4.12



인하대학교
INHA UNIVERSITY

산업경영공학과 임세실

Overview

- 리스트 자세히 이해하기
- 튜플 이해하기
- 딕셔너리 이해하기



리스트 자세히 이해하기



리스트(list)란? (복습)

- 자료의 목록을 저장할 수 있는 데이터형
- 숫자, 문자열, bool 등 모두 저장 가능
- 대괄호 [] 내부에 개별 데이터 (요소; element)를 쉼표로 구분해 넣고 정의
 - 예: list1=[1, 2, 3, 4, 5]
list2=["고양이", "개", "토끼", "쥐"]
list3=[253, "안녕", 462, True, "밤"]



리스트 연산자: 인덱싱, 슬라이싱 (복습)

- list1=[1, 2, 3, 4, 5]
- 인덱싱: 특정 위치 요소 선택
- 슬라이싱: 특정 범위 요소 선택

	list1[-5]	list1[-4]	list1[-3]	list1[-2]	list1[-1]
	list1[0]	list1[1]	list1[2]	list1[3]	list1[4]
list1	1	2	3	4	5

```
list1 = [1,2,3,4,5]
print(list1[3])
print(list1[-3])
print(list1[2:4])
```

[1] ✓ 3.4s Python

... 4
3
[3, 4]

리스트 연산자: + 연산자와 * 연산자 (복습)

- + 연산자: 리스트 두 개의 자료를 합쳐 새로운 리스트 생성
- * 연산자: 리스트를 연산자 뒤의 숫자만큼 반복한 새로운 리스트 생성

```
list1 = [1,3,5,7,9]
list2=[2,4,6,8,10]
list3 = list1+list2
list4=list1*3
print(list3)
print(list4)
```

[2] ✓ 0.1s Python

... [1, 3, 5, 7, 9, 2, 4, 6, 8, 10]
[1, 3, 5, 7, 9, 1, 3, 5, 7, 9, 1, 3, 5, 7, 9]

리스트 연산자: in과 len() (복습)

- in: list 내부에 값 있으면 True, 없으면 False
- len(list): list 내부의 요소의 개수 세서 반환

```
list1=[1,3,5,7,9]
print(2 in list1)
print(len(list1))
```

[3] ✓ 0.1s Python

... False

5

리스트 내부에 특정 값의 위치 찾아 반환하기: index() (1)

- 리스트 안에 특정 값과 일치하는 데이터를 찾아 그 인덱스를 반환
 - 일치하는 값이 없으면 오류를 반환함
- **리스트이름.index(찾는값)**과 같은 식으로 사용함
 - list1.index(2)

리스트 내부에 특정 값의 위치 찾아 반환하기: index() (2)

```
list1=[1,2,3,4,5]
print(list1.index(2))
print(list1.index(7))
```

[15] 1.8s Python

1

```
-----
ValueError                                Traceback (most recent call last)
Untitled-1.ipynb Cell 9' in <cell line: 3>()
      1 list1=[1,2,3,4,5]
      2 print(list1.index(2))
----> 3 print(list1.index(7))

ValueError: 7 is not in list
```

리스트에 요소 하나 추가하기: append(), insert() (1)

- 리스트에 요소 추가할 때 사용
 - append: 리스트 뒤에 요소 추가
 - insert: 리스트 중간에 정해진 위치에 요소 추가
- **리스트이름.함수이름()**과 같은 식으로 사용함
 - list1.append(4)
 - list1.insert(3,5)

리스트에 요소 하나 추가하기: append(), insert() (2)

```
list1=[1,3,5,7,9]
print(list1)
list1.append(11)
print(list1)
list1.insert(2,4)
print(list1)
```

[7] ✓ 0.6s Python

... [1, 3, 5, 7, 9]
[1, 3, 5, 7, 9, 11]
[1, 3, 4, 5, 7, 9, 11]

리스트에 다른 리스트 이어붙이기: extend() (1)

- 리스트 뒤에 다른 리스트의 요소 이어붙일 때 사용
 - + 연산자와 비슷한 결과를 반환함
 - + 연산자는 원래 리스트의 내용에는 변화 없음
 - extend()는 원래 리스트의 내용을 변화시킴
- **리스트1이름.extend(리스트2이름)**과 같은 식으로 사용함

리스트에 다른 리스트 이어붙이기: extend() (2)

```
list1=[1,2,3,4,5]
list2=[6,7,8,9,10]
print(list1+list2)
print(list1)
list1.extend(list2)
print(list1)
```

[8] ✓ 0.9s Python

... [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

리스트에서 특정 위치의 요소 제거하기: del, pop() (1)

- 리스트에서 특정 위치에 있는 요소를 제거할 때 사용
 - del: 인덱싱 연산자와 함께 써서 해당 위치의 요소를 제거하는 연산자
 - pop: 인덱스를 매개변수로 받아 그 위치의 요소를 제거하는 함수
-> 매개변수 입력 안 하면 마지막 요소를 제거
- 사용예시
 - del: del list1[0]
 - pop: list1.pop(0)

리스트에서 특정 위치의 요소 제거하기: del, pop() (2)

```
list1=[1,2,3,4,5]
del list1[0]
print(list1)

list1=[1,2,3,4,5]
del list1[0:2]
print(list1)

list1=[1,2,3,4,5]
list1.pop(0)
print(list1)

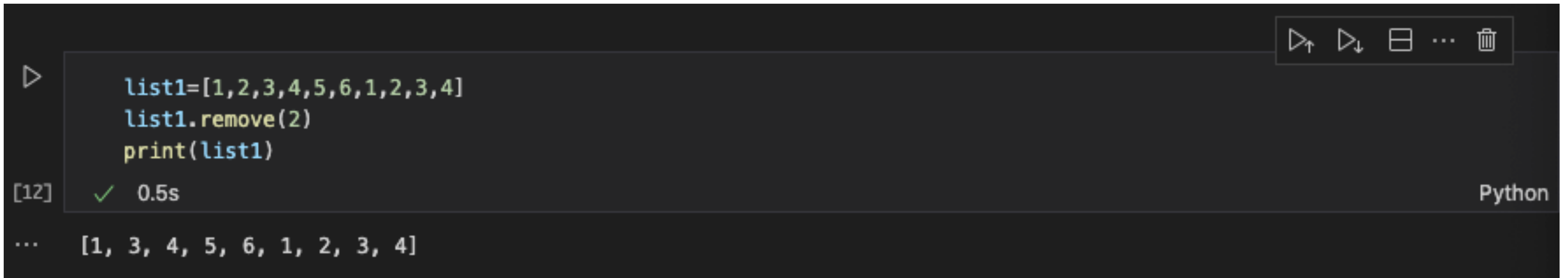
list1=[1,2,3,4,5]
list1.pop()
print(list1)
```

[11] ✓ 0.1s Python

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

리스트에서 특정 값의 요소 제거하기: remove()

- 리스트에서 특정 값을 가지는 요소를 제거할 때 사용
 - 여러 개 존재해도 가장 먼저 발견되는 값만 제거함

A screenshot of a Python REPL (Jupyter Notebook) interface. The code cell contains three lines: list1=[1,2,3,4,5,6,1,2,3,4], list1.remove(2), and print(list1). The output cell shows [12] followed by a green checkmark, 0.5s execution time, and the word 'Python'. Below the output, the resulting list is displayed: ... [1, 3, 4, 5, 6, 1, 2, 3, 4]. The interface includes a toolbar with navigation icons (back, forward, search, etc.) in the top right corner.

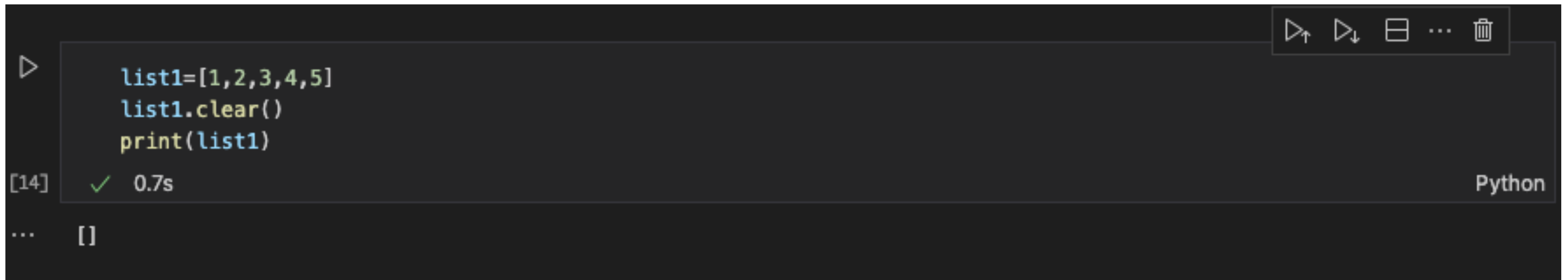
```
list1=[1,2,3,4,5,6,1,2,3,4]
list1.remove(2)
print(list1)
```

[12] ✓ 0.5s Python

... [1, 3, 4, 5, 6, 1, 2, 3, 4]

리스트 내부의 모든 요소 제거: clear()

- 리스트, 딕셔너리, 튜플 등 목록형 데이터의 모든 요소 제거

A screenshot of a Python IDE with a dark theme. The editor shows three lines of code: list1=[1,2,3,4,5], list1.clear(), and print(list1). Below the code, the output [14] is shown with a green checkmark and a 0.7s execution time. The Python logo is in the bottom right corner of the IDE window.

```
list1=[1,2,3,4,5]
list1.clear()
print(list1)
```

[14] ✓ 0.7s Python

... []

리스트 안의 요소를 정렬하기: sort()

- 리스트의 요소를 순서대로 정렬하기
- reverse=True 옵션 입력하면 내림차순, 아무것도 입력 안 하면 오름차순

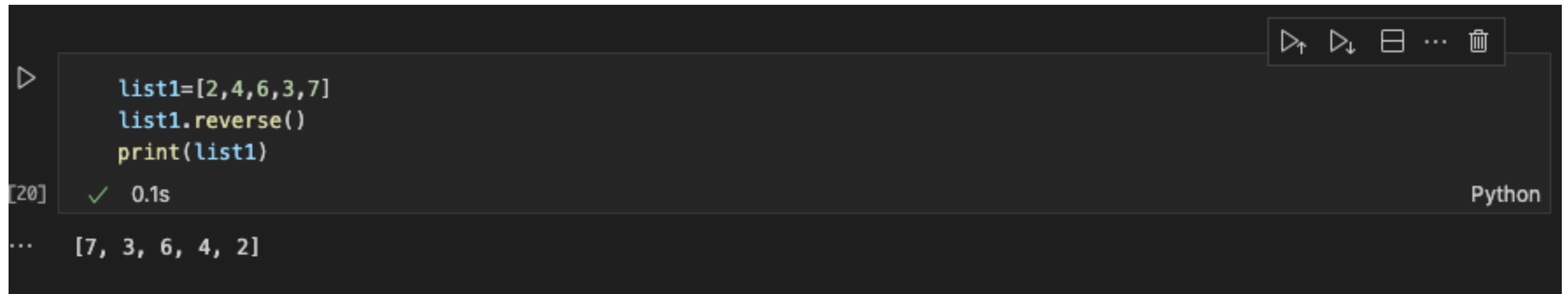
```
list1=[1,5,3,2,4]
list1.sort()
print(list1)
list1.sort(reverse=True)
print(list1)
list2=["a","c","e","d","b"]
list2.sort()
print(list2)
list2.sort(reverse=True)
print(list2)
```

[18] ✓ 0.6s Python

... [1, 2, 3, 4, 5]
[5, 4, 3, 2, 1]
['a', 'b', 'c', 'd', 'e']
['e', 'd', 'c', 'b', 'a']

리스트 내 요소의 순서를 반대로 정렬하기: reverse()

- 리스트 내 요소의 순서를 반대로 정렬하기

A screenshot of a Python code editor showing a list reversal operation. The code defines a list 'list1' with values [2, 4, 6, 3, 7], calls 'list1.reverse()' to reverse it, and then prints 'list1'. The output shows the reversed list [7, 3, 6, 4, 2]. The interface includes a toolbar with navigation icons and a status bar indicating the execution was successful in 0.1s.

```
list1=[2,4,6,3,7]
list1.reverse()
print(list1)
```

[20] ✓ 0.1s Python

... [7, 3, 6, 4, 2]

enumerate() 함수와 반복문의 조합 (1)

- 리스트 list1 = ['a','b','c','d']가 있을 때 다음과 같이 출력하고 싶다고 가정

```
0번째 요소는 a입니다  
1번째 요소는 b입니다  
2번째 요소는 c입니다  
3번째 요소는 d입니다
```

enumerate() 함수와 반복문의 조합 (2)

- 일반적인 해법:

```
list1=['a','b','c','d']

for i in range(len(list1)):
    print("{}번째 요소는 {}".format(i,list1[i]))
```

[52] ✓ 0.6s Python

... 0번째 요소는 a입니다
1번째 요소는 b입니다
2번째 요소는 c입니다
3번째 요소는 d입니다

enumerate() 함수와 반복문의 조합 (3)

- enumerate(): 리스트의 각 요소와 인덱스 쌍을 튜플로 묶은 순서열 객체를 반환

```
list1=['a','b','c','d']
print(enumerate(list1))
print(list(enumerate(list1)))

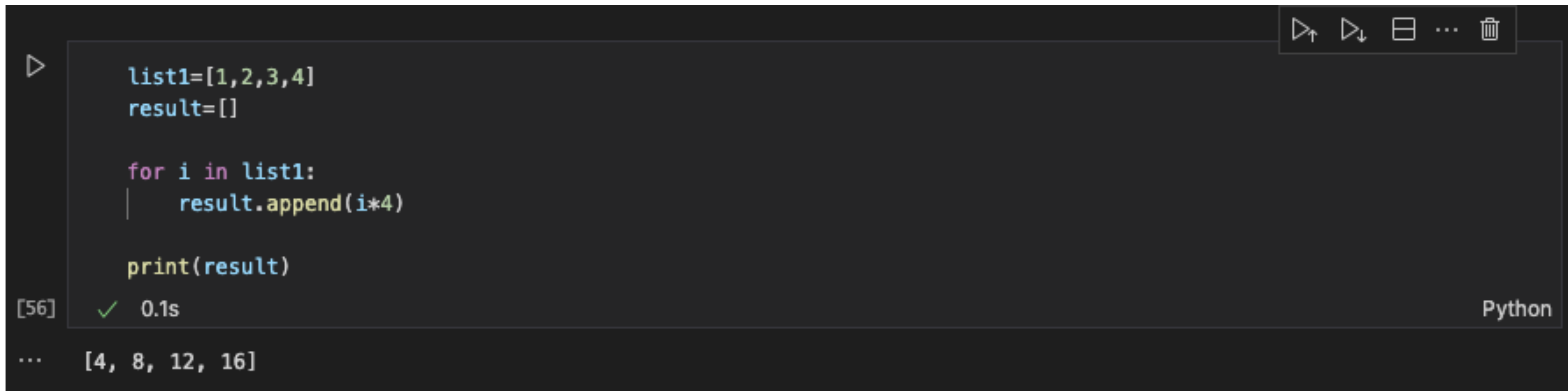
for i,value in enumerate(list1):
    print("{}번째 요소는 {}".format(i,value))
```

[54] ✓ 0.6s Python

... <enumerate object at 0x10ae1dd80>
[(0, 'a'), (1, 'b'), (2, 'c'), (3, 'd')]
0번째 요소는 a입니다
1번째 요소는 b입니다
2번째 요소는 c입니다
3번째 요소는 d입니다

리스트 내포 사용하기 (1)

- list1=[1,2,3,4]일 때, 각 요소에 4 곱해 새 리스트 만들기
 - 일반적인 방법



A screenshot of a Python IDE window. The code editor contains the following Python code:

```
list1=[1,2,3,4]
result=[]

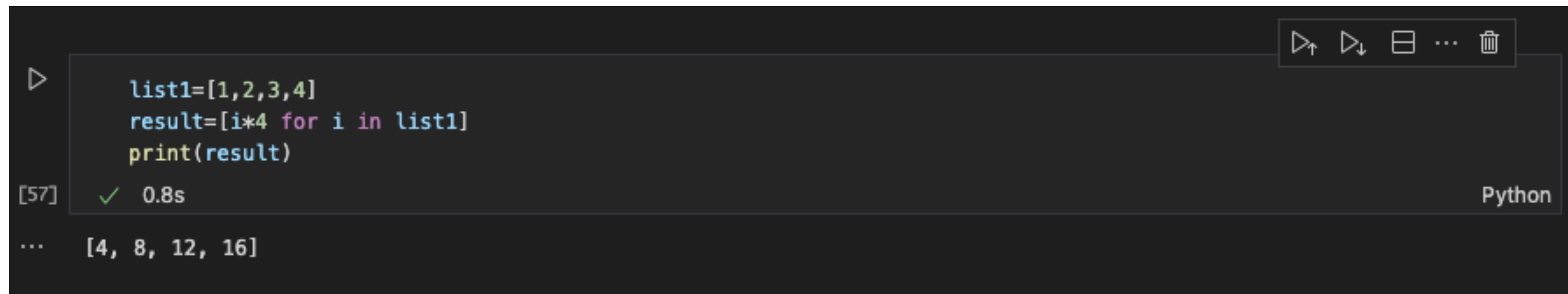
for i in list1:
    result.append(i*4)

print(result)
```

Below the code editor, the output of the execution is shown: [56] ✓ 0.1s Python [4, 8, 12, 16]. The IDE interface includes a toolbar with icons for running, stepping through, and other development tools.

리스트 내포 사용하기 (2)

- list1=[1,2,3,4]일 때, 각 요소에 4 곱해 새 리스트 만들기 (C'd)
- 새 리스트 이름=[표현식 for 반복변수 in 리스트]



A screenshot of a Python IDE interface. The code editor shows the following Python code:

```
list1=[1,2,3,4]
result=[i*4 for i in list1]
print(result)
```

Below the code, the output is displayed: `[4, 8, 12, 16]`. The IDE also shows a status bar with a green checkmark, the text `[57] 0.8s`, and the word `Python`. The interface includes standard IDE controls like a play button, a search icon, and a trash icon.

리스트 내포 사용하기 (3)

- list1=[1,2,3,4]일 때, 홀수에만 각 요소에 4 곱해 새 리스트에 담기
 - 새 리스트 이름=[표현식 for 반복변수 in 리스트 if 조건표현식]

```
list1=[1,2,3,4]
result=[i*4 for i in list1 if i%2!=0]
print(result)
```

[58] ✓ 0.8s Python

... [4, 12]

튜플 이해하기



튜플(Tuple)이란?

- 리스트와 마찬가지로 자료의 목록을 저장할 수 있는 데이터형
- 숫자, 문자열, bool 등 모두 저장 가능
- 괄호 () 내부에 개별 데이터 (요소; element)를 쉼표로 구분해 넣고 정의
 - 요소 하나만을 있을 때에도 반드시 쉼표를 붙여야 함
 - 괄호 생략가능
- 요소의 수정 불가



튜플 만들기

```
tuple1=(1,)
tuple2=("a","b")
tuple3=1,3,5
```

[23] ✓ 0.1s Python



튜플과 리스트의 차이

```
list1=[1,3,5,7,9]
list1[2]=6
print(list1)

tuple1=(1,3,5,7,9)
tuple1[2]=6
print(tuple1)
```

[24] 0.5s Python

... [1, 3, 6, 7, 9]

```
</>
-----
TypeError                                Traceback (most recent call last)
Untitled-1.ipynb Cell 13' in <cell line: 6>()
      3 print(list1)
      5 tuple1=(1,3,5,7,9)
----> 6 tuple1[2]=6
      7 print(tuple1)

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



튜플 연산자

- 인덱싱, 슬라이싱, +, * 연산자, len() 연산자

```
tuple1=1,3,5,7,9
print(tuple1[1])
print(tuple1[0:2])
tuple2=2,4,6,8,10
print(tuple1+tuple2)
print(tuple1*2)
len(tuple1)
```

[27] ✓ 0.1s Python

```
... 3
(1, 3)
(1, 3, 5, 7, 9, 2, 4, 6, 8, 10)
(1, 3, 5, 7, 9, 1, 3, 5, 7, 9)

5
```



튜플 함수

- `index()`: 튜플 안에 특정 값과 일치하는 데이터를 찾아 그 인덱스를 반환

```
tuple1 = 1,3,5,7,9
print(tuple1.index(5))
print(tuple1.index(6))
```

[28] 0.7s Python

... 2

```
</> -----
ValueError                                Traceback (most recent call last)
Untitled-1.ipynb Cell 15' in <cell line: 3>()
      1 tuple1 = 1,3,5,7,9
      2 print(tuple1.index(5))
----> 3 print(tuple1.index(6))

ValueError: tuple.index(x): x not in tuple
```

for 반복문과 튜플 함께 사용하기

- 튜플 내 모든 요소 출력하기

```
tuple1 = 3,6,8,8,9,3,2,1
for i in tuple1:
    print(i)
```

[29] ✓ 0.9s Python

... 3
6
8
8
9
3
2
1



딕셔너리 이해하기



인하대학교
INHA UNIVERSITY

산업경영공학과 임세실

딕셔너리(Dictionary)이란?

- 자료 저장 시 키 (key)에 대응해서 값 (value)이 한쌍으로 저장
 - 예: 이름 (key) = 임세실 (value), 학과 (key) = 산업경영공학과 (value)
- 괄호 {} 내부에 [key]:[value] 형태를 쉼표로 구분해 넣고 정의
 - key는 문자열, 숫자, 불 등으로 선언 가능하지만 일반적으로 문자열 사용



딕셔너리 만들기

```
dic1 = {"이름": "임세실", "학과": "산업경영공학과", "사무실": "2N487"}
dic2 = {"브랜드": ["애플", "삼성", "hp"]}
```

[32] ✓ 0.1s Python

dic1

Key	Value
이름	임세실
학과	산업경영공학과
사무실	2N487



딕셔너리 사용하기 (1)

- key 사용해서 특정 위치의 값 접근하기

```
dic1 = {"이름": "임세실", "학과": "산업경영공학과", "사무실": "2N487"}
print(dic1["이름"])
dic1["이름"]="나모름"
print(dic1)
```

[35] ✓ 0.4s Python

... 임세실
{ '이름': '나모름', '학과': '산업경영공학과', '사무실': '2N487' }

dic1

Key	Value
이름	임세실
학과	산업경영공학과
사무실	2N487



딕셔너리 사용하기 (2)

- key 사용해서 특정 위치의 값 접근하기

```
dic2 = {"브랜드": ["애플", "삼성", "hp"], "가격": [240, 230, 100]}
print(dic2["가격"])
print(dic2["가격"][1])
dic2["가격"][2]=202
print(dic2)
```

[36] ✓ 0.9s Python

... [240, 230, 100]
230
{'브랜드': ['애플', '삼성', 'hp'], '가격': [240, 230, 202]}



딕셔너리에 값 추가하기

- 새로운 key와 함께 값을 추가

```
dic1 = {"이름": "임세실", "학과": "산업경영공학과", "사무실": "2N487"}
dic1["전화번호"] = "0328607362"
print(dic1)
```

[37] ✓ 0.1s Python

... {'이름': '임세실', '학과': '산업경영공학과', '사무실': '2N487', '전화번호': '0328607362'}

+ Code + Markdown



딕셔너리에 중복된 key 사용할 경우

- 제일 나중에 입력되는 값 이외의 나머지 값들은 무시

```
dic1 = {"이름": "임세실", "학과": "산업경영공학과", "이름": "2N487"}
print(dic1)
```

[38] ✓ 0.5s Python

... {'이름': '2N487', '학과': '산업경영공학과'}



딕셔너리에 값 제거하기

- del 연산자 사용

```
dic1 = {"이름": "임세실", "학과": "산업경영공학과", "사무실": "2N487"}
del dic1["학과"]
print(dic1)
```

[39] ✓ 0.5s Python

... {'이름': '임세실', '사무실': '2N487'}



딕셔너리에 존재하는 key와 value 목록 만들기 (1)

- `dic1.keys()`: d와c1 딕셔너리의 key만 모아서 `dict_keys` 객체로 반환
- `dic1.values()`: dic1 딕셔너리의 value만 모아서 `dict_values` 객체로 반환
- `dic1.items()`: dic1 딕셔너리의 key-value 쌍을 튜플로 묶어서 `dict_items` 객체로 반환
- 각각의 결과를 리스트로 반환하려면 명령어를 `list()`로 감싸서 객체를 리스트로 변환



딕셔너리에 존재하는 key와 value 목록 만들기 (2)

```
dic1={"이름":"임세실", "학과":"산업경영공학과", "사무실":"2N487"}
print(dic1.keys())
print(dic1.values())
print(dic1.items())
print(list(dic1.keys()))
```

[41] ✓ 0.1s Python

```
... dict_keys(['이름', '학과', '사무실'])
dict_values(['임세실', '산업경영공학과', '2N487'])
dict_items([('이름', '임세실'), ('학과', '산업경영공학과'), ('사무실', '2N487')])
['이름', '학과', '사무실']
```



딕셔너리에서 key가 존재하는지 확인

- in 연산자: 딕셔너리 내부에 key 있으면 True 반환, 없으면 False 반환
- get 함수: 딕셔너리 내부에 key 있으면 값 반환, 없으면 None 반환

```
dic1={"이름":"임세실", "학과":"산업경영공학과", "사무실":"2N487"}
print("학과" in dic1)
print("전화번호" in dic1)
print(dic1.get("학과"))
print(dic1.get("전화번호"))
```

[42] ✓ 0.5s Python

... True
False
산업경영공학과
None



for 반복문과 딕셔너리 함께 사용하기

- 딕셔너리 내 모든 값 출력하기

```
dic1={"이름":"임세실", "학과":"산업경영공학과", "사무실":"2N487"}
for key in dic1:
    print(dic1[key])
```

[45] ✓ 0.1s Python

... 임세실
산업경영공학과
2N487



실습 (4/14)

- 강의실에서 출석 확인 예정
 - 온라인 출석확인 대상자는 과제 제출 여부로 출석 확인
- 실습과제는 실습시간 정시에 iClass에 업로드될 예정
- 4월 15일 자정까지 제출해야 과제 점수 인정

