

파이썬(python)

- 1991년 출시 / 반 로섬
- 특징

1. Interpreter(인터프리터) 방식 : source -> 인터프리터 -> 한 줄씩 처리(대화형) / 실행 속도가 느리다는 단점 but 소스파일만 있으면 되기 때문에 다른 OS에서도 돌아감

컴파일러는 source -> compiler -> .exe / 하지만 다른 OS에선 돌아가지 않는다는 단점

2. Object Oriented(객체지향)

3. Dynamic Typed : 타입 선언 필요 없음(타입이 runtime 때 결정) / a = "String"

컴파일러는 타입 선언이 필요 / int a;

설치

Python 3.7.8

Release Date: June 27, 2020

Note

Python 3.8 is now the latest feature release series of Python 3. [Get the latest release of 3.8.x here.](#)

Python 3.7.8 is the latest *bugfix* release of Python 3.7. 3.7.8 is also expected to be the last bugfix release before 3.7 enters the *security-fix* phase of its life cycle. We plan to provide *security fixes* until mid 2023, five years after its initial release.

Please see the [Full Changelog](#) link for more information about the contents of this release and see [What's New In Python 3.7](#) for more information about 3.7 features.

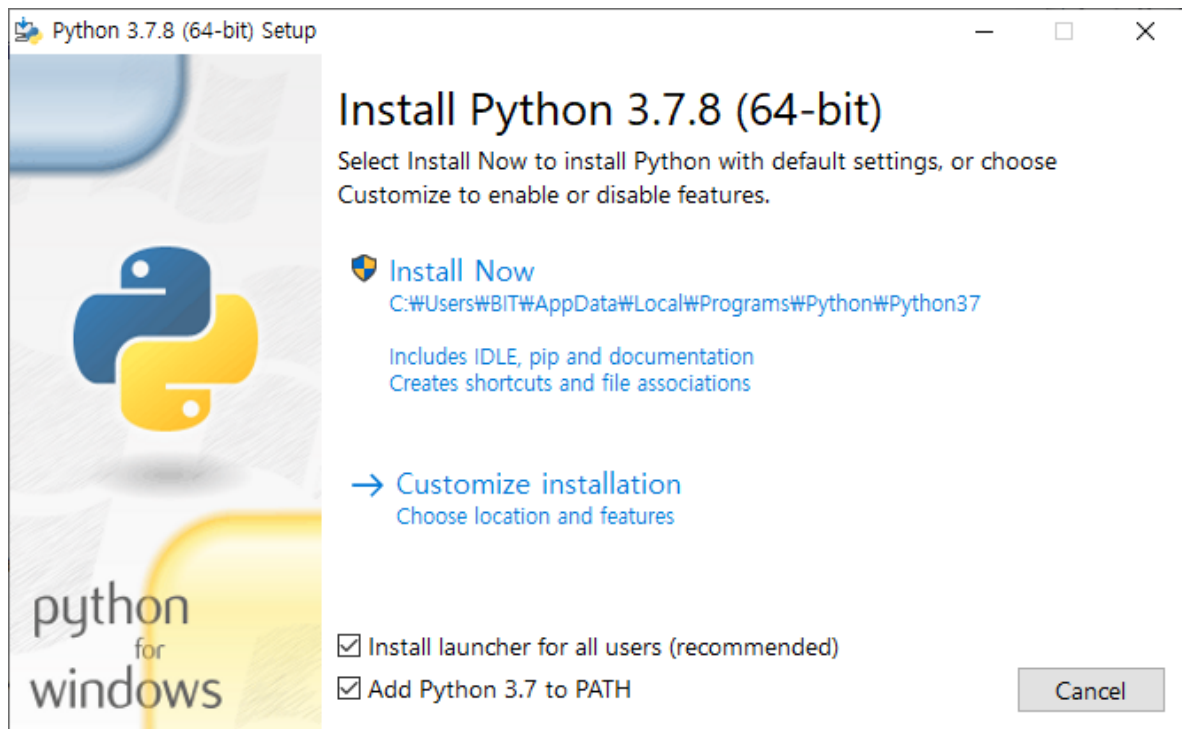
More resources

- [Online Documentation](#)
- [PEP 537, 3.7 Release Schedule](#)
- Report bugs at <https://bugs.python.org>.
- [Help fund Python and its community.](#)

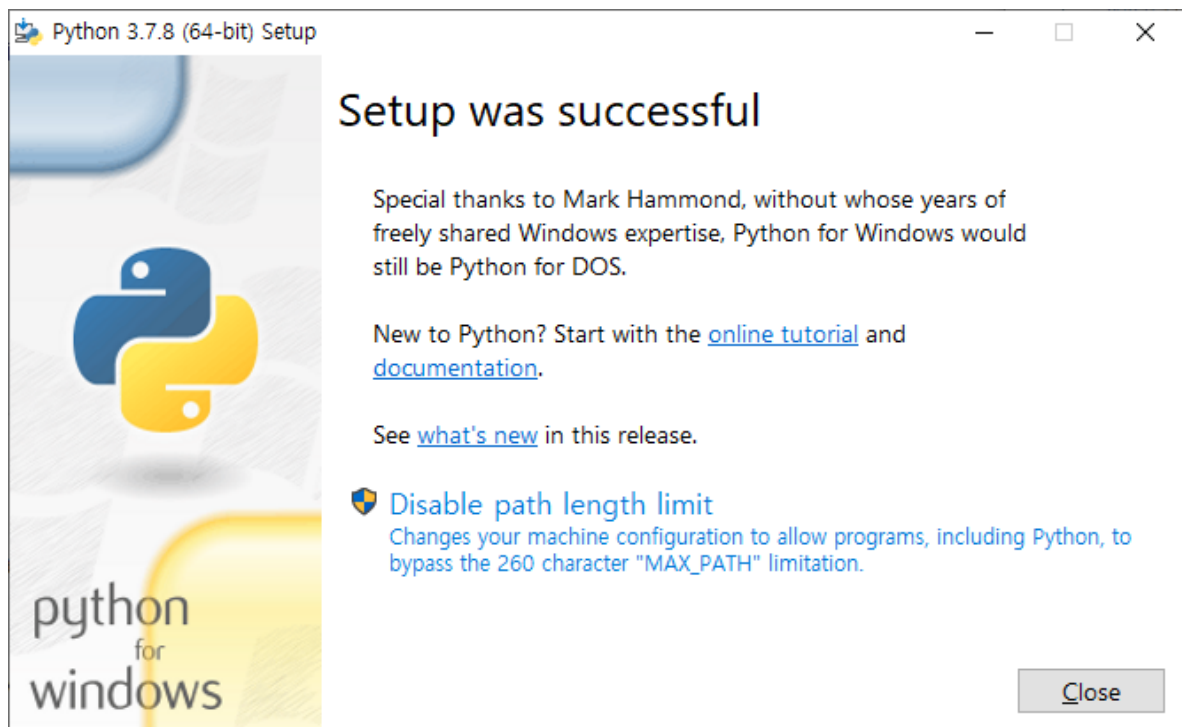
Windows users

- The binaries for AMD64 will also work on processors that implement the Intel 64 architecture. (Also known as the "x64" architecture, and formerly known as both "EM64T" and "x86-64".)

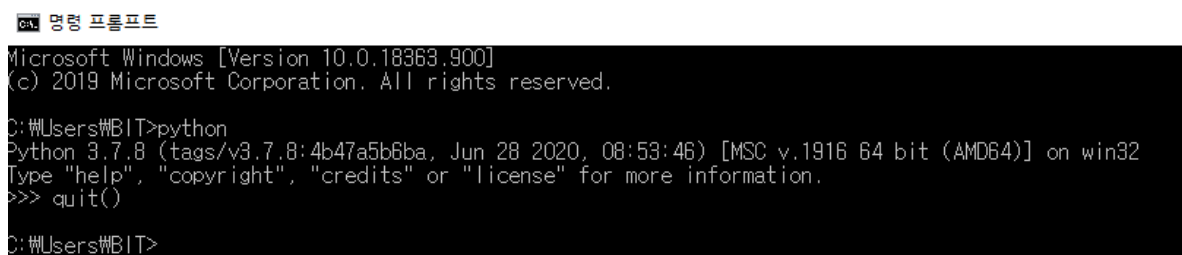
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	70b08ab8e75941da7f5bf2b9be58b945	26993432	SIG
---	---------	---------------------	----------------------------------	----------	---------------------



기본 설치 경로로 Install



Disable ~ 선택(OS 자체에서 긴 경로명을 막기 때문)



정상 설치 모습

```

C:\Users\WBIT>d:
D:\>cd khy
D:\KHY>cd Oracle_Ex
D:\KHY\Oracle_Ex>cd khy
지정된 경로를 찾을 수 없습니다.
D:\KHY\Oracle_Ex>quit
'quit'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.
D:\KHY\Oracle_Ex>cd ..
D:\KHY>work
'work'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.
D:\KHY>cd work
D:\KHY\work>python
Python 3.7.8 (tags/v3.7.8:4b47a5b6ba, Jun 28 2020, 08:53:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>

```

사용자의 work 폴더에서 작업

실습

```

D:\KHY\work>python
Python 3.7.8 (tags/v3.7.8:4b47a5b6ba, Jun 28 2020, 08:53:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> book = "The hunt for red October"
>>> book
'The hunt for red October'
>>> print(book)
The hunt for red October
>>>

```

```

>>> a = 7
>>> b = 5
>>> print(a + b)
12
>>>

```

a = 7이라는 리터럴 객체가 생성된다고 보면 됨

```

>>> print("a" + "b")
ab
>>> a
7
>>>

```

```

>>> del a
>>> a
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'a' is not defined
>>>

```

del로 a 정보가 삭제된 모습

Pythonic Code

- 파이썬 핵심 철학

"아름다운게 추한 것보다 낫다." (Beautiful is better than ugly)
 "명시적인 것이 암시적인 것 보다 낫다." (Explicit is better than implicit)
 "단순함이 복잡함보다 낫다." (Simple is better than complex)
 "복잡함이 난해한 것보다 낫다." (Complex is better than complicated)
 "가독성은 중요하다." (Readability counts)

- number_of_cylinder: 파이썬식 변수 선언

numberOfCylinder : 자바식 변수 선언

- 파이썬은 대소문자 구분
- 문자열 선언 방법

1. book = "~~~~"(더블 쿼테이션) : 내용 자체에서 어퍼스트로피를 쓸 경우 더블로
2. book = '~~~~'(싱글 쿼테이션) : 내용 자체에서 말하는 문장을 쓸 경우 싱글로
3. book = """~~~~""" : 여러 줄을 한 개의 라인으로 처리할 때

```
>>> book = 'My name is Tom'
>>> book
'My name is Tom'
>>> book = "My name is Tom"
>>> book
'My name is Tom'
>>> book = """asdfasdf
... asdfasdf
... asdfasdf"""
>>> book
'asdfasdf\nasdfasdf\nasdfasdf'
>>>
```

- boolean 선언 방법

```
>>> bool_type = True
>>> bool_type
True
>>>
```

True / False 와 같이 첫 글자 대문자

```
>>> True == 1
True
>>> True == 0
False
>>>
```

1과 0으로도 bool 판별

- 사칙연산

```
>>> print(2 ** 3)
8
>>> print(7 / 2)
3.5
>>> print(7 // 2)
3
>>> print(7 % 2)
1
>>>
```

제곱 / 나누기 / 몫 / 나머지

- 모듈러 연산자(나머지 연산자)

```
>>> print(-1 % 5)
4
>>>
```

A % B = (A + B) % B 으로 내부에서 처리하기 때문에 4가 출력 됨

```
>>> print(1 % -5)
-4
>>>
```

따라서 음수값 또한 출력 가능

- += / -= / *= / /= 연산

```
>>> a = 1
>>> a = a + 1
>>> print(a)
2
>>> a += 1
>>> print(a)
3
>>>
```

- 형변환

```
>>> a = 10
>>> b = float(a)
>>> print(a, b)
10 10.0
>>>
```

```
>>> c = int(b)
>>> print(c)
10
>>>
```

```
>>> a = "76.3"
>>> b = float(a)
>>> print(a, b)
76.3 76.3
>>>
```

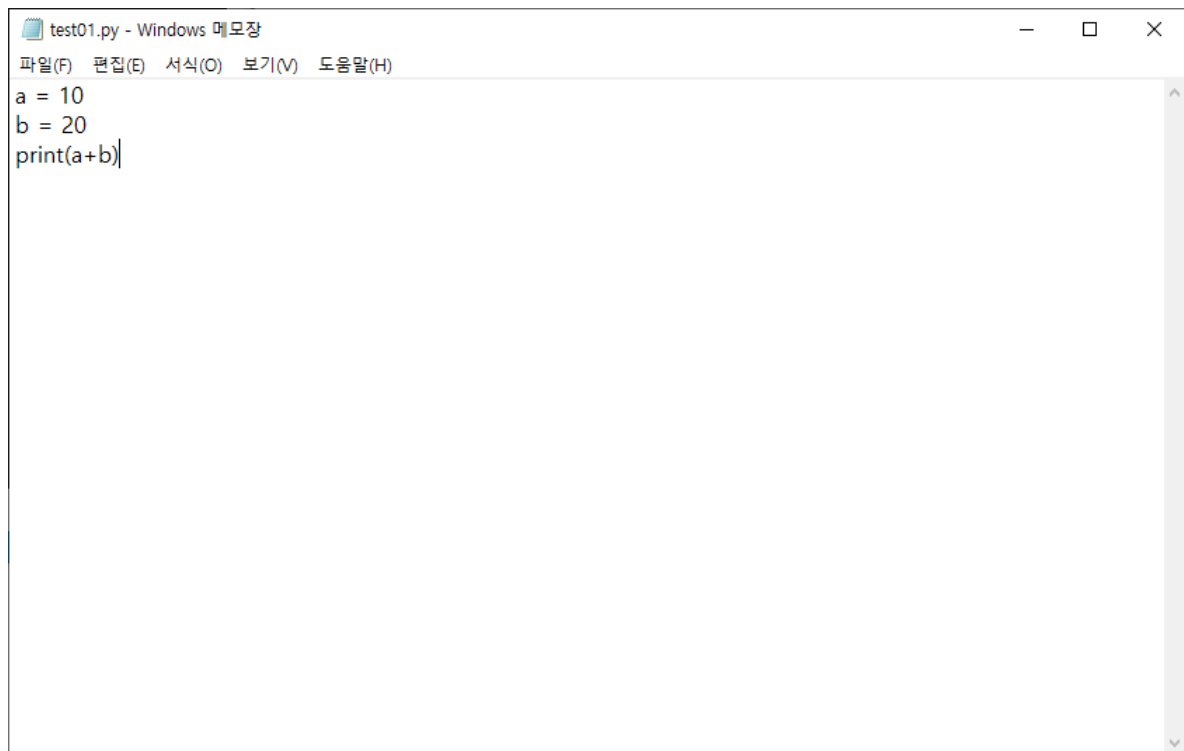
```
>>> a = 10.5
>>> b = 21.8
>>> c = str(a)
>>> d = str(b)
>>> print(a + b, c + d)
32.3 10.521.8
>>>
```

숫자는 더해서, 문자열은 붙어서 출력

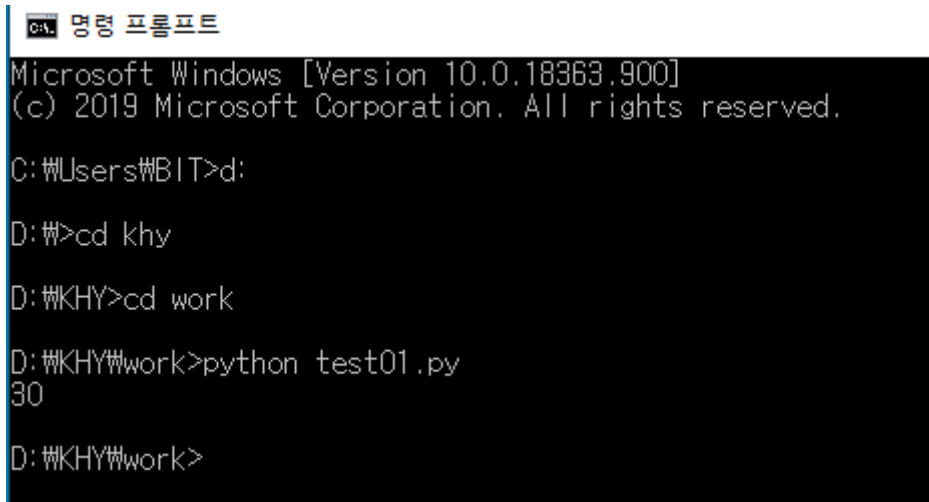
- 타입 확인

```
>>> a = 10.3
>>> type(a)
<class 'float'>
>>>
```

```
>>> b = "23"
>>> type(b)
<class 'str'>
>>>
```



```
test01.py - Windows 메모장
파일(F)  편집(E)  서식(O)  보기(V)  도움말(H)
a = 10
b = 20
print(a+b)|
```



```
C:\> 명령 프롬프트
Microsoft Windows [Version 10.0.18363.900]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\BIT>d:
D:\>cd khy
D:\KHY>cd work
D:\KHY\work>python test01.py
30
D:\KHY\work>
```

파일로 실행할 땐 파이썬셸이 아니기 때문에 print로 출력을 해줘야함

따라서 IDE를 사용

IDE (Integrated Development Enviroment: 통합개발 환경)

- 설치

20

years

[도구](#)
[언어](#)
[솔루션](#)
[지원](#)
[회사](#)
[스토어](#)

PyCharm

[새로운 기능](#)
[기능](#)
[구매](#)

다운로드

버전: 2020.1.2

빌드: 201.7846.77

2020년 6월 3일

시스템 요구 사항

다운로드 PyCharm

[Windows](#)
[Mac](#)
[Linux](#)

Professional

과학 및 웹 Python 개발용. HTML, JS, SQL 지원.

다운로드

무료 평가판

Community

순수 Python 개발용

다운로드

무료, 오픈 소스

Feedback

커뮤니티 버전 다운로드

인스톨러 실행

옵션들 전체 체크(확실하지 않음)

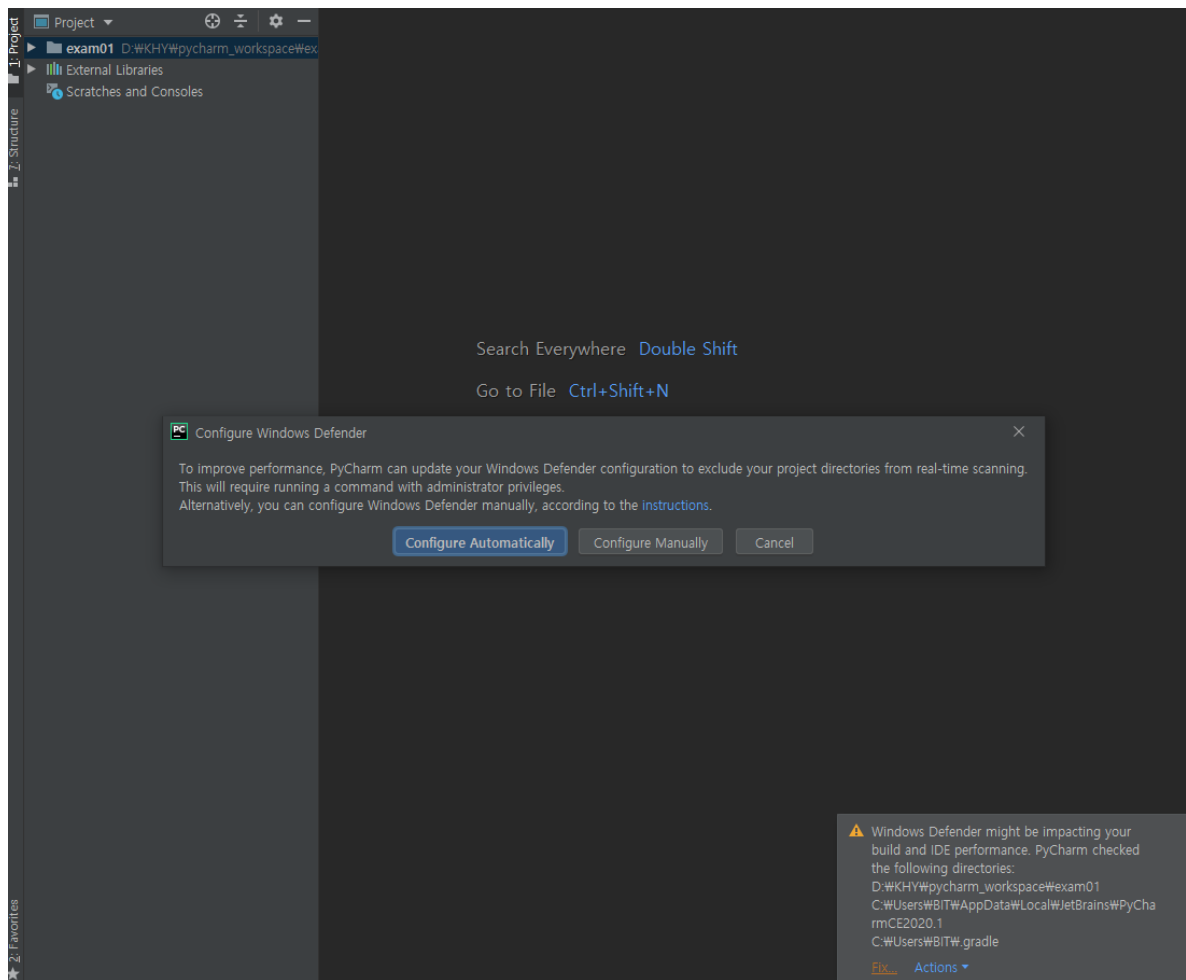
- 파일 경로 설정

The screenshot shows the PyCharm IDE interface with the 'Create Project' dialog open. The dialog is configured for a new Virtualenv environment. The location is set to 'D:\KHY\pycharm_workspace\exam01'. The base interpreter is 'C:\Users\BIT\AppData\Local\Programs\Python\Python37\python.exe'. The 'Make available to all projects' checkbox is checked. A warning message from Windows Defender is visible in the bottom right corner.

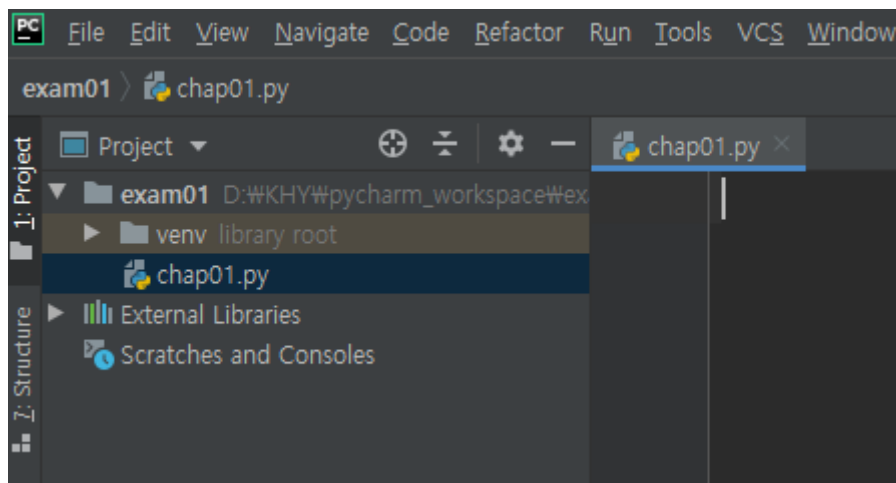
new Environment를 버추얼~ 로 선택

Base Interpreter를 파이썬(경로 또는 이름을 되어있음)으로 선택

create 선택 -> This window 선택

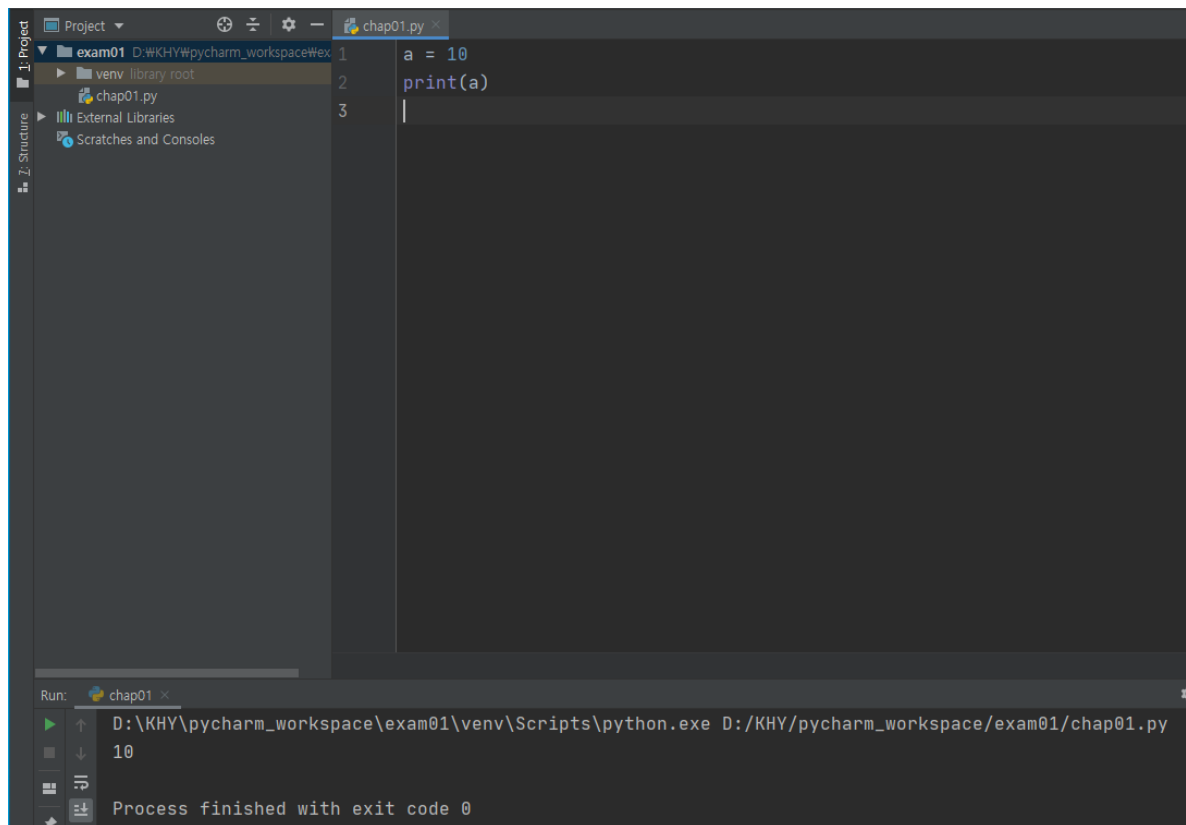


디펜더 픽스 -> 오토메티컬리



프로젝트 파일 -> new -> 파일 -> 파일명.py(확장명 안 하면 선택하라고 창이 뜬)

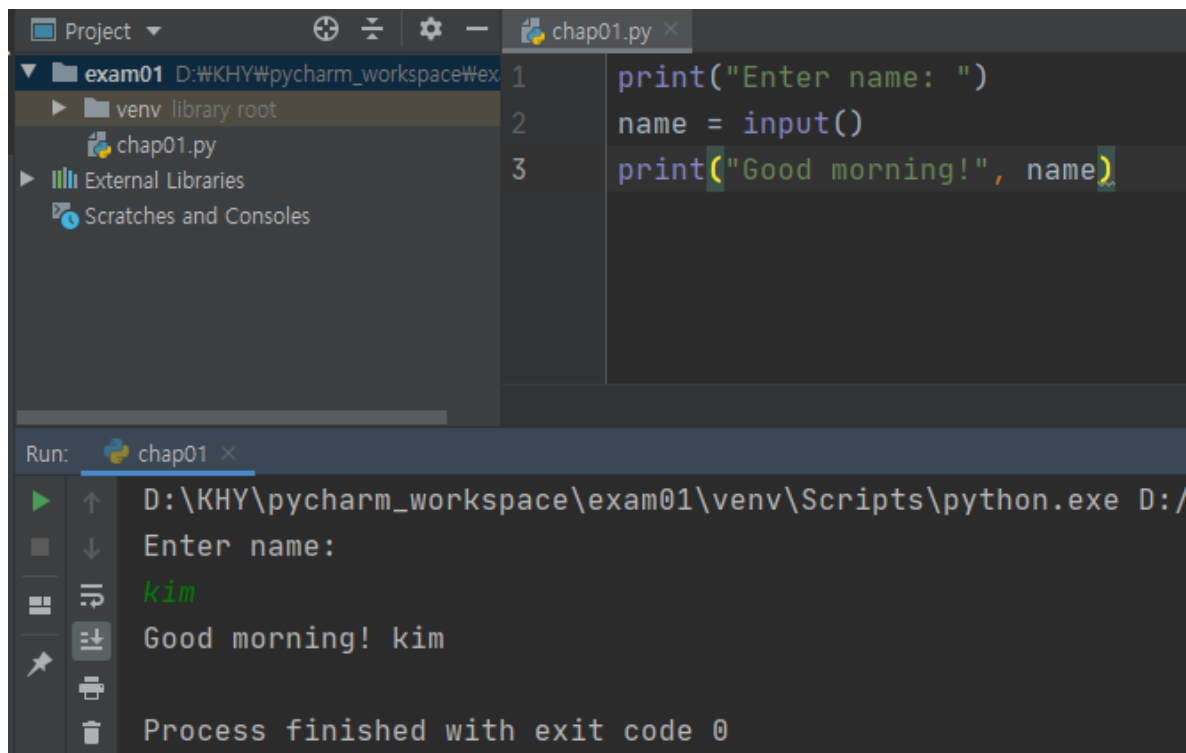
- 실행



처음엔 직접 Run (알트+시프트+F10)

이후엔 시프트+F10으로 가능

input()



콤마(,)로 출력할 콘텐츠들을 나열할 때 자동적으로 한 칸을 띄어줌

주석

한 줄 주석 : #

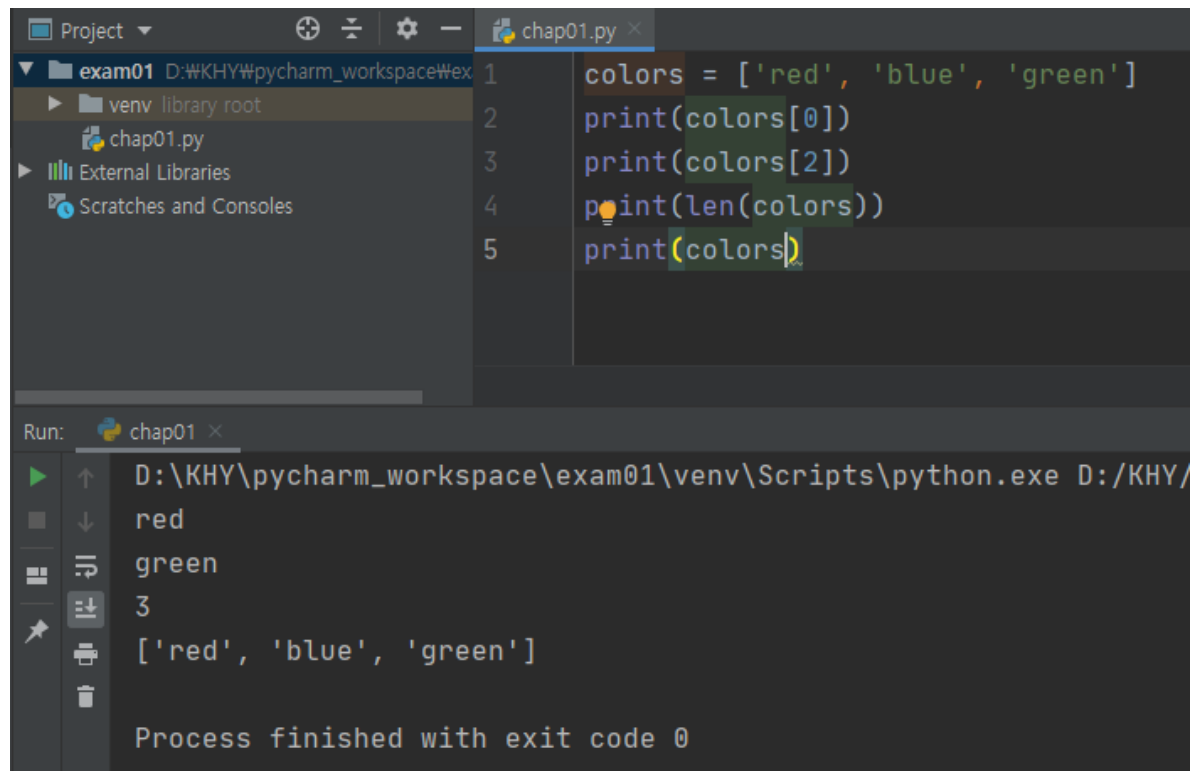
```
print(colors[2]) # asdfasdf
```

주석 전 띄어쓰기 최소 2칸, 주석 후 띄어쓰기 1칸

여러 줄 주석 : """ ... """

```
"""sdfasdfasdf"""  
colors = ['red', 'blue', 'green']
```

리스트



The screenshot shows the PyCharm IDE interface. The top pane displays a Python script named `chap01.py` with the following code:

```
1 colors = ['red', 'blue', 'green']  
2 print(colors[0])  
3 print(colors[2])  
4 print(len(colors))  
5 print(colors)
```

The bottom pane shows the output of running the script. The output is as follows:

```
Run: chap01 ×  
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/  
red  
green  
3  
['red', 'blue', 'green']  
  
Process finished with exit code 0
```

```
1 mix_list = [1, 2, 'Three', 3.14, True]
2 print(mix_list)
```

Run: chap01 x

D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pych

[1, 2, 'Three', 3.14, True]

Process finished with exit code 0

리스트 내부에 어떤 타입이던 삽입 가능

슬라이싱

```
1 cities = ['서울', '부산', '인천', '대구', '대전', '광주', '울산', '수원']
2 print(cities[0:6])
3 print(cities[:6])
4 print(cities[2:])
5 print(cities[-7:])
6 print(cities[:-2])
7 print(cities[:])
8 print(cities)
9 print(cities[1:7:2])
10 print(cities[::-1])
11 print(cities[::2])
```

Run: chap01 x

D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pycharm_workspace/exam01/chap01.p

['서울', '부산', '인천', '대구', '대전', '광주']

['서울', '부산', '인천', '대구', '대전', '광주']

['인천', '대구', '대전', '광주', '울산', '수원']

['부산', '인천', '대구', '대전', '광주', '울산', '수원']

['서울', '부산', '인천', '대구', '대전', '광주']

['서울', '부산', '인천', '대구', '대전', '광주', '울산', '수원']

['서울', '부산', '인천', '대구', '대전', '광주', '울산', '수원']

['부산', '대구', '광주']

['수원', '울산', '광주', '대전', '대구', '인천', '부산', '서울']

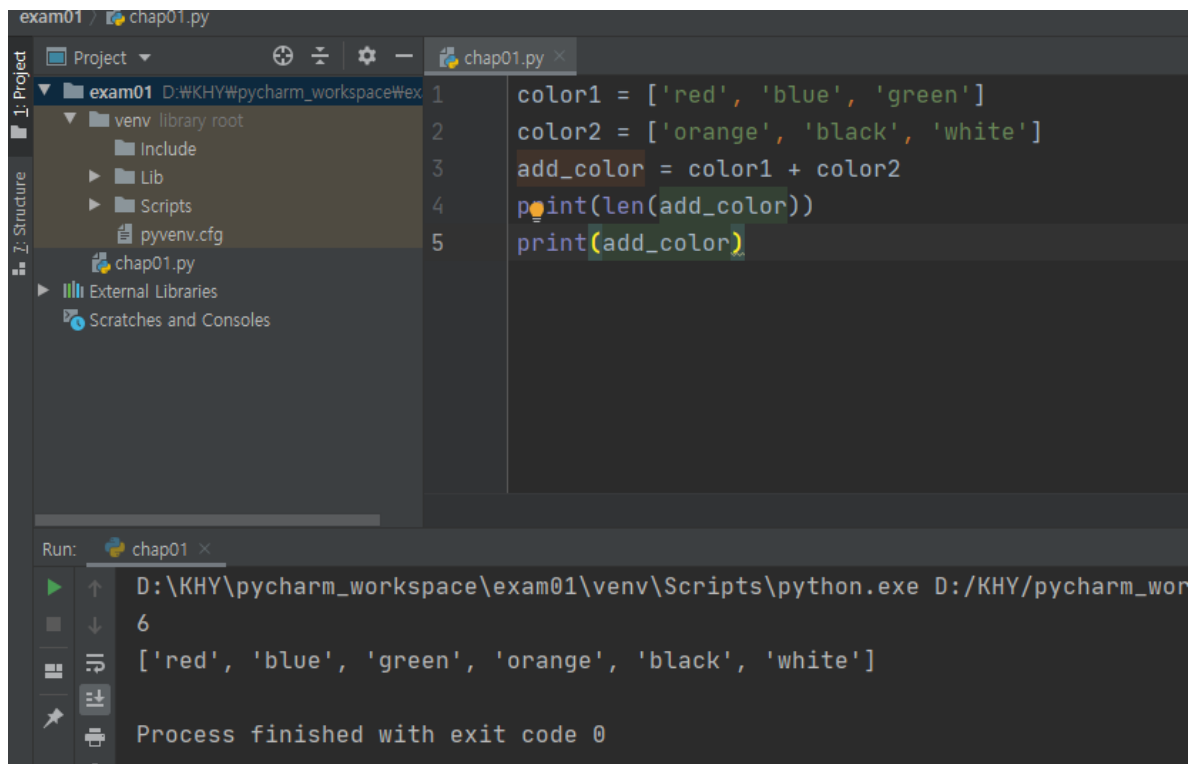
['서울', '인천', '대전', '울산']

Process finished with exit code 0

음수 : 거꾸로 (선형리스트이기 때문), 실제 갯수는 몰라도 마지막 인덱스에 접근가능

[1:7:2] : 1번째부터 6번째까지(7번은 포함 안됨) 2개씩 점프

리스트 연산



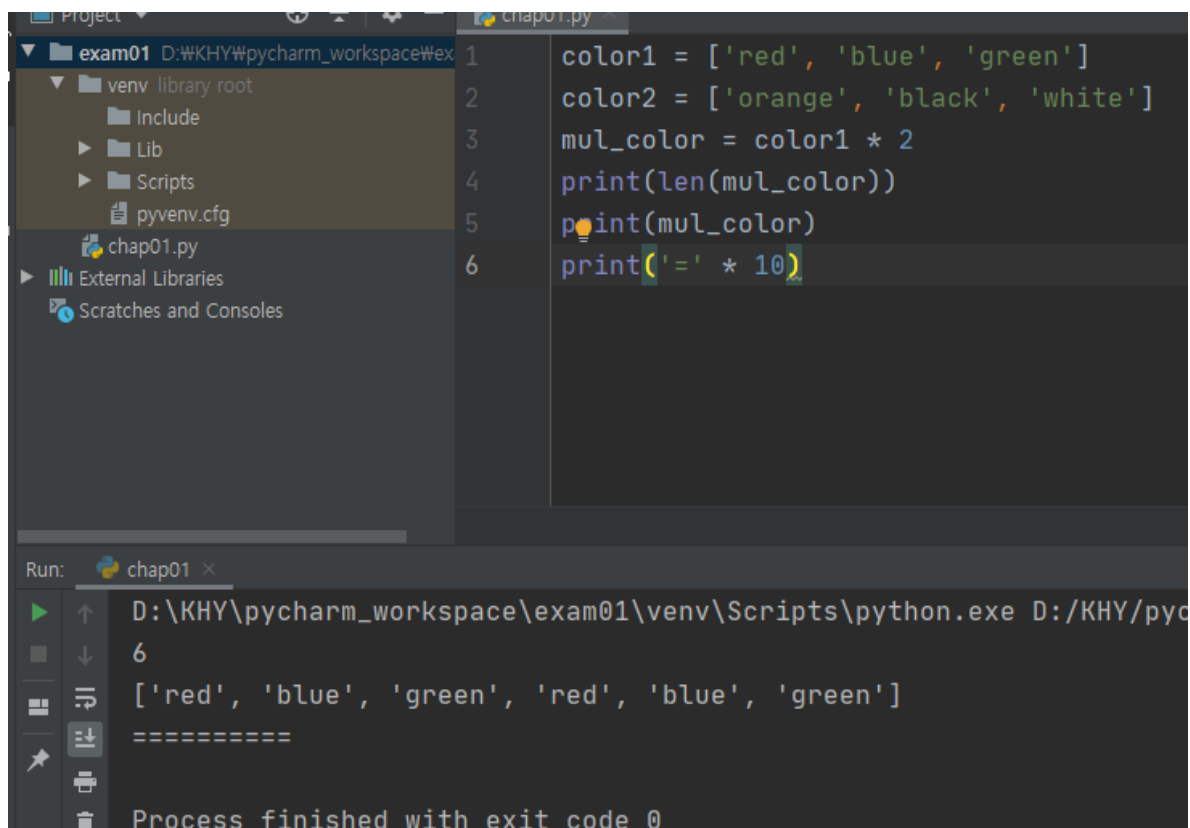
The screenshot shows the PyCharm IDE with a project named 'exam01'. The file explorer on the left shows a directory structure with 'venv', 'Include', 'Lib', 'Scripts', and 'pyvenv.cfg'. The main editor displays a Python file 'chap01.py' with the following code:

```
1 color1 = ['red', 'blue', 'green']
2 color2 = ['orange', 'black', 'white']
3 add_color = color1 + color2
4 print(len(add_color))
5 print(add_color)
```

The Run console at the bottom shows the output of the program:

```
Run: chap01 x
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pycharm_wor
6
['red', 'blue', 'green', 'orange', 'black', 'white']
Process finished with exit code 0
```

더하기



The screenshot shows the PyCharm IDE with the same project 'exam01'. The file explorer shows the same directory structure. The main editor displays a Python file 'chap01.py' with the following code:

```
1 color1 = ['red', 'blue', 'green']
2 color2 = ['orange', 'black', 'white']
3 mul_color = color1 * 2
4 print(len(mul_color))
5 print(mul_color)
6 print('=' * 10)
```

The Run console at the bottom shows the output of the program:

```
Run: chap01 x
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pyc
6
['red', 'blue', 'green', 'red', 'blue', 'green']
=====
Process finished with exit code 0
```

곱하기

```
1 color1 = ['red', 'blue', 'green']
2 color2 = ['orange', 'black', 'white']
3 print('blue' in color2)
```

Run: chap01 ×

```
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/p
False
```

in 연산 / 출력은 불리언 타입

append()

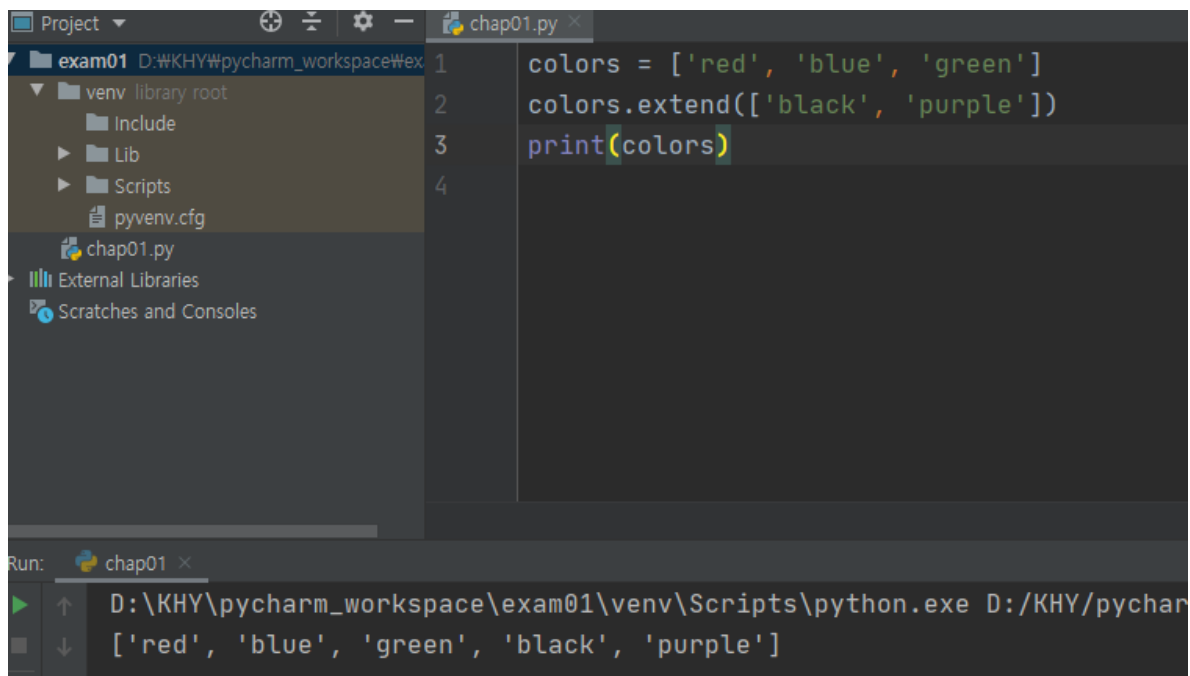
```
1 colors = ['red', 'blue', 'green']
2 colors.append('white')
3 print(colors)
4
```

Run: chap01 ×

```
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KH
['red' 'blue' 'green' 'white']
```

기존 리스트에 요소 추가

extend()



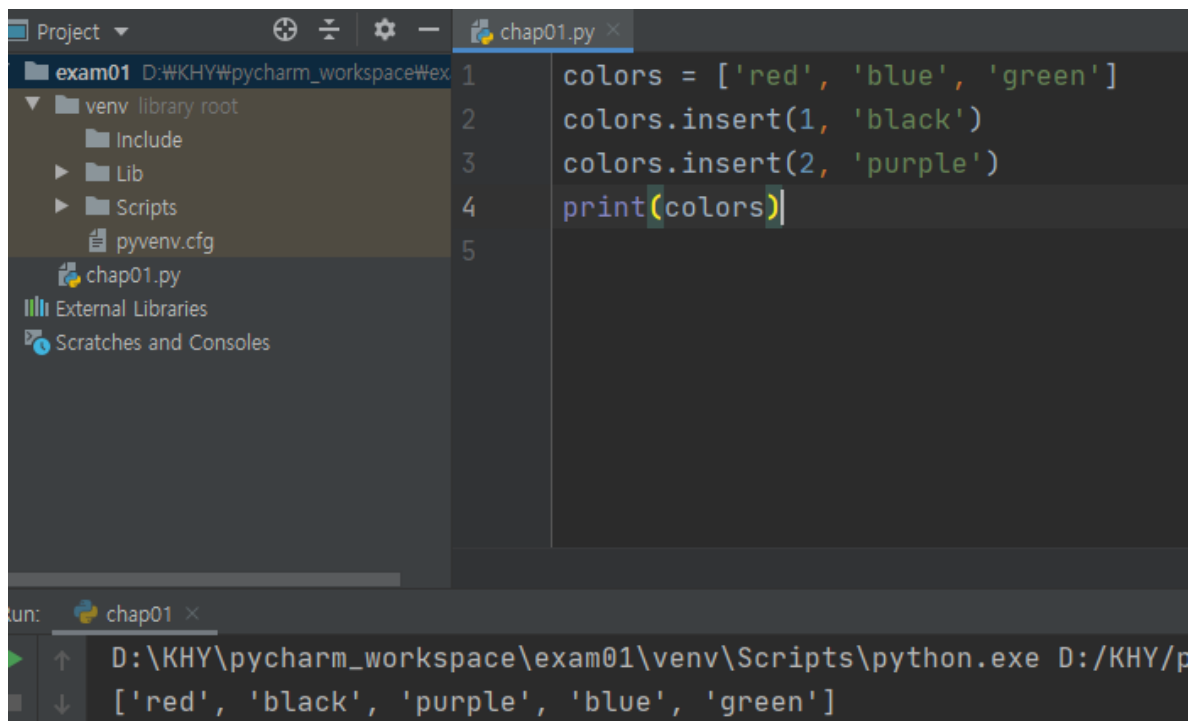
The screenshot shows the PyCharm IDE with a project named 'exam01'. The file explorer on the left shows the project structure, including a 'venv' directory and a 'chap01.py' file. The main editor window displays the following Python code in 'chap01.py':

```
1 colors = ['red', 'blue', 'green']
2 colors.extend(['black', 'purple'])
3 print(colors)
4
```

The Run window at the bottom shows the command executed: `D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pychar` and the output: `['red', 'blue', 'green', 'black', 'purple']`.

다른 리스트를 기존 리스트에 추가

insert()



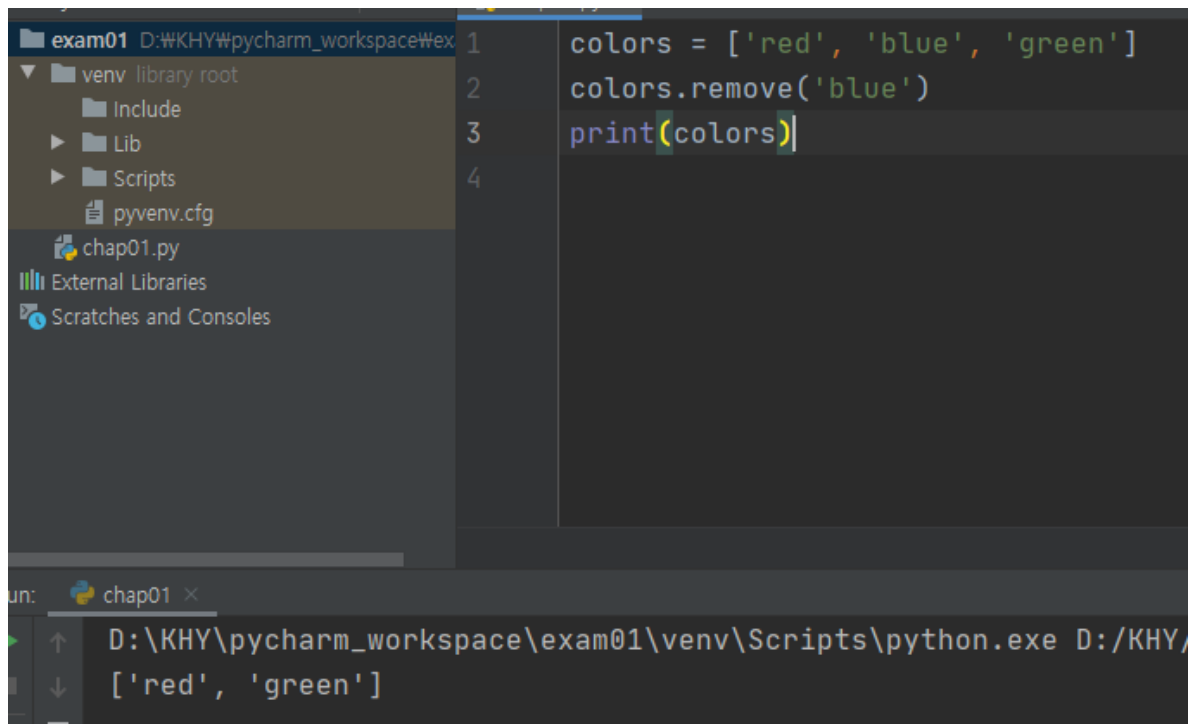
The screenshot shows the PyCharm IDE with the same project 'exam01'. The file explorer on the left shows the project structure, including a 'venv' directory and a 'chap01.py' file. The main editor window displays the following Python code in 'chap01.py':

```
1 colors = ['red', 'blue', 'green']
2 colors.insert(1, 'black')
3 colors.insert(2, 'purple')
4 print(colors)
5
```

The Run window at the bottom shows the command executed: `D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/p` and the output: `['red', 'black', 'purple', 'blue', 'green']`.

특정 인덱스에 요소 추가

remove()



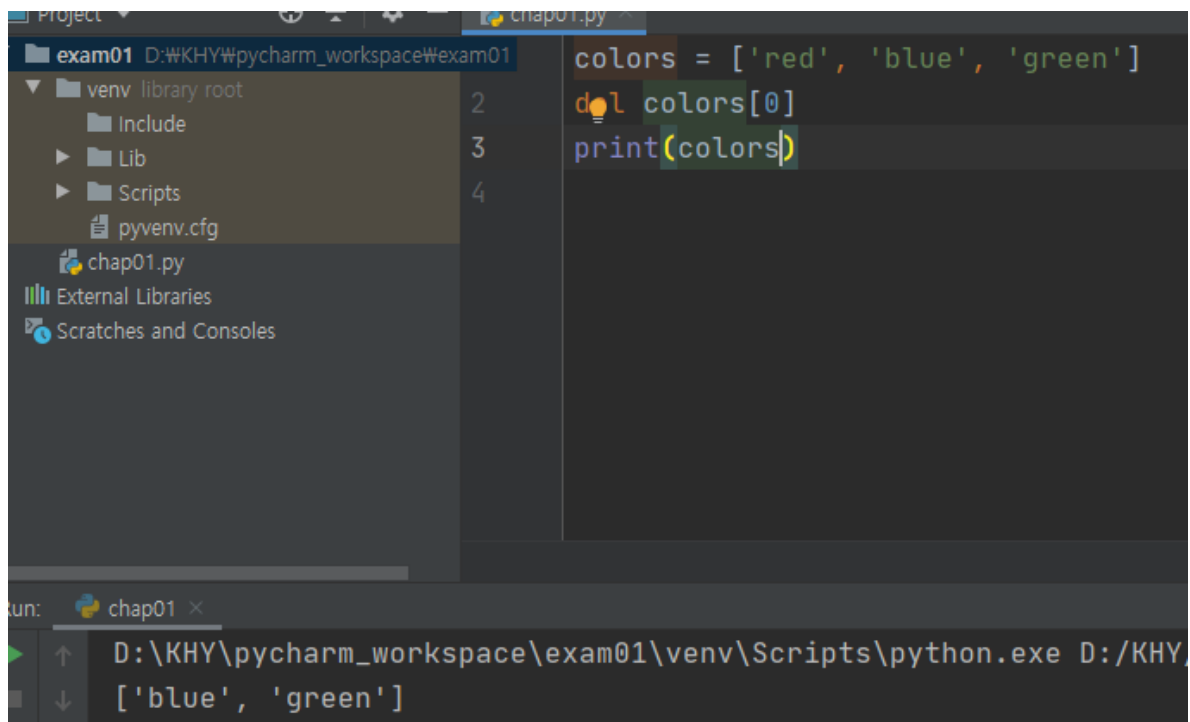
The screenshot shows the PyCharm IDE with a project named 'exam01'. The file explorer on the left shows a virtual environment 'venv' with folders 'Include', 'Lib', and 'Scripts', and a file 'pyvenv.cfg'. The main editor displays a Python script 'chap01.py' with the following code:

```
1 colors = ['red', 'blue', 'green']
2 colors.remove('blue')
3 print(colors)
4
```

The console at the bottom shows the execution of the script, outputting: `['red', 'green']`.

리스트 내에서 특정 값 삭제

del



The screenshot shows the PyCharm IDE with the same project 'exam01'. The file explorer is the same. The main editor displays the same Python script 'chap01.py' with the following code:

```
1 colors = ['red', 'blue', 'green']
2 del colors[0]
3 print(colors)
4
```

The console at the bottom shows the execution of the script, outputting: `['blue', 'green']`.

특정 인덱스 요소 삭제

특정 인덱스 덮어쓰기(대체)

```
Project: exam01 D:\KHY\pycharm_workspace\exam01
venv library root
  Include
  Lib
  Scripts
  pyvenv.cfg
chap01.py
External Libraries
Scratches and Consoles

1 colors = ['red', 'blue', 'green']
2 colors[2] = 'white'
3 print(colors)
4

Run: chap01 x
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/p
['red', 'blue', 'white']
```

패킹 & 언패킹

```
Project: exam01 D:\KHY\pycharm_workspace\exam01
venv library root
  Include
  Lib
  Scripts
  pyvenv.cfg
chap01.py
External Libraries
Scratches and Consoles

1 colors = ['red', 'blue', 'green'] # packing
2 a, b, c = colors # unpacking
3 print(a, b, c)
4

Run: chap01 x
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pycharm_work
red blue green
```

리스트의 해당 위치의 값이 각각 언패킹됨

이 때 변수의 갯수와 리스트 요소의 갯수가 동일해야 함

The screenshot shows the PyCharm IDE with a project named 'exam01'. The file explorer on the left shows the project structure, including a 'venv' directory. The main editor displays a Python file 'chap01.py' with the following code:

```
1 colors = ['red', 'blue', 'green'] # packing
2 a, _, c = colors # unpacking
3 print(a, c)
4
```

The Run console at the bottom shows the command used to execute the script: `D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pycharm_w...`. The output of the script is `red green`.

언더바(_) 코드로 해당 위치의 언패킹한 값을 날림(변수할당 x)

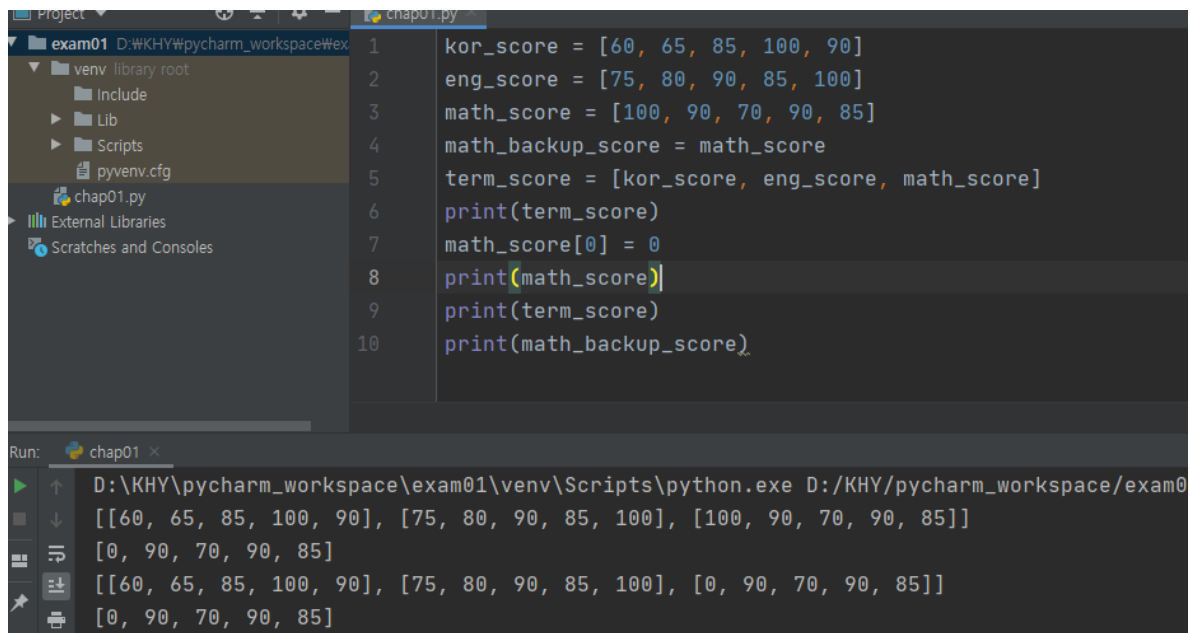
이중리스트(중첩리스트)

The screenshot shows the PyCharm IDE with a project named 'exam01'. The file explorer on the left shows the project structure, including a 'venv' directory. The main editor displays a Python file 'chap01.py' with the following code:

```
1 kor_score = [60, 65, 85, 100, 90]
2 eng_score = [75, 80, 90, 85, 100]
3 math_score = [100, 90, 70, 90, 85]
4 term_score = [kor_score, eng_score, math_score]
5 print(term_score)
6 print(term_score[0]) # [row]
7 print(term_score[0][2]) # [row][column]
```

The Run console at the bottom shows the command used to execute the script: `D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pycharm_workspac...`. The output of the script is:

```
[[60, 65, 85, 100, 90], [75, 80, 90, 85, 100], [100, 90, 70, 90, 85]]
[60, 65, 85, 100, 90]
85
```



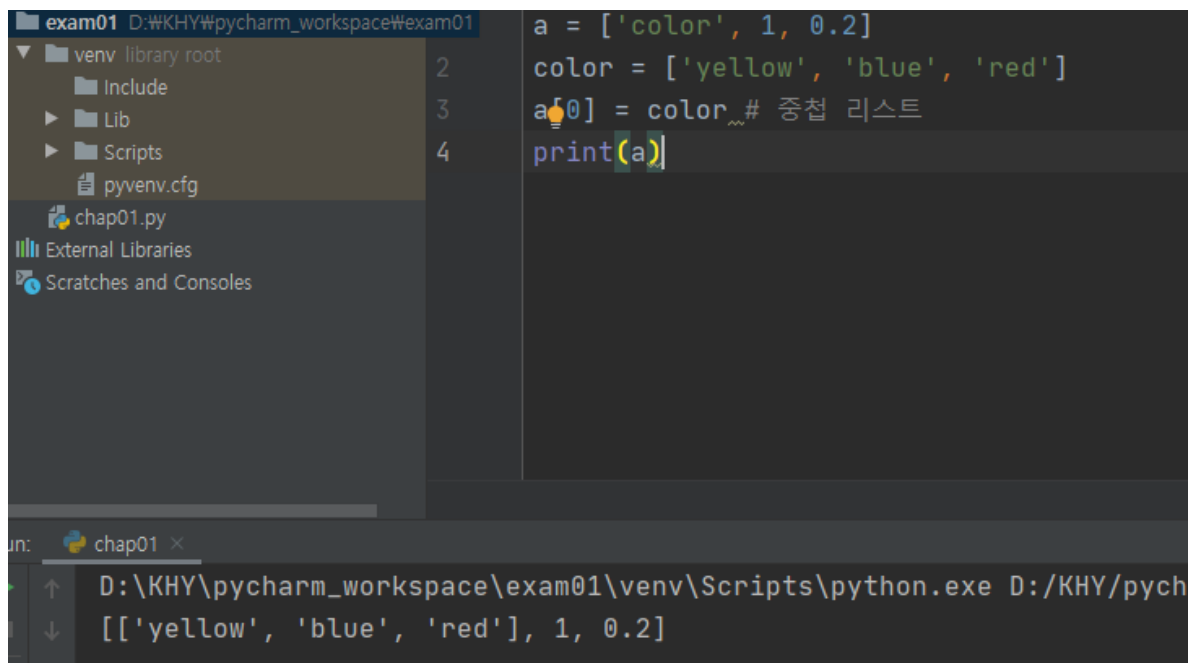
The image shows a PyCharm IDE window with a project named 'exam01'. The left sidebar displays the project structure, including a virtual environment 'venv' and a file 'chap01.py'. The main editor shows the following Python code:

```
1 kor_score = [60, 65, 85, 100, 90]
2 eng_score = [75, 80, 90, 85, 100]
3 math_score = [100, 90, 70, 90, 85]
4 math_backup_score = math_score
5 term_score = [kor_score, eng_score, math_score]
6 print(term_score)
7 math_score[0] = 0
8 print(math_score)
9 print(term_score)
10 print(math_backup_score)
```

Below the editor, the 'Run' console shows the output of the script:

```
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pycharm_workspace/exam0
[[60, 65, 85, 100, 90], [75, 80, 90, 85, 100], [100, 90, 70, 90, 85]]
[0, 90, 70, 90, 85]
[[60, 65, 85, 100, 90], [75, 80, 90, 85, 100], [0, 90, 70, 90, 85]]
[0, 90, 70, 90, 85]
```

얕은 복사



The image shows a PyCharm IDE window with a project named 'exam01'. The left sidebar displays the project structure, including a virtual environment 'venv' and a file 'chap01.py'. The main editor shows the following Python code:

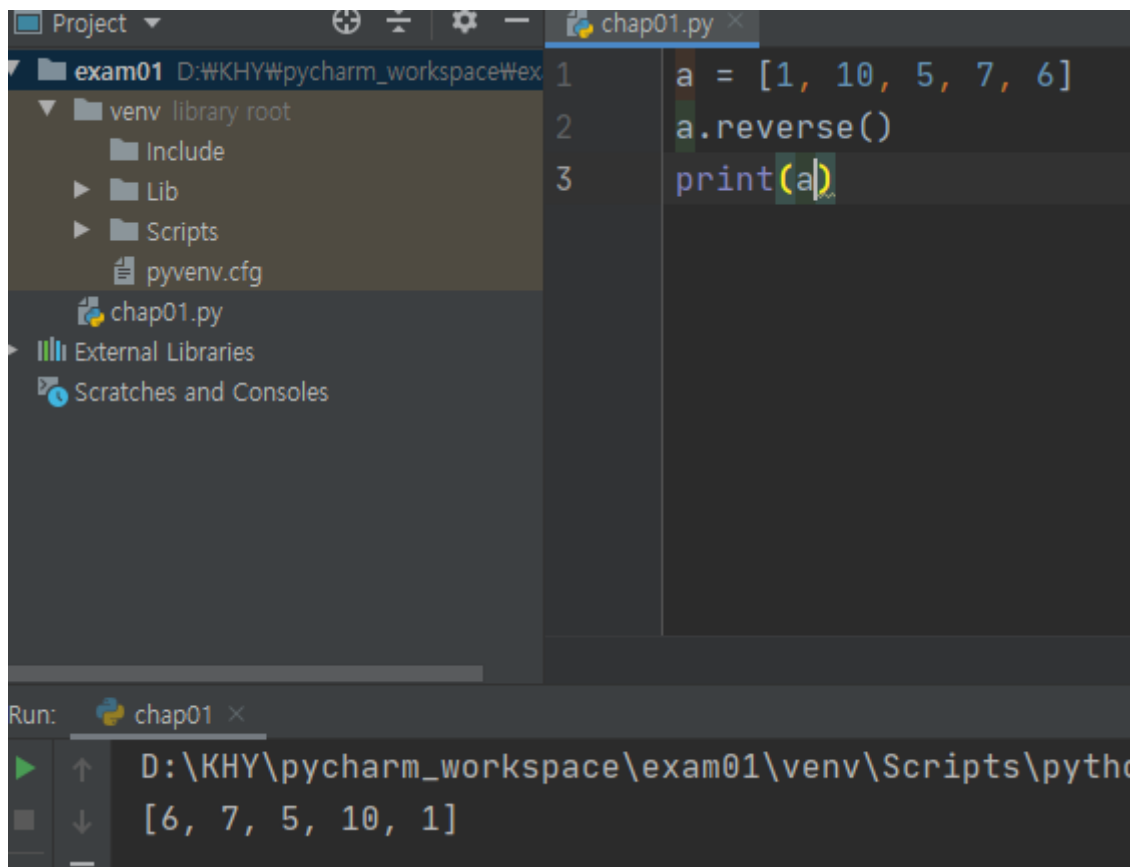
```
1 a = ['color', 1, 0.2]
2 color = ['yellow', 'blue', 'red']
3 a[0] = color # 중첩 리스트
4 print(a)
```

Below the editor, the 'Run' console shows the output of the script:

```
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pych
[['yellow', 'blue', 'red'], 1, 0.2]
```

다른 리스트로 리스트 내부 값 변경

reverse()



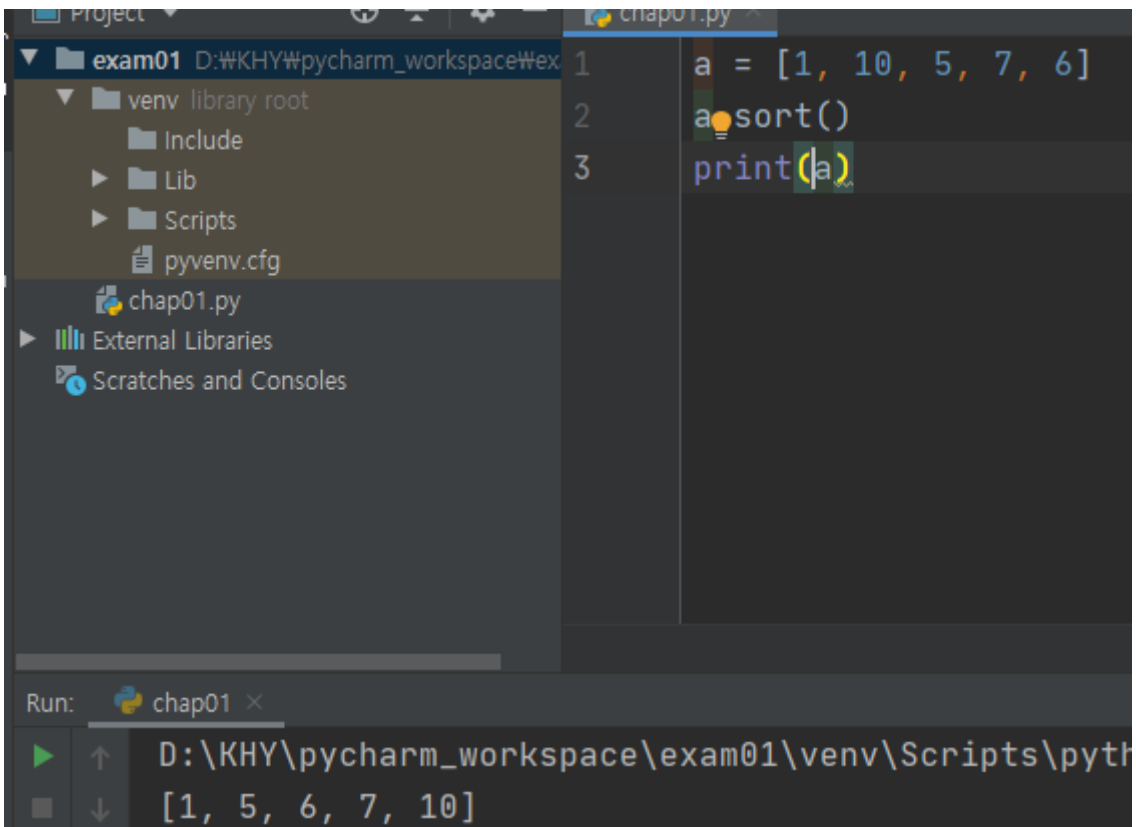
The screenshot shows the PyCharm IDE with a project named 'exam01'. The file explorer on the left shows a 'venv' directory with 'library root', 'Include', 'Lib', 'Scripts', and 'pyvenv.cfg'. The main editor window shows a file named 'chap01.py' with the following code:

```
1 a = [1, 10, 5, 7, 6]
2 a.reverse()
3 print(a)
```

The Run console at the bottom shows the output of the program:

```
Run: chap01 x
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe
[6, 7, 5, 10, 1]
```

sort()



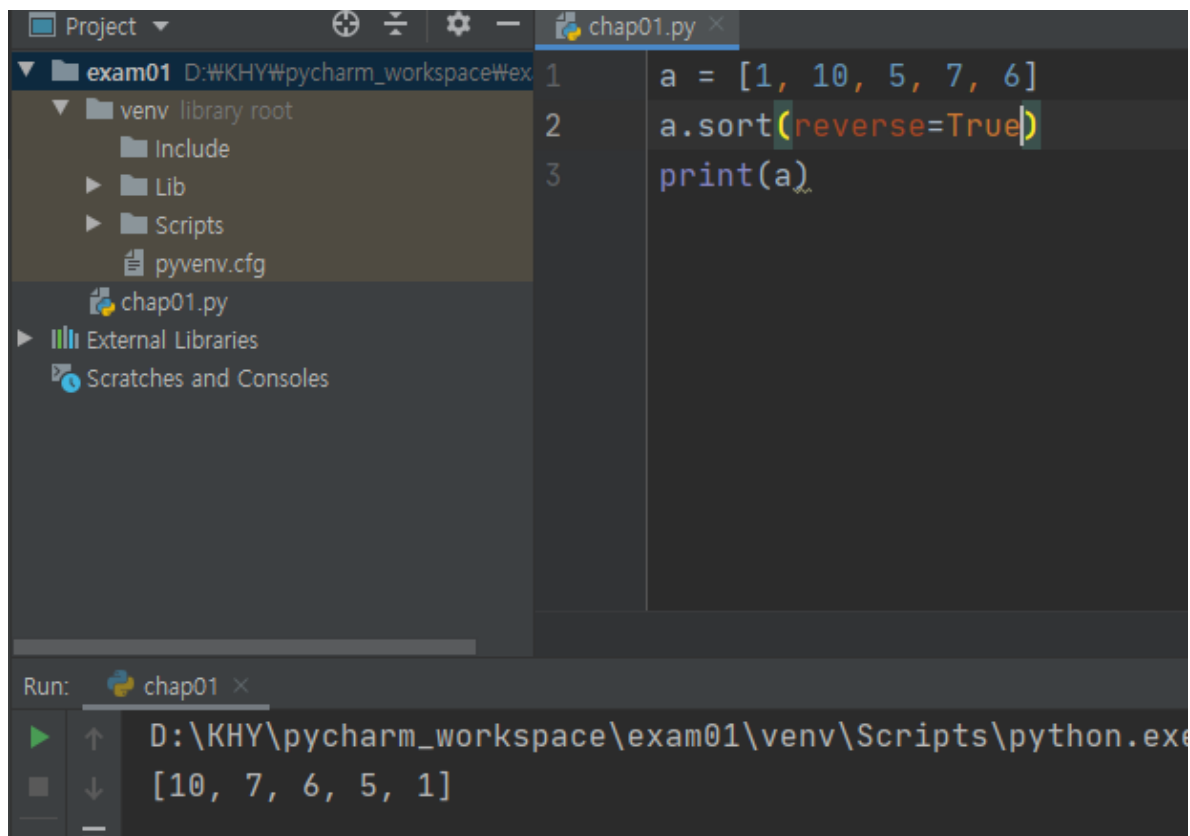
The screenshot shows the PyCharm IDE with the same project 'exam01'. The file explorer on the left shows the 'chap01.py' file. The main editor window shows the following code:

```
1 a = [1, 10, 5, 7, 6]
2 a.sort()
3 print(a)
```

The Run console at the bottom shows the output of the program:

```
Run: chap01 x
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe
[1, 5, 6, 7, 10]
```

디폴트로 작성 시 오름차순



The screenshot shows the PyCharm IDE with a project named 'exam01'. The file explorer on the left shows the project structure, including a 'venv' directory and a 'chap01.py' file. The main editor window displays the following Python code:

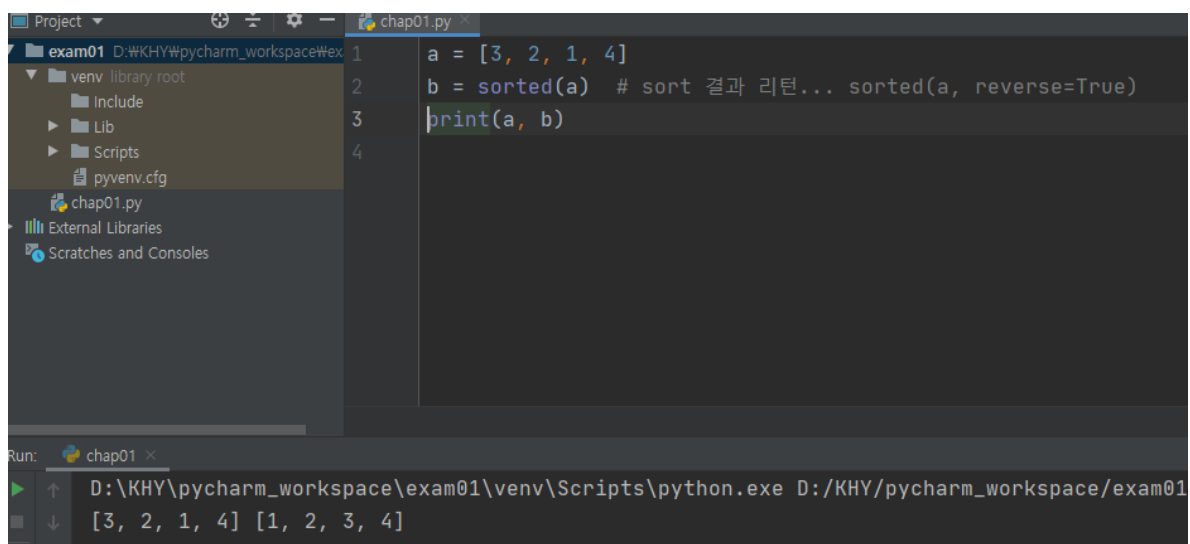
```
1 a = [1, 10, 5, 7, 6]
2 a.sort(reverse=True)
3 print(a)
```

The Run window at the bottom shows the execution of 'chap01.py' using the Python interpreter located at 'D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe'. The output of the program is '[10, 7, 6, 5, 1]'.

내림차순으로 작성한 sort() 코드

sort()는 원본을 수정함

sorted()



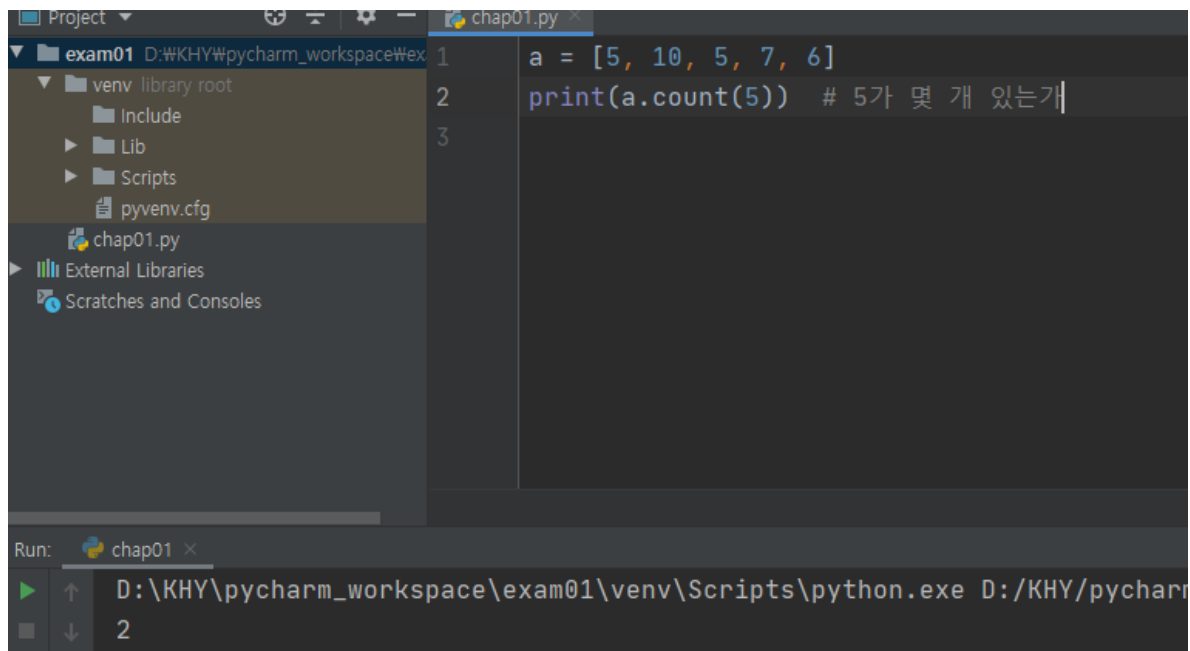
The screenshot shows the PyCharm IDE with a project named 'exam01'. The file explorer on the left shows the project structure, including a 'venv' directory and a 'chap01.py' file. The main editor window displays the following Python code:

```
1 a = [3, 2, 1, 4]
2 b = sorted(a) # sort 결과 리턴... sorted(a, reverse=True)
3 print(a, b)
4
```

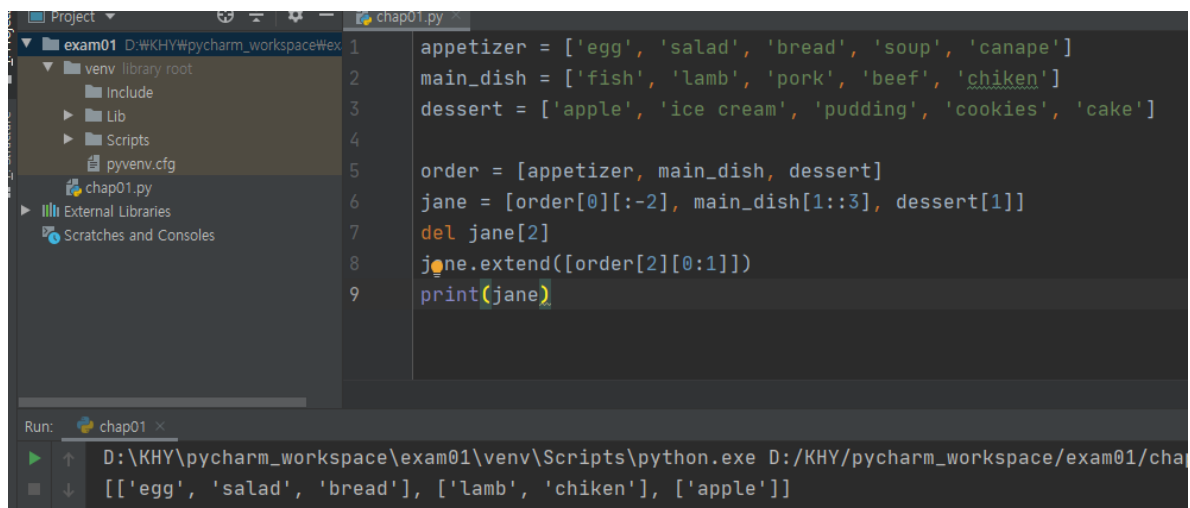
The Run window at the bottom shows the execution of 'chap01.py' using the Python interpreter located at 'D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe'. The output of the program is '[3, 2, 1, 4] [1, 2, 3, 4]'.

sorted()는 원본을 유지

count()



실습 ※※※※※※※※



시험 답 쓸 때 리스트 형태 그대로 써야 정답

indentation (들여쓰기)

- 파이썬에선 들여쓰기가 매우 중요

The screenshot shows the PyCharm IDE interface. The top pane displays a Python script named `chap01.py` with the following code:

```
1 a = ['color', 1, 0.2]
2     color = ['yellow', 'blue', 'red']
3 a[0] = color # 중첩 리스트
4 print(a)
```

The bottom pane shows the console output of the script execution, which resulted in an error:

```
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/py
File "D:/KHY/pycharm_workspace/exam01/chap01.py", line 2
    color = ['yellow', 'blue', 'red']
    ^
IndentationError: unexpected indent
```

타 언어에선 들여쓰기 정도를 신경 쓰진 않지만 파이썬은 인식함

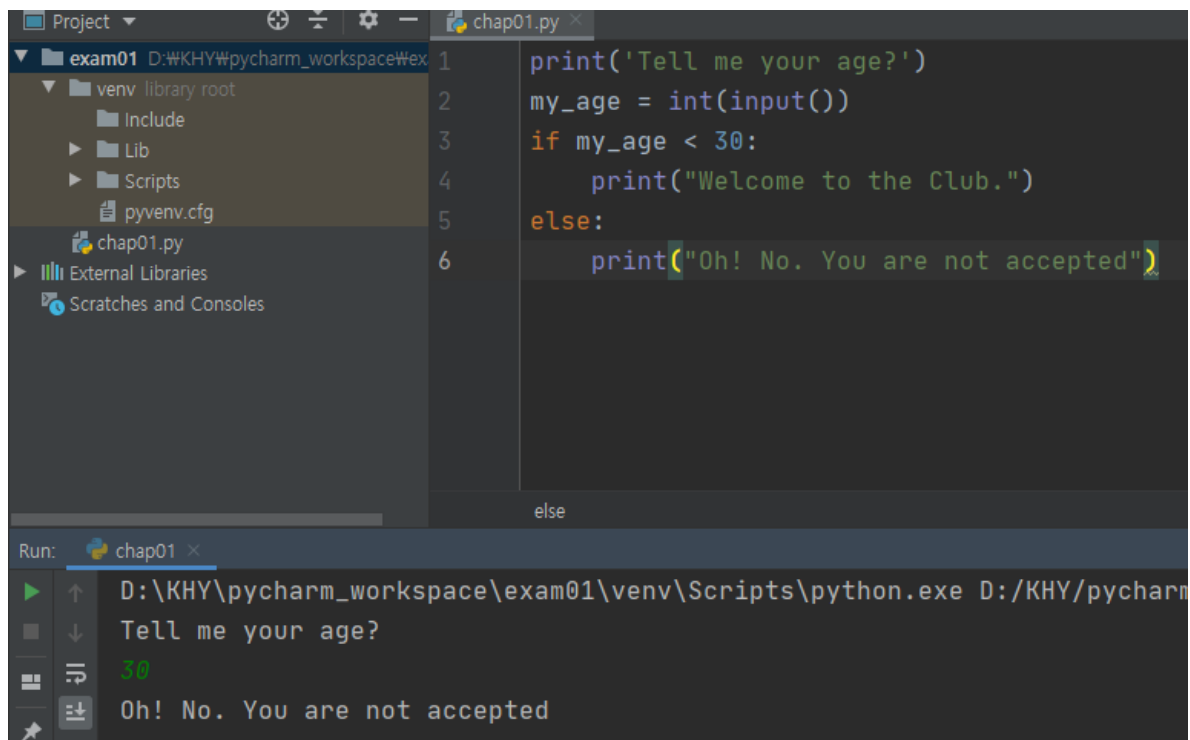
들여쓰기 구분을 위해서 탭 사용을 지양함(탭과 스페이스 혼용 또한 금지, 하지만 자동적으로 탭을 스페이스로 파이참에서 바꿔줌)

4개의 스페이스로 구분

쉬프트 + 탭 버튼으로 이전 단계 grade로 돌입 가능

조건문과 반복문

- if



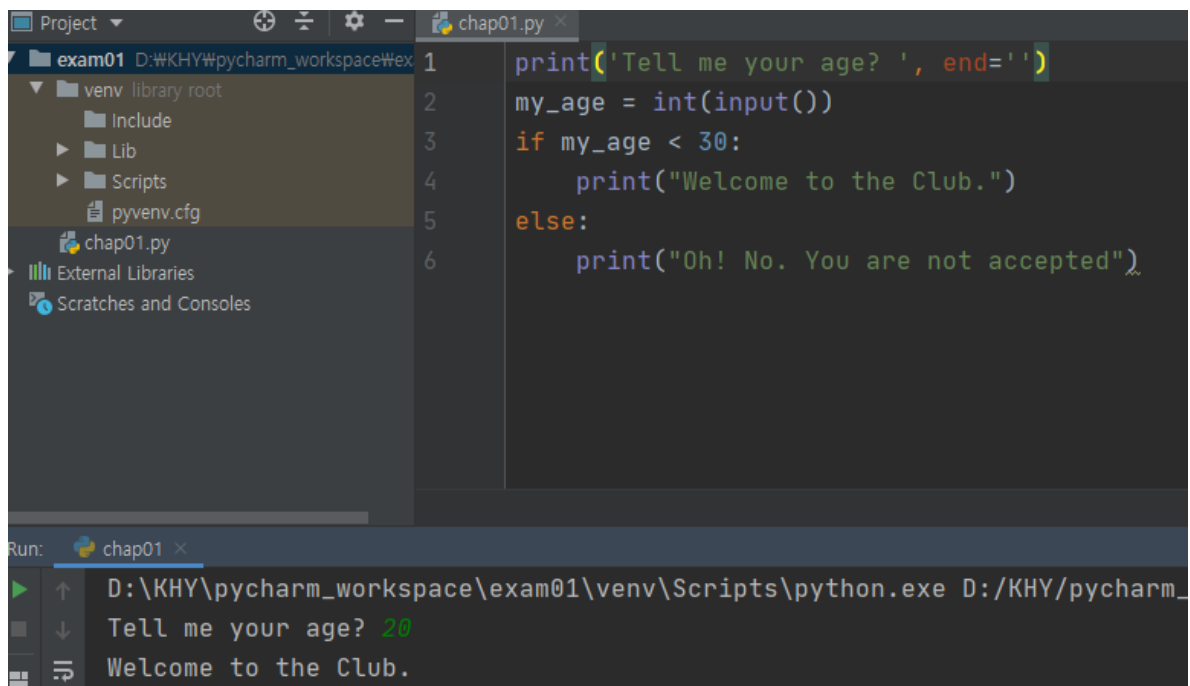
The image shows the PyCharm IDE with a project named 'exam01'. The file explorer on the left shows the project structure, including a 'venv' directory and a 'chap01.py' file. The main editor displays the following Python code:

```
1 print('Tell me your age?')
2 my_age = int(input())
3 if my_age < 30:
4     print("Welcome to the Club.")
5 else:
6     print("Oh! No. You are not accepted")
```

The Run console at the bottom shows the execution of 'chap01.py'. The output is:

```
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pycharm_
Tell me your age? 30
Oh! No. You are not accepted
```

파이썬은 콜론(:)으로 구분



The image shows the PyCharm IDE with the same project and file structure. The 'chap01.py' file now contains the following Python code:

```
1 print('Tell me your age? ', end='')
2 my_age = int(input())
3 if my_age < 30:
4     print("Welcome to the Club.")
5 else:
6     print("Oh! No. You are not accepted")
```

The Run console shows the execution of 'chap01.py'. The output is:

```
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pycharm_
Tell me your age? 20
Welcome to the Club.
```

print()의 구분자?(end) 값을 디폴트가 아닌 직접 사용자 지정한 모습

- 조건 판단과 논리 연산자

2. 조건 판단

1) 비교 연산자

비교 연산자	비교 상태	설명
$x < y$	보다 작음	x가 y보다 작은지 검사
$x > y$	보다 큼	x가 y보다 큰지 검사
$x == y$	같음	x와 y의 값이 같은지 검사
$x != y$	같지 않음	x와 y의 값이 같지 않은지 검사
$x \text{ is } y$	같음(reference)	x와 y의 reference가 같은지 검사
$x \text{ is not } y$	같지 않음(reference)	x와 y의 reference가 같지 않은지 검사
$x \geq y$	크거나 같음	x가 y보다 크거나 같은지 검사
$x \leq y$	작거나 같음	x가 y보다 작거나 같은지 검사

2) 논리 연산자

연산자	설명	예시
and	두 값이 모두 참일 경우 True, 그렇지 않을 경우 False	$(7 > 5) \text{ and } (10 > 5) ==> \text{True}$ $(7 < 5) \text{ and } (10 < 5) ==> \text{False}$
or	두 값 중 하나 이상 참일 경우 True, 두 값 모두 거짓일 경우 False	$(7 < 5) \text{ or } (10 > 5) ==> \text{True}$ $(7 < 5) \text{ or } (10 < 5) ==> \text{False}$
not	논리의 역	$\text{not } (7 < 5) ==> \text{True}$ $\text{not } (7 > 5) ==> \text{False}$

- elif

```

1 score = int(input("Enter your score: "))
2
3 if score >= 90:
4     grade = 'A'
5 elif score >= 80:
6     grade = 'B'
7 elif score >= 70:
8     grade = 'C'
9 elif score >= 60:
10    grade = 'D'
11 else:
12    grade = 'F'
13
14 print(grade)

```

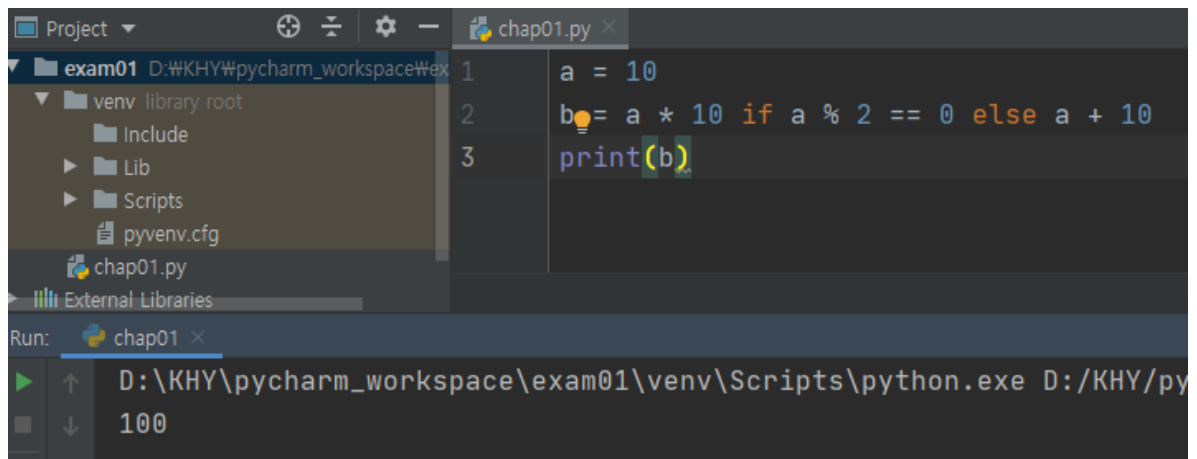
Run: chap01

```

D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pychar
Enter your score: 80
B

```

- One-Line if (한 줄 코드)



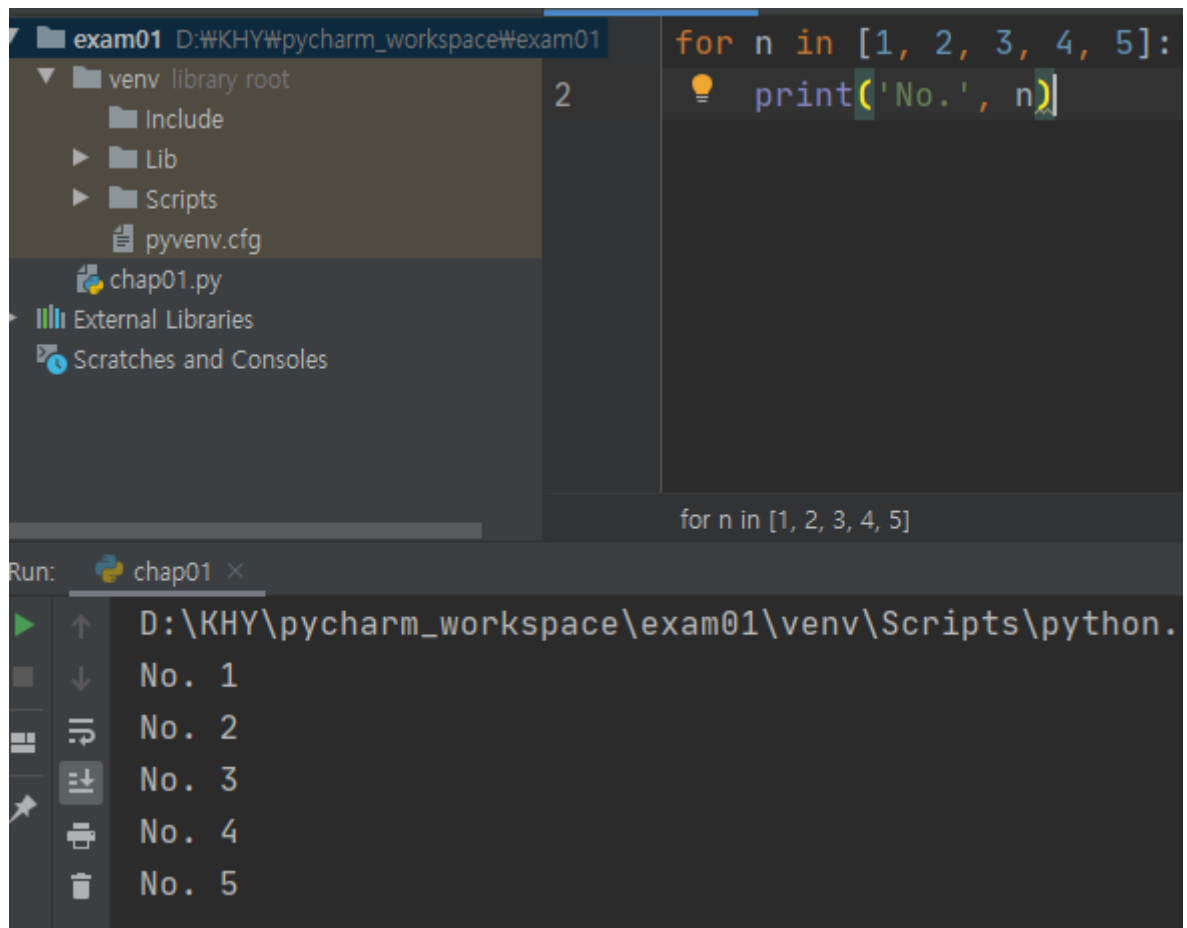
```

safe_x = x or 0
safe_x = x if x else 0

```

원-라인 이프문으로 변환하여 작성한 모습(하단)

- for



in 연산자를 활용한 코드

- range()

```
Project ▾
exam01 D:\KHY\pycharm_workspace\exam01
├── venv library root
│   ├── Include
│   ├── Lib
│   ├── Scripts
│   └── pyvenv.cfg
├── chap01.py
├── External Libraries
└── Scratches and Consoles

Run: chap01 ×
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe
No. 0
No. 1
No. 2
No. 3
No. 4
```

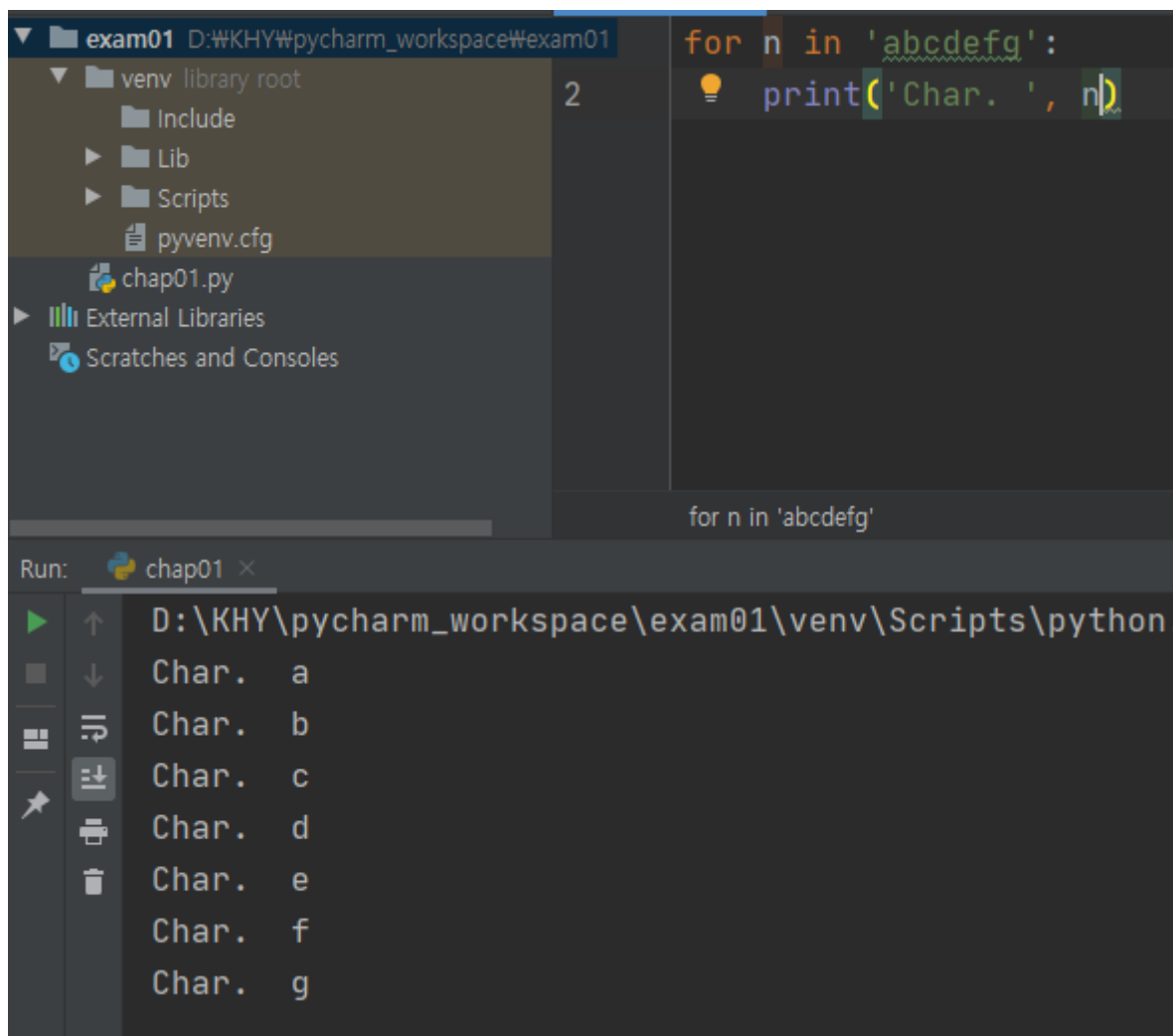
for n in range(5):
 print('No. ', n)

```
Project ▾
exam01 D:\KHY\pycharm_workspace\exam01
├── venv library root
│   ├── Include
│   ├── Lib
│   ├── Scripts
│   └── pyvenv.cfg
├── chap01.py
├── External Libraries
└── Scratches and Consoles

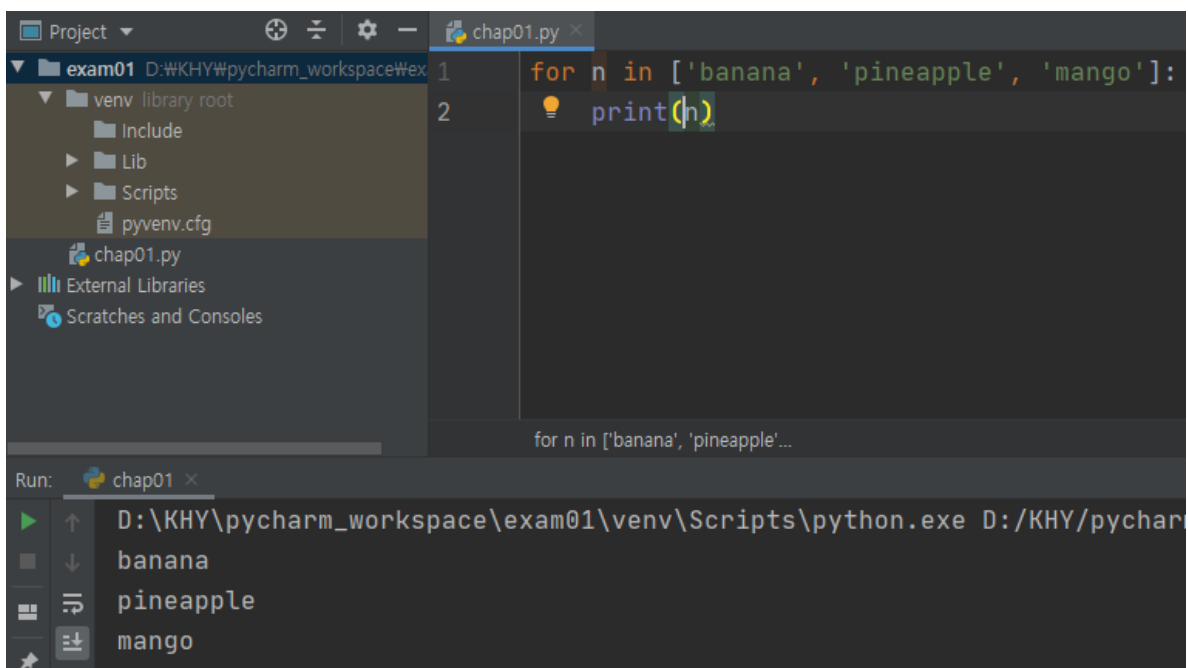
Run: chap01 ×
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe
No. 1
No. 3
No. 5
```

for n in range(1, 6, 2):
 print('No. ', n)

range(시작, 끝(-1), 점프)



문자열도 가능



배열도 가능

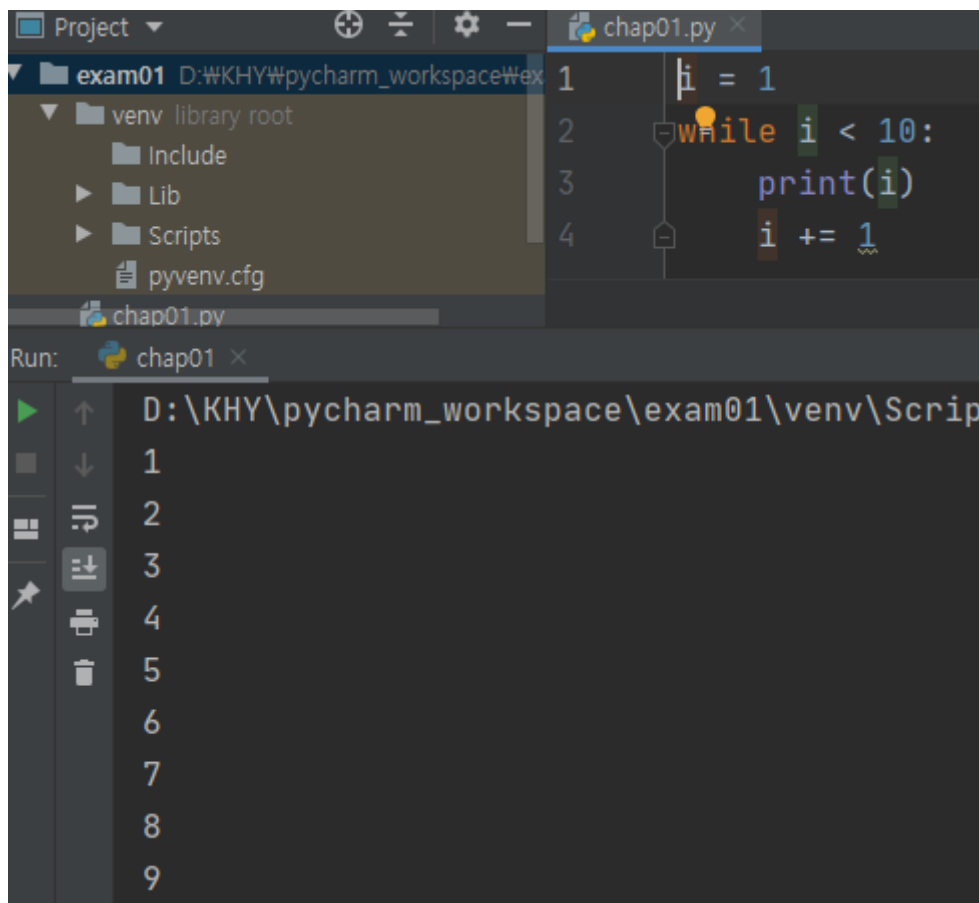
- 이중 for문과 in

The screenshot shows the PyCharm IDE interface. The top toolbar includes icons for running, debugging, and settings. The 'Project' view on the left shows a workspace named 'exam01' located at 'D:\WKHY\pycharm_workspace\exam01'. It contains a virtual environment 'venv' with subfolders 'Include', 'Lib', and 'Scripts', and a file 'pyvenv.cfg'. The main editor displays a file 'chap01.py' with the following Python code:

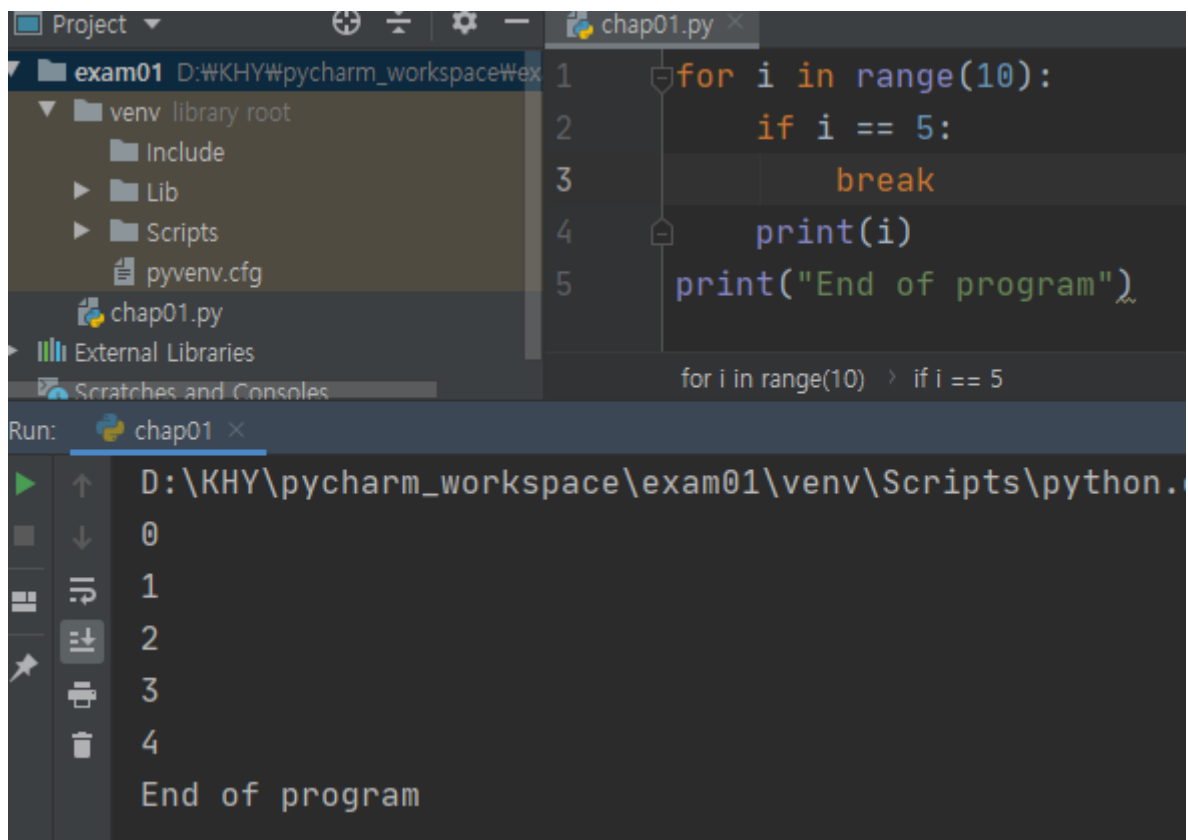
```
1 for n in ['banana', 'pineapple', 'mango']:
2     for k in n:
3         print(k)
4     print()
```

Below the editor, the 'Run' console shows the command used to execute the script: 'D:\KH\Y\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KH\Y/pychar'. The output of the script is displayed in the console, showing the characters of the words 'banana', 'pineapple', and 'mango' printed on separate lines, with an additional blank line after each word.

- while



- break



자신과 가장 가까운 반복문으로 부터 break

- continue

The image shows a PyCharm IDE window with a project named 'exam01'. The file explorer on the left shows the project structure, including a 'venv' directory and a 'chap01.py' file. The main editor displays the code in 'chap01.py':

```
1 for i in range(10):
2     if i == 5:
3         continue
4     print(i)
5 print("End of program")
```

The 'Run' tab at the bottom shows the execution output for 'chap01'. The output is as follows:

```
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.
0
1
2
3
4
6
7
8
9
End of program
```

- 파이썬은 for문에 else사용 가능(하지만 거의 안 씀, 결과가 똑같기 때문)

```
1 for i in range(10):
2     print(i)
3 else:
4     print("End of program")
```

Run: chap01 ×

D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:

0
1
2
3
4
5
6
7
8
9
End of program

```
1 for i in range(10):
2     if i == 5: break
3     print(i)
4 else:
5     print("End of program")
```

Run: chap01 ×

D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe

0
1
2
3
4

for문이 끝나고 else 실행(정상적으로 끝났을 때만 실행)

위 예제의 경우 정상적으로 끝낸 것인지, break문으로 끝난 것인지 비교하기 위한 것

실습

The image shows a PyCharm IDE window with a project named 'exam01'. The file explorer on the left shows the project structure, including a 'venv' directory and a 'chap01.py' file. The main editor displays the code for 'chap01.py':

```
1 print("몇 단?")
2 dan = input()
3 print("구구단 ", dan, " 단 계산")
4 int_dan = int(dan)
5 for i in range(1, 10):
6     result = int_dan * i
7     print(dan, " X ", i, " = ", result)
8
```

The 'Run' tab at the bottom shows the execution of 'chap01.py'. The command prompt displays the following output:

```
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pycha
몇 단?
3
구구단 3 단 계산
3 X 1 = 3
3 X 2 = 6
3 X 3 = 9
3 X 4 = 12
3 X 5 = 15
3 X 6 = 18
3 X 7 = 21
3 X 8 = 24
3 X 9 = 27
```

구구단 출력

The image shows a PyCharm IDE window with a project named 'exam01'. The file explorer on the left shows the project structure, including a 'venv' directory and a 'chap01.py' file. The main editor displays the code for 'chap01.py':

```
1 sentence = 'I love you'
2 reverse_sentence = ''
3 for char in sentence:
4     """char에 하나씩 들어간 값을 reverse_sentence의 전방에 붙여 줌"""
5     reverse_sentence = char + reverse_sentence
6 print(reverse_sentence)
7
```

The 'Run' tab at the bottom shows the execution of 'chap01.py'. The command prompt displays the following output:

```
D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pycharm_workspace/exam01/chap01.py
uoy evol I
```

함수 사용하지 않고 로직으로 작성한 reverse


```
1 decimal = 14
2 result = ''
3 while decimal > 0:
4     remainder = decimal % 2
5     decimal = decimal // 2
6     result = str(remainder) + result
7 print(result)
8 """
9 de = 14 -> rem = 0 de = 7 res = '0' + ''
10 de = 7 -> rem = 1 de = 3 res = '1' + '0'
11 de = 3 -> rem = 1 de = 1 res = '1' + '10'
12 de = 1 -> rem = 1 de = 0 res = '1' + '110'
13 de = 0 -> out of while -> res = '1110'
14 """
```

Run: chap01 x

D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pycharm_wor

1110

10진수를 2진수로 변환

```
1 kor_score = [49, 80, 20, 100, 80]
2 math_score = [43, 60, 85, 30, 90]
3 eng_score = [49, 82, 48, 50, 100]
4 term_score = [kor_score, math_score, eng_score]
5 student_score = [0, 0, 0, 0, 0] # student_score[idx] += j 파트를 위해 0값으로 초기화
6 idx = 0 # index
7 for i in term_score: # 1.kor_score, 2.math_score, 3.eng_score 자체를 i에 가져옴
8     for j in i: # *_score의 내부 리스트 값을 가져옴
9         student_score[idx] += j # student_score[idx] = student_score[idx] + j
10        idx += 1 # 인덱스를 이 부분에서 증가시켜 줌
11    idx = 0 # 이 부분에서 인덱스 다시 0으로 초기화
12    n_s = len(term_score)
13    a, b, c, d, e = student_score # 언패킹
14    student_average = [a/n_s, b/n_s, c/n_s, d/n_s, e/n_s]
15    print(student_average)
```

Run: chap01 x

D:\KHY\pycharm_workspace\exam01\venv\Scripts\python.exe D:/KHY/pycharm_workspace/exam01/chap01.py

[47.0, 74.0, 51.0, 60.0, 90.0]

이중 for문으로 평균값 출력

3) None

3) None

None is for non-exist value

- x = None
- assert x == None, "this is the not the Pythonic way to check for None"
- assert x is None, "this is the Pythonic way to check for None"

파이썬이나 사용하는 False

False, None, [], {}, "", set(), 0, 0.0

```
s = some_function_that_returns_a_string()
if s:
    first_char = s[0]
else:
    first_char = ""
```

None 예시

```
first_char = s and s[0]
```

위 코드를 한 줄로 작성한 코드

s에 문자열을 받았을 경우 and하게 되면 값이 있느냐 없느냐에 따라 연산을 함