1-1. [Practice01-1.py]

풀이)

1) C:\PythonStudy\Section01 디렉터리를 만듭니다.

2) 메모장을 열고 아래와 같은 코드를 작성합니다.

```
01  print('안녕 파이썬!')
02  print('앞으로 잘 부탁해!')
```

3) 작성한 파일을 C:\PythonStudy\Section01 디렉터리에 Practice01-1.py 파일명으로 저장합니다.

4) Practice01-1.py 파일을 닫습니다.

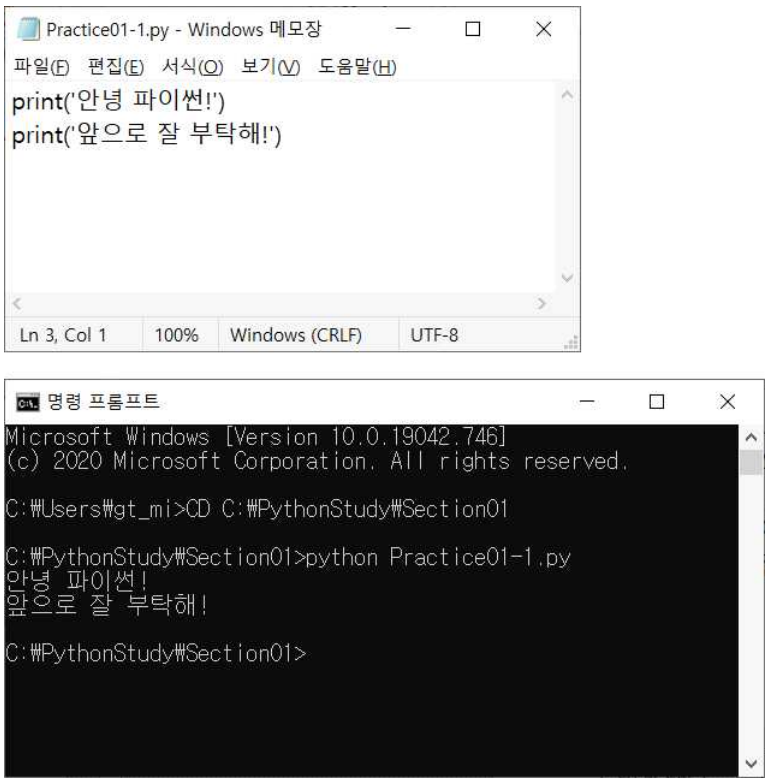
5) 명령 프롬프트를 실행합니다. (시작 메뉴에서 검색하거나 실행 창을 열고 'cmd'를 입력합니다.)

6) C:\PythonStudy\Section01 디렉터리로 이동하는 명령을 입력하고 엔터를 누릅니다.

```
CD C:\PythonStudy\Section01
```

7) Practice01-1.py 파일의 실행 명령을 입력하고 엔터를 누릅니다.

```
python Practice01-1.py
또는
python.exe Practice01-1.py
```



[그림 01. Practice01-1]

1-2. [Practice01-2.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

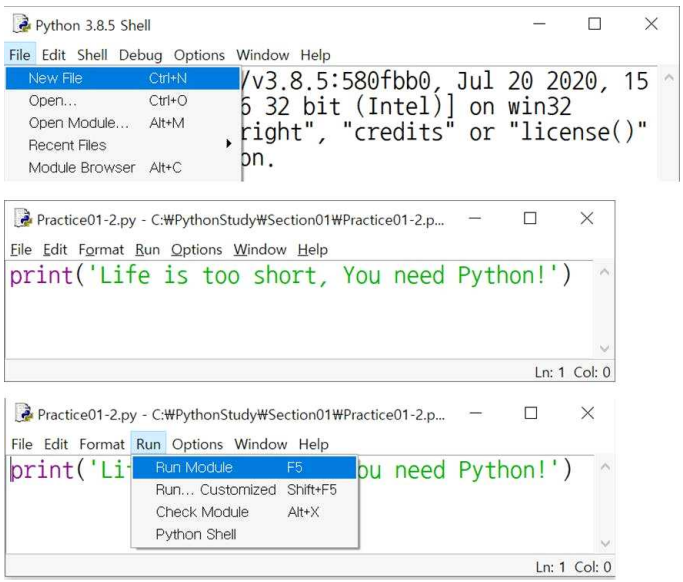
2) 아래와 같은 코드를 작성합니다.

```
01  print('Life is too short, You need Python!')
```

3) 작성한 파일을 C:\PythonStudy\Section01 디렉터리에 Practice01-2.py 파일명으로 저장합니다.

4) [Run] - [Run Module] 메뉴를 실행합니다. 또는 바로 가기 키 F5를 누릅니다.

5) Practice01-2.py 파일을 닫습니다.



[그림 02. Practice01-2]

2-1. [Practice02-1.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01 | student = '31025'
02 | grade_no = student[0]
03 | class_no = student[1:3]
04 | stu_no = student[3:]
05 | print(grade_no, '학년', class_no, '반', stu_no, '번')
```

3) C:\PythonStudy\Section02 디렉터리에 Practice02-1.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice02-1.py 파일을 닫습니다.

해설)

01 : student 변수에 학번을 문자열(str) 타입으로 저장합니다.
02 : 인덱스는 0부터 시작하므로 student[0]은 첫 번째 문자 '3'을 의미합니다.
03 : student[1:3]은 인덱스 1 이상 3 미만을 의미합니다. 즉 인덱스 1과 2의 문자 '10'을 의미합니다.
04 : student[3:]은 인덱스 3부터 끝까지를 의미합니다. 즉 인덱스 3과 4의 문자 '25'를 의미합니다. 마이너스 인덱스도 지원하므로 student[-2:]도 가능합니다.

참고)

student 변수의 각 문자와 인덱스를 확인해 보시길 바랍니다.

student	'3'	'1'	'0'	'2'	'5'
인덱스	0	1	2	3	4

2-2. [Practice02-2.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01 car1 = '서울2가1234'
02 car2 = '10가1234'
03 car3 = '288가1234'
04 car_no1 = car1[-4:]
05 car_no2 = car2[-4:]
06 car_no3 = car3[-4:]
07 print(car1, '의 차량번호 4자리는', car_no1, '입니다.')
08 print(car2, '의 차량번호 4자리는', car_no2, '입니다.')
09 print(car3, '의 차량번호 4자리는', car_no3, '입니다.')
```

3) C:\PythonStudy\Section02 디렉터리에 Practice02-2.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice02-2.py 파일을 닫습니다.

해설)

01 ~ 03 : 형식이 서로 다른 임의의 자동차 번호 3개를 변수에 저장합니다.

04 ~ 06 : 마이너스 인덱스를 지원하므로 뒤에서부터 4번째 글자부터 끝까지 출력하는 슬라이싱을 사용합니다.

참고)

각 변수의 마이너스 인덱스 값을 확인해 보시길 바랍니다.

car1	'서'	'울'	'2'	'가'	'1'	'2'	'3'	'4'
인덱스	-8	-7	-6	-5	-4	-3	-2	-1

car2	'1'	'0'	'가'	'1'	'2'	'3'	'4'
인덱스	-7	-6	-5	-4	-3	-2	-1

car3	'2'	'8'	'8'	'가'	'1'	'2'	'3'	'4'
인덱스	-8	-7	-6	-5	-4	-3	-2	-1

2-3. [Practice02-3.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  s = 'maple'
02  middle = s[len(s) // 2]
03  print(s, '의 가운데 글자는', middle, '입니다.')
```

3) C:\PythonStudy\Section02 디렉터리에 Practice02-3.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice02-3.py 파일을 닫습니다.

해설)

02 : 문자열의 길이를 2로 나눈 몫을 구해서 가운데 글자의 위치로 사용하고 있습니다.
문자열의 길이가 홀수이면 정확히 가운데 글자를 구할 수 있고,
문자열의 길이가 짝수이면 가운데 글자의 다음 글자를 구할 수 있습니다.

참고)

1) 길이가 4(짝수)인 경우:

s	'l'	'e'	'f'	't'
인덱스	0	1	2	3

2) 길이가 5(홀수)인 경우:

s	'm'	'a'	'p'	'l'	'e'
인덱스	0	1	2	3	4

2-4. [Practice02-4.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01 my_list = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
02 print('3번째 요소부터 7번째 요소 =', my_list[2:7])
03 print('3번째 요소부터 7번째 요소 중 2번째 요소 =', my_list[2:7][1])
```

3) C:\PythonStudy\Section02 디렉터리에 Practice02-4.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice02-4.py 파일을 닫습니다.

해설)

02 : my_list[2:7]은 인덱스 2 이상 7 미만을 의미합니다. 따라서 인덱스 2부터 6까지입니다. 인덱스는 0부터 시작하므로 3번째 요소부터 7번째 요소까지가 됩니다.

03 : my_list[2:7][1]은 3번째 요소부터 7번째 요소들 중에서 인덱스가 1인 요소입니다. 따라서 3번째 요소부터 7번째 요소 중 2번째 요소가 됩니다.

2-5. [Practice02-5.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  menu = {
02      '금': '탕수육',
03      '토': '유산슬',
04      '일': '팔보채'
05      }
06  print('금요일 :', menu['금'])
07  print('토요일 :', menu['토'])
08  print('일요일 :', menu['일'])
```

3) C:\PythonStudy\Section02 디렉터리에 Practice02-5.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice02-5.py 파일을 닫습니다.

해설)

01 ~ 04 : 딕셔너리 menu입니다. key는 '금', '토', '일'이고 각각의 value는 '탕수육', '유산슬', '팔보채'입니다.

04 : 딕셔너리는 key를 인덱스처럼 사용하고 해당 value를 알아낼 수 있습니다. 따라서 menu['금']은 '탕수육'입니다.

3-1. [Practice03-1.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  a = float(input('첫 번째 실수를 입력하세요 >>> '))
02  b = float(input('두 번째 실수를 입력하세요 >>> '))
03  total = a + b
04  print('{}와 {}의 합은 {}입니다.'.format(a, b, total))
```

3) C:\PythonStudy\Section03 디렉터리에 Practice03-1.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice03-1.py 파일을 닫습니다.

해설)

01 ~ 02 : input 함수로 입력 받은 모든 값의 타입은 문자열(str)입니다. 따라서 타입을 실수로 변경해주는 float() 함수가 필요합니다.

3-2. [Practice03-2.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  days = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
02  month = int(input('1~12 사이의 월을 입력하세요 >>> '))
03  day = days[month - 1]
04  print('{}월은 {}일까지 있습니다.'.format(month, day))
```

3) C:\PythonStudy\Section03 디렉터리에 Practice03-2.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice03-2.py 파일을 닫습니다.

해설)

01 : days 리스트에 1월부터 12월까지의 모든 일수를 순서대로 저장합니다.
02 : input() 함수로 입력 받은 값을 정수로 변경하기 위해서 int() 함수를 사용합니다.
03 : days 리스트의 인덱스는 0 ~ 11이지만 입력 받은 month는 1 ~ 12입니다. 이를 같은 값으로 조정하기 위해서 month - 1을 합니다.

3-3. [Practice03-3.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  english_dict = {
02      'flower': '꽃',
03      'fly': '날다',
04      'floor': '바닥'
05  }
06  word = input('영어 단어를 입력하세요 >>> ')
07  print('{}: {}'.format(word, english_dict[word]))
```

3) C:\PythonStudy\Section03 디렉터리에 Practice03-3.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice03-3.py 파일을 닫습니다.

해설)

01 ~ 04 : 딕셔너리 english_dict입니다. key는 영어 단어인 'flower', 'fly', 'floor'입니다. value는 각 영어 단어의 뜻인 '꽃', '날다', '바닥'입니다.

07 : english_dict['flower'] == '꽃'입니다. 영어 단어를 인덱스처럼 사용하면 해당 영어 단어의 저장된 뜻이 나타납니다.

3-4. [Practice03-4.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  s = set()
02  s.add(input('희망하는 수학여행지를 입력하세요 >>> '))
03  s.add(input('희망하는 수학여행지를 입력하세요 >>> '))
04  s.add(input('희망하는 수학여행지를 입력하세요 >>> '))
05  print('조사된 수학여행지는 {}입니다.'.format(s))
```

3) C:\PythonStudy\Section03 디렉터리에 Practice03-4.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice03-4.py 파일을 닫습니다.

해설)

01 : set를 생성합니다. s = {}를 이용하는 경우 세트 대신 딕셔너리가 만들어지기 때문에 주의해야 합니다.

02 ~ 04 : 입력된 값을 세트 s에 추가하기 위해서 add() 메소드를 사용했습니다. 세트의 특성상 같은 입력은 저장되지 않고 무시됩니다.

참고)

s.add(input('희망하는 수학여행지를 입력하세요 >>> ')) 이 코드가 3번 연속 반복되기 때문에 비효율적으로 보이는데 이것은 제어문을 통해서 간단히 해결할 수 있습니다. 아래 코드는 나중에 배우는 코드이니 지금은 참고만 해 주시길 바랍니다.

```
for i in range(3):
    s.add(input('희망하는 수학여행지를 입력하세요 >>> '))
```

4-1. [Practice04-1.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  number = int(input('10 ~ 99 사이의 정수를 입력하세요 >>> '))
02  tens = number // 10
03  units = number % 10
04  print('십의 자리: {}'.format(tens))
05  print('일의 자리: {}'.format(units))
```

3) C:\PythonStudy\Section04 디렉터리에 Practice04-1.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

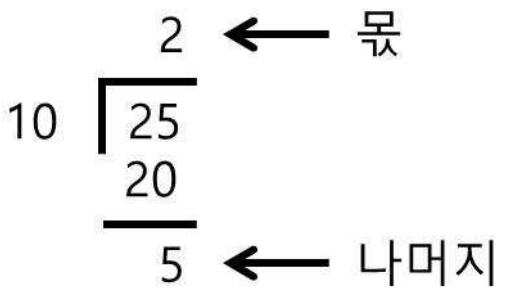
5) Practice04-1.py 파일을 닫습니다.

해설)

```
01 : 하나의 정수를 입력 받습니다.
02 : 입력된 정수를 10으로 나눈 몫을 tens 변수에 저장합니다.
03 : 입력된 정수를 10으로 나눈 나머지를 units 변수에 저장합니다.
```

참고)

2자리 정수의 경우 10으로 나눈 몫은 십의 자리를 의미하고 10으로 나눈 나머지는 일의 자리를 의미합니다.



[그림 03. Practice04-1]

4-2. [Practice04-2.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01 times = int(input('초를 입력하세요 >>> '))
02 hour = times // 3600
03 minute = times % 3600 // 60
04 second = times % 60
05 print('변환 결과는 {}시간 {}분 {}초입니다.'.format(hour, minute, second))
```

3) C:\PythonStudy\Section04 디렉터리에 Practice04-2.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice04-2.py 파일을 닫습니다.

해설)

01 : 정수 타입의 전체 시간을 초 단위로 입력 받습니다.
02 : 1시간은 3600초이므로 입력 받은 시간을 3600으로 나눈 몫은 시간(hour)이 됩니다.
03 : 입력 받은 시간을 3600으로 나누고 남은 나머지 시간은 1시간미만의 시간입니다. 그 시간을 60으로 나눈 몫은 분(minute)이 됩니다.
04 : 1분은 60초이므로 입력 받은 시간을 60으로 나눈 나머지는 모두 초(second)가 됩니다.

4-3. [Practice04-3.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  emp_no = int(input('4자리 사원번호를 입력하세요 >>> '))
02  emp_last_no = emp_no % 10
03  is_am = emp_last_no >= 5
04  print('근무 시간은 {}입니다.'.format('오전' if is_am else '오후'))
```

3) C:\PythonStudy\Section04 디렉터리에 Practice04-3.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice04-3.py 파일을 닫습니다.

해설)

01 : 4자리 숫자로 된 사원번호를 입력 받습니다.

02 : 사원번호를 10으로 나눈 나머지는 사원번호의 일의 자리(마지막 숫자)입니다.

03 : 사원번호의 마지막 숫자가 5 이상이면 is_am에는 True가 저장되고, 아니면 False가 저장됩니다.

04 : is_am 값이 True이면 '오전', 아니면 '오후'가 반환되는 조건 연산자를 사용하고 있습니다.

4-4. [Practice04-4.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01 print('한 박스에 20개의 라면을 담을 수 있습니다.')
02 print('라면의 개수를 입력하시면 필요한 박스 수를 알려드립니다.')
03 ramen = int(input('라면의 개수를 입력하세요 >>> '))
04 box = ramen // 20 if ramen % 20 == 0 else ramen // 20 + 1
05 print('{}개 라면을 담으려면 {}박스가 필요합니다.'.format(ramen, box))
```

3) C:\PythonStudy\Section04 디렉터리에 Practice04-4.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice04-4.py 파일을 닫습니다.

해설)

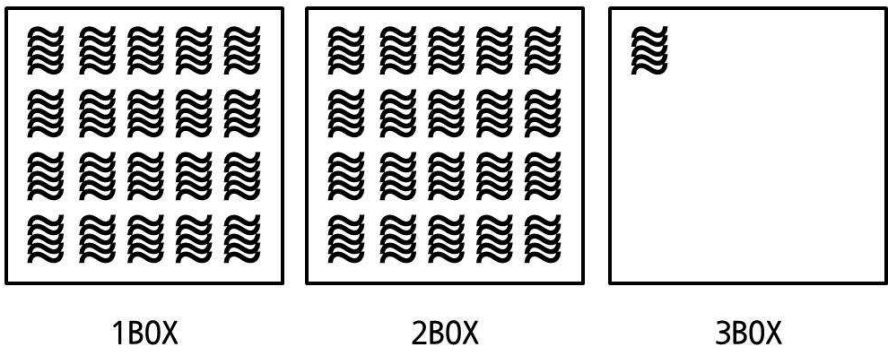
03 : 라면의 개수를 정수 타입으로 입력 받습니다.

04 : 라면의 개수가 20, 40, 60 과 같이 20의 배수이면 남는 라면 없이 모두 박스에 넣을 수 있습니다. 라면의 개수가 20의 배수가 아니라면 박스에 들어가지 못하고 남는 라면이 존재한다는 의미로 그 나머지 라면들을 담을 박스가 1개 더 필요합니다.

참고)

1) 라면이 40개가 있다면 필요한 박스는 (40 // 20)입니다.

2) 라면이 41개가 있다면 필요한 박스는 (41 // 20) + 1입니다.



[그림 04. Practice04-4]

4-5. [Practice04-5.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01 kor = int(input('국어 점수를 입력하세요 >>> '))
02 eng = int(input('영어 점수를 입력하세요 >>> '))
03 mat = int(input('수학 점수를 입력하세요 >>> '))
04 avg = (kor + eng + mat) / 3
05 result = '합격' if avg >= 80 else '불합격'
06 print('평균은 {}점이고, 결과는 {}입니다.'.format(avg, result))
```

3) C:\PythonStudy\Section04 디렉터리에 Practice04-5.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice04-5.py 파일을 닫습니다.

해설)

01 ~ 03 : 국어(kor), 영어(eng), 수학(mat) 점수를 정수로 입력 받습니다.
04 : 국어, 영어, 수학 점수의 평균을 구해서 avg 변수에 저장합니다.
05 : 평균(avg)이 80점 이상이면 '합격'이 result 변수에 저장되고 아니면(80점 미만이면) '불합격'이 result 변수에 저장됩니다.

5-1. [Practice05-1.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  score = int(input('점수를 입력하세요 >>> '))
02  if score >= 90:
03      grade = 'A'
04  elif score >= 80:
05      grade = 'B'
06  elif score >= 70:
07      grade = 'C'
08  elif score >= 60:
09      grade = 'D'
10  else:
11      grade = 'F'
12  print('점수는 {}점이고, 학점은 {}학점입니다.'.format(score, grade))
```

3) C:\PythonStudy\Section05 디렉터리에 Practice05-1.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice05-1.py 파일을 닫습니다.

해설)

01 : 정수 타입의 점수를 입력 받습니다.

02 : 점수(score)가 90 이상이면 grade 변수에 'A'를 저장합니다.

04 : elif는 else if의 줄임말입니다. 점수(score)가 90 이상이 아닌 경우에만 elif문이 동작합니다. 따라서 elif score >= 80:은 score < 90 and score >= 80:을 의미합니다.

06 : score < 80 and score >= 70:을 의미합니다.

08 : score < 70 and score >= 60:을 의미합니다.

10 : 나머지 모든 경우라는 의미로 score < 60:을 의미합니다.

5-2. [Practice05-2.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  n = int(input('정수를 입력하세요 >>> '))
02  if n % 3 == 0 and n != 0:
03      print('{}는 3의 배수입니다.'.format(n))
04  else:
05      print('{}는 3의 배수가 아닙니다.'.format(n))
```

3) C:\PythonStudy\Section05 디렉터리에 Practice05-2.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice05-2.py 파일을 닫습니다.

해설)

```
01 : 임의의 정수를 하나 입력 받습니다.
02 : 3의 배수는 0이 아닌 정수 중에서 3으로 나눈 나머지가 0인 수를 의미합니다.
```

5-3. [Practice05-3.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  n1 = int(input('정수1 입력 >>> '))
02  n2 = int(input('정수2 입력 >>> '))
03  n3 = int(input('정수3 입력 >>> '))
04  if n1 >= n2 and n1 >= n3:
05      print('가장 큰 수는 {}입니다.'.format(n1))
06  elif n2 >= n1 and n2 >= n3:
07      print('가장 큰 수는 {}입니다.'.format(n2))
08  else:
09      print('가장 큰 수는 {}입니다.'.format(n3))
```

3) C:\PythonStudy\Section05 디렉터리에 Practice05-3.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice05-3.py 파일을 닫습니다.

해설)

```
01 ~ 03 : 3개의 임의의 정수를 입력 받습니다.
04 : n1이 가장 큰 수인지 검사합니다.
06 : n2가 가장 큰 수인지 검사합니다.
08 : n1이나 n2가 가장 큰 수가 아니라면 n3가 가장 큰 수입니다.
```

5-4. [Practice05-4.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  car_no = input('차량번호를 입력하세요 >>> ')
02  if int(car_no[-1]) % 2 == 0:
03      result = '운행가능'
04  else:
05      result = '운행불가'
06  print('차량번호 {} 는 오늘 {}입니다.'.format(car_no, result))
```

3) C:\PythonStudy\Section05 디렉터리에 Practice05-4.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice05-4.py 파일을 닫습니다.

해설)

01 : 차량번호를 문자열 타입으로 입력 받습니다.
02 : 차량번호의 마지막 문자(car_no[-1])를 정수로 변환한 뒤 짝수인지 검사합니다.
정수로 변환하지 않으면 나머지 연산(%)이 되지 않습니다.

6-1. [Practice06-1.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  count = int(input('정수를 입력하세요 >>> '))
02  if count <= 0:
03      print('잘못된 입력입니다.')
04  else:
05      n = 0
06      while n < count:
07          print('{}번째 Hello'.format(n + 1))
08          n += 1
```

3) C:\PythonStudy\Section06 디렉터리에 Practice06-1.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice06-1.py 파일을 닫습니다.

해설)

01 : 임의의 정수를 하나 입력 받습니다.
02 : 0 이하의 정수는 처리하지 않습니다.
05 ~ 08 : print('{}번째 Hello'.format(n + 1))를 count만큼 반복하는 while문입니다.

참고)

사용자가 입력한 count 값이 5인 경우로 가정

n	n < count	'{}번째 Hello'.format(n + 1)	비고
0	True	1번째 Hello	while문 시작
1	True	2번째 Hello	
2	True	3번째 Hello	
3	True	4번째 Hello	
4	True	5번째 Hello	
5	False	X	while문 종료

6-2. [Practice06-2.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다. 2개의 코드를 준비했습니다.

```
01  n = 1
02  while n <= 100:
03      if n % 7 == 0:
04          print(n)
05      n += 1
```

```
01  n = 7
02  while n <= 100:
03      print(n)
04      n += 7
```

3) C:\PythonStudy\Section06 디렉터리에 Practice06-2.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice06-2.py 파일을 닫습니다.

해설)

01 : 1부터
02 : 100까지
05 : 1씩 증가하는 n을 사용합니다.
03 : n을 7로 나눈 나머지가 0인 수는 7의 배수입니다.

01 : 7부터
02 : 100까지
04 : 7씩 증가하는 n을 사용합니다. 7의 배수만 생성되기 때문에 별도로 검사할 필요가 없습니다.

6-3. [Practice06-3.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  coffee = 0
02  money = int(input('자판기에 얼마를 넣을까요? >>> '))
03  while money >= 300:
04      coffee += 1
05      money -= 300
06      print('커피 {}잔, 잔돈 {}원'.format(coffee, money))
```

3) C:\PythonStudy\Section06 디렉터리에 Practice06-3.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice06-3.py 파일을 닫습니다.

해설)

```
01 : 커피 잔 수(coffee)는 0으로 시작합니다.
02 : 임의의 돈(money)을 정수 타입으로 입력 받습니다.
03 : 커피 1잔이 300원이므로 money가 300 이상이면 커피를 계속 뽑을 수 있습니다.
04 : 커피를 1잔 추가할 때마다,
05 : 돈은 300원씩 줄어듭니다.
```

6-4. [Practice06-4.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  numbers = set()
02  while len(numbers) < 5:
03      n = int(input('0 ~ 9 사이 정수를 입력하세요 >>> '))
04      numbers.add(n)
05  print('모두 입력되었습니다.')
06  print('입력된 값은 {}입니다.'.format(numbers))
```

3) C:\PythonStudy\Section06 디렉터리에 Practice06-4.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice06-4.py 파일을 닫습니다.

해설)

01 : 비어 있는 세트 numbers를 생성합니다.

02 : 세트 numbers에 저장된 데이터가 5개 미만이면(4개까지 허용) 계속 작업을 수행합니다.

03 : 0 ~ 9 사이의 임의의 정수를 입력 받습니다.

04 : 입력 받은 정수를 세트 numbers에 저장합니다. 이 때 동일한 값은 저장되지 않고 무시됩니다. 중복된 데이터의 저장이 불가능한 것이 대표적인 세트의 특징입니다.

6-5. [Practice06-5.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  n = 1
02  while n <= 100:
03      if n % 10 == 0:
04          print(n)
05      else:
06          print(n, end='\t')
07      n += 1
```

3) C:\PythonStudy\Section06 디렉터리에 Practice06-5.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice06-5.py 파일을 닫습니다.

해설)

01 : 1부터
02 : 100까지
07 : 1씩 증가하는 n을 사용합니다.
03 : 10, 20, 30 과 같은 10의 배수를 출력한 뒤에는 줄 바꿈이 필요합니다. print(n)은 n을 출력한 뒤 줄을 바꿉니다.
05 : 10의 배수가 아닌 값을 출력한 뒤에는 줄 바꿈 대신 간격 조정을 위해서 탭 문자('\t')를 출력합니다.

6-6. [Practice06-6.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  n = 1
02  while n <= 9:
03      dan = 2
04      while dan <= 9:
05          print('{}×{}={}'.format(dan, n, dan * n), end='\t')
06          dan += 1
07      print()
08      n += 1
```

3) C:\PythonStudy\Section06 디렉터리에 Practice06-6.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice06-6.py 파일을 닫습니다.

해설)

가장 먼저 출력 형태를 분석해야 합니다. 출력의 첫 번째 줄을 살펴보겠습니다.

```
2x1=2   3x1=3   4x1=4   5x1=5   6x1=6   7x1=7   8x1=8   9x1=9
```

첫 번째 줄의 모든 출력은 n = 1로 고정되어 있습니다. n = 1로 고정시킨 상태로 dan을 2~9까지 모두 사용하므로 바깥쪽 while문에 n을 배치시키고, 안쪽 while문에 dan을 배치시키는 형태로 코드를 구성합니다.

```
01 : 1부터
02 : 9까지
08 : 1씩 증가하는 n을 사용합니다.
03 : 2부터
04 : 9까지
06 : 1씩 증가하는 dan을 사용합니다.
05 : 출력 후 줄 바꿈이 되지 않도록 end 속성을 지정합니다. 간격 조정을 위해 탭 문자('\t')를 사용합니다.
07 : 줄을 바꾸는 코드입니다.
```

7-1. [Practice07-1.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  for n in range(5, 0, -1):
02      print(n)
```

3) C:\PythonStudy\Section07 디렉터리에 Practice07-1.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice07-1.py 파일을 닫습니다.

해설)

01 : range(5, 0, -1) 함수는 5부터 0까지 -1씩 증가하는 값을 생성합니다. 실제로는 5는 포함하지만 0은 포함하지 않기 때문에 5부터 1까지 값이 생성됩니다.

7-2. [Practice07-2.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  total = 0
02  end = int(input('임의의 양수를 입력하세요 >>> '))
03  for n in range(0, end + 1):
04      total += n
05  print('1부터 {}사이 모든 정수의 합계는 {}입니다.'.format(n, total))
```

3) C:\PythonStudy\Section07 디렉터리에 Practice07-2.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice07-2.py 파일을 닫습니다.

해설)

01 : 합계를 구하는 변수 total은 0으로 초기화를 한 뒤에 사용해야 합니다.
02 : 임의의 양수를 하나 입력 받습니다.
03 : range(0, end + 1) 함수는 0부터 end까지 1씩 증가하는 값을 생성합니다. (step 값이 생략되면 1씩 증가한다는 의미입니다. range(0, end + 1, 1) 함수와 같은 의미입니다.)
04 : total이 n만큼 증가합니다.

참고)

만약 음수나 0이 입력되면 정상적으로 진행되지 않습니다. 따라서 재입력을 요구하도록 아래와 같이 코드를 구성할 수 있습니다.

```
01  total = 0
02  while True:
03      end = int(input('임의의 양수를 입력하세요 >>> '))
04      if end > 0:
05          break
06      else:
07          print('양수만 입력할 수 있습니다. 다시 입력하세요.')
08  for n in range(0, end + 1):
09      total += n
10  print('1부터 {}사이 모든 정수의 합계는 {}입니다.'.format(n, total))
```

7-3. [Practice07-3.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01 | count = int(input('몇 개의 과일을 보관할까요? >>> '))
02 | basket = []
03 | for n in range(count):
04 |     fruit = input('{}번째 과일을 입력하세요 >>> '.format(n + 1))
05 |     basket.append(fruit)
06 | print('입력 받은 과일들은 {}입니다.'.format(basket))
```

3) C:\PythonStudy\Section07 디렉터리에 Practice07-3.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice07-3.py 파일을 닫습니다.

해설)

01 : 임의의 정수를 입력 받습니다.

02 : 비어 있는 리스트 basket을 생성합니다.

03 : range(count) 함수는 range(0, count) 또는 range(0, count, 1) 과 같은 의미입니다. 0부터 count - 1까지 1씩 증가하는 값을 생성합니다. 예를 들어 count가 5라고 가정하면 range(count)는 '0, 1, 2, 3, 4'를 생성합니다.

04 : 과일(fruit)을 문자열 타입으로 입력 받습니다.

05 : 입력 받은 과일을 basket 리스트에 추가하는 append() 메소드가 사용되었습니다.

7-4. [Practice07-4.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  exam = [99, 78, 100, 91, 81, 85, 54, 100, 71, 50]
02  score = [min(n + 5, 100) for n in exam]
03  print(score)
```

3) C:\PythonStudy\Section07 디렉터리에 Practice07-4.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice07-4.py 파일을 닫습니다.

해설)

01 : 10개의 점수를 저장하고 있는 exam 리스트를 생성합니다.

02 : for n in exam을 통해서 모든 점수는 n에 저장됩니다. 모든 점수 n은 min(n + 5, 100) 함수에서 사용됩니다. min 함수는 기존 점수보다 5점이 증가한 값(n + 5)과 100점 중에서 작은 값을 반환합니다. 만약 기존 점수보다 5점이 증가한 값이 100 보다 크다면 min 함수의 결과는 100이 됩니다.

7-5. [Practice07-5.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  for n in range(1, 100):
02      units = n % 10
03      tens = n // 10
04      condition1 = units % 3 == 0 and units != 0
05      condition2 = tens % 3 == 0 and tens != 0
06      if condition1 and condition2:
07          print('짝짝', end='\t')
08      elif condition1 or condition2:
09          print('짝', end='\t')
10      else:
11          print(n, end='\t')
12      if n % 10 == 0:
13          print()
```

3) C:\PythonStudy\Section07 디렉터리에 Practice07-5.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice07-5.py 파일을 닫습니다.

해설)

01 : range(1, 100) 함수는 1부터 99까지 생성합니다.
02 : n % 10은 일의 자리를 의미합니다. 따라서 units는 일의 자리를 저장합니다.
03 : n // 10은 십의 자리를 의미합니다. 따라서 tens는 십의 자리를 저장합니다.
04 : 일의 자리가 3, 6, 9인지 검사합니다. 일의 자리가 3, 6, 9중 하나이면 condition1에는 True가 저장되고 아니면 False가 저장됩니다.
05 : 십의 자리가 3, 6, 9인지 검사합니다. 십의 자리가 3, 6, 9중 하나이면 condition2에는 True가 저장되고 아니면 False가 저장됩니다.
06 : 일의 자리와 십의 자리가 모두 3, 6, 9중 하나이면 '짝짝'을 출력합니다.
08 : 일의 자리와 십의 자리 중 하나가 3, 6, 9중 하나이면 '짝'을 출력합니다.
10 : 일의 자리와 십의 자리가 모두 3, 6, 9가 아니면 숫자를 그대로 출력합니다.
12 : 10, 20, 30 과 같이 10의 배수를 출력하고 나면 줄을 바꿉니다.

8-1. [Practice08-1.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  money = 10000
02  while True:
03      print('현재 {}원이 있습니다.'.format(money))
04      if money == 0:
05          break
06      spend = int(input('사용할 금액 입력 >>> '))
07      if spend <= 0:
08          print('0 이하의 금액은 사용할 수 없습니다.')
09      elif spend > money:
10          print('{}원이 부족합니다.'.format(spend - money))
11      else:
12          money -= spend
```

3) C:\PythonStudy\Section08 디렉터리에 Practice08-1.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice08-1.py 파일을 닫습니다.

해설)

```
01 : money는 10000에서 시작합니다.
02 : 무한루프입니다. 종료를 위한 break문이 내부에 존재해야 합니다.
04 : money가 0인 경우가 종료 조건입니다.
06 : 사용할 금액을 입력 받습니다.
07 : 0 이하의 금액은 사용하지 않습니다.
08 : money보다 큰 금액은 사용하지 않습니다.
09 : 0보다 크고 money보다 작은 금액을 사용합니다.
```

8-2. [Practice08-2.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  while True:
02      grade = int(input('이번 영화의 평점을 입력하세요 >>> '))
03      if grade >= 1 and grade <= 5:
04          print('평점: {}'.format('★' * grade))
05          break
06      else:
07          print('평점은 1~5 사이만 입력할 수 있습니다.')
```

3) C:\PythonStudy\Section02 디렉터리에 Practice08-2.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice08-2.py 파일을 닫습니다.

해설)

01 : 무한루프로 구성되면 break문을 통해서 종료시켜야 합니다.
02 : 정수 타입의 평점(grade)을 하나 입력 받습니다.
03 : 1부터 5사이의 평점만 사용합니다.
04 : '★' * grade는 '★'을 grade만큼 반복한 결과를 반환합니다. 예를 들어 '★' * 5는 '★★★★★'입니다.

8-3. [Practice08-3.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  answer = 'qwerty'
02  count = 0
03  while True:
04      if count == 5:
05          print('비밀번호 입력 횟수를 초과했습니다.')
06          break
07      pw = input('비밀번호를 입력하세요 >>> ')
08      if pw == answer:
09          print('비밀번호를 맞혔습니다.')
10          break
11      count += 1
```

3) C:\PythonStudy\Section08 디렉터리에 Practice08-3.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice08-3.py 파일을 닫습니다.

해설)

01 : 비밀번호 정답(answer)은 'qwerty'입니다.
02 : 입력 시도 횟수(count)는 0에서 시작합니다.
03 : 무한루프입니다.
04 : 입력 시도 횟수(count)는 0부터 4까지만 허용됩니다. 5는 입력 횟수 초과를 의미하므로 곧바로 종료됩니다.
07 : 비밀번호를 하나 입력 받습니다.
08 : 비밀번호를 맞히면 곧바로 종료됩니다.
11 : 입력 시도 횟수가 증가합니다.

8-4. [Practice08-4.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  for dan in range(2, 10):
02      if dan % 2 == 0:
03          print()
04          continue
05      for n in range(1, 10):
06          if dan < n:
07              break
08          print('{}x{}={}'.format(dan, n, dan * n))
```

3) C:\PythonStudy\Section08 디렉터리에 Practice08-4.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice08-4.py 파일을 닫습니다.

해설)

01 : range(2, 10) 함수는 2부터 9까지 생성합니다. 따라서 2단부터 9단까지 사용됩니다.
02 : 2, 4, 6, 8단(2의 배수)는 continue문을 실행합니다. continue문은 루프로 되돌아가는 코드입니다. 따라서 2, 4, 6, 8단은 05번 라인 이하로 내려가지 못하므로 출력되지 않습니다.
05 : range(1, 10) 함수는 1부터 9까지 생성합니다. 따라서 1부터 9까지 사용됩니다.
06 : dan과 n이 같을 때까지만 출력되고 dan보다 n이 커지면 바로 종료됩니다.

9-1. [Practice09-1.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  # 첫 번째 풀이입니다.
02  rainbow = ['red', 'orange', 'yellow', 'green', 'blue', 'navy', 'purple']
03  for no, color in enumerate(rainbow):
04      print('무지개의 {}번째 색은 {}입니다.'.format(no + 1, color))
05
06  print()
07
08  # 두 번째 풀이입니다.
09  for idx in range(len(rainbow)):
10      print('무지개의 {}번째 색은 {}입니다.'.format(idx + 1, rainbow[idx]))
```

3) C:\PythonStudy\Section09 디렉터리에 Practice09-1.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice09-1.py 파일을 닫습니다.

해설)

문제 해결을 위한 2가지 해결 방법을 각각 제시하고 있습니다. 첫 번째 해결 방법은 enumerate() 함수를 이용하는 방법이고 두 번째 방법은 range()와 len() 함수를 이용하는 방법입니다.

02 : 7개의 색상을 저장하고 있는 rainbow 리스트를 생성합니다.

03 : enumerate(rainbow) 함수는 (0, 'red') 와 같은 형식으로 리스트에 저장된 각 요소들의 인덱스와 요소를 모두 반환합니다. 인덱스는 no에 저장되고 'red'는 color에 저장됩니다.

09 : len(rainbow) 함수의 결과는 7입니다. 따라서 range(len(rainbow)) 함수는 range(7)과 같습니다. range(7)은 0부터 6까지를 생성하므로 rainbow 리스트의 인덱스를 만드는 코드입니다.

9-2. [Practice09-2.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  exam = []
02  print('점수를 입력하세요.')
03  print('더 이상 입력할 점수가 없으면 음수를 아무거나 입력하세요.')
04  while True:
05      score = int(input('점수 입력 >>> '))
06      if score < 0:
07          break
08      else:
09          exam.append(score)
10  average = sum(exam) / len(exam)
11  maximum = max(exam)
12  minimum = min(exam)
13  print('평균 = {}, 최대 = {}, 최소 = {}'.format(average, maximum, minimum))
```

3) C:\PythonStudy\Section09 디렉터리에 Practice09-2.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice09-2.py 파일을 닫습니다.

해설)

01 : 점수를 저장할 비어 있는 exam 리스트를 준비합니다.
04 : 무한루프로 구성합니다.
05 : 정수 타입의 점수를 하나 입력 받습니다.
06 : 입력된 정수가 0보다 작은 음수이면 바로 종료됩니다.
08 : 입력된 정수가 0 이상이면 exam 리스트에 추가합니다.
10 : sum(exam) 함수는 exam 리스트의 모든 요소를 더한 합계를 반환합니다. len(exam) 함수는 exam 리스트에 저장된 모든 요소의 개수입니다. 따라서 sum(exam) / len(exam) 의 결과는 평균입니다.
11 : max(exam) 함수는 exam 리스트의 모든 요소 중에서 가장 큰 값을 반환합니다.
12 : min(exam) 함수는 exam 리스트의 모든 요소 중에서 가장 작은 값을 반환합니다.

10-1. [Practice10-1.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다. 2개의 코드를 준비했습니다.

```
01 phone = input('전화번호를 입력하세요 >>> ')
02 start = phone.index('-') + 1
03 end = phone.index('-', start)
04 code = phone[start:end]
05 print(code)
```

```
01 phone = input('전화번호를 입력하세요 >>> ')
02 code = phone.split('-')[1]
03 print(code)
```

3) C:\PythonStudy\Section10 디렉터리에 Practice10-1.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

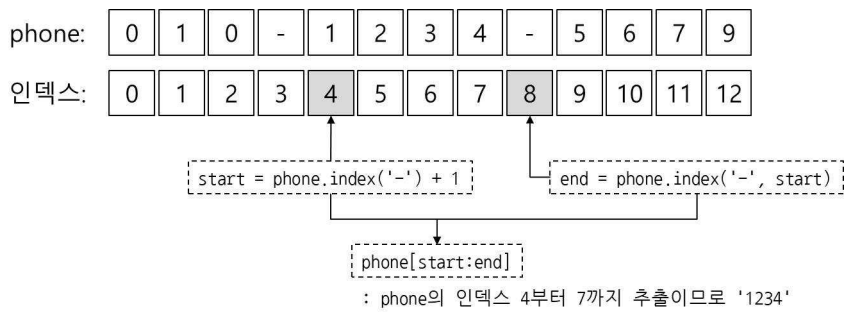
5) Practice10-1.py 파일을 닫습니다.

해설)

01 : 문자열 타입의 전화번호를 입력 받습니다.
02 : phone.index('-') 메소드는 phone에서 첫 번째 하이픈('-')의 인덱스를 반환합니다.
03 : phone.index('-', start) 메소드는 phone의 start 인덱스부터 찾기 시작하여 첫 번째로 나타나는 하이픈('-')의 인덱스를 반환합니다.
04: phone[start:end] 는 phone의 start 인덱스부터 end 이전 인덱스까지 추출합니다.

01 : 문자열 타입의 전화번호를 입력 받습니다.
02 : phone이 '010-1234-5678'인 경우 phone.split('-') 메소드의 결과는 ['010', '1234', '4567']입니다. 국번은 2번째 데이터이므로 인덱스 1을 추가하여 phone.split('-')[1]이라고 작성합니다.

참고)



[그림 05. Practice10-1]

10-2. [Practice10-2.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  no = input('사업자등록번호를 입력하세요(예: 123-45-67890) >>> ')
02
03  condition1 = (no.find('-') == 3)
04  condition2 = (no.find('-', 4) == 6)
05  condition3 = (len(no) == 12)
06  condition4 = (no.replace('-', '').isdecimal())
07  if condition1 and condition2 and condition3 and condition4:
08      print('올바른 형식입니다.')
09  else:
10      print('올바른 형식이 아닙니다.')
```

3) C:\PythonStudy\Section10 디렉터리에 Practice10-2.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice10-2.py 파일을 닫습니다.

해설)

01 : 문자열 타입의 사업자등록번호를 입력 받아서 no에 저장합니다.

03 : no.find('-') 메소드는 사업자등록번호를 처음부터 검색하여 발견되는 첫 번째 하이픈('-')의 인덱스를 반환합니다. 올바른 형식의 사업자등록번호는 3이 반환됩니다.

04 : 사업자등록번호가 올바른 형식이라면 첫 번째 하이픈의 인덱스가 3이기 때문에 두 번째 하이픈은 인덱스 4부터 검색해야 합니다. no.find('-', 4) 메소드는 사업자등록번호의 인덱스 4 위치부터 검색하여 발견되는 첫 번째 하이픈('-')의 인덱스를 반환합니다.

05 : len(no) 함수는 사업자등록번호의 길이를 반환합니다. 올바른 형식의 사업자등록번호는 12입니다.

06 : no.replace('-', '') 메소드를 통해서 사업자등록번호의 모든 하이픈을 제거합니다. 그런 다음 isdecimal() 메소드를 통해서 숫자인지 아닌지 검사합니다. isdecimal() 메소드는 숫자의 경우 True를 반환합니다.

07 : 4개의 모든 조건을 만족하는 경우 08라인이 실행됩니다.

10-3. [Practice10-3.py]

풀이)

1) IDLE를 실행하고 [File] - [New File] 메뉴를 선택해서 스크립트 모드 창을 엽니다.

2) 아래와 같은 코드를 작성합니다.

```
01  student = '"김철수",85'
02  name = student.split(',')[0].strip('"')
03  age = student.split(',')[1]
04  print('이름은 {}이고, 점수는 {}점입니다.'.format(name, age))
```

3) C:\PythonStudy\Section10 디렉터리에 Practice10-3.py 파일명으로 저장합니다.

4) 바로 가기 키 F5를 눌러서 실행합니다.

5) Practice10-3.py 파일을 닫습니다.

해설)

01 : student 변수 값은 이름과 나이가 콤마(,)를 이용해 연결된 상태입니다. 어떤 문자열의 값을 분리하는 split() 메소드를 사용하는 것이 좋습니다. student.split(',') 메소드를 사용하면 student 변수 값이 콤마(,)로 분리되어 ['"김철수"', '85'] 리스트가 반환됩니다.

02 : student.split(',')[0] 의 결과는 '"김철수"'입니다. 앞뒤로 불필요한 큰 따옴표가 남아 있기 때문에 이를 제거하기 위해서 strip('"') 메소드가 사용됩니다. strip() 메소드는 지정된 문자를 문자열의 앞뒤에서 모두 제거하는 메소드입니다.

03 : student.split(',')[1] 의 결과는 '85'입니다. 만약 숫자로 변환해야 한다면 int() 함수를 추가로 사용해서 age = int(student.split(',')[1])로 처리해야 합니다.

11-1. [Practice11-1.py]

*** Section11부터는 모범답안과 해설만 제시합니다. 풀이 과정은 기존과 동일합니다.**

모범답안) C:\PythonStudy\Section11\Practice11-1.py

```
01  def vending_machine(money):
02      price = 700
03      count = money // price
04      for drink in range(count + 1):
05          change = money - drink * price
06          print('음료수 = {}개, 잔돈 = {}원'.format(drink, change))
07
08  vending_machine(3000)
```

해설)

01 : vending_machine() 함수를 정의합니다. 매개변수는 money입니다.
02 : 변수 price는 음료수 한 잔의 가격입니다.
03 : 변수 count는 최대로 구매 가능한 음료수의 개수입니다.
04 : 변수 drink는 음료수를 의미합니다. range(count + 1) 함수는 0부터 count까지 생성하므로 drink를 0개부터 count개까지 사용할 수 있습니다.
05 : 변수 change는 잔돈을 의미합니다.
08 : vending_machine() 함수를 호출합니다. 인수 3000은 매개변수는 money로 전달되므로 vending_machine() 함수는 money를 3000으로 설정하고 실행됩니다.

11-2. [Practice11-2.py]

모범답안) C:\PythonStudy\Section11\Practice11-2.py

```
01  def get_average(marks):
02      total = 0
03      for subject in marks:
04          total += marks[subject]
05      average = total / len(marks)
06      return average
07
08  marks = {'국어': 90, '영어': 80, '수학': 85}
09  average = get_average(marks)
10  print('평균은 {}점입니다.'.format(average))
```

해설)

01 : get_average() 함수를 정의합니다. 매개변수는 marks입니다.
02 : 변수 total은 합계를 저장할 변수이므로 초기화 값이 0입니다.
03 : marks에 딕셔너리가 전달되면 subject에는 딕셔너리의 key 값이 전달됩니다.
04 : marks가 딕셔너리이고 subject가 key이므로 marks[subject]는 value입니다. 점수가 value에 저장되어 있으므로 모든 점수의 합계를 구하는 코드입니다.
05 : 점수의 합계가 total에 저장되어 있고 len(marks)는 과목의 개수를 반환하므로 변수 average는 평균입니다.
06 : 평균(average)을 반환합니다. 09번 라인의 get_average() 함수를 호출한 곳으로 average를 반환합니다.
08 : marks는 과목명이 key이고 점수가 value인 딕셔너리입니다.
09 : get_average() 함수를 호출합니다. 인수 marks는 매개변수는 marks로 전달됩니다.

11-3. [Practice11-3.py]

모범답안) C:\PythonStudy\Section11\Practice11-3.py

```
01  total = 0
02
03  def gift(dic, who, money):
04      global total
05      total += money
06      dic[who] = money
07
08  wedding = {}
09  gift(wedding, '영희', 5)
10  gift(wedding, '철수', 3)
11  gift(wedding, '이모', 10)
12  print('축의금 명단: {}'.format(wedding))
13  print('전체 축의금: {}'.format(total))
```

해설)

01 : 전역변수 total입니다. 합계를 구하기 위해서 0으로 초기화되어 있습니다.
03 : gift() 함수를 정의합니다. 매개변수는 dic, who, money입니다. 각각 딕셔너리, key, value가 전달될 변수들입니다.
04 : 전역변수 total의 값을 함수 내부에서 변경하려면 global 키워드를 통해서 선언해야 합니다.
05 : 변수 money는 축의금입니다. total에 축의금의 누적 합계를 구합니다.
06 : dic은 딕셔너리로 축의금을 낸 사람 이름(who)과 금액(money)을 함께 보관합니다.
08 : wedding은 비어 있는 딕셔너리입니다.
09 : gift() 함수를 호출합니다. 인수 wedding, '영희', 5는 매개변수 dic, who, money에 전달됩니다.

12-1. [Practice12-1.py]

모범답안) C:\PythonStudy\Section12\Practice12-1.py

```
01  import random
02  import time
03
04  pot = [n for n in range(1, 46)]
05  jackpot = []
06
07  for n in range(1, 7):
08      random.shuffle(pot)
09      pick = pot.pop()
10      print('{}번째 당첨번호는 {}입니다.'.format(n, pick))
11      jackpot.append(pick)
12      time.sleep(2)
13
14  jackpot.sort()
15  print('이번 당첨번호는 {} 입니다.'.format(jackpot))
```

해설)

01 : shuffle() 메소드를 호출하기 위해서 추가된 import입니다.
02 : sleep() 메소드를 호출하기 위해서 추가된 import입니다.
04 : pot은 리스트이며 1부터 45까지 정수 값을 가지고 있습니다.
05 : jackpot은 리스트입니다. 당첨번호 6개를 저장합니다.
07 : 6번 반복합니다. 매번 실행될 때마다 당첨번호가 1개씩 결정됩니다.
08 : pot 리스트를 임의로 섞습니다. pot에 저장된 숫자들이 어떤 순서로 저장되어 있는지 알 수 없습니다.
09 : pot 리스트에서 마지막 요소를 하나 빼서 당첨번호(pick) 변수에 저장합니다.
11 : 당첨번호(pick)를 당첨번호를 저장할 jackpot 리스트에 저장합니다.
12 : 2초간 잠시 시스템이 멈춥니다.
14 : 당첨번호들을 크기순으로 정렬합니다.

12-2. [Practice12-2.py]

모범답안) C:\PythonStudy\Section12\Practice12-2.py

```
01  import random
02  import time
03  import math
04
05  answer = random.randint(1, 100)
06
07  print('UpDown게임을 시작합니다.')
08  start = time.time()
09  while True:
10      n = int(input('어떤 값일까요? >>> '))
11      if answer == n:
12          print('정답입니다.')
13          break
14      elif answer < n:
15          print('Down')
16      else:
17          print('Up')
18  end = time.time()
19
20  elapse = end - start
21  print('{}초 만에 성공했습니다.'.format(math.floor(elapse)))
```

해설)

```
01 : randint() 메소드를 호출하기 위해서 추가된 import입니다.
02 : time() 메소드를 호출하기 위해서 추가된 import입니다.
03 : floor() 메소드를 호출하기 위해서 추가된 import입니다.
05 : 1 ~ 100사이의 난수를 생성합니다. 이 난수(answer)를 사용자가 맞혀야 합니다.
08 : 게임 시작 시간을 저장해 둡니다. 시간을 재기 위함입니다.
09 : 난수(answer)를 맞힐 때까지 무한루프로 동작합니다.
10 : 사용자로부터 임의의 정수를 하나 입력 받습니다.
11 : 난수를 맞히면 게임이 종료됩니다.
14 : 입력된 값보다 난수가 작으면 'Down'을 출력합니다.
16 : 입력된 값보다 난수가 크면 'Up'을 출력합니다.
18 : 게임 종료 시간을 저장해 둡니다.
20 : 게임에 걸린 시간(elapse)을 구합니다.
21 : 게임에 걸린 시간은 밀리초 단위(소수 이하 자리)도 있으므로 이를 내림 처리합니다.
```

13-1. [Practice13-1.py]

모범답안1) C:\PythonStudy\Section13\Practice13-1.py

```
01  nation = ['그리스', '아테네', '독일', '베를린', '러시아', '모스크바', '미국', '워싱턴']
02  file = open('nation.txt', 'wt')
03  for i in range(0, len(nation), 2):
04      file.write('{}-{}\n'.format(nation[i], nation[i + 1]))
05  file.close()
06
07  print('생성된 nation.txt 파일의 내용은 다음과 같습니다.')
08  file = open('nation.txt', 'rt')
09  line_list = file.readlines()
10  for line in line_list:
11      print(line, end='')
12  file.close()
```

해설1)

01 : 국가와 수도 순으로 저장된 nation 리스트입니다.
02 : nation.txt 파일을 새로 생성합니다.
03 : range(0, len(nation), 2) 함수는 0, 2, 4, 6을 생성합니다. 모두 국가가 저장된 인덱스입니다. 따라서 변수 i는 국가가 저장된 인덱스입니다.
04 : 변수 i는 국가의 인덱스이므로 nation[i]는 국가이고, nation[i + 1]은 해당 국가의 수도입니다. 따라서 '국가-수도' 형식으로 nation.txt 파일이 생성됩니다.
08~12 : nation.txt 파일을 읽어서 화면에 출력합니다. readlines() 메소드는 파일 전체를 읽어서 한 줄씩 리스트에 저장하는 메소드입니다.

모범답안2) C:\PythonStudy\Section13\Practice13-1.py

```
01 nation = ['그리스', '아테네', '독일', '베를린', '러시아', '모스크바', '미국', '워싱턴']
02 file = open('nation.txt', 'wt')
03 for i, country in enumerate(nation):
04     if i % 2 == 0:
05         file.write('{}-{}\n'.format(nation[i], nation[i + 1]))
06 file.close()
07
08 print('생성된 nation.txt 파일의 내용은 다음과 같습니다.')
09 file = open('nation.txt', 'rt')
10 line_list = file.readlines()
11 for line in line_list:
12     print(line, end='')
13 file.close()
```

해설2)

03 : enumerate(nation) 함수는 nation 리스트의 모든 인덱스와 내용을 추출합니다.
04 : 인덱스(i)가 짝수인지 검사합니다. 인덱스가 짝수이면 국가입니다.
05 : 변수 i는 짝수이므로 국가의 인덱스입니다. 따라서 모범답안1과 같은 형식으로 nation.txt 파일을 생성합니다.
08~12 : nation.txt 파일을 읽어서 화면에 출력합니다. readlines() 메소드는 파일 전체를 읽어서 한 줄씩 리스트에 저장하는 메소드입니다.

13-2. [Practice13-2.py]

모범답안) C:\PythonStudy\Section13\Practice13-2.py

```
01  prev_file = open('연락처.txt', 'rt')
02  buffer = prev_file.read()
03  n = buffer.count('"011-')
04  buffer = buffer.replace('"011-', '"010-')
05  print('총 {}건의 011 데이터를 찾았습니다.'.format(n))
06  prev_file.close()
07
08  new_file = open('연락처.txt', 'wt')
09  new_file.write(buffer)
10  new_file.close()
11  print('모든 데이터를 수정했습니다.')
```

해설)

```
01 : 연락처.txt 파일을 읽기 모드로 엽니다.
02 : 연락처.txt 파일의 전체 내용을 변수 buffer에 저장합니다.
03 : 변수 buffer에 포함된 모든 '"011-'의 개수를 구해서 변수 n에 저장합니다.
04 : 변수 buffer에 포함된 모든 '"011-'을 '"010-'으로 수정합니다.
06 : 연락처.txt 파일의 읽기 모드를 종료합니다.
08 : 연락처.txt 파일을 쓰기 모드로 다시 엽니다.
09 : 모든 수정 내용이 저장된 변수 buffer의 내용으로 연락처.txt 파일을 새로 생성합니
다. 기존의 연락처.txt 파일은 삭제됩니다.
10 : 연락처.txt 파일의 쓰기 모드를 종료합니다.
```

14-1. [Practice14-1.py]

모범답안) C:\PythonStudy\Section14\Practice14-1.py

```
01  while True:
02      filename = input('복사할 파일명을 입력하세요 >>> ')
03      extname = filename[filename.rfind('.') + 1:]
04      if extname != 'txt':
05          print('복사할 수 없는 파일입니다.')
06      else:
07          break
08
09  with open(filename, 'rt') as source:
10      with open('복사본-' + filename, 'wt') as copy:
11          while True:
12              buffer = source.read(1)
13              if not buffer:
14                  break
15              copy.write(buffer)
16
17  print('복사본-' + filename + ' 파일이 생성되었습니다.')
```

해설)

02 : 사용자로부터 복사할 파일의 이름(filename)을 입력 받습니다.
03 : 입력 받은 파일의 이름에서 확장자(extname)를 분리합니다.
04 : 확장자(extname)가 'txt'가 아니면 에러 메시지를 출력하고 복사할 파일 이름을 다시 입력 받습니다.
06 : 확장자(extname)가 'txt'이면 복사할 파일명 입력을 종료합니다.
09 : 복사할 원본 파일을 읽기 모드로 엽니다. with문으로 열면 close()가 자동으로 진행되기 때문에 구현할 필요가 없습니다.
10 : 복사본 파일을 쓰기 모드로 엽니다. 새로 생성되는 복사본 파일의 이름은 원본 파일명 앞에 '복사본-'이 추가된 형태입니다.
12 : 복사할 원본 파일에서 한 글자를 읽어서 변수 buffer에 저장합니다.
13 : 저장된 글자가 없으면 복사할 원본 파일을 모두 읽은 것이므로 파일 복사를 종료합니다.
15 : 변수 buffer에 저장된 내용으로 복사본 파일을 생성합니다.

14-2. [Practice14-2.py]

모범답안) C:\PythonStudy\Section14\Practice14-2.py

```
01  import csv
02
03  with open('cctv.csv', 'r') as csvfile:
04      buffer = csv.reader(csvfile, delimiter=',', quotechar='"')
05      totalcctv = 0
06      for i, line in enumerate(buffer):
07          if i != 0:
08              totalcctv += int(line[4])
09
10  print('서울특별시 마포구에 설치된 CCTV는 총 {}대입니다.'.format(totalcctv))
```

해설)

01 : csv 모듈을 import하면 csv.reader() 메소드를 통해서 쉽게 CSV 데이터를 읽을 수 있습니다.
03 : cctv.csv 파일을 읽기 모드로 엽니다.
04 : cctv.csv 파일을 읽어 들입니다. CSV파일을 읽을 때는 구분자(delimiter)는 쉼표(,)로 설정하고 쉼표(,)가 구분자가 아닌 경우에는 큰 따옴표(")로 데이터가 묶여 있으므로 quotechar를 큰 따옴표(")로 설정합니다. cctv.csv 파일을 읽어 들인 결과인 buffer는 cctv.csv의 전체 데이터를 저장하고 있는 객체입니다.
05 : cctv의 합계를 구할 변수 total입니다.
06 : 객체 buffer는 cctv.csv 파일의 모든 라인을 각각의 요소로 가지고 있는 리스트와 비슷합니다. 따라서 enumerate(buffer) 함수로 각 라인의 인덱스와 데이터를 가져올 수 있습니다.
07 : 첫 번째 라인은 제목 라인으로 원하는 데이터가 없으므로 제외합니다.
08 : 각 라인의 5번째 데이터가 카메라대수입니다. 따라서 line[4]가 카메라대수입니다.

참고)
cctv.csv 파일의 일부 모습입니다. 카메라대수는 5번째 데이터임을 알 수 있습니다.

```
"관리기관명",소재지도로명주소,소재지지번주소,설치목적구분,카메라대수,카메라화소수,
"서울특별시 마포구청",양화로 72,서교동 395-43,주정차단속,1,200,360도전방면,30,2005
```

[그림 06. Practice14-2]

14-3. [Practice14-3.py]

모범답안) C:\PythonStudy\Section14\Practice14-3.py

```
01  import json
02
03  with open('cctv.json', 'r', encoding='utf-8') as jsonfile:
04      buffer = jsonfile.read()
05      cctv_list = json.loads(buffer)
06      cctv_purpose = set()
07      for place in cctv_list:
08          cctv_purpose.add(place['설치목적구분'])
09
10  print(cctv_purpose)
```

해설)

01 : json 모듈을 import합니다.
03 : cctv.json 파일을 읽기 모드로 엽니다. cctv.json 파일은 UTF-8 형식으로 인코딩이 되어 있기 때문에 encoding='utf-8' 옵션을 추가해야 합니다. (encoding='UTF-8'이나 encoding='utf-8' 모두 가능합니다.)
04 : read() 메소드를 통해 cctv.json의 전체 내용을 읽어서 변수 buffer에 저장합니다.
05 : loads() 메소드를 통해 변수 buffer의 내용을 파이썬이 해석할 수 있는 정보로 변환합니다. 각 cctv 정보는 딕셔너리에 저장되고 이것들은 모두 리스트에 저장됩니다.
06 : cctv설치목적을 저장할 비어 있는 세트 cctv_purpose를 생성합니다.
07 : 각 cctv 정보를 place에 저장합니다. place는 딕셔너리 형식입니다.
08 : 딕셔너리는 key값으로 데이터를 가져옵니다. cctv설치목적은 '설치목적구분'이라는 key값을 가지고 있으므로 place['설치목적구분']으로 cctv설치목적을 가져올 수 있습니다. 세트는 중복 저장이 불가능하므로 중복은 제거된 상태로 저장됩니다.

15-1. [Practice15-1.py]

모범답안) C:\PythonStudy\Section15\Practice15-1.py

```
01  class Book:
02
03      def set_info(self, title, author):
04          self.title = title
05          self.author = author
06
07      def print_info(self):
08          print('책 제목: {}'.format(self.title))
09          print('책 저자: {}'.format(self.author))
10
11
12  # book1, book2 인스턴스의 생성
13  book1 = Book()
14  book2 = Book()
15
16  # book1, book2 제목과 저자 정보 저장
17  book1.set_info('파이썬', '민경태')
18  book2.set_info('어린왕자', '생텍쥐페리')
19
20  book_list = [book1, book2]
21  for book in book_list:
22      book.print_info()
```

해설)

01 : 클래스 Book입니다.
03 : set_info() 메소드입니다. title과 author를 받아서 인스턴스 변수(self.title, self.author)에 저장합니다.
07 : print_info() 메소드입니다. 인스턴스 변수의 정보를 출력합니다.
13 : book1 객체를 생성합니다.
14 : book2 객체를 생성합니다.
17 : book1 객체의 title을 '파이썬', author를 '민경태'로 설정합니다.
18 : book2 객체의 title을 '어린왕자', author를 '생텍쥐페리'로 설정합니다.
20 : book1과 book2 객체를 가지는 book_list 리스트를 생성합니다.
21~22 : book_list 리스트의 각 요소를 book으로 지정한 뒤 각 book의 정보를 출력합니다.

15-2. [Practice15-2.py]

모범답안) C:\PythonStudy\Section15\Practice15-2.py

```
01  class Watch:
02
03      def set_time(self):
04          now = input('시간을 입력하세요 >>> ')
05          hms = now.split(':')
06          self.hour = int(hms[0])
07          self.minute = int(hms[1])
08          self.second = int(hms[2])
09
10      def add_hour(self, hour):
11          if hour <= 0:
12              return
13          self.hour += hour
14          self.hour %= 24
15
16      def add_minute(self, minute):
17          if minute <= 0:
18              return
19          self.minute += minute
20          self.add_hour(self.minute // 60)
21          self.minute %= 60
22
23      def add_second(self, second):
24          if second <= 0:
25              return
26          self.second += second
27          self.add_minute(self.second // 60)
28          self.second %= 60
29
30      def see(self):
31          print('계산된 시간은 ', end='')
32          print('{}시 {}분 {}초입니다.'.format(self.hour, self.minute, self.second))
33
34
```

```
35  watch = Watch()
36  watch.set_time()
37  watch.add_hour(int(input('계산할 시간을 입력하세요 >>> ')))
38  watch.add_minute(int(input('계산할 분을 입력하세요 >>> ')))
39  watch.add_second(int(input('계산할 초를 입력하세요 >>> ')))
40  watch.see()
```

해설)

03 : set_time() 메소드는 시간을 h:mm:ss 형식으로 입력 받아서 이를 hour, minute, second로 분리합니다.
10 : add_hour() 메소드는 전달 받은 hour만큼 시간을 증가시키는 메소드입니다. hour는 0~23 사이의 값만 가질 수 있으므로 24로 나눈 나머지로 처리해야 합니다.
16 : add_minute() 메소드는 전달 받은 minute만큼 분을 증가시키는 메소드입니다. 전달 받은 minute은 hour와 minute으로 분리해서 처리해야 하므로(minute이 90이라면 1hour 30minute으로 변환해서 처리합니다.) hour 처리를 위한 add_hour() 메소드를 호출하고 있습니다. minute은 0~59 사이의 값만 가질 수 있으므로 60으로 나눈 나머지로 처리해야 합니다.
23 : add_second() 메소드는 전달 받은 second만큼 초를 증가시키는 메소드입니다. 전달 받은 second은 minute과 second으로 분리해서 처리해야 하므로(second이 90이라면 1minute 30second으로 변환해서 처리합니다.) minute 처리를 위한 add_minute() 메소드를 호출하고 있습니다. second은 0~59 사이의 값만 가질 수 있으므로 60으로 나눈 나머지로 처리해야 합니다.
30 : 계산된 시간을 출력하는 see() 메소드입니다.

15-3. [Practice15-3.py]

모범답안) C:\PythonStudy\Section15\Practice15-3.py

```
01  class Song:
02
03      def set_song(self, title, genre):
04          self.title = title
05          self.genre = genre
06
07      def print_song(self):
08          print('노래제목: {}({})'.format(self.title, self.genre))
09
10
11  class Singer:
12
13      def set_singer(self, name):
14          self.name = name
15
16      def hit_song(self, song):
17          self.song = song
18
19      def print_singer(self):
20          print('가수이름: {}'.format(self.name))
21          self.song.print_song()
22
23
24  song = Song()
25  song.set_song('취중진담', '발라드')
26  singer = Singer()
27  singer.set_singer('김동률')
28  singer.hit_song(song)
29  singer.print_singer()
```

해설)

01 : 클래스 Song입니다. title과 genre를 저장할 수 있는 set_song() 메소드와 출력할 수 있는 print_song() 메소드로 구성되어 있습니다.
11 : 클래스 Singer입니다. name을 저장할 수 있는 set_singer() 메소드와 song을 저장할 수 있는 hit_song() 메소드로 구성되어 있습니다.
21 : song은 클래스 Song의 객체이므로 print_song() 메소드를 사용할 수 있습니다.

16-1. [Practice16-1.py]

모범답안) C:\PythonStudy\Section16\Practice16-1.py

```
01  class Person:
02
03      population = 0
04
05      def __init__(self, name):
06          self.name = name
07          Person.population += 1
08          print('{} is born.'.format(self.name))
09
10      def __del__(self):
11          Person.population -= 1
12          print('{} is dead.'.format(self.name))
13
14      @classmethod
15      def get_population(cls):
16          return cls.population
17
18
19  man = Person('james')
20  woman = Person('emily')
21  print('전체 인구수: {}명'.format(Person.get_population()))
22  del man
23  print('전체 인구수: {}명'.format(Person.get_population()))
```

해설)

03 : population은 Person 클래스의 클래스 변수입니다. Person이 생성될 때마다 함께 증가하는 변수로 인구수를 의미합니다.
05 : 클래스 Person의 생성자입니다. name을 받아서 인스턴스 변수 self.name에 저장하고 클래스 변수 population을 증가시킵니다.
10 : 클래스 Person의 소멸자입니다. 클래스 변수 population을 감소시킵니다.
14 : 클래스 메소드에 추가하는 데코레이터입니다.
15 : get_population() 메소드는 클래스 메소드입니다. 첫 번째 매개변수인 cls는 클래스 Person을 의미합니다. 인스턴스 변수 name을 사용하지 않고 클래스 변수 population만 사용하기 때문에 클래스 메소드로 만드는 것이 좋습니다.
21 : 클래스 메소드는 객체가 아니라 클래스를 통해서 호출합니다. 따라서 man이나 woman이 아닌 Person.get_population()으로 호출합니다.

16-2. [Practice16-2.py]

모범답안) C:\PythonStudy\Section16\Practice16-2.py

```
01  class Shop:
02
03      total = 0
04      menu_list = [
05          {'떡볶이': 3000},
06          {'순대': 3000},
07          {'튀김': 500},
08          {'김밥': 2000}
09      ]
10
11      @classmethod
12      def sales(cls, food, count):
13          for menu in cls.menu_list:
14              if food in menu:
15                  cls.total += (menu[food] * count)
16
17
18  Shop.sales('떡볶이', 1)
19  Shop.sales('김밥', 2)
20  Shop.sales('튀김', 5)
21  print('매출: {}원'.format(Shop.total))
```

해설)

03 : 클래스 Shop의 클래스 변수 total입니다. 메뉴 구분 없이 전체 매출액을 저장합니다.
04 : 클래스 Shop에서 판매하는 menu입니다.
12 : sales() 메소드는 판매할 메뉴 이름인 food와 판매할 개수인 count를 매개변수로 가지고 있습니다. 전체 메뉴인 menu_list 리스트의 메뉴를 하나씩 비교해서 판매하는 메뉴를 찾고 가격을 알아냅니다. 그렇게 알아낸 가격과 판매할 개수인 count를 곱해서 전체 매출액 total에 누적합니다.

16-3. [Practice16-3.py]

모범답안) C:\PythonStudy\Section16\Practice16-3.py

```
01  class Car:
02
03      max_oil = 50
04
05      def __init__(self, oil):
06          self.oil = oil
07
08      def add_oil(self, oil):
09          if oil <= 0:
10              return
11          self.oil += oil
12          if self.oil > Car.max_oil:
13              self.oil = Car.max_oil
14
15      def car_info(self):
16          print('현재 주유상태: {}'.format(self.oil))
17
18
19  class Hybrid(Car):
20
21      max_battery = 30
22
23      def __init__(self, oil, battery):
24          super().__init__(oil)
25          self.battery = battery
26
27      def charge(self, battery):
28          if battery <= 0:
29              return
30          self.battery += battery
31          if self.battery > Hybrid.max_battery:
32              self.battery = Hybrid.max_battery
33
```

```
34        def hybrid_info(self):
35            super().car_info()
36            print('현재 충전상태: {}'.format(self.battery))
37
38
39    car = Hybrid(0, 0)
40    car.add_oil(100)
41    car.charge(50)
42    car.hybrid_info()
```

해설)

03 : Car 객체는 최대 max_oil만큼 주유할 수 있습니다.
05 : Car 클래스의 생성자입니다. Car 객체는 생성될 때마다 oil값을 받아서 저장합니다.
08 : add_oil() 메소드입니다. 0 이상의 oil만 사용할 수 있고 최대 max_oil만큼 주유할 수 있습니다.
15 : car_info() 메소드입니다. Car 객체의 주유 상태를 출력합니다.
19 : Car 클래스를 상속 받는 Hybrid 클래스입니다. Hybrid 객체는 Car 클래스의 메소드인 add_oil()과 car_info()를 사용할 수 있습니다.
21 : Hybrid 객체는 최대 max_battery만큼 충전할 수 있습니다.
23 : Hybrid 클래스의 생성자입니다. Car 클래스를 상속 받았으므로 Car 클래스의 생성자를 먼저 호출해야 합니다.
27 : charge() 메소드입니다. 0 이상의 battery만 사용할 수 있고 최대 max_battery만큼 충전할 수 있습니다.
34 : hybrid_info() 메소드입니다. 주유 상태와 충전 상태를 출력합니다. 주유 상태 출력은 Car 클래스(super())의 car_info() 메소드를 호출해서 확인합니다.

17-1. [Practice17-1.py]

모범답안) C:\PythonStudy\Section17\Practice17-1.py

```
01  class Quiz:
02
03      answer = ['경기도', '강원도', '충청남도', '충청북도', '전라남도', '전라북도',
04                '경상남도', '경상북도', '제주특별자치도']  # 한 줄로 작성
05
06      @classmethod
07      def challenge(cls):
08          if not cls.answer:
09              print('모든 도를 맞혔습니다. 성공입니다.')
10              return
11          do = input('정답은? >>> ')
12          if do not in cls.answer:
13              raise Exception('틀렸습니다.')
14          for i, answer_do in enumerate(cls.answer):
15              if do == answer_do:
16                  print('정답입니다.')
17                  cls.answer.pop(i)
18                  break
19          cls.challenge()
20
21
22  try:
23      print('우리나라의 9개 모든 도를 맞히는 퀴즈입니다. 하나씩 대답하세요.')
24      Quiz.challenge()
25  except Exception as e:
26      print(e)
```

해설)

03 : 우리나라의 모든 도를 저장하고 있는 answer 리스트입니다.
07 : challenge() 메소드입니다. 클래스 메소드로 구성했습니다.
08 : 올바른 정답을 입력할 때마다 answer 리스트에서 해당 정답을 제거합니다. 따라서 answer 리스트가 비어 있으면 모든 정답을 맞혔다는 의미입니다.
11 : 사용자로부터 정답을 입력 받습니다.
12 : 사용자가 입력한 정답이 answer 리스트에 없는 오답인 경우 강제로 예외를 발생시킵니다. 이 예외는 25번 라인의 except문에서 처리됩니다.
14 : 사용자가 입력한 정답을 answer 리스트에서 찾아서 제거합니다.
19 : challenge() 메소드 호출이 끝날 때 다시 challenge() 메소드를 호출합니다. 그러면 다시 challenge() 메소드가 처음부터 실행됩니다. 이와 같은 방식의 호출을 재귀 호출이라고 합니다.
22 : try문에서 실행할 코드를 작성합니다. try문의 코드를 실행하다가 예외가 발생되면 25번 라인의 except문에서 해당 예외를 처리합니다.
24 : challenge() 메소드는 Quiz 클래스의 클래스 메소드이므로 Quiz.challenge() 과 같은 방식으로 호출합니다.
25 : try문에서 발생된 예외를 받아서 처리하는 except문입니다.

17-2. [Practice17-2.py]

모범답안) C:\PythonStudy\Section17\Practice17-2.py

```
01  import random
02
03  class UpDown:
04
05      def __init__(self):
06          self.answer = random.randint(1, 100)
07          self.count = 0
08
09      def challenge(self):
10          self.count += 1
11          n = int(input('입력(1~100) >>> '))
12          if n < 1 or n > 100:
13              raise Exception('1~100 사이만 입력하세요.')
14          return n
15
16      def play(self):
17          while True:
18              try:
19                  n = self.challenge()
20              except Exception as e:
21                  print(e)
22              else:
23                  if self.answer < n:
24                      print('Down!')
25                  elif self.answer > n:
26                      print('Up!')
27                  else:
28                      print('{}번만의 정답입니다.'.format(self.count))
29                      break
30
31
32  game = UpDown()
33  game.play()
```

해설)

01 : randint() 메소드를 사용하려면 random 모듈을 import해야 합니다.
05 : UpDown 객체가 생성될 때 자동으로 호출되는 생성자입니다. 1~100 사이의 임의의 난수가 생성되서 인스턴스 변수 answer에 저장됩니다. 또한 도전 횟수를 의미하는 count 변수도 0으로 초기화합니다.
09 : challenge() 메소드입니다. 1~100 사이의 값을 입력 받아서 해당 값을 반환합니다. 만약 1~100 사이의 값이 입력되지 않으면 강제로 예외를 발생시킵니다.
16 : play() 메소드입니다.
18 : try문에서는 사용자로부터 1~100 사이의 값을 입력 받는 challenge() 메소드가 호출됩니다. 1~100 사이의 값이 입력되지 않으면 예외가 발생되면서 20번 라인의 예외 처리 except문으로 예외가 전달됩니다.
22 : try문에서 예외가 발생되지 않는 경우에 처리되는 else문입니다. UpDown 게임의 규칙대로 사용자가 입력한 값보다 정답이 작으면 Down을 출력하고 사용자가 입력한 값보다 정답이 크면 Up을 출력합니다. 사용자가 정답을 맞히면 몇 번의 시도만에 정답인지 출력하고 종료합니다.

17-3. [Practice17-3.py]

모범답안) C:\PythonStudy\Section17\Practice17-3.py

```
01  class BankError(Exception):
02
03      def __init__(self, message):
04          super().__init__(message)
05
06
07  class BankAccount:
08
09      def __init__(self, acc_no, balance):
10          self.acc_no = acc_no
11          self.balance = balance
12
13      def deposit(self, money):
14          if money <= 0:
15              raise BankError('{}원 입금 불가'.format(money))
16          self.balance += money
17
18      def withdraw(self, money):
19          if money <= 0:
20              raise BankError('{}원 출금 불가'.format(money))
21          if money > self.balance:
22              raise BankError('잔액 부족')
23          self.balance -= money
24          return money
25
26      def transfer(self, your_acc, money):
27          your_acc.deposit(self.withdraw(money))
28
29      def inquiry(self):
30          print('계좌번호: {}'.format(self.acc_no))
31          print('통장잔액: {}원'.format(self.balance))
32
33
```

```
34  me = BankAccount('012-34-56789', 50000)
35  you = BankAccount('987-65-43210', 50000)
36
37  try:
38      # me.deposit(-1000)  # me로 -1000원 입금
39      # me.withdraw(-1000)  # me에서 -1000원 출금
40      # me.withdraw(100000)  # me에서 100000원 출금
41      me.transfer(you, 5000)  #me에서 you로 5000원 이체
42  except BankError as e:
43      print(e)
44  finally:
45      me.inquiry()
46      you.inquiry()
```

해설)

01 : Exception 클래스를 상속 받는 BankError 예외 클래스입니다.
03 : BankError 예외 클래스의 생성자입니다. 예외 메시지를 Exception 클래스의 생성자에 전달합니다.
07 : BankAccount 클래스(은행계좌)입니다.
09 : BankAccount 클래스의 생성자입니다. 계좌번호(acc_no)와 통장잔액(balance)를 인수로 받아서 각각을 인스턴스 변수에 저장합니다.
13 : deposit() 메소드입니다. 입금 처리를 합니다.
14 : 마이너스 금액을 입금하려고 하면 강제로 BankError 예외를 발생시킵니다
16: 정상적인 금액이 입금되면 통장잔액에 추가됩니다.
18 : withdraw() 메소드입니다. 출금 처리를 합니다.
19 : 마이너스 금액을 출금하려고 하면 강제로 BankError 예외를 발생시킵니다.
21 : 통장잔액보다 큰 금액을 출금하려고 하면 강제로 BankError 예외를 발생시킵니다.
23 : 정상적인 금액이 출금되면 통장잔액이 줄어듭니다.
24 : 실제로 출금된 금액을 반환합니다.
26 : transfer() 메소드입니다. 이체 처리를 합니다. 상대방 계좌 정보(your_acc)와 이체할 금액(money)을 매개변수로 선언합니다.
27 : 내 통장에서 출금을 먼저하고 상대방 통장에 입금하는 방식으로 동작합니다.
29 : 계좌 정보를 조회하는 inquiry() 메소드입니다. 계좌번호와 통장잔액을 출력합니다.

18-1. [Practice18-1.py]

모범답안) C:\PythonStudy\Section18\Practice18-1.py

```
01  from bs4 import BeautifulSoup
02  import requests
03
04  url = 'https://movie.naver.com/movie/sdb/rank/rpeople.nhn'
05  response = requests.get(url)
06  html = response.text
07  soup = BeautifulSoup(html, 'html.parser')
08  result_list = soup.find_all('td', class_='title')
09
10  movie_in = []
11  for result in result_list:
12      movie_in.append(result.text.strip())
13  for person in movie_in:
14      print(person)
```

해설)

01 : 웹 페이지를 쉽게 해석하는 BeautifulSoup() 함수를 import합니다.
02 : 웹 페이지를 가져오는 requests 모듈을 import합니다.
04 : 분석해야 할 url입니다.
05 : 분석할 사이트의 정보를 가져와서 변수 response에 저장합니다.
06 : response.text는 변수 response에 저장된 전체 소스코드를 가져옵니다.
07 : html.parser를 이용해서 웹 페이지 분석이 용이한 BeautifulSoup 객체 soup을 생성합니다.
08 : 영화인의 정보가 나타난 HTML 소스코드는 다음과 같습니다.

```
<td  class="title"><a  href="/movie/bi/pi/basic.nhn?st=1&amp;code=1937">임창정
</a></td>
```

영화인의 정보가 포함된 <td class="title"></td> 태그를 가져옵니다.
result_list = soup.find_all('td', attrs={'class': 'title'})과 같은 방식도 가능합니다.
10 : 모든 영화인을 저장할 movie_in 리스트를 선언합니다.
11 : 가져온 태그들에서 텍스트 정보를 추출합니다. 불필요한 공백을 제거하기 위해서 추가로 strip() 메소드를 사용합니다.
13 : 가져온 영화인의 정보를 모두 출력합니다.

18-2. [Practice18-2.py]

모범답안) C:\PythonStudy\Section18\Practice18-2.py

```
01  from bs4 import BeautifulSoup
02  import requests
03
04  url = 'https://movie.naver.com/movie/sdb/rank/rmovie.nhn'
05  response = requests.get(url)
06  html = response.text
07  soup = BeautifulSoup(html, 'html.parser')
08  movie_list = soup.find_all('div', class_='tit3')
09
10  for idx, movie in enumerate(movie_list):
11      print('{}위: {}'.format(idx + 1, movie.text.strip()))
```

해설)

01 : 웹 페이지를 쉽게 해석하는 BeautifulSoup() 함수를 import합니다.
02 : 웹 페이지를 가져오는 requests 모듈을 import합니다.
04 : 분석해야 할 url입니다.
05 : 분석할 사이트의 정보를 가져와서 변수 response에 저장합니다.
06 : response.text는 변수 response에 저장된 전체 소스코드를 가져옵니다.
07 : html.parser를 이용해서 웹 페이지 분석이 용이한 BeautifulSoup 객체 soup을 생성합니다.
08 : 조회순으로 영화 랭킹 정보가 표시된 HTML 소스코드는 다음과 같습니다.

```
<div class="tit3">
<a href="/movie/bi/mi/basic.nhn?code=193194" title="도굴">도굴</a>
</div>
```

영화 제목이 나타난 태그를 가져옵니다.
10 : enumerate(movie_list) 함수는 movie_list 요소와 해당 요소의 인덱스를 함께 반환합니다. 이 때 인덱스(idx)는 0~49로 구성되므로 영화 랭킹은 인덱스 + 1(idx + 1)을 통해서 생성합니다.

18-3. [Practice18-3.py]

모범답안) C:\PythonStudy\Section18\Practice18-3.py

```
01  from bs4 import BeautifulSoup
02  import requests
03
04  url = 'https://movie.naver.com/movie/sdb/rank/rmovie.nhn'
05  response = requests.get(url)
06  html = response.text
07  soup = BeautifulSoup(html, 'html.parser')
08  movie_list = soup.find_all('tr')
09  up_list = []
10  for movie in movie_list:
11      target_list = movie.find_all('td', class_='ac')
12      if target_list:
13          if target_list[1].find('img', class_='arrow').get('alt') == 'up':
14              up_list.append(movie.find('td', class_='title').text.strip())
15  for up_movie in up_list:
16      print(up_movie)
```

해설)

08 : 영화 제목과 순위 변동을 모두 포함하고 있는 태그는 <tr>태그입니다.
11 : 순위 변동을 알아내기 위해 <td class="ac"></td> 태그를 가져옵니다.
12 : 모든 <tr></tr>태그에 <td class="ac"></td> 태그가 존재하지는 않습니다.
13 : <tr></tr>태그에 포함된 2번째 <td class="ac"></td> 태그에 <img alt="up">인 요소가 포함되어 있으면 순위가 상승한 영화입니다.
14 : 영화 제목은 <tr></tr>태그에 포함된 <td class="title"></td> 태그에 있습니다.

참고)

[그림 07. Practice18-3]

19-1. [Practice19-1.py]

모범답안) C:\PythonStudy\Section19\Practice19-1.py

```
01  from matplotlib import font_manager, rc
02  import matplotlib.pyplot as plt
03
04  font_path = 'C:/Windows/Fonts/malgun.ttf'
05  font_name = font_manager.FontProperties(fname=font_path).get_name()
06  rc('font', family=font_name)
07
08  figure = plt.figure()
09  axes = figure.add_subplot(111)
10  data = [0.18, 0.3, 3.33, 3.75, 0.38, 25, 0.25, 2.75, 0.1]
11  vitamin = ['비타민 A', '비타민 B1', '비타민 B2', '나이아신', '비타민 B6',
12             '비타민 C', '비타민 D', '비타민 E', '엽산']
13  axes.pie(data, labels=vitamin, autopct='%0.1f%%')
14  plt.axis('equal')
15  plt.show()
```

해설)

01 : 한글 사용을 위해서 font_manager와 rc를 import합니다.
02 : 그래프를 그리기 위해서 matplotlib.pyplot을 plt로 이름을 바꿔 import합니다.
04 : 그래프에서 사용할 '맑은 고딕' 폰트의 경로와 파일명을 작성해 둡니다.
05 : 사용하려고 지정해 둔 폰트 이름을 font_manager가 알아냅니다.
06 : font_manager가 알아낸 폰트를 등록합니다.
08 : 그래프가 작성될 전체 구역을 생성합니다.
09 : 전체 구역에서 1개의 subplot을 생성합니다. 그래프는 1개를 생성할 수 있습니다.
10 : 비타민 함량을 리스트로 준비합니다.
11 : 비타민 종류를 리스트로 준비합니다.
13 : pie() 메소드를 이용해서 원형 그래프를 생성합니다. 비타민 종류가 화면에 나타나고 비타민 함량이 소수 1자리의 백분율로 표시됩니다.
14 : 타원이 아닌 정원으로 원형 그래프를 생성합니다.
15 : 화면에 그래프를 나타냅니다.

19-2. [Practice19-2.py]

모범답안) C:\PythonStudy\Section19\Practice19-2.py

```
01  import random
02  import matplotlib.pyplot as plt
03
04  figure = plt.figure()
05  axes = figure.add_subplot(111)
06  x = [n for n in range(101)]
07  y1 = []
08  y2 = []
09  for i in range(101):
10      y1.append(random.randint(0, 100))
11      y2.append(random.randint(0, 100))
12  axes.plot(x, y1, color='r', marker='.')
13  axes.bar(x, y2, color='g')
14  plt.show()
```

해설)

01 : randint() 메소드를 사용하기 위해서 random 모듈을 import합니다.
02 : 그래프를 그리기 위해서 matplotlib.pyplot을 plt로 이름을 바꿔 import합니다.
04 : 그래프가 작성될 전체 구역을 생성합니다.
05 : 전체 구역에서 1개의 subplot을 생성합니다. 그래프는 1개를 생성할 수 있습니다.
06 : 가로 축(x축)에서 사용할 1~100을 생성합니다.
10 : y1 리스트에 꺾은선그래프로 나타낼 임의의 100개의 값을 생성합니다.
11 : y2 리스트에 막대그래프로 나타낼 임의의 100개의 값을 생성합니다.
12 : x 리스트와 y1 리스트를 이용해서 꺾은선그래프를 생성합니다. 색상은 빨간색이고 • 모양의 marker가 생성됩니다.
13 : x 리스트와 y2 리스트를 이용해서 막대그래프를 생성합니다. 색상은 녹색입니다.
14 : 화면에 그래프를 나타냅니다.

20-1. [addressBook.py 추가 작업]

풀이)

1) IDLE를 실행하고 [File] - [Open] 메뉴를 선택해서 C:\PythonStudy\Section20 디렉터리에 있는 addressBook.py 파일을 선택하고 열기를 선택합니다.

2) class AddressBook 내부에 아래 search() 메소드를 추가합니다. 추가되는 위치는 상관이 없습니다.

```
01  def search(self):
02      print('=== 주소록 검색 ===')
03      name = input('찾을 이름을 입력 >>> ')
04      if not name:
05          print('입력된 이름이 없어 검색을 취소합니다.')
06          return
07      exist = False
08      for person in self.address_list:
09          if name == person.name:
10              person.info()
11              exist = True
12      if not exist:
13          print('{}의 정보가 없습니다.'.format(name))
```

3) 바로 가기 키 F5를 눌러서 실행합니다.

5) addressBook.py 파일을 닫습니다.

해설)

03 : 검색할 이름을 입력 받습니다.
04 : 이름을 입력하지 않은 경우입니다. if name == '': 과 동일합니다.
07 : 검색되었는지 유무를 판단할 플래그 변수입니다. 초기화는 False입니다.
08 : address_list 리스트의 모든 요소를 person으로 옮겨서 비교합니다.
09 : 이름(name)이 일치하는 사람의 정보를 출력하고 exist 변수를 True로 변경합니다.
12 : if exist == False: 와 동일합니다.