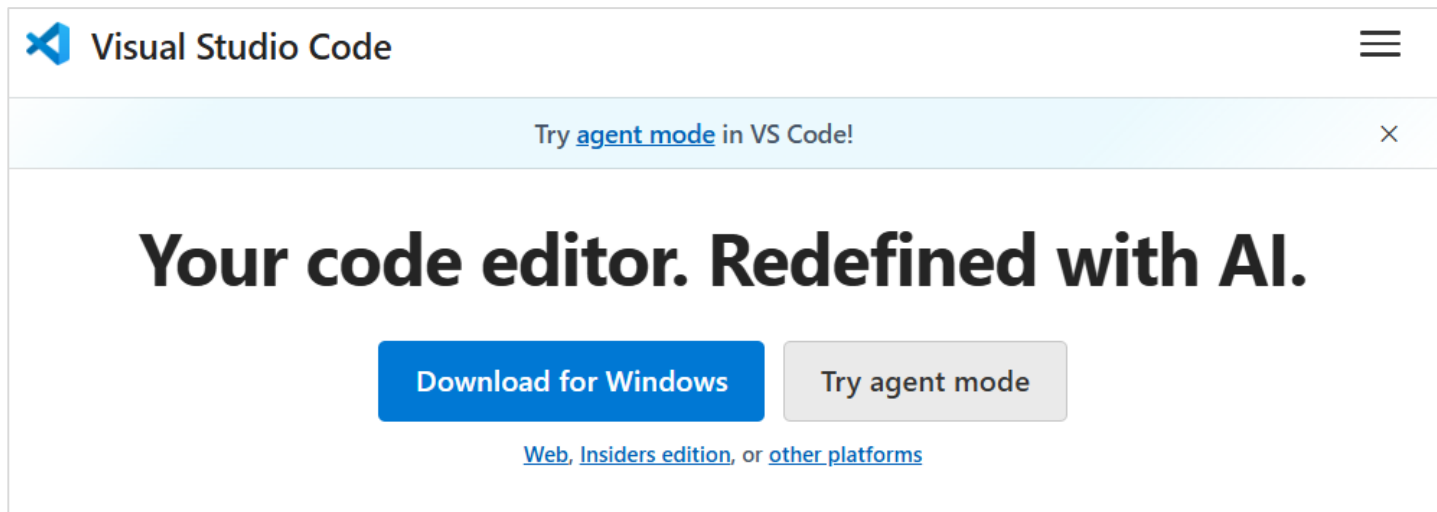


3장. 리스트 자료구조



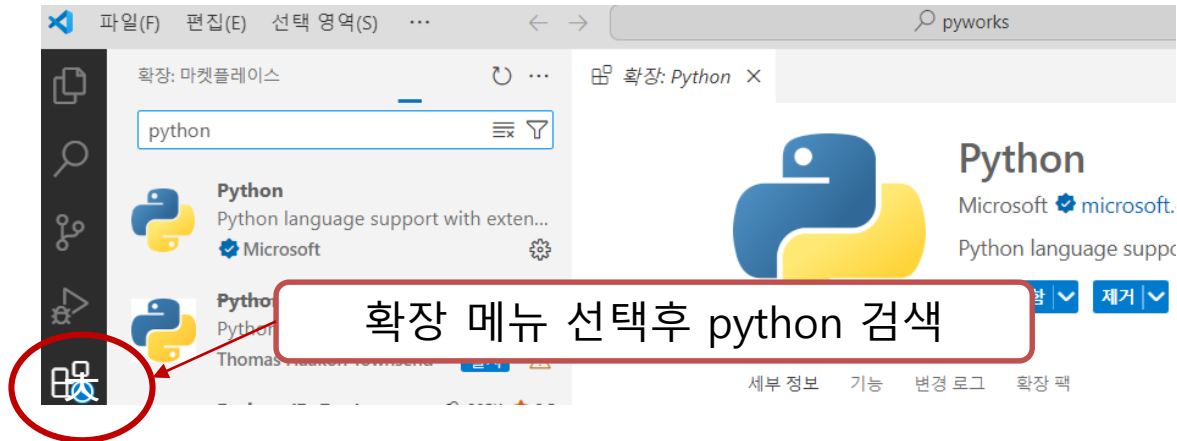
비주얼 스튜디오 코드 설치

◆ 비주얼 스튜디오 코드(VS code) 다운로드

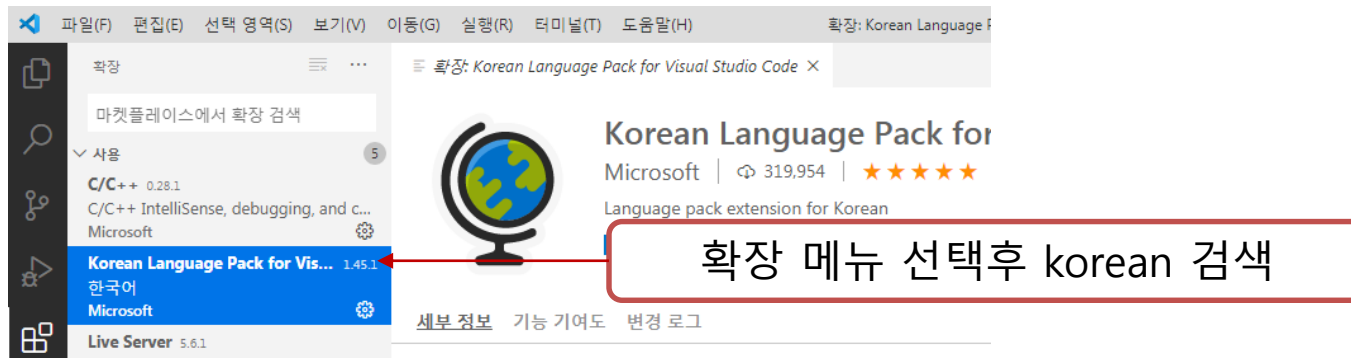


비주얼 스튜디오 코드 환경 설정

◆ Python 언어 지원 확장 팩 설치하기

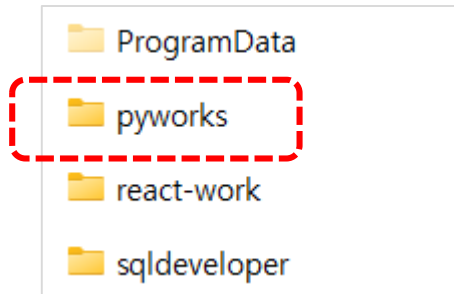


◆ 한국어 팩 설치

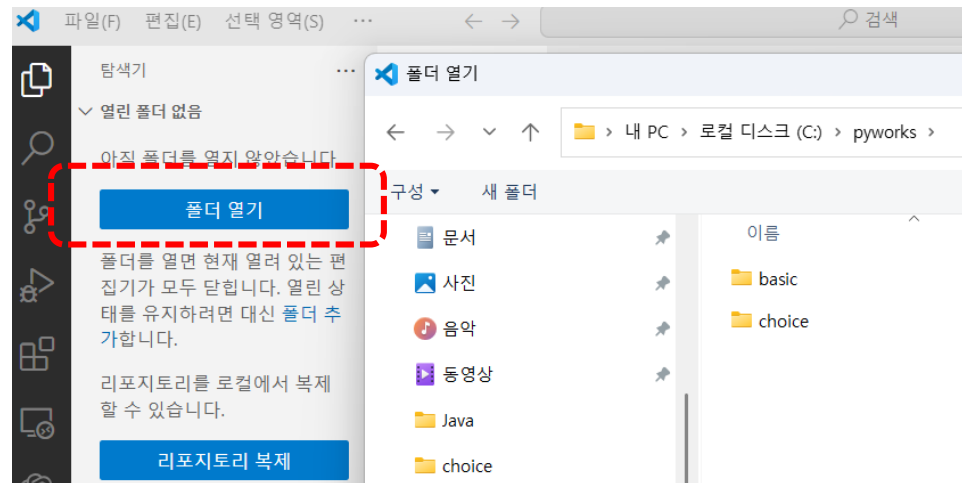


VS code – 작업 폴더 설정하기

◆ VS code – 작업 폴더 설정



① 작업영역 폴더 만들기



② 작업영역 폴더 설정

VS code – 설정 화면

◆ VS code – 관리 도구

The image shows the VS Code interface. On the left, the Activity Bar has the 'Settings' gear icon circled in red. A red arrow points from this icon to a red-bordered box containing the text '관리 > 설정'. The Explorer view on the left shows a file tree with 'PYWORKS' and 'basic' folders, and files 'hello.py' and 'input.py'. The Command Palette is open, showing '설정' (Settings) with the keyboard shortcut 'Ctrl+,'. On the right, the 'Settings' page is displayed under the '사용자 작업 영역' (User Workspace) tab. The '일반적으로 사용되는 설정' (Commonly Used Settings) section is highlighted. It includes three settings: 'Files: Auto Save' with a dropdown menu set to 'afterDelay' (circled in red with an arrow pointing to a red-bordered box labeled '자동 저장'), 'Editor: Font Size' with a text input field set to '17' (circled in red with an arrow pointing to a red-bordered box labeled '글꼴 크기'), and 'Editor: Font Family' with a text input field containing 'Consolas, 'Courier New', monospace'.

관리 > 설정

일반적으로 사용되는 설정

Files: Auto Save
저장되지 않은 변경 사항이 있는 편집기의 자동 저장을 제어합니다.
afterDelay

자동 저장

Editor: Font Size
글꼴 크기(픽셀)를 제어합니다.
17

글꼴 크기

Editor: Font Family
글꼴 패밀리를 제어합니다.
Consolas, 'Courier New', monospace

파이썬 파일 실행

◆ 파일 실행 및 터미널 출력

The image shows a screenshot of a Python IDE interface. On the left, a code editor displays a file named `hello.py` with the following content:

```
basic > hello.py
1 # print() 함수
2 # 문자 출력
3 print("Hello~ World!")
4 print("안녕~ 세계야!")
5 print('010-1234-5678')
6
7 #숫자 출력
8 print(12)
9 print(2.54)
10 print(10 + 20)
11 print(10 - 20)
12
```

On the right, a toolbar contains a play button (run icon) which is circled in red. A red arrow points from this button to a red-bordered box containing the text "실행" (Run). A dropdown menu is open below the play button, titled "Python 파일 실행" (Run Python File). The menu options are:

- Python 파일 실행 (Run Python File)
- 전용 터미널에서 Python 파일 실행 (Run Python File in Dedicated Terminal)
- 대화형 창에서 현재 파일 실행 (Run Current File in Interactive Window)
- Python 디버거: Python 파일 디버그 (Python Debugger: Python File Debug)
- Python 디버거: launch.json을 사용하여 디버그 (Python Debugger: Debug using launch.json)

Below the menu, a terminal window is open with the following output:

```
문제   출력   디버그 콘솔   터미널   포트
PS C:\pyworks> & C:/Users/LG/AppData/Local/Programs/Python/Python38-64/python.exe hello.py
Hello~ World!
안녕~ 세계야!
010-1234-5678

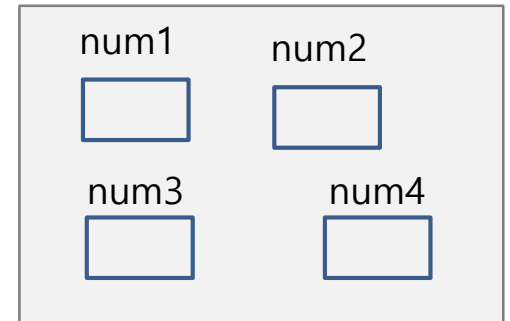
12
2.54
30
-10
PS C:\pyworks>
```

리스트(배열) 사용의 필요성

● 리스트(배열) 사용의 필요성

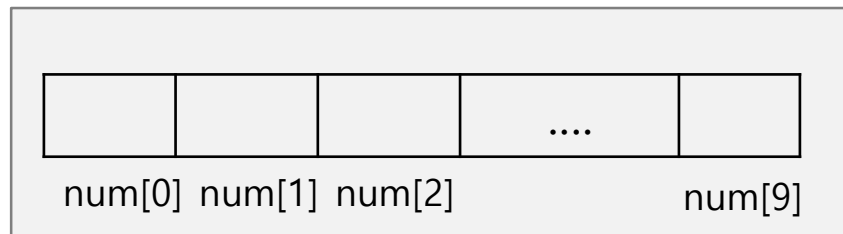
- 정수 10개를 이용한 학생의 성적 프로그램을 만들때 10개의 변수를 선언
(num1, num2, num3... num10)
- 정보가 흩어진 채 저장되고, 변수 이름이 많아 비효율적이고 관리하기 어렵다.

메모리



● 리스트(배열) 사용의 장점

- 인덱스를 이용하여 순차(순서)적으로 관리할 수 있다 -> 효율적이다.



리스트(list)란?

- 리스트(list)란?

- 여러 개의 연속적인 값을 저장하고자 할 때 사용하는 자료형이다.
- 변수는 1개의 값만을 저장하고 변경할 수 있다.

- 리스트의 생성

리스트 이름 = [요소1, 요소2, 요소3...]

season = ["봄", "여름", "가을", "겨울"]

number = [1, 2, 3, 4, 5]

리스트(list)의 생성

■ 문자형 리스트

```
# 리스트(배열) 생성  
carts = ["라면", "커피", "계란", "토마토"]
```

```
# 리스트 객체 출력  
print(carts)  
print(type(carts)) #자료형
```

```
# 요소 접근(인덱싱)  
print(carts[0])  
print(carts[3])  
print(carts[-1])
```

```
# 요소 수정  
carts[1] = "우유"
```

```
# 요소 삭제  
del carts[2]
```

```
# 리스트 출력  
print(carts)
```

```
['라면', '커피', '계란', '토마토']  
<class 'list'>  
라면  
토마토  
토마토  
['라면', '우유', '토마토']
```

리스트(list)의 활용

■ 정수형 리스트

```
# 리스트 생성 - 요소 중복 가능
numbers = [10, 40, 30, 10, 30]

# 리스트 출력
print(numbers)

# 리스트의 크기
print("리스트의 크기: ", len(numbers))

# 요소 수정
numbers[2] = 10

# 요소 삭제
del numbers[0]

# 리스트 출력
print(numbers)
```

```
[10, 40, 30, 10, 30]
리스트의 크기: 5
[40, 10, 10, 30]
```

리스트(list) 반복 - in 사용

▪ for 변수 in 리스트:

```
# for 반복문
# in 리스트 - 리스트 존재 유무 확인
print(30 in numbers)
print(50 in numbers)
print(50 not in numbers)
print()

# 전체 요소 출력
for num in numbers:
    print(num, end=' ')
print()

# 40보다 작은 값 출력
for num in numbers:
    if num < 40:
        print(num, end=' ')
print()
```

```
True
False
True

40 10 10 30
10 10 30
```

리스트(list) 반복 - in 사용

▪ if 변수 in [list]

리스트 내부에 값이 있으면 True, 없으면 False

```
# 음식 분류하기 - 한식, 일식, 중식
foods = ["비빔밥", "짜장면", "초밥", "김치찌게"]

for food in foods:
    if food in ["짜장면", "짬뽕"]:
        print(f'{food}는(은) 중식입니다.')
    elif food in ["초밥", "우동"]:
        print(f'{food}는(은) 일식입니다.')
    else:
        print(f'{food}는(은) 한식입니다.')
```

비빔밥는(은) 한식입니다.
짜장면는(은) 중식입니다.
초밥는(은) 일식입니다.
김치찌게는(은) 한식입니다.

리스트(list)의 연산

◆ 리스트의 연산 – 개수, 합계, 평균 구하기

```
score = [70, 80, 50, 60, 90, 40]
total = 0
count = len(score)

for i in score:
    total += i

avg = total / count

print("개수:", count)
print("합계:", total)
print("평균:", avg)

# 내장함수 sum()과 비교
print("합계:", sum(score))
```

리스트(list)의 연산

◆ 리스트의 연산 – 최고점수, 최저점수

```
# 최고 점수, 최저 점수
max_v = score[0] #최대값 설정
for i in score:
    if max_v < i:
        max_v = i

min_v = score[0] #최소값 설정
for i in score:
    if min_v > i:
        min_v = i

print("최고 점수:", max_v)
print("최저 점수:", min_v)

# 내장함수 - max(), min()과 비교
print("최고 점수:", max(score))
print("최저 점수:", min(score))
```

리스트(list)의 연산

◆ 리스트의 최대값과 최대값 위치 찾기

```
# 최고 점수, 최저 점수 위치 찾기
max_idx = 0 #최대값 위치 설정
for i in range(count):
    if score[max_idx] < score[i]:
        max_idx = i

min_idx = 0 #최소값 위치 설정
for i in range(count):
    if score[min_idx] > score[i]:
        min_idx = i

print("최고 점수 위치:", max_idx)
print("최저 점수 위치:", min_idx)
```

개수: 6
합계: 390
평균: 65.0
합계: 390

최고 점수: 90
최저 점수: 40
최고 점수: 90
최저 점수: 40

최고 점수 위치: 4
최저 점수 위치: 5

리스트(list)의 주요 함수

■ 리스트의 주요 메서드(함수)

함수	기능	사용 예
append()	요소 추가	a = [1, 2, 3] a.append(4) a = [1, 2, 3, 4]
insert()	특정 위치에 추가	a = [2, 4, 5] a.insert(1,3) #1번 위치에 3 삽입 a = [2, 3, 4, 5]
pop()	요소 삭제	a = [1, 2, 3, 4, 5] a.pop() # 마지막 위치의 요소 제거 a = [1, 2, 3, 4] a.pop(1) #1 위치의 2 제거 a = [1, 3, 4]
remove()	특정 요소 삭제	s = ['모닝', 'BMW', 'BENZ', '스포티지'] s.remove('BMW') #요소 직접 삭제 s = ['모닝', 'BENZ', '스포티지']

리스트(list)의 주요 함수

■ 리스트의 주요 메서드(함수)

함수	기능	사용 예
sort()	정렬	<code>a = [1, 4, 2, 3]</code> <code>a.sort()</code> <code>[1, 2, 3, 4]</code>
reverse()	뒤집기	<code>lower = ['b', 'c', 'a']</code> <code>lower.reverse()</code> <code>['a', 'b', 'c']</code>
extend(리스트)	리스트의 끝에 리스트 추가	<code>li = ['a', 'b'],</code> <code>li.extend(['c','d'])</code> <code>['a', 'b', 'c', 'd'],</code>
copy()	리스트 복사	<code>n = [1, 2, 3]</code> <code>m = n.copy()</code>

리스트(list)의 주요 함수

■ 리스트의 주요 메서드(함수)

```
a = [1, 2, 3]
print(a)

# 리스트의 크기
print(len(a))

# 맨 뒤에 추가
a.append(4)

# 1번 위치에 추가
a.insert(1, 5)

print(a)
```

```
# 맨 뒤에서 삭제
a.pop()

# 1번 위치에서 삭제
a.pop(1)

print(a)
print("-----")
```

```
[1, 2, 3]
3
[1, 5, 2, 3, 4]
[1, 2, 3]
-----
```

리스트(list)의 주요 함수

- 리스트의 주요 메서드(함수)

```
car = ["Sonata", "BMW", "EV3", "IONIC6"]  
print(car)  
  
# 리스트의 크기  
print(len(car))  
  
# 추가  
car.append("모닝")  
  
# 삭제  
car.pop()  
  
# 특정 요소 삭제  
car.remove("BMW")  
print(car)
```

```
['Sonata', 'BMW', 'EV3', 'IONIC6']  
4  
['Sonata', 'EV3', 'IONIC6']
```

리스트(list)의 주요 함수

■ 리스트의 주요 메서드(함수)

```
# 리스트의 정렬과 뒤집기
n = [1, 4, 3, 2]
n.sort()
print(n) #[1, 2, 3, 4]

lower = ['b', 'c', 'a']
lower.reverse()
print(lower) #['a', 'c', 'b']

n2 = [1, 3, 5, 4, 2]
n2.sort()
n2.reverse()
print(n2) #[5, 4, 3, 2, 1]
```

```
# 리스트 추가
li = ['a', 'b']
li.extend(['c', 'd'])
print(li) #['a', 'b', 'c', 'd']

# 리스트 복사
n = [1, 2, 3] #원본
print(n) #[1, 2, 3]
m = n.copy() #복사본
print(m) #[1, 2, 3]

# 요소 수정
n[1] = 5
print(n) #[1, 5, 3]
print(m) #[1, 2, 3]
```

리스트(list) 복사

◆ 리스트의 복사

```
a1 = [1, 2, 3, 4, 5]
a2 = []
a3 = []

print("a1 =", a1)

# a1을 a2에 복사
for i in a1:
    a2.append(i)

print("a2 =", a2)
```

리스트(list) 복사

◆ 리스트의 복사

```
# a1의 요소중 홀수만 저장
total = 0
for i in a1:
    if i % 2 == 1:
        a3.append(i)
        total += i

print("a3 =", a3)
print("홀수의 합계:", total)
```

```
a1 = [1, 2, 3, 4, 5]
a2 = [1, 2, 3, 4, 5]
a3 = [1, 3, 5]
홀수의 합계: 9
```

리스트(list) 내포

◆ 리스트 내포 사용하기

[**표현식** for 항목(요소) in 리스트]

```
arr1 = [1, 2, 3, 4, 5]
arr2 = []
arr3 = []
arr4 = []

# arr1의 요소를 3의 배수로 저장
for i in arr1:
    arr2.append(i * 3)
print("arr2 =", arr2)

# 리스트 내포 - 3의 배수로 저장
arr3 = [i * 3 for i in arr1]
print("arr3 =", arr3)

# arr1에서 홀수만 저장
arr4 = [i for i in arr1 if i % 2 == 1]
print("arr4 =", arr4)
```

```
arr2 = [3, 6, 9, 12, 15]
arr3 = [3, 6, 9, 12, 15]
arr4 = [1, 3, 5]
```

리스트(list)의 정렬

◆ 오름차순 정렬 – 버블 정렬

인접한 두 요소를 비교하며, 큰 값을 오른쪽으로 이동 시킴

```
n_list = [60, 40, 90, 50, 80]
n = len(n_list)

for i in range(0, n):
    for j in range(0, n-1):
        if n_list[j] > n_list[j+1]:
            # 교환 방법 1
            temp = n_list[j]
            n_list[j] = n_list[j+1]
            n_list[j+1] = temp

            # 교환 방법 2
            # n_list[j], n_list[j+1] = n_list[j+1], n_list[j]
```

*** 오름차순 정렬 ***
40 50 60 80 90

리스트(list)의 정렬

◆ 오름차순 정렬 – 버블 정렬

```
'''
    i=0, j=0, 40, 60, 90, 50, 80 #교환
        j=1, 40, 60, 90, 50, 80 #유지
        j=2, 40, 60, 50, 90, 80 #교환
        j=3, 40, 60, 50, 80, 90 #교환
        j=4, 40, 60, 50, 80, 90 [임시 저장]
    i=1, j=0, 40, 60, 50, 80, 90 #유지
        j=1, 40, 50, 60, 80, 90 #교환
        j=2, 40, 50, 60, 80, 90 #유지
        j=3, 40, 50, 60, 80, 90 #유지
        j=4, 40, 50, 60, 80, 90 [임시 저장]
    i=2, 교환 없음
    i=3, 교환 없음
    i=4, 교환 없음
'''

print("*** 오름차순 정렬 ***")
for i in n_list:
    print(i, end=' ')
```

리스트 슬라이싱

◆ 리스트의 슬라이싱(범위 검색)

```
carts = ["라면", "커피", "계란", "토마토"]
```

```
print(carts[0:4])
```

```
print(carts[:])
```

```
print(carts[0:3])
```

```
print(carts[0:-1])
```

```
print(carts[0:2])
```

```
print(carts[0:-2])
```

```
['라면', '커피', '계란', '토마토']  
['라면', '커피', '계란', '토마토']  
['라면', '커피', '계란']  
['라면', '커피', '계란']  
['라면', '커피']  
['라면', '커피']
```

문자열 인덱싱, 슬라이싱

◆ 문자열은 특별한 1차원 리스트이다.

문자열(시작번호:끝번호)

※ 끝번호는 (끝번호 -1)과 같다

```
# 문자열은 1차원 리스트이다.  
say = "Have a nice day"  
  
print(say[0])  
print(say[-1])  
print(say[0:4])  
print(say[0])  
print(say[7:])
```

문자열 인덱싱, 슬라이싱

◆ 문자열은 특별한 1차원 리스트이다.

문자열(시작번호:끝번호)

※ 끝번호는 (끝번호 -1)과 같다

```
say = "Have a nice day"
```

```
print(say[0])  
print(say[-1])  
print(say[0:4])  
print(say[0])  
print(say[7:])
```

```
s = "20240621Rainy"
```

```
year = s[:4]  
print(year)
```

```
day = s[4:8]  
print(day)
```

```
weather = s[8:]  
print(weather)
```

```
H  
y  
Have  
H  
nice day  
2024  
0621  
Rainy
```

챗봇(chatbot)

◆ 챗봇 프로그램

단어가 포함되어 있으면 문장을 완성해주는 챗봇 프로그램 만들기

```
사용자(exit 입력시 종료): 안녕  
챗봇: 안녕하세요! 반가워요!  
사용자(exit 입력시 종료): 이름  
챗봇: 저는 python 챗봇입니다.  
사용자(exit 입력시 종료): 날씨  
챗봇: 날씨앱이나 검색 기능을 이용하세요.  
사용자(exit 입력시 종료): 시간  
챗봇: 죄송해요. 잘 이해하지 못했어요.  
사용자(exit 입력시 종료): exit  
챗봇: 대화를 종료합니다. 안녕히 가세요!
```

챗봇(chatbot)

◆ 챗봇 프로그램

```
#단어가 포함되어 있으면 출력하는 프로그램
animal = "dog"

# in 명령어 - 있다/없다를 확인하는 명령어임
print('d' in animal) #True
print('c' in animal) #False
print('g' not in animal) #False

animals = "dog cat horse"
print('cat' in animals) #True
print('cow' in animals) #False
```

챗봇(chatbot)

◆ 챗봇 프로그램

```
while True:
    user_input = input("사용자(exit 입력시 종료): ")

    if user_input == "exit":
        print("챗봇: 대화를 종료합니다. 안녕히 가세요!")
        break
    elif "안녕" in user_input:
        print("챗봇: 안녕하세요! 반가워요!")

    elif "이름" in user_input:
        print("챗봇: 저는 Python 챗봇입니다.")

    elif "날씨" in user_input:
        print("챗봇: 날씨앱이나 검색 기능을 이용하세요.")

    else:
        print("챗봇: 죄송해요. 잘 이해하지 못했어요.")
```

실습 문제 – 문자열 처리

2025년 민생회복 지원금 신청 프로그램을 아래와 같이 구현하세요.

출생연도 끝자리	접종 요일
1 또는 6	월요일
2 또는 7	화요일
3 또는 8	수요일
4 또는 9	목요일
5 또는 0	금요일

👉 실행 결과

출생연도 입력: 2023
신청일은 수요일입니다.

출생연도 입력: 20004
입력 오류: 출생연도는 4자리여야 합니다.

문자열 함수

■ 문자열 함수(메서드) 정리

메서드	설명
split()	<pre>s = 'banana, grape, kiwi' s = fruit.split(',') [구분기호로 나누고 리스트로 만듦] s ['banana', ' grape', ' kiwi']</pre>
replace()	<pre>s = 'Hello, World' s = s.replace('World', 'Korea') [문자를 변경함] 'Hello, Korea'</pre>
find()	<pre>s = "Hello" s.find('H') 0 s.find('k') -1 [문자열이 존재하는 위치 반환. 없으면 -1반환]</pre>

문자열 함수

- 문자열 함수(메서드)

```
# split()
fruit = "banana, grape, apple"
fruit = fruit.split(',')
print(fruit)

# replace()
msg = "대한민국의 수도는 부산이다."
msg = msg.replace("부산", "서울")
print(msg)
```

```
['banana', ' grape', ' apple']
대한민국의 수도는 서울이다.
```

문자열 함수

■ 문자열 함수(메서드)

```
# find()
s1 = "smile"
print(s1.find('m'))
print(s1.find('k'))

# 대소문자 구분함
txt = "Welcome to my House!!"
x = txt.find("welcome")
print(x)
```

```
# strip() - 공백 문자 제거
s2 = "  Hi, han"
print(s2.lstrip())
print(s2.strip())

s3 = "Hi, han  "
print(s3.rstrip())
print(s3.strip())
```

```
1
-1
-1
Hi, han
Hi, han
Hi, han
Hi, han
```

문자열 응용 – 실습

◆ 문자열 처리

```
print("Happy Birthday!!을 입력하세요.")
x = input()
x = x.capitalize()
print(x)
y = x.split()
print(y)

# print(y[0][:1], end='*')
print(y[0][:2], end='*')
print(y[1][5:8])
```

```
Happy Birthday!!을 입력하세요.
Happy Birthday!!
Happy birthday!!
['Happy', 'birthday!!']
Hpy*day
```

2차원 리스트(list)

- 2차원 리스트의 선언 및 생성
 - 리스트 내부에 리스트를 가진 자료 구조이다.
 - 행과 열의 표(테이블) 형태를 이루고 있다.

리스트 이름 = [요소1, 요소2, [요소1, 요소2, 요소3]]

	열1	열2
행1	a[0][0]	a[0][1]
행2	a[1][0]	a[1][1]
행3	a[2][0]	a[2][1]

2차원 리스트(list)

■ 2차원 리스트 생성 및 출력

```
a = [  
    [10, 20],  
    [30, 40],  
    [50, 60]  
]  
  
# 리스트 출력  
print(a)  
  
# 리스트의 크기  
print("리스트의 크기(행) : ", len(a))  
print("리스트의 크기(열) : ", len(a[0]))  
print("리스트의 크기(열) : ", len(a[1]))  
  
# 특정 요소 검색  
print(a[0][1])  
print(a[1][0])
```

```
[[10, 20], [30, 40], [50, 60]]  
리스트의 크기(행) : 3  
리스트의 크기(열) : 2  
리스트의 크기(열) : 2  
20  
30
```

2차원 리스트(list)

■ 2차원 리스트 생성 및 출력

```
# 전체 출력
for x, y in a:
    print(x, y)
print()

# 전체 출력 - range() 사용
for i in range(len(a)):
    for j in range(len(a[i])):
        print(a[i][j], end=' ')
    print()
```

```
10 20
30 40
50 60

10 20
10 20
30 40
50 60
```

2차원 리스트(list)

■ 2차원 리스트의 추가 및 수정

```
# 리스트 추가
a.append([70, 80])
print(a)
```

```
# 리스트 수정
a[1][1] = 100
```

```
# 리스트 삭제
del a[2] #2행 삭제
print(a)
```

```
# 열 삭제
for row in a:
    del row[1]
print(a)
```

```
a.clear() #a 리스트 삭제
print(a)
```

```
[[10, 20], [30, 40], [50, 60], [70, 80]]
[[10, 20], [30, 100], [70, 80]]
[[10], [30], [70]]
[]
```


2차원 리스트(list)

■ 2차원 리스트의 연산

```
score = [  
    [10, 20],  
    [30, 40],  
    [50, 60],  
    [70, 80]  
]  
  
# 2차원 리스트의 연산  
total = 0 # 합계  
count = 0 # 개수  
  
for i in range(len(score)):  
    for j in range(len(score[0])):  
        count += 1  
        total += score[i][j]  
avg = total / count  
  
print("합계:", total)  
print("개수:", count)  
print("평균:", avg)
```

합계: 360
개수: 8
평균: 45.0
평균: 45.0

2차원 리스트(list)

- 2차원 리스트 성적 통계

```
# 5명의 수학, 영어 과목의 총점과 평균
# 과목별 총점과 평균
score = [
    [80, 70],
    [70, 80],
    [60, 93],
    [50, 75],
    [75, 70]
]

n = len(score)

# 개인별 총점과 평균
total = 0
print("총점 평균")
for i in range(0, n):
    total = score[i][0] + score[i][1]
    print(total, total / 2)
```

2차원 리스트(list)

- 2차원 리스트 성적 통계

```
# 과목별 총점
sum_subject = [0, 0]      # 수학 총점
avg_subject = [0.0, 0.0]  # 영어 총점

for i in range(0, n):
    sum_subject[0] += score[i][0]
    sum_subject[1] += score[i][1]

avg_subject[0] = sum_subject[0] / n # 수학 평균
avg_subject[1] = sum_subject[1] / n # 영어 평균

print("수학 총점 : %d점" % sum_subject[0])
print("영어 총점 : %d점" % sum_subject[1])
print("수학 평균 : %.1f점" % avg_subject[0])
print("영어 평균 : %.1f점" % avg_subject[1])
```