

응용 프로그래밍

python

데이터 타입
- 리스트 -

수업내용

- 복합 자료형
 - ◆ List

List 메소드

- range
- append, insert
- del, clear, pop
- count
- sort, reverse
- index
- slice
- concatenation
- repetition
- length
- membership test

- 파이썬에서 다루는 주요 데이터 타입 중 하나인 리스트(list)에 대해서 공부한다.
- 각각의 자료 구조들은 데이터를 생성, 삽입, 읽기, 수정, 삭제를 하기 위한 방법을 요구한다.
- 파이썬은 각각의 자료 구조를 위해서 '메소드'를 사용한다.
- '메소드'는 '클래스'의 일부이며, '클래스'는 '객체지향 프로그래밍'의 일부이다. 이것은 강의 뒷부분에서 살펴 볼 주제이다.
- 클래스안에서 구현된 함수를 "메소드" 라 한다. 그것은 데이터 구조에 대한 '연산' 이라고 생각하고, 뒷부분에서 '메소드'에 대해 자세히 다룬다.

- List는 유연한(flexible) 데이터 타입이다.
- 리스트의 저장 항목은 요소(element)라 하며, 연속적으로 저장 가능하다.
- 리스트에 저장되는 항목들은 각각 다른 형식일 수 있다. 리스트에는 정수, 실수, 부울, 문자열 입력이 가능하다.
- 리스트는 중복된 요소를 허용한다.
- 리스트는 대괄호([])로 묶고, 내부의 요소(elements)를 쉼표로 구분한다.

```
listname = [10, 20, 'john', 36.2, True, 'john']
```

- List 만들기과 확인하기

```
>>> a = [10, 20, 30, 40, 50]
```

```
# 리스트 만들기
```

```
>>> print(a)
```

```
# 리스트 확인하기
```

```
[10, 20, 30, 40, 50]
```

```
>>> a = [ ]
```

```
# 빈 리스트 만들기
```

```
>>> print(a)
```

```
# 리스트 확인하기
```

```
[ ]
```

- 작성한 List의 요소 중 특정 요소 선택하기. 이때, 대괄호([]) 사용.

a =

10	20	30	40	50
a[0]	a[1]	a[2]	a[3]	a[4]

↑
index

```
>>> print(a[1])
```

```
20
```

인덱스 1 출력하기

```
>>> print(a[4])
```

```
50
```

인덱스 4 출력하기

```
>>> print(a[5])
```

인덱스 범위 밖 출력하기

```
IndexError: list index out of range
```

리스트 구조 보기

- <http://pythontutor.com/live.html#mode=edit>
`li = [10, 20, "john", 36.2, True, "john"]`

Live Programming Mode - Python x +

← → ↻ 주의 요함 | pythontutor.com/live.html#mode=edit

This mode is experimental. Use the [regular Python Tutor](#) to **get live help** and use more features.

Write code in Python 3.6 (drag lower right corner to resize code editor)

```

1 li = [10, 20, "john", 36.2, True, "john"]
2

```

Frames

Global frame

li

Objects

list

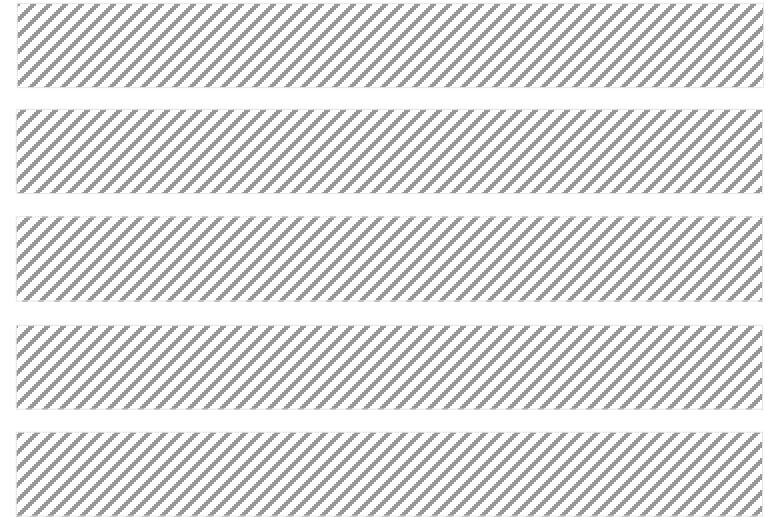
0	1	2	3	4	5
10	20	"john"	36.2	True	"john"

Example & Solution

- 7명의 성적 리스트를 작성하시오.

`gradeList = [75, 90, 80, 100, 85, 95, 60]`

- 1) gradeList의 전체 요소를 출력하시오.
- 2) gradeList의 index 0 요소를 출력하시오.
- 3) gradeList의 index 3 요소는 출력하시오.
- 4) gradeList의 마지막 요소를 출력하시오.
- 5) 빈 리스트 emptyList를 만드시오.



- “name” 리스트를 만드시오.
 - ◆ 버킷 리스트에 항목을 추가하거나 삭제, 변경하시오.
 - ◆ *append()*, *insert()*, *remove()*, *del*, *pop()*, *len()* 등

```
>>> name = ['john', 'rosa', 'maria']
```

append(), insert(), len()

```
>>> name = ['john', 'rosa', 'maria']      # 리스트 만들기
>>> name.append('danny')                  # ['john', 'rosa', 'maria', 'danny']
>>> name.insert(2, 'alex')                 # ['john', 'rosa', 'alex', 'maria', 'danny']
>>> name[1]='selly'                        # ['john', 'selly', 'alex', 'maria', 'danny']
>>> len(name)                             # 5
```

- `append()` 메소드는 리스트의 마지막에 엘리먼트 하나를 추가
- `insert()`는 지정한 index 위치에 element를 추가하고 싶을 때 사용
사용 형식: `insert(index, element)`
- `len()`는 리스트에 포함된 요소(element)의 개수를 반환

remove(), del, pop(), clear()

```
>>> name.remove('alex')           # ['john', 'rosa', 'alex', 'maria', 'danny']
>>> del name[1]                   # ['john', 'rosa', 'maria', 'danny']
>>> name.pop()                    # ['john', 'maria', 'danny']
>>> name.clear()                  # ['john', 'maria']
>>>                               # [ ]
```

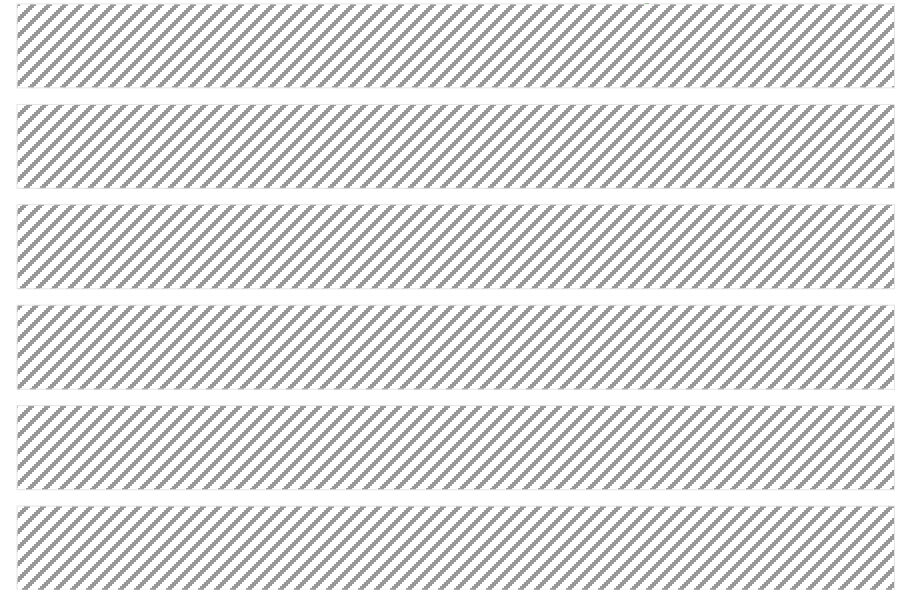
- *remove()*는 지정한 요소(element)를 리스트에서 제거함.
- 만약 리스트에서 삭제하고 싶은 항목의 인덱스 번호를 알고 있다면 *del*을 사용해도 됨.
- *pop()*은 리스트에서 마지막 번째 항목을 삭제하거나, 괄호 안에 인덱스를 입력하여 특정 항목을 삭제할 수도 있음.
- *clear()* 연산은 리스트에 저장된 모든 요소(element)를 삭제함.

Example & Solution

- 방문하고 싶은 장소의 "bucketList"를 만들고 아래 문제에 답하시오.

```
bucketList = ['서울', '대구', '부산 ']
```

- 1) 리스트의 끝에 '제주'를 추가하시오.
- 2) 리스트의 길이를 출력하시오.
- 3) 인덱스1에 '대전'을 추가하시오.
- 4) 리스트에서 '서울'을 삭제하시오.
- 5) 리스트에서 마지막 요소(element)를 삭제하시오.
- 6) 리스트의 모든 요소(element)를 삭제하시오.



- *slicing* 또는 *slice* 연산은 시퀀스(sequence) 자료 중 범위를 지정하여 특정 요소를 가져오는 방법 및 표기법을 의미함.
- 분할 영역의 시작 인덱스, 종료 인덱스, 간격을 지정할 수 있음

리스트이름[0:6:2]

↓ 시작 ↓ 간격(증가, 감소)
↓ 끝(이 숫자는 포함 안함)

```
>>> bucketList = ['김가천', 22, '서울', '속초', '부산', '제주']    # 리스트 만들기
                  0      1      2      3      4      5

>>> bucketList[0:6:2]                                           # 슬라이스 하기
['김가천', '서울', '부산']
```

```
>>> bucketList = ['김가천', 22, '서울', '속초', '부산', '제주']
```

```
>>>
```

```
>>> bucketList[2:6]
```

```
['서울', '속초', '부산', '제주']
```

인덱스2부터 5까지 출력

```
>>> bucketList[3:]
```

```
['속초', '부산', '제주']
```

인덱스3부터 끝까지 출력

```
>>> bucketList[:3]
```

```
['김가천', 22, '서울']
```

처음부터 인덱스2까지 출력

- 음수 인덱싱(Negative indexing)은 파이썬 시퀀스 자료 끝에서 시작하는 목록 요소에 대한 액세스에 사용
 - ◆ 인덱스 -1은 리스트의 마지막 요소를 액세스. `my_list[-1]`
 - ◆ 인덱스 -2는 리스트의 끝에서 두번째 요소를 액세스. `my_list[-2]`

```
>>> my_list = ["hello", "world", "hi", "bye"]
>>> my_list[-1]           # bye
>>> my_list[-3]           # world
>>> my_list[-4]           # hello
```


- 문자열 a를 만들고 원하는 글자를 슬라이싱 해보자.

문자열[0:6:2]

↓ 시작 ↓ 간격(증가, 감소)
↓
↓ 끝(이 숫자는 포함 안함)

```
>>> a = "Life is too short, you need python"
```

```
>>> a[0:4]
```

```
'Life'
```

```
>>> a[-6:]
```

```
'python'
```

리스트 연산(+, *)

- 연결(Concatenation): '+' 연산자를 사용하여 두 리스트를 병합
- 반복(Repetition): '*' 연산자를 사용하여 지정된 횟수 만큼 리스트 요소 반복

goodPlace + city
city * 3

```
>>> goodPlace = ['Rome', 'Paris', 'Hanoi']
>>> city = ['Tokyo', 'Danang']
>>> goodPlace + city
['Rome', 'Paris', 'Hanoi', 'Tokyo', 'Danang']
>>> city*3
['Tokyo', 'Danang', 'Tokyo', 'Danang', 'Tokyo', 'Danang']
```

Example1 & Solution

ex5-1.py

- 다음 코드의 실행 결과는 무엇일까요?

```
>>> fruits = ['사과', '바나나', '체리', '포도', '오렌지', '딸기', '멜론']  
>>> print(fruits[-3:], fruits[1::3])
```

Result>



Example2 & Solution

ex5-2.py

- 다음 코드의 실행 결과는 무엇일까요?

```
>>> a = [5, 7, 3]
>>> b = [3, 9, 1]
>>> c = a + b
>>> c.sort()
>>> print(c)
```

Result>



Example3 & Solution

ex5-3.py

- 다음 year에 연도, population에 서울시 인구수가 저장되어 있다.
다음 코드를 완성하여 최근 3년간의 연도와 인구수가 리스트로 출력되도록 하시오.

```
year = [2011,2012,2013,2014,2015,2016,2017,2018,2019, 2020]
population=[10249679,10195318,10143645,10103233,10022181,9930616,
            9857426,9838892,9789523, 9715429]
print( )
print( )
```

Result>

[2018, 2019, 2020]

[9838892, 9789523, 9715429]

- *index()* 연산은 리스트에서 지정된 항목이 몇 번째 요소인지 인덱스 번호를 반환
- 대소문자 구별

goodPlace.index('Paris')

```
>>> goodPlace = ['Rome', 'Paris', 'Hanoi']
```

인덱스 만들기

```
>>> goodPlace.index('Paris')
```

인덱스 번호 출력

```
1
```

```
>>> goodPlace.index('paris')
```

없는 요소 출력 결과

```
Traceback (most recent call last)
```

대소문자 구별함

```
File "<pyshell#42>", line 1, in <module>
```

```
    goodPlace.index('paris')
```

```
ValueError: 'paris' is not in list
```

Membership test

- **in** 키워드는 '앞에 적은 항목'이 '뒤에 적은 리스트' 내에 포함되어 있는지, 아닌지 여부를 판별하여 True, False로 반환.
 - ♦ 아래 예제는 goodPlace 리스트에 'Room'이라는 요소가 포함되어 있는지 여부 판별

'Rome' in goodPlace

```
>>> goodPlace = ['Rome', 'Paris', 'Hanoi']
```

```
>>> 'Rome' in goodPlace
```

```
True
```

#goodPlace에 'Rome'이 있나?

```
>>> 'New York' in goodPlace
```

```
False
```

#goodPlace에 'New York'이 있나?

- `count()` 연산은 리스트에 '해당 항목'이 몇 번 포함되어 있는지 반환함.
 - ♦ 아래 예제는 `goodPlace` 리스트에 'Room' 항목이 몇 번 포함되어 있는지 확인하는 예제이다. 결과는 2이다.

`goodPlace.count('Rome')`

```
>>> goodPlace = ['Rome', 'Paris', 'Hanoi', 'Rome']
```

```
>>> goodPlace.count('Rome')
```

```
2
```

```
>>> goodPlace.count(20)
```

```
0
```


sort(), reverse()

- `sort()` : 오름차순(**ascending** order) 정렬을 수행하는 연산
- `reverse()` : 리스트 거꾸로 뒤집기

```
>>> name = ['john', 'anna', 'maria']
>>> name.sort()                                # ['anna', 'john', 'maria']
>>> name.reverse()                             # ['maria', 'john', 'anna']
>>> name.append(22)                             # ['maria', 'john', 'anna', 22]
>>> name.sort()
```

```
name.sort()
TypeError: '<' not supported between instances of 'str' and 'int'
```

문자열과 정수 요소 사이에는
관계 연산자를 사용할 수 없다.
즉, 정렬을 할 수 없다.

Exercise & Solution

- 리스트 "bucketList"와 "goodPlace" 를 만드시오.

```
bucketList = ['Seoul', 'Daejon', 'Daegu', 'Busan', 'jeju']
```

```
goodPlace = ['Rome', 'Paris', 'Hanoi']
```

- 1) "bucketList"의 인덱스2부터 인덱스 4 요소를 슬라이스하시오.



- 2) "bucketList"의 인덱스1부터 마지막 요소까지 2간격으로 슬라이스하시오.



- 3) 두 리스트 "bucketList"와 "goodPlace"를 연결하시오.



- 4) 'Paris'가 "goodPlace"에 포함되어 있는지 확인하시오.



- 5) "goodPlace"에서 'Rome'의 발생횟수를 출력하시오.



- 6) "goodPlace"를 오름차순 정렬하시오.



for 문을 사용하여 리스트 반복하기

- *for* 반복문을 사용하여 목록의 모든 요소를 반복할 수 있다.

```
>>> bucketList = ['Busan', 'Jeju', 'Sokcho', 'Daejeon']
```

```
>>> for i in bucketList:
```

```
    print(i)
```

```
['Busan', 'Jeju', 'Sokcho', 'Daejeon']
```

Result>

Busan

Jeju

Sokcho

Daejeon

while Statement을 사용하여 리스트 반복하기


- *while* 반복문을 사용하여 목록의 모든 요소를 반복할 수 있다.

```
>>> bucketList = ['Busan', 'Jeju', 'Sokcho', 'Daejeon']
>>> i = 0
>>> while i < len(bucketList):
    print(bucketList[i])
    i += 1
```


```
Result>
Busan
Jeju
Sokcho
Daejeon
```

Exercise & Solution

- test 리스트 요소 값의 합계를 계산하시오.

```
>>> test = [75, 90, 66, 100, 50]
>>> total = 0
>>> for i in test :
    
>>> print('total = ', total)
```

Result>
total = 381

```
>>> test = [75, 90, 66, 100, 50]
>>> print('total = ', )
```

Result>
total = 381

Exercise

- 빈 리스트 temp를 만들고 온도를 5회 입력 받으시오.
- temp 리스트의 평균 온도를 구하시오.

Result>

온도를 입력하세요 : 22

온도를 입력하세요 : 23

온도를 입력하세요 : 22

온도를 입력하세요 : 25

온도를 입력하세요 : 24

평균 온도는 23.2 입니다.

Exercise & Solution

ex5-4.py

- 빈 리스트 temp를 만들고 온도를 5회 입력 받으시오.
- temp 리스트의 평균 온도를 구하시오.

```
temp = []  
for i in range(1,6):  
      
  
print("평균 온도는 {} 입니다.".format()))
```

python

파이썬 고급 기능

- `split()` ↔ `join()`
- `enumerate()`
- 구구단

- **split()** 은 문자열이나 리스트에 저장된 '문자열(string)'을 지정된 구분자(separator)를 사용해서 '요소(element)'를 나누어 List로 만드는 메소드 (2개의 매개 변수를 가지며, 모두 선택 사항임.)

split([separator], [max])

- ◆ 구분자(Separator) : 문자열을 분할하는 기호. 생략하면 기본값은 '**공백**(white space)'.
- ◆ max : 구분자로 사용할 개수

split() : string

- *split()* 은 구분자를 사용하여 문자열을 **분리**하여 **리스트**로 만듦.

```
>>> quiz = 'python.program.good'
```

```
>>> sp = quiz.split('.')
```

```
#[ 'python', 'program', 'good']
```

```
>>> sp = quiz.split('.',1)
```

```
#[ 'python', 'program.good']
```

```
>>> sp = quiz.split('.',2)
```

```
#[ 'python', 'program', 'good']
```

```
>>> quiz = 'python program good'
```

```
>>> sp = quiz.split(' ')
```

```
#[ 'python', 'program', 'good']
```

```
>>> quiz = 'python/program/good'
```

```
>>> sp = quiz.split('/')
```

```
#[ 'python', 'program', 'good']
```

split() : List

- 리스트는 'split()'이 없다. 하나의 항목으로 묶여 있다면 반복문을 사용하여 쉼표(,)로 구분하여 리스트 요소로 나눌 수 있다.

['부산,제주,속초,대전'] → ['부산','제주','속초','대전']

리스트는 'split()'이 없다.

```
>>> bucketList = ['부산,제주,속초,대전']
```

```
>>> for i in bucketList:
```

```
    sp2 = i.split(',')
```

구분자 쉼표(,)

```
>>> bucketList
```

['부산,제주,속초,대전']

```
>>> sp2
```

['부산', '제주', '속초', '대전']

join() : string, List

- `join()` 은 리스트내 구분자로 구분되어진 요소를 **추가**하여 **합쳐서 문자열**로 만듦
- 문자열은 지정한 구분자로 합치기

`','`.join(리스트 이름)

```
>>> sp = [ 'python', 'program', 'good']      # 확인
>>> sp2 = ['부산', '제주', '속초', '대전']
>>> quiz = ','.join(sp)                     # 'python,program,good'
>>> bucketList = '/'.join(sp2)               # '부산/제주/속초/대전'
>>> ': '.join('abcd')                       # 'a: b: c: d'
>>> '/ '.join('abcd')                       # 'a/ b/ c/ d'
```

Example

- 아래와 같이 “Quiz” list를 만드시오 :

```
quiz = ['1.중국의 수도를 선택하세요./1)칭따오/2)상하이/3)베이징']
```

- 1) quiz 리스트를 구분자 '/'를 사용하여 항목을 분리하시오.
- 2) 구분된 리스트를 다음과 같이 출력하시오.

```
Result>  
1.중국의 수도를 선택하세요.  
1)칭따오  
2)상하이  
3)베이징
```

Example : Solution

ex5-split.py

```
quiz = ['1.중국의 수도를 선택하세요./1)칭따오/2)상하이/3)베이징']
```

```
for i in quiz :
```



```
    print(quizSplit[0])
```

```
    print(quizSplit[1])
```

```
    print(quizSplit[2])
```

```
    print(quizSplit[3])
```

```
# ['1.중국의 수도를 선택하세요.', '1)칭따오', '2)상하이', '3)베이징']
```

```
# 출력
```

Result>


1.중국의 수도를 선택하세요.

1)칭따오

2)상하이

3)베이징

- `enumerate()` 연산은 순서가 있는 자료형(리스트, 튜플, 딕셔너리 등)을 입력 받아 **인덱스**와 **요소**를 반환



```
for i, value in enumerate(시퀀스 자료형 이름, [sequence]):  
    print(i, value)
```

- 2개의 매개 변수가 필요함.
 - ♦ 시퀀스 자료형 이름 : 리스트, 튜플, 딕셔너리 등
 - ♦ `[sequence]`는 인덱스 시작 번호 지정, 생략하면 기본값은 0.
➔ `i`는 인덱스 번호, `value`는 요소

example

- *for*문과 *enumerate*문을 사용해서 인덱스와 리스트에 저장된 값을 출력하시오.

```
>>> bucketList = ['부산', '제주', '속초', '대전']  
>>> for i, value in enumerate(bucketList):  
    print(i, value)
```

Result>

0 부산

1 제주

2 속초

3 대전

example2

- *for*문과 *enumerate*문을 사용해서 인덱스와 리스트에 저장된 값을 출력하시오.

```
>>> bucketList = bucketList = ['부산', '제주', '속초', '대전']  
>>> for i, value in enumerate(bucketList, 1):  
    print(i, value)
```

Result>

```
1 부산  
2 제주  
3 속초  
4 대전
```

Exercise 1: *for* 문과 *enumerate* 연산

- 다음과 같이 test 리스트를 만들고, 처리하는 프로그램을 작성하시오.

(1) 점수 리스트를 작성하시오.

```
test =[57, 99, 78, 85, 60]
```

(2) 리스트에 저장된 **인덱스**와, **점수**를
오른쪽 결과와 같이 출력하시오.

Result>

```
번호1. 학생의 점수는 57점 입니다.  
번호2. 학생의 점수는 99점 입니다.  
번호3. 학생의 점수는 78점 입니다.  
번호4. 학생의 점수는 85점 입니다.  
번호5. 학생의 점수는 60점 입니다.
```

Exercise 1 : Solution

ex5-score.py

```
test = [57, 99, 78, 85, 60]

for i, value in enumerate(test, 1):
    print('번호{}. 학생의 점수는 {}점 입니다.'.format(i, value))
```

Exercise 2: *for* 문과 *enumerate* 연산

- exercise1을 수정하여 Result>와 같이 출력되도록 프로그램을 수정하십시오.

(1) 점수 리스트를 그대로 사용하십시오.

test = [57, 99, 78, 85, 60]

(2) 리스트에 저장된 **인덱스 번호**와, 점수
그리고, 점수에 대한 **합격/불합격**을
오른쪽 결과와 같이 출력하십시오.

(if문을 사용하여 70 이상이면 "합격",
그렇지 않으면 "불합격")

Result>

번호1. 학생의 점수는 57점 입니다. 불합격!
번호2. 학생의 점수는 99점 입니다. 합격!
번호3. 학생의 점수는 78점 입니다. 합격!
번호4. 학생의 점수는 85점 입니다. 합격!
번호5. 학생의 점수는 60점 입니다. 불합격!

Exercise 2 : Solution

```
test = [57, 99, 78, 85, 60]
```

ex5-pass_fail.py

```
for i, value in enumerate(test, 1):
```

```
    if value >= 70:
```

```
        print('번호{}. 학생의 점수는 {}점 입니다. 합격!'.format(i, value))
```

```
    else :
```

```
        print('번호{}. 학생의 점수는 {}점 입니다. 불합격!'.format(i, value))
```

■ 최대값, 최소값, 합계 구하기

```
>>> test = [38, 23, 57, 99, 65]
```

```
>>> max(test)
```

최대값 구하기

```
99
```

```
>>> min(test)
```

최소값 구하기

```
23
```

```
>>> sum(test)
```


합계 구하기

```
282
```

Exercise 3 & Solution

- 다음 midterm 리스트를 작성하고 평균값을 구하시오.

ex5-midterm.py

```
>>> midterm = [75, 90, 66, 100, 50]
>>> print('Average : ', )
```

Result>

Average : 83.0

■ 2단 만들기

ex5-timesTable.py

```
dan = 2
for i in range(1,10):
    print('{} x {} = {}'.format(dan, i, dan*i))
```

2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18

■ 2단~9단 만들기

ex5-timesTable2.py

```
for dan in range(2,10):  
    for i in range(1,10):  
        print('{} x {} = {}'.format(dan, i, dan*i))  
    print()
```

2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18

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

4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32

⋮
⋮

Example4

- 구구단 테이블 만들기(단이 밖에 for문)

ex5-timesTable3.py

```
for dan in range(2,10):  
    for i in range(1,10):  
        print('{}x{}= {}'.format(dan, i, dan*i), end='\\t')  
    print()
```

2 x 1 = 2	2 x 2 = 4	2 x 3 = 6	2 x 4 = 8	2 x 5 = 10	2 x 6 = 12	2 x 7 = 14	2 x 8 = 16	2 x 9 = 18
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
4 x 1 = 4	4 x 2 = 8	4 x 3 = 12	4 x 4 = 16	4 x 5 = 20	4 x 6 = 24	4 x 7 = 28	4 x 8 = 32	4 x 9 = 36
5 x 1 = 5	5 x 2 = 10	5 x 3 = 15	5 x 4 = 20	5 x 5 = 25	5 x 6 = 30	5 x 7 = 35	5 x 8 = 40	5 x 9 = 45
6 x 1 = 6	6 x 2 = 12	6 x 3 = 18	6 x 4 = 24	6 x 5 = 30	6 x 6 = 36	6 x 7 = 42	6 x 8 = 48	6 x 9 = 54
7 x 1 = 7	7 x 2 = 14	7 x 3 = 21	7 x 4 = 28	7 x 5 = 35	7 x 6 = 42	7 x 7 = 49	7 x 8 = 56	7 x 9 = 63
8 x 1 = 8	8 x 2 = 16	8 x 3 = 24	8 x 4 = 32	8 x 5 = 40	8 x 6 = 48	8 x 7 = 56	8 x 8 = 64	8 x 9 = 72
9 x 1 = 9	9 x 2 = 18	9 x 3 = 27	9 x 4 = 36	9 x 5 = 45	9 x 6 = 54	9 x 7 = 63	9 x 8 = 72	9 x 9 = 81

Example5

- 구구단 테이블 만들기(단이 안에 for문)

```
for i in range(1,10):  
    for dan in range(2,10):  
        print('{}x{}= {}'.format(dan, i, dan*i), end='\\t')  
    print()
```

2 x 1 = 2	3 x 1 = 3	4 x 1 = 4	5 x 1 = 5	6 x 1 = 6	7 x 1 = 7	8 x 1 = 8	9 x 1 = 9
2 x 2 = 4	3 x 2 = 6	4 x 2 = 8	5 x 2 = 10	6 x 2 = 12	7 x 2 = 14	8 x 2 = 16	9 x 2 = 18
2 x 3 = 6	3 x 3 = 9	4 x 3 = 12	5 x 3 = 15	6 x 3 = 18	7 x 3 = 21	8 x 3 = 24	9 x 3 = 27
2 x 4 = 8	3 x 4 = 12	4 x 4 = 16	5 x 4 = 20	6 x 4 = 24	7 x 4 = 28	8 x 4 = 32	9 x 4 = 36
2 x 5 = 10	3 x 5 = 15	4 x 5 = 20	5 x 5 = 25	6 x 5 = 30	7 x 5 = 35	8 x 5 = 40	9 x 5 = 45
2 x 6 = 12	3 x 6 = 18	4 x 6 = 24	5 x 6 = 30	6 x 6 = 36	7 x 6 = 42	8 x 6 = 48	9 x 6 = 54
2 x 7 = 14	3 x 7 = 21	4 x 7 = 28	5 x 7 = 35	6 x 7 = 42	7 x 7 = 49	8 x 7 = 56	9 x 7 = 63
2 x 8 = 16	3 x 8 = 24	4 x 8 = 32	5 x 8 = 40	6 x 8 = 48	7 x 8 = 56	8 x 8 = 64	9 x 8 = 72
2 x 9 = 18	3 x 9 = 27	4 x 9 = 36	5 x 9 = 45	6 x 9 = 54	7 x 9 = 63	8 x 9 = 72	9 x 9 = 81

Homework 1 : 쇼핑리스트 만들기

- 다음과 같이 목록을 생성하고 처리하는 프로그램을 작성하시오 : shoppinglist.py

- (1) 빈 리스트를 만드시오. `slist = []`
- (2) *while* 문을 사용하여 참일때까지 계속하여 리스트에 요소를 추가하고 정렬하시오.
- (3) 만일, "quit"를 입력하면 프로그램을 종료하고, 항목을 입력할 때마다 리스트를 출력하시오.

```
원하는 물건을 입력하세요 : 컴퓨터
1 컴퓨터
원하는 물건을 입력하세요 : 스피커
1 스피커
2 컴퓨터
원하는 물건을 입력하세요 : 마우스
1 마우스
2 스피커
3 컴퓨터
원하는 물건을 입력하세요 : quit
```

Homework 1 : Solution

```
slist = []
```

```
while True :
```

```
    item= input('원하는 물건을 입력하세요 : ')
```

```
    if item == 'quit' :
```

```
        break
```

```
# slist에 항목 추가하기
```

```
# 오름차순 정렬하기
```

```
for
```

```
    print(i, value)
```

```
# 인덱스번호와 요소 출력
```

Homework 2 : 짝수, 홀수 리스트 만들기

- 다음과 같이 목록을 만들고 짝수, 홀수 리스트를 구분하여 리스트를 작성하시오 :

(1) 리스트를 만드시오.

number = [4, 13, 15, 70, 51, 23, 38, 9, 12, 5]

(2) number리스트를 읽고 짝수이면 even리스트에,
홀수이면 odd 리스트에 추가하시오.

(3) 리스트를 정렬하여 출력하시오.

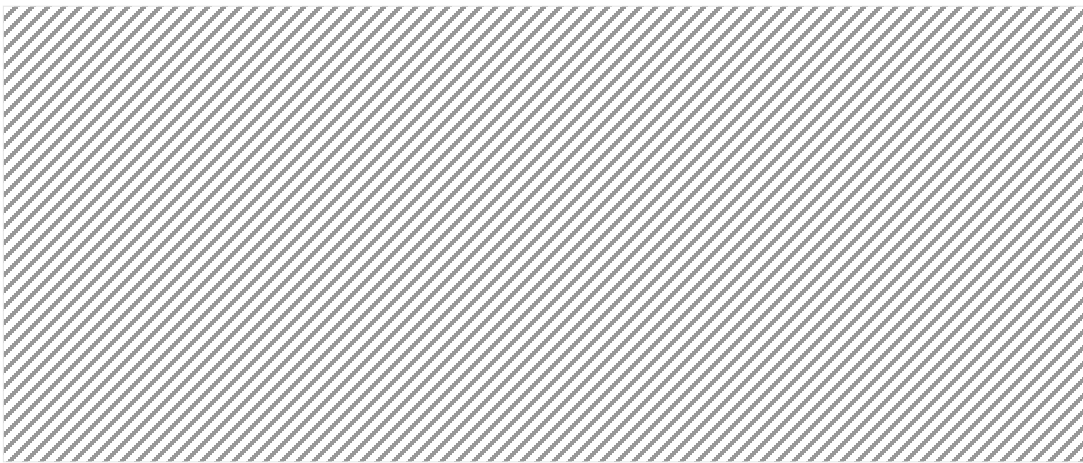
짝수 리스트 = [4, 12, 38, 70]
홀수 리스트 = [5, 9, 13, 15, 23, 51]

Homework 2 : Solution

```
number = [4, 13, 15, 70, 51, 23, 38, 9, 12, 5]
```

```
even = []
```

```
odd = []
```



```
even.sort()
```

```
odd.sort()
```

```
print('짝수 리스트 = ', even)
```

```
print('홀수 리스트 = ', odd)
```

짝수, 홀수 리스트 정렬하기

Q&A