학습 내용

1부. 프로그래밍 언어 기본

1장. 파이썬 개요 및 개발환경 구성

2장. 자료형과 연산자

3장. 데이터 구조

- **1. 리스트**
- **2. 튜플**
- **3. 딕셔너리**
- **4. 셋**
- **5. enumerate**

4장. 제어문

5장. 함수

1절. 리스트

- 리스트를 이용하면 여러 개 값을 저장
- 리스트를 만들려면 대괄호('['와 ']')를 이용
- 인덱스를 이용해 읽기와 쓰기를 지원
- 부분 데이터셋을 뽑아내는 슬라이싱(slicing)을 지원
- 파이썬의 인덱스는 0부터 시작

표 1. 파이썬에서 리스트 다루기

1절. 리스트

방법	설명
listData = [ ]	리스트를 만들어 줌
len(listData)	리스트의 항목의 수를 반환
mix(listData), max(listData)	리스트에서 가장 작은(min) 항목과 가장 큰(max) 항목을 반환
listData[start:stop]	리스트의 start 위치부터 stop 위치까지 부분 데이터를 추출(stop 위치의 항목은 포함 안 됨)
listData.append(value)	list에 value를 추가
listData.clear()	list의 모든 항목을 삭제
listData.count(value)	리스트에서 value의 개수를 반환
listData.extend(newList)	list에 newList를 추가
+	두 리스트를 연결함
listData.index(value, position=0)	position위치 이후에서 value의 값이 있는 인덱스를 반환
listData.insert(index, value)	list의 index위치에 value를 삽입
listData.remove(value)	리스트에서 해당 값을 삭제
del listData[index]	리스트에서 인덱스를 이용해 항목을 삭제
listData.pop()	리스트에서 가장 마지막 항목을 반환하고 삭제
listData.reverse()	리스트의 항목들의 순서를 반대로 함
listData.sort(reverse=False)	리스트의 항목들을 정렬. reverse 속성을 True로 하면 내림차순으로 정렬

1) 1차원 리스트

- 리스트는 [ ]를 이용해 만듬
- [ ]안에 [ ]를 넣으면 차원이 증가
  - 변수명 = [...] : 1차원 리스트
  - 변수명 = [[...], ...] : 2차원 리스트
  - 변수명 = [[[...], ...], ...] : 3차원 리스트

```
1  fruits = ["banana", "apple", "orange", "grape"]
2  print(fruits)
```

```
['banana', 'apple', 'orange', 'grape']
```

```
1  numbers = [1,2,3,4,5]
2  print(numbers)
```

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

2) 다차원 리스트

다차원 리스트 중에서 2차원 구조가 가장 많이 사용됨

```
1  numbers_2d = [[1,2,3,4,5], [10,20,30,40,50], [1,3,5,7,9], [2,4,6,8,10]]
2  print(numbers_2d)
```

```
[[1, 2, 3, 4, 5], [10, 20, 30, 40, 50], [1, 3, 5, 7, 9], [2, 4, 6, 8, 10]]
```

행 \ 열	0	1	2	3	4
0	1	2	3	4	5
1	10	20	30	40	50
2	1	3	5	7	9
3	2	4	6	8	10

```
1  numbers_2d_v = [[1,2,3], [10,20,30,40], [1,3,5,7,9], [2,4,6,8,10,12]]
2  print(numbers_2d_v)
```

```
[[1, 2, 3], [10, 20, 30, 40], [1, 3, 5, 7, 9], [2, 4, 6, 8, 10, 12]]
```

행 \ 열	0	1	2	3	4	5
0	1	2	3			
1	10	20	30	40		
2	1	3	5	7	9	
3	2	4	6	8	10	12

1.2. 기본 정보 조회

- len() : 길이(항목의 수)
- min(), max() : 최댓값, 최솟값

```
1  numbers = [1,2,3,4,5]
2  numbers_2d = [[1,2,3,4,5], [10,20,30,40,50], [1,3,5,7,9], [2,4,6,8,10]]
```

```
1  len(numbers), len(numbers_2d), len(numbers_2d[0])
```

```
(5, 4, 5)
```

```
1  min(numbers), max(numbers)
```

```
(1, 5)
```

```
1  max(numbers_2d)
```

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

1) +에 의한 연결, 2) *에 의한 반복

- + : 두 리스트를 연결
- * : 리스트를 곱한 수만큼 반복

```
1  numbers = [1,2,3,4,5]
2  numbers_2d = [[1,2,3,4,5], [10,20,30,40,50], [1,3,5,7,9], [2,4,6,8,10]]
```

```
1  new_numbers = numbers + numbers
2  new_numbers
```

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

```
1  3*numbers
```

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

3) 항목 추가하기, 4) 리스트 추가하기, 5) 중간에 삽입하기

- append()
  - 단일 항목을 맨 뒤에 추가
  - 리스트를 append 하면 리스트가 항목으로 추가됨
- extend()
  - 리스트를 항목별로 맨 뒤에 추가
- insert()
  - 지정한 인덱스 위치에 삽입

```
1  numbers.append(10)
2  numbers
```

```
[1, 2, 3, 4, 5, 10]
```

```
1  numbers.append([20, 30, 40, 50])
2  numbers
```

```
[1, 2, 3, 4, 5, 10, [20, 30, 40, 50]]
```

```
1  numbers = [1,2,3,4,5]
2  numbers.extend([10,20,30,40,50])
3  numbers
```

```
[1, 2, 3, 4, 5, 10, 20, 30, 40, 50]
```

```
1  numbers.insert(5, 100)
2  numbers
```

```
[1, 2, 3, 4, 5, 100, 10, 20, 30, 40, 50]
```

1) 데이터 수 세기, 2) 항목의 위치 반환하기

- count() : 리스트에서 데이터의 개수를 반환
- index() : 해당 항목의 위치 반환
- 항목을 찾지 못하면 에러 발생

```
1  numbers = [1,3,5,7,9]
2  numbers.index(5)
```

```
2
```

```
1  numbers = [1,3,5,7,9,1,2,3,4,5]
2  numbers.index(5,3)
```

```
9
```

```
1  numbers = [1,3,5,7,9,1,2,3,4,5]
2  numbers.index(6,3)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback
<ipython-input-24-f30a67067ef4> in <module>()
      1 numbers = [1,3,5,7,9,1,2,3,4,5]
----> 2 numbers.index(6,3)

ValueError: 6 is not in list
```

3) 인덱스를 이용한 직접 접근

- [index] : 인덱스를 이용한 직접 접근

```
1  numbers = [1,3,5,7,9,1,2,3,4,5]
```

```
1  numbers[0], numbers[2]
```

```
(1, 5)
```

```
1  numbers[20]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-32-48ce5c7b3e82> in <module>()
----> 1 numbers[20]

IndexError: list index out of range
```

4) 다차원 리스트에서 인덱스 사용

- 다차원 리스트에서 인덱스를 이용해 데이터를 참조하려면 차원 별로 [index]를 지정

```
1  a = [ 1, 2, [ 'a', 'b', [ 'hello', 'world' ] ] ]
2  print(a[2][2][0])
```

```
hello
```

1) [start : stop]

- [start : stop] : start부터 stop까지 부분 리스트 추출

```
1  numbers = [1,2,3,4,5,6,7,8,9,10]
2  numbers_2d = [[1,2,3,4,5], [10,20,30,40,50], [1,3,5,7,9], [2,4,6,8,10]]
```

```
1  numbers[2:4]
```

```
[3, 4]
```

```
1  numbers[2:]
```

```
[3, 4, 5, 6, 7, 8, 9, 10]
```

```
1  numbers[:4]
```

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

```
1  numbers[:]
```

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

```
1  numbers[-5:-1]
```

```
[6, 7, 8, 9]
```

```
1  numbers[:-1]
```

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

```
1  numbers[-1:-5]
```

```
[]
```

```
1  numbers_2d[1:3]
```

```
[[10, 20, 30, 40, 50], [1, 3, 5, 7, 9]]
```

```
1  numbers_2d[1][1:4]
```

```
[20, 30, 40]
```

2) [start : stop : step]

- [*start* : *stop* : *step*] 형식은 매 *step* 번째 아이템을 추출

```
1  numbers = [1,2,3,4,5,6,7,8,9,10]
2  numbers_2d = [[1,2,3,4,5], [10,20,30,40,50], [1,3,5,7,9], [2,4,6,8,10]]
```

```
1  numbers[::2]
```

```
[1, 3, 5, 7, 9]
```

```
1  numbers[::-1]
```

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

```
1  numbers[-5::-1]
```

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

1) 인덱싱으로 수정하기

- 인덱싱으로 데이터 수정하기
  - 리스트 데이터는 인덱싱 방법으로 수정할 수 있음
  - 지정한 인덱스 위치의 데이터를 다른 데이터로 바꿈
- 슬라이싱으로 데이터 수정하기
  - 지정한 범위의 리스트 항목을 다른 항목들로 한꺼번에 바꿈
  - 바꾸려는 항목이 더 많거나 더 적어도 항목들을 일괄적으로 변경

```
1  numbers[2] = 2
2  numbers[4] = 40
3  numbers[6] = 600
```

```
1  numbers
```

```
[1, 3, 2, 7, 40, 1, 600, 3, 4, 5]
```

```
1  numbers = [1,3,5,7,9,1,2,3,4,5]
2  numbers[0:5] = [2,4,6,8,10]
3  numbers
```

```
[2, 4, 6, 8, 10, 1, 2, 3, 4, 5]
```

1) pop()

- pop() : 맨 뒤의 항목 반환 및 삭제

```
1  numbers = [1,2,3,4,5,6,7,8,9,10]
2  numbers_2d = [[1,2,3,4,5], [10,20,30,40,50], [1,3,5,7,9], [2,4,6,8,10]]
```

```
1  numbers.pop()
```

```
10
```

```
1  numbers
```

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

```
1  numbers_2d.pop()
```

```
[2, 4, 6, 8, 10]
```

```
1  numbers_2d
```

```
[[1, 2, 3, 4, 5], [10, 20, 30, 40, 50], [1, 3, 5, 7, 9]]
```

2) remove()

- remove() : 해당 항목 삭제

```
1  numbers = [1,2,3,4,5,6,7,8,9,10]
```

```
1  numbers.remove(3)
2  numbers
```

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

```
1  numbers_2d = [[1,2,3,4,5], [10,20,30,40,50], [1,3,5,7,9], [2,4,6,8,10]]
```

```
1  numbers_2d.remove([1,2,3,4,5])
2  numbers_2d
```

```
[[10, 20, 30, 40, 50], [1, 3, 5, 7, 9], [2, 4, 6, 8, 10]]
```

3) del

- del : 지정한 위치 항목 삭제, 변수 삭제

```
1  numbers = [1,2,3,4,5,6,7,8,9,10]
```

```
1  del numbers[3]
2  numbers
```

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

```
1  numbers = [1,2,3,4,5,6,7,8,9,10]
```

```
1  del numbers[3:8]
2  numbers
```

```
[1, 2, 3, 9, 10]
```

```
1  del numbers
```

4) clear()

- clear() : 모든 항목 삭제

```
1  numbers = [1,2,3,4,5,6,7,8,9,10]
```

```
1  numbers.clear()
2  numbers
```

```
[]
```

1.8. 정렬하기

- sort() : 정렬(reverse=True 속성을 이용하면 내림차순 정렬)
- reverse() : 역순으로 나열(내림차순 정렬이 아님)
- [::-1] : 역순으로 나열

❖ sort()와 reverse()는 원본데이터를 변경하지만 [::-1]은 역순으로 출력하고 원본데이터는 바꾸지 않습니다.

```
1  numbers = [6, 2, 7, 4, 10, 8, 3, 9, 1, 5]
```

```
1  numbers.sort()
2  numbers
```

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

```
1  numbers = [6, 2, 7, 4, 10, 8, 3, 9, 1, 5]
```

```
1  numbers.sort(reverse=True)
2  numbers
```

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

```
1  numbers = [6, 2, 7, 4, 10, 8, 3, 9, 1, 5]
```

```
1  numbers.reverse()
2  numbers
```

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

```
1  numbers = [6, 2, 7, 4, 10, 8, 3, 9, 1, 5]
```

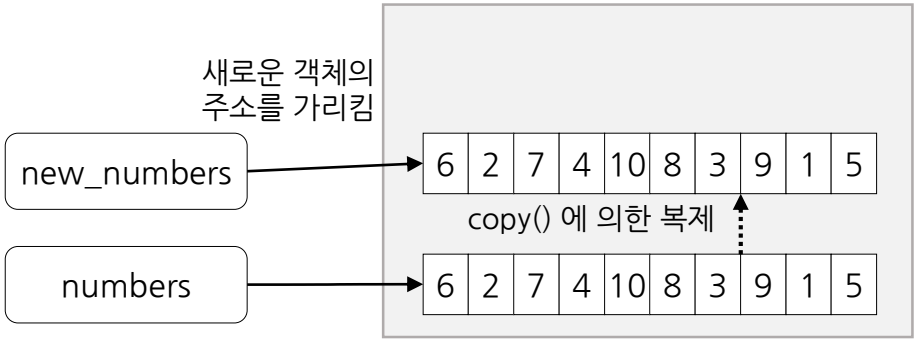
```
1  new_numbers = numbers[::-1]
2  new_numbers
```

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

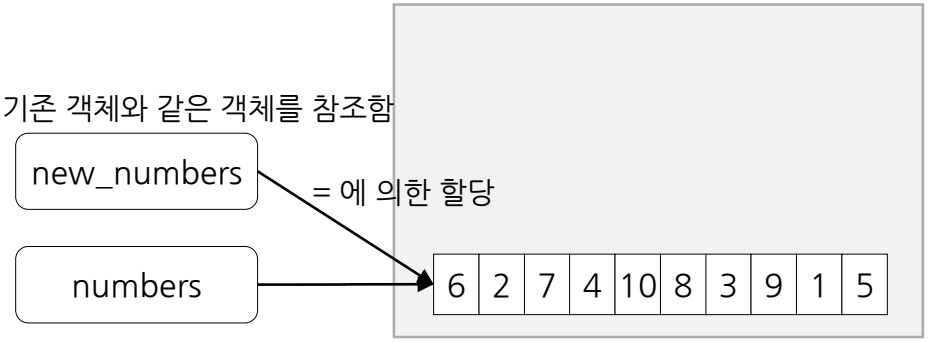
리스트 복제

- copy() : 복제된 새로운 객체를 생성
- = : 주소를 복사해 같은 객체를 참조

```
1  numbers = [6, 2, 7, 4, 10, 8, 3, 9, 1, 5]
```



copy() 함수를 이용한 복사



할당연산자(=)를 이용한 복사

```
1  new_numbers = numbers.copy()
2  print(numbers)
3  print(new_numbers)
```

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

```
1  new_numbers = numbers
2  print(numbers)
3  print(new_numbers)
```

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

2.1. 튜플 만들기

2절. 튜플

- 튜플(tuple)은 소괄호('('와 ')')를 이용해 만듬
- 읽기 전용
  - 튜플은 속도가 빨라 수정이 필요 없는 배열 형태의 데이터 타입에 사용
  - 데이터를 수정할 수 없기 때문에 제공되는 함수가 많지 않음

방법	설명
tupleData = ( )	튜플을 만들어 줍니다.
len(tupleData)	튜플의 항목 수를 반환합니다.
min(tuple), max(tuple)	튜플에서 가장 작은 값(min)과 가장 큰 값(max)을 반환합니다.
tupleData.count(value)	튜플에서 value의 개수를 반환합니다.
tupleData.index(value, position)	position 위치 이후에서 value가 있는 인덱스를 반환합니다.

```
1  numbers1 = (1,2,3,4,5)
2  type(numbers1)
```

```
tuple
```

```
1  numbers2 = (1,)
2  type(numbers2)
```

```
tuple
```

```
1  numbers_2d = ((1,2,3), (4,5,6))
2  numbers_2d
```

```
((1, 2, 3), (4, 5, 6))
```

3.1. 딕셔너리 만들기

3절. 딕셔너리

- 키(key)와 값(value)의 쌍으로 구성된 자료 구조
- 딕셔너리를 만들기 위해서는 중괄호('{'와 '}')를 이용
- 키는
  - 중복이 없이 유일한 값이어야 함
  - 리스트 타입을 사용할 수 없지만 튜플 타입은 사용할 수 있음
- 값은 중복이 가능하며 모든 타입이 가능
- 인덱스를 이용한 데이터의 참조는 지원 안함
- 딕셔너리 키 목록에 없는 데이터를 사용하여 참조하면 에러가 발생

방법	설명
`dictData = {"key":"value", ... }`	딕셔너리를 만들어 줍니다.
`len(dictData)`	딕셔너리의 항목의 수를 반환합니다.
`dictData.items()`	딕셔너리의 각 항목들을 (key, value) 형식의 튜플들로 반환합니다.
`dictData.keys()`	딕셔너리의 키(key)들을 반환합니다.
`dictData.values()`	딕셔너리의 값(value)들을 반환합니다.

```
dictData = dict()
dictData['one'] = '하나'
dictData['two'] = '둘'
dictData['three'] = '셋'

word = input("영단어 ?")
print(dictData.get(word, "없음"))
```

4.1. 셋 만들기

4절. 셋

- 순서가 정해지지 않고, 중복을 허용하지 않는 집합
- 중괄호('{'와 '}')를 이용하여 정의

```
1  fruits = {"apple", "orange", "banana", "apple", "bear"}
```

```
1  fruits
```

```
{'apple', 'banana', 'bear', 'orange'}
```

```
1  fruits.add("apple")
```

```
1  fruits
```

```
{'apple', 'banana', 'bear', 'orange'}
```

```
1  fruits.add("mango")
```

```
1  fruits
```

```
{'apple', 'banana', 'bear', 'mango', 'orange'}
```

enumerate

5절. enumerate

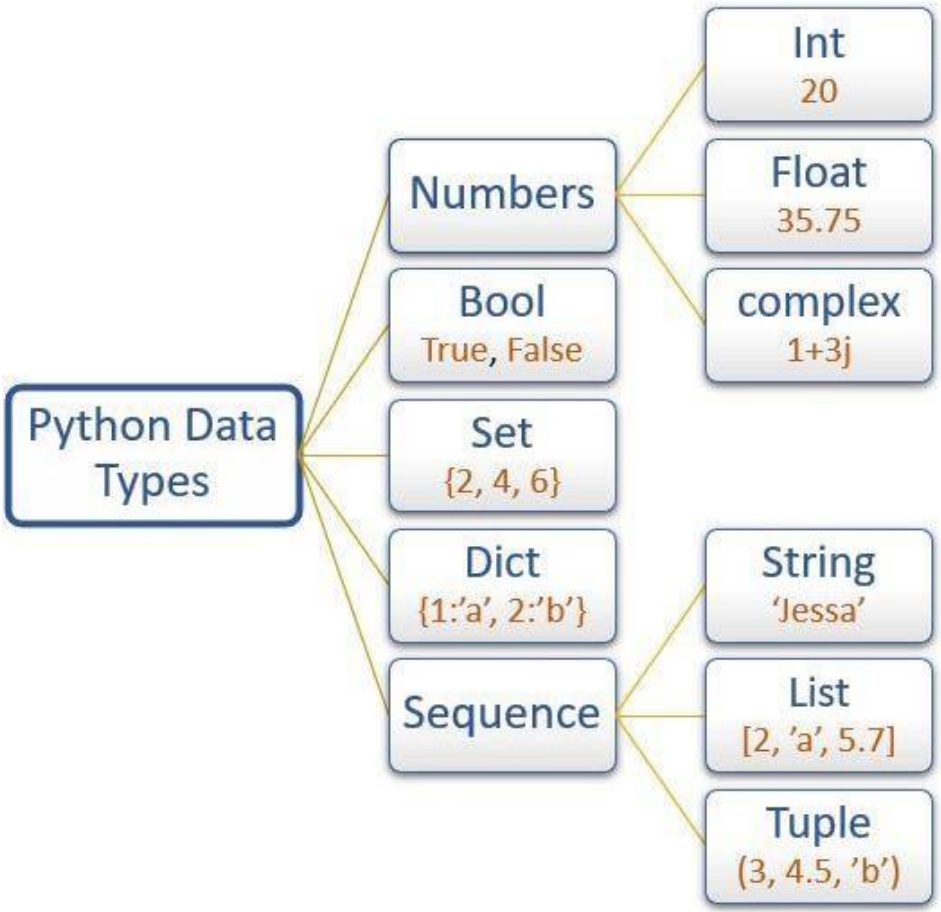
- 반복자(iterator) 또는 순서(sequence) 객체를 인수로 받음
- enumerate(iter) 이라고 사용했을 경우 이 함수는 iter 객체를 (0, iter[0]), (1, iter[1]), (2, iter[2]), ... 이런 형식으로 반환

```
1  fruits = ['watermelon', 'orange', 'mango', 'grape', 'banana', 'apple']
```

```
1  for index, value in enumerate(fruits):
2      print("{}번째 데이터는 {}입니다.".format(index+1, value))
3
```

```
1번째 데이터는 watermelon입니다.
2번째 데이터는 orange입니다.
3번째 데이터는 mango입니다.
4번째 데이터는 grape입니다.
5번째 데이터는 banana입니다.
6번째 데이터는 apple입니다.
```

변수가 담을 수 있는 기본 타입들



출처 : https://pynative.com/python-data-types/

연습문제(실습형)

- 다음 리스트가 주어졌을 경우 요구사항대로 코드를 작성하시오

```
numbers = [1,2,3,4,5,6,7,8,9,10]
```

1. 숫자 100을 맨 뒤에 추가하세요
2. 다음 리스트를 numbers리스트 맨 뒤에 추가하세요

```
data = [200, 300, 400, 500]
```

3. 처음 다섯 개 숫자만 출력하세요
4. 리스트에서 짝수 번째 데이터만 출력하세요
5. 짝수번째 데이터를 모두 0으로 바꾸세요
6. 데이터를 역순으로 나열하세요(내림차순 정렬이 아닙니다)

연습문제

- 다음 딕셔너리 데이터가 주어졌을 경우 요구사항대로 코드를 작성하세요.

```
member_info = {“name”:”홍길동”, “age”:20, address”:”서울시 마포구“,
                “score”:90}
```

7. address값을 출력하세요
8. score 를 출력하고 member_info 딕셔너리에서 삭제하세요.
9. address를 “서울시 서대문구”로 변경하세요.
10. member_info 딕셔너리 데이터의 값을 리스트로 출력하세요

연습문제(서술형)

1. 다음 코드를 실행했을 때 출력되는 것은?

```
L1 = (“orange”,”apple”,”banana”,”kiwi”)
new_list = [i for i in L1 if len(i)>5]
 print(new_list)
```

2. 다음 코드와 실행결과가 잘못 짝지어진 것은?

① print(list(range(10))) - [0,1,2,3,4,5,6,7,8,9]

② print(list(range(5,10))) - [5,6,7,8,9]

③ print(list(range(10,0,-1))) - [9,8,7,6,5,4,3,2,1,0]

④ print(list(range(10,20,2))) - [10,12,14,16,18]

연습문제

3. 다음 구문에 의해 출력되는 것은?

```
numbers_2d = [ [1,2,3,4,5], [10,20,30,40],[1,3,5],[2,] ]
len(numbers_2d[3])
```

4. 다음 코드의 실행결과가 나올 수 있는 빈칸에 들어갈 함수 이름은?

```
numbers = [1,2,3,4,5]
numbers._____([10,20,30,40,50])
numbers
```

결과 : [1,2,3,4,5,10,20,30,40,50]

연습문제

5. 다음 구문을 실행할 때 결과는 ?

```
numbers = list(range(10))
numbers[ : : 2]= [0] * len(numbers[ : : 2])
print(numbers)
```

6. 다음 구문을 실행할 때 결과는 ?

```
numbers = [0,1,2,3,4,5,6,7,8,9]
numbers[ : : 2]= numbers[5:]
print(numbers)
```

연습문제

7. 다음 구문을 실행할 때 결과는 ?

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
numbers[ : : 2]= numbers[:5]
print(numbers)
```

8. 다음 데이터에서 10을 출력할 수 없는 구문은?

```
my_dic = {“a”:10, “b”:20, “c”:30}
```

① my_dic[0]

② my_dic[‘a’]

③ list(my_dic.items())[0][1]

④ list(my_dic.values())[0]

⑤ my_dic.get(‘a’)

연습문제

9. 다음 프로그램의 결과를 출력하시오

```
fruits = [ ‘apple’, ‘banana’, ‘orange’]
x, y, z = fruits
print(x)
print(y)
print(z)
```

10. 다음 프로그램의 결과를 출력하시오

```
fruits = [ ‘apple’, ‘banana’, ‘orange’]
print(len(fruits))
```

11. 파이썬에서 함수를 만들기 위해 선언하는 단어는?