(Ch 2-A) 아래 문제들을 Ch 2 강의자료를 참고하여 조별구성원들이 협력하여 풀고 결과를 PPT file로 만들어서 조교에게 제출

(1번, 10점) 다음 코드의 수행결과를 보이시오.

(a) `>>> 2 + 4 % 3`

정답: 3

(b) `>>> 3.2 // 1.5`

정답: 2.0

(c) `>>> int(2.5) + round(3.4)`

정답: 5

(d) `>>> float("123") == 123`

정답: True

(2번, 10점)　다음 코드의 수행결과를 보이시오.

(a)
```
>>> “abc” < “bcd”
```
정답: True

(b)
```
>>> ord(“C”) – ord(“A”)
```
정답: 2

(c)
```
>>> ls = [1, 2, 3]
>>> ls[-1] = ls[1:2]
>>> ls
```
정답: [1, 2, [2]]

(d)
```
>>> ls = [1, 2, 3]
>>> ls.append([4, 5])
>>> ls
```
정답: [1, 2, 3, [4, 5]]

(3번, 10점)　다음 코드의 수행결과를 보이시오.

(a)
```
>>> ls = [1, 2, {2: 3, 1: 4, 0: 5}]
>>> ls[2][2]
```
정답: 3

(b)
```
>>> s = "hello, world!"
>>> s[-6:12]
```
정답: 'world'

(c)
```
>>> s = ["hello", "world!"]
>>> s[1][1]
```
정답: 'o'

(d)
```
>>> “123” * 2
```
정답: '123123'

(4번, 10점)　　다음 코드의 수행결과를 보이시오.

(a, 2점)　>>> lst = [1, 2, 3, 4, 5]

>>> lst[1:3]

>>> lst[0:-3]

정답:
[2, 3]
[1, 2]

(b, 4점) 다음 코드의 수행결과를 보이고 그 이유를 설명하시오.

>>> t = (1, 2, 3)

>>> t[2] = 4

정답: error

설명: tuple 함수는 list 함수와 다르게 메모리를 적게 사용하는 대신, data의 수정이 불가능하다. 아예 새로 정의를 해야 하는 것이 아니라면, 2번째 명령처럼 data의 일부분만 수정하는 것이 불가능한 함수이므로 error가 뜬다.

(c, 4점) 다음 코드의 수행결과를 보이고 그 이유를 설명하시오.

>>> s = {1, 2, 3, 4, 5}

>>> s[2]

정답: error

설명: set 함수는 list함수나 tuple함수로 변환한 후에 []안에 숫자를 입력해야 그 순서에 맞는 data가 출력된다. Set함수는 집합 개념이므로, 순서의 개념이 없기 때문이다.

(5번, 10점) 아래 코드의 수행결과를 보이고 함수( ) 의 정체를 설명하시오

(a) >>> abs(-1.5)

정답: **1.5**

abs함수는 괄호 안의 숫자의 절댓값을 반환한다.

(b) >>> eval("2+3")

정답: **5**

eval함수는 식을 입력 받아 실행하는 함수이다.
다음 코드에서는 문자열 "2+3" 이 나타내는 식 2+3을 실행하여 그 결과인 5를 출력하게 된다.

(c) >>> pow(2, 4)

정답: **16**

pow(a, b)는 a의 b제곱을 계산하여 반환한다.

(d) >>> len("korea")

정답: **5**

len함수(length를 의미한다)는 data값의 길이를 반환한다.

(6번, 10점) 수행결과를 보이고 왜 그런지를 설명하시요

(a, 3점) >>> print("3" * 3, "3" + "3")

정답: **333 33**

이유: "3" * 3은 "3"을 세 번 반복해서 이어 붙인다는 뜻(Repetition)으로 "333"을 의미한다. "3" + "3"은 "3"에 "3"을 이어 붙인 "33"을 의미한다.(Concatenation) 또한 print 함수에 comma를 사용하면 두 문자열을 한 칸 띄워서 출력할 수 있으므로, 결과적으로 333 33이 출력된다.

(b, 4점)
```
>>> a = 1
>>> b = 2
>>> print(a, b)
>>> c = b
>>> b = a
>>> a = c
>>> print(a, b)
```

정답: **1 2**
2 1

이유: 3번째 줄의 print명령에서는 첫번째 줄과 두번째 줄에서 대입한 a와 b의 값이 그대로 출력되는데, 4번째 줄에서 c의 값에 2가 저장되고, 5번째 줄에서 b의 값에 1이 저장되고, 6번째 줄에서 a의 값에 2가 저장되어 7번째 줄의 print명령에서는 a에 2, b에 1이 저장되어 출력된다.

(c, 3점)
```
>>> x, y = 1, 2
>>> x, y = y, x
>>> print(x, y)
```

정답: **2 1**

이유: 첫 줄에서 x라는 문자에는 1의 값을, y라는 문자에는 2의 값을 입력했지만, 두 번째 줄에서 x자리에 y를, y자리에 x를 대입하였으므로 print(x,y)라는 명령에서는 첫 줄에서 대입한 x와 y의 값이 서로 swap되어 출력된다.

(7번 10점) For문장을 range( )를 써서 다음과 같은 결과를 보이시요

(a) 출력: 5, 10, 15, 20, 25, 30

정답:

```
# 7-(a)
for i in range(1,6):
    print(5*i,end=", ")
print("30")
```

(b) 출력: 0, 7, 14, 21, 28, 35, 42, 49

정답:

```
# 7-(b)
for k in range(7):
    print(7*k,end=", ")
print("49")
```

*이 문제는 스크립트 파일(.py)로 만들어 실행하였습니다.

[8번, 10점]　아래 코드 실행 결과를 작성하시오

(a)
```
>>> x=2
>>> x-=-2
>>> print(x)
```
정답: 4

(b)　다음 코드의 수행결과를 보이고 그 이유를 설명하시오.
```
>>> ls = {1, 2, 3, 2, 1}
>>> print(len(ls))
```
정답:　3

설명: ls 의 data type이 set이기에 중복되는 값이 있을 수 없으므로, ls = {1,2,3}으로 저장 되어있다. len 함수에 set를 넣으면 원소의 개수를 세는데, ls의 원소는 3개 이므로 3이 출력된다.

(c)
```
>>> for i in range(1, 5, 2):
        print(i, end=' ')
```
정답:1 3

설명 : [1, 5)의 수를 간격 2로 출력하므로 1과 3이 출력된다.

(d)
```
>>> i = 0
>>> while i >= 10:
        print(i)
```
정답:

설명:while 뒤의 조건문의 bool값이 False이므로 print(i)는 실행 되지 않고, 아무것도 출력되지 않는다.

[9번 10점] 아래 코드 실행 결과를 작성하시오

(a)
```
>>> text = "Hello,How,Are,You,Fine"
>>> words = text.split(",")
>>> print(words)
```

정답: ['Hello', 'How', 'Are', 'You', 'Fine']

(b)
```
>>> lst1 = [1, 2, 3]
>>> lst2 = [4, 5, 6]
>>> lst3 = lst1 + lst2
>>> print(lst3)
```

정답: [1, 2, 3, 4, 5, 6]

(c)
```
>>> lst1.extend(lst2)
>>> print(lst1)
```

정답: [1, 2, 3, 4, 5, 6]

(d)
```
>>> lst1.remove(5)
>>> print(lst1)
```

정답: [1, 2, 3, 4, 6]

[10번 10점] 아래 코드 실행 결과를 작성하시오

(a)
```
>>> lst = [“h”, “i”, “!”]
>>> lst.remove(“!”)
>>> print(lst)
```
정답: ['h', 'i']

(b)
```
>>> lst = [“h”, “i”, “!”]
>>> lst.pop(2)
>>> print(lst)
```
정답: ['h', 'i']

*이 문제는 스크립트 파일(.py)로 만들어 실행하였습니다.