과목평가 대비 문제 - 답o

1,2번. `List Slice`

- 다음의 코드를 print 한 결과를 작성하시오

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

# 참고 : a[10]은 IndexError

(1)aList[2:10]

(1-2)aList[2:12]

(2)aList[:5]

(3)aList[5:]

(4)aList[0:8:2]

(5)aList[:8:2]

(6)aList[::2]

(7)aList[5::2]

(8)aList[:]

(9)aList[2:2]

(10)aList[:-2]

(11)aList[-4:]

(12)aList[::-1]

(13)aList[::-2]

(14)aList[::-3]
```

▼ answer

```
(1)aList[2:10]
# [3, ... 10]

(1-2)aList[2:12]
# [3, ... 10]
# aList[10]은 존재하지 않는 인덱스이기 때문에 참조할 수 없다
```

```
# 하지만 aList[2:10]에서 10은 끝 값이며 실제로는 참조하는 범위에 포함되지 않는다
# 따라서 2번 인덱스부터 끝까지라는 의미이며, 이 경우에도 마찬가지로 에러가 아니다

(2)aList[:5]
# [1,2,3,4,5]

(3)aList[5:]
# [6,7,8,9,10]

(4)aList[0:8:2]
# [0,2,4,6]

(5)aList[:8:2]
# [0,2,4,6]

(6)aList[::2]
# [1,3,5,7,9]
# 간격이 명시되면 start, end 값은 동시에 생략될 수 있다

(7)aList[5::2]
# [6, 8, 10]

(8)aList[:]
# [1,2,3,4,5,6,7,8,9,10]
# step 까지 쓰지 않으면 [:] 으로 참조할 수 있다
# 이는 리스트의 처음과 끝까지를 의미하므로 전체 리스트를 의미한다
# 및, 리스트 전체에 대한 사본을 얻는 셈이 된다(복사)

(9)aList[2:2]
# []

(10)aList[:-2]
# [1, 2, 3, 4, 5, 6, 7, 8]

(11)aList[-4:]
# [7, 8, 9, 10]

(12)aList[::-1]
# [10,9,8,7,6,5,4,3,2,1]

(13)aList[::-2]
# [10, 8, 6, 4, 2]

(14)aList[::-3]
# [10, 7, 4, 1]
```

3번. 내장함수 - 줄맞춤

- 다음의 코드를 print 한 결과를 작성하시오

```
str = 'abc'

(1)str.ljust(8, '+')

(2)str.rjust(8, '+')

(3)str.rjust(8, 'lm') # hint: 에러가 날까 안날까?

(4)str.center(9, '+')

# ------------------------
str2 = '-123'

(5)str2.rjust(8, '0')

(5-2)str2.zfill(8)) # 보너스 문제
# 답: -0000123
# 오.른.쪽 정렬을 하면서 0을 채워주는 메서드로 zfill() 이 있다
# rjust() 와 zfill() 메서드 차이점은 + - 기호를 포함해서 인식하는지 차이가 있다
# 금액 표시같은 경우에는 zfill()를 사용하지만 우편번호나 전화번호등,
# 문자열 그대로 사용하고 싶은 경우에는 rjust()를 사용하는 것이 좋다
```

▼ answer

```
str = 'abc'

(1)str.ljust(8, '+')
# abc+++++

(2)str.rjust(8, '+')
#+++++abc

(3)str.rjust(8, 'lmn') -> 에러가 날까 안날까?
# TypeError

(4)str.center(9, '+')
# +++abc+++

# ------------------------
str2 = '-123'

(5)str2.rjust(8, '0')
# 0000-123
```

4번. 논리 연산자

- 다음의 코드를 print 한 결과를 작성하시오

```
(1)a = 5 and 4

(2)b = 5 or 3

(3)c = 0 and 5

(4)d = 5 or 0

(5)e = 0 or 1

(6)f = '' or 2
```

▼ answer

```
(1)a = 5 and 4
print(a) #4

(2)b = 5 or 3
print(b) #5

(3)c = 0 and 5
print(c) #0

(4)d = 5 or 0
print(d) #5

(5)e = 0 or 1
print(d) # 1

(6)f = '' or 1
print(d) # 1

# 특이사항 : 결과가 확실한 경우 두 번째 값은 확인하지 않는다
    # and 연산에서 첫번째 값이 False인 경우 무조건 False ⇒ 첫번째 값 반환
    # or 연산에서 첫번째 값이 True인 경우 무조건 True ⇒ 첫번째 값 반환
```

5번. Global , Local 변수

- 다음 중, 에러가 발생하는 코드를 **모두** 고르시오

```
# (1)
a = 10
def func1():
  global a
  a = 3

print(a)
func1()
print(a)

----------------------------------------
# (2)
a = 10
def func1():
  print(a)
  global a
  a = 3

print(a)
func1()
print(a)

----------------------------------------
# (3)

a = 10
def func1():
  global a
  a = 3

print(a)
func1(3)
print(a)

----------------------------------------
# (4)

def ham():
    a = 'spam'
    return a

ham()
print(a)
```

▼ answer

```
#함수 내부에서 글로벌 변수 변경하기
a = 10
def func1():
  global a
  a = 3

print(a)
func1()
```

```
print(a)

#10
#3
----------------------------------------
#global 관련 에러1 - 사용제한예시
a = 10
def func1():
  print(a)
  global a
  a = 3

print(a)
func1()
print(a)

#SyntaxError : name 'a' is used prior to global declaration
----------------------------------------
#global 관련 에러2 - 사용제한예시

a = 10
def func1():
  global a
  a = 3

print(a)
func1(3) ##여기가 문제인듯 ㅇㅇ.
print(a)

# TypeError: func1() takes 0 positional arguments but 1 was given
----------------------------------------------
def ham():
    a = 'spam'
    return a

ham()
print(a)

# NameError: name 'a' is not defined
# 함수를 실행시켜도 ,애초에 a가 정의되어 있지 않으므로 아무 의미 없는 것
```

6번. Positional, Keyword Packing

- 다음의 코드를 print 한 결과를 작성하시오

```
(1)
def add(*args):
  print(args, type(args))

print(add(1, 2, 3))
```

```
print(add(1))

# (1, 2, 3) <class 'tuple'>
# (1,) <class 'tuple'>  # (1)아님. (1,)임

# 포지셔널 패킹 : 별1개
# 인자에 *를 넣기만 했는데 바로 튜플로 만들어진다
------------------------
(2)
def family(**kwargs):
  print(kwargs, type(kwargs))

family(father = '고길동', son='둘리') #참고로, 여기에서 딕셔너리 형태로 안 써줘도 된다.

# 키워드 패킹 : 별2개! - 그러면 딕셔너리로 묶여서 키/값쌍이 만들어짐
# {'father': '고길동', 'son': '둘리'} <class 'dict'>

------------------------

# 즉, 함수에서 → 튜플로 반환하고 싶으면 *패킹 / 딕셔너리로 반환하고 싶으면 **패킹
```

7번. Join메서드에 대한 이해

- 다음의 코드를 print 한 결과를 작성하시오

```
numbers = [1,2,3]
words = ['안녕', 'hello']

(1)'!'.join('ssafy')

(2)' '.join(['3', '5'])

(3)numbers.join(' ')

(4)' '.join(numbers)

(5)words[0].join(words[1])

(6)"".join(words)
```

▼ answer

```
(1)'!'.join('ssafy')    # 's!s!a!f!y'

(2)' '.join(['3', '5']) # '3 5'

(3)numbers.join(' ')
# 주의점 : join은 string 메서드이다
```

```
(4)' '.join(numbers)
# 요소가 문자열이 아닌 경우
# typeerror : sequence item 0 : expected str instance, int found
# 따라서, print(' '.join(map(str, numbers))) 이렇게 써야함.

(5)words[0].join(words[1]) # 'h안녕e안녕l안녕l안녕o'

(6)"".join(words) # '안녕hello'
```

8번. 얕은 복사, 깊은 복사

- 다음의 코드를 print 한 결과를 작성하시오

```
(1)할당

original_list = [1, 2, 3]
copy_list = original_list
copy_list[0] = 'hello'
print(original_list == copy_list)

---------------------------------------

(2)얕은 복사1

a = [1, 2, ['a', 'b']]
b = a[:]
print(a, b)
b[2][0] = 0
print(a, b)

---------------------------------------

(3)얕은 복사2
original_list = [1, 2, [0, 1]]
copy_list = original_list[:]

copy_list[2] = 'h'
print(copy_list, original_list)

---------------------------------------

(4)얕은 복사3
original_list = [1, 2, {'a': 'apple'}]
copy_list = original_list[:]

copy_list[2]['a'] = 'h'
print(copy_list, original_list)

---------------------------------------

(5)깊은 복사
import copy
```

```
a = [1, 2, ['a', 'b']]
b = copy.deepcopy(a)
print(a, b)
b[2][0] = 0
print(a, b)
```

▼ answer

```
(1)True
# (할당)대입 연산자(=)를 통한 복사는 해당객체에 대한 객체 참조(원본 통)를 복사한다


(2)얕은 복사1
# [1, 2, ['a', 'b']] [1, 2, ['a', 'b']]
# [1, 2, [0, 'b']] [1, 2, [0, 'b']]

# id를 통해서도 확인해보자
# 분명a와 b의 id는 다르다는 것을 확인하였지만,
# 내부 값은 영향을 받게 되었다.
# 이유는 : a[2]와 b[2]가 서로 같은 주소를 바라보고 있기 때문에 벌어진 일이다
# 즉, 1차원 리스트면 그냥 얕은 복사하면 되는데,
# 이렇게 2차원 경우에는 깊은 복사를 해야 문제가 해결된다

(3)얕은 복사2
# [1, 2, 'h'] [1, 2, [0, 1]]
# 이건 2차원 리스트를 파괴한 경우
# 그냥 덮어씌운거라 얕은 복사의 문제점이 드러나지 않는다

(4)얕은 복사3
# [1, 2,{'a': 'h'}] [1, 2,{'a': 'h'}]
# 이렇게 하면 원본도 바뀌기 때문에, deep copy 필요!


(5)깊은 복사
# [1, 2, ['a', 'b']] [1, 2, ['a', 'b']]
# [1, 2, ['a', 'b']] [1, 2, [0, 'b']]
```

9번. 재귀기초(복습)

- 다음의 코드를 print 한 결과를 작성하시오

```
(1)
def bbq(level):

  print(level)
  if level == 2:
```

```
    return

  bbq(level+1)
  print(level)

bbq(0)

-----------------------------------
(2)
def bbq(level):

    if level == 5:
        print(level)
        return

    print(level)
    bbq(level+1)
    print(level)

bbq(1)

-----------------------------------
(3)
def bbq(level):

    if level == 5:
        print(level)
        print(level)
        return

    print(level)
    bbq(level+1)
    print(level)

bbq(1)
```

▼ answer

```
(1)
# 0 1 2 1 0(세로로)

(2)
# 1 2 3 4 5 4 3 2 1(세로로)

(3)
# 1 2 3 4 5 5 4 3 2 1(세로로)
```

10번. 상속관련 메서드 - Super()

- name은 가져오고 싶고, age는 가져오기 싫을 때, Super()메서드를 사용할 수 있는가?

- 아래 코드를 살펴보고, 에러가 뜨는지 안 뜨는지 답하시오.

```
class Person:
    def __init__(self, name, age, number, email):
        self.name = name
        self.age = age
        self.number = number
        self.email = email

class Student(Person):
    def __init__(self, name, age, number, email, student_id):
        super().__init__(name, number, email)
        self.age = age
        self.student_id = student_id
```

▼ answer

- 답 : 탭에러가 뜬다 (TabError: inconsistent use of tabs and spaces in indentation)
  - 위 질문대로, 일부만(age빼고) 가져오면 하면 탭에러가 뜬다
  - super를 사용하실 경우 선택적으로 일부만 가져오긴 힘들고,
  - 우선 다 받아와서 덮어씌우는 형태로 작성해야 한다

str.ljust(8, '+')