AI전문가과정 - Python Review

Practice Session #3

Structed Types (2/2)

Packing/Unpacking : Basic

- "packing" a tuple : tuple object를 생성
- "unpacking" a tuple (multiple assignments) : extract the values back into variables

```
coordinates = (10,5) #packing
print('packed: ',coordinates)

x,y = coordinates # unpacking
print('unpacked: ',x,y)

x,y = y,x #packing & unpacking
print('packed and unpacked: ', x,y)
```

Run Code Visualize

- 함수들 중에는 여러 objects를 tuple로 packing하여 리턴하기도 합니다.(ex.divmod)

```
dividend, divisor = input('dividend & divisor:').split() # list unpacking
quotient, remainder = divmod(int(dividend),int(divisor)) # packing & unpacking
print(divmod(int(dividend),int(divisor)), quotient, remainder)
```

Run Code Visualize

Packing/Unpacking : Asterisk *

- *를 활용하면 여러 values를 하나의 변수에 list의 형태로 assign 할 수 있습니다.

```
fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")
(green, yellow, *red) = fruits
print(green,yellow,red)
```

Run Code Visualize

- *가 마지막 변수에 추가되지 않더라도 파이썬은 할당 해야할 남은 변수의 수와 values 수가 일치할 때까지 packing하여 해당 변수에 list로 할당합니다.

```
fruits = ("apple", "mango", "papaya", "pineapple", "cherry")
(green, *tropic, red) = fruits
print(green, tropic, red)
print(*tropic) #unpacking
```

Run Code Visualize

Packing/Unpacking : `enumerate()`

- `enumerate()`함수를 이용하면 **iterable object**를 입력으로 받아 **current iteration, value**로 unpacking 하여 반복문에서 활용할 수 있습니다.

```
my_str = 'hello'

for idx,char in enumerate(my_str):
    print(idx,'번째 문자:',char)
```

Run Code | Visualize

Packing/Unpacking : `zip()`

- `zip()` 함수를 활용하면 두 개 이상의 list들을 쉽게 결합하고 해체할 수 있습니다.
- `zip()` 함수는 여러 개의 순회가능한(iterable)객체를 인자로 받아 각 객체가 담고 있는 원소를 튜플의 형태로, 차례로 접근할 수 있는 반복자(iterator)를 반환합니다.

```
kor = ['안녕', '고마워', '잘자']
eng = ['Hello', 'Thanks', 'Good night']
frn = ['Bonjour', 'Merci', 'Bonne nuit']

for lang_tuple in zip(kor,eng,frn):
    print(lang_tuple)

for i in range(len(kor)):
    print((kor[i],eng[i],frn[i])) # same as above

pairs = list(zip(kor,eng,frn))
print(pairs)
kors, engs, frns = zip(*pairs) #unzip
print(kors, engs, frns)
```

Run Code | Visualize

Packing/Unpacking : zip()

- zip() 함수를 이용하면 두 개의 리스트나 튜플로부터 쉽게 dictionary를 만들 수 있습니다.

```
product = ['짜파게티','불닭볶음면','진라면','괄도네넴띤']
company = ['NS','SY','ODG','PD']
ramen_info = dict(zip(product,company))
print(ramen_info)

for key in ramen_info: # another method to obtain keys of dictionary
    print(key)

hot_items = ['짜파게티','불닭볶음면','참깨라면','진짬뽕']

# key check method: `not in`
for item in hot_items:
    print(item,'is','not'*(item not in ramen_info),'hot_ramen.')
```

Run Code | Visualize

nested list

- list의 원소로 list를 사용한 nested list 도 가능합니다.

```
matrix = [[0,1,2],[3,4,5],[6,7,8]]
print('nested list:',matrix)
print('*** 3 x 3 matrix ***')
for row in matrix:
    print(row)

print('*** diagonals ***')
for i in range(len(matrix)):
    print(matrix[i][i])

print('*** anti-diagonals ***')
for i,row in enumerate(matrix):
    print(row[-1-i])

print('*** Lower Triangular ***')
for i,row in enumerate(matrix):
    print(*row[:i+1])
```

Run Code Visualize

nested list

list comprehension

- list comprehension syntax를 활용하면 비교적 짧은 코드로 기존에 있던 `iterable` object로 부터 새로운 `list`를 생성할 수 있습니다. (아래 두 문법은 동일하게 작동합니다.)

```
new_list = [ expression for item in iterable if condition] # condition 생략 가능
```

```
new_list = []
for item in iterable:
  if condition:
    new_list.append(`expression`)
```

nested list

list comprehension

- `if-else`문도 함께 활용할 수 있습니다. (아래 두 문법은 동일하게 작동합니다.)

```
new_list = [ expression1 if condition else expression2 for item in iterable ]
```

```
new_list = []
for item in iterable:
  if condition:
    new_list.append(expression1)
  else:
    new_list.append(expression2)
```

```
print([input().lower() if i%2 else input().upper() for i in range(5)])
```

Run Code | Visualize

nested list

list comprehension

- `zip`, `enumerate` 등 과도 쓸 수 있습니다.

```
# 떠오르는 대로 형용사, 명사, 동사를 입력하여 재밌는 문장들을 만들어 봅시다:)

itr = 3
adjs = [input(str(i) + '번째 형용사 입력:') for i in range(itr)]
nouns = [input(str(i) + '번째 명사 입력:') for i in range(itr)]
verbs = [input(str(i) + '번째 동사 입력:') for i in range(itr)]
sentences = [ str(idx) + '. ' + adj +' '+ noun + '(이)가 ' + verb
              for idx, (adj, noun, verb) in enumerate(zip(adjs, nouns, verbs))]

for sentence in sentences:
    print(sentence)
```

Run Code | Visualize

nested list

nested list comprehension

- nested list comprehension도 가능합니다.

```
three_by_five = [[ (5*i+j)%10 for j in range(5)] for i in range(3) ]
for row in three_by_five:
    print(*row)
```

Run Code Visualize

- 지나친 중첩구조의 list comprehension 사용은 코드의 가독성을 떨어뜨릴 수 있습니다.

Mutable/imutable Types

mutable/imutable types

```
dccp_spring = {
    'COURSE_NAME' : 'Digital Computer Concept and Practice', # String
    'CREDIT_INFO' : (3,2,2), # Tuple : '총학점-이론-실습'
    'LECTURE_PLANS' : ['Introduction','Data Types, Input, Output',
                       'Dceision Structures and Repetition Structuress'], # list
    'ALTERNATIVE_SUBJECTS' : {'DCCP','Principles of Computer Science'} # set
} # dictionary

dccp_fall = dict(zip(dccp_spring.keys(),dccp_spring.values()))

print(dccp_spring == dccp_fall, dccp_spring is dccp_fall)

for key in dccp_spring:
    print(key, 'not same values' if dccp_spring[key] != dccp_fall[key] else 'same values')
    print(key, 'not same identity' if dccp_spring[key] is not dccp_fall[key] else 'same identity')
```

Run Code | Visualize

Mutable/imutable Types

mutable/imutable types

```
dccp_spring = {
    'COURSE_NAME' : 'Digital Computer Concept and Practice', # String
    'CREDIT_INFO' : (3,2,2), # Tuple : '총학점-이론-실습'
    'LECTURE_PLANS' : ['Introduction','Data Types, Input, Output',
                       'Dceision Structures and Repetition Structuress'], # list
    'ALTERNATIVE_SUBJECTS' : {'DCCP','Principles of Computer Science'} # set
} # dictionary

dccp_fall = dict(zip(dccp_spring.keys(),dccp_spring.values()))
dccp_fall['COURSE_NAME'] += ' Advanced' #dccp_fall['CREDIT_INFO'][0] += 1 #try!
dccp_fall['CREDIT_INFO'] = tuple([val+1 if idx==0 else val for idx,val in enumerate(dccp_fall['CREDIT_INFO'])])

print(dccp_spring == dccp_fall, dccp_spring is dccp_fall )

for key in ['COURSE_NAME','CREDIT_INFO']:
    print(key, 'not same values' if dccp_spring[key] != dccp_fall[key] else 'same values')
    print(key, 'not same identity' if dccp_spring[key] is not dccp_fall[key] else 'same identity')
```

Run Code | Visualize

Mutable/imutable Types

basic properties

Mutable/imutable Types

mutable vs immutable

- 목적에 맞는 타입의 자료형과 메소드를 사용하면 효율적으로 자원을 사용할 수 있습니다.

Mutable/imutable Types

mutable vs immutable

- 목적에 맞는 타입의 자료형과 메소드를 사용하면 효율적으로 자원을 사용할 수 있습니다.

```
# local 환경이나 다른 서버환경에서 확인해보세요:)

import sys

print(sys.getsizeof(tuple(range(10))))
print(sys.getsizeof(list(range(10))))
```

Mutable/imutable Types

mutable vs immutable

- 조작의 편의성 vs 변하지 않는 값에 대한 보호/보장
  - 길이, 값의 변화가 잦은 데이터
  - 읽기용 데이터, 하지만 크기가 매우 크다면 ex) deep learning model을 구성하는 수백만개의 tensor
  - 순서가 명확한 데이터 (좌표, 함수 arguments,..)
  - dictionary의 key 가 mutable이 가능하면...?
    - hashable type

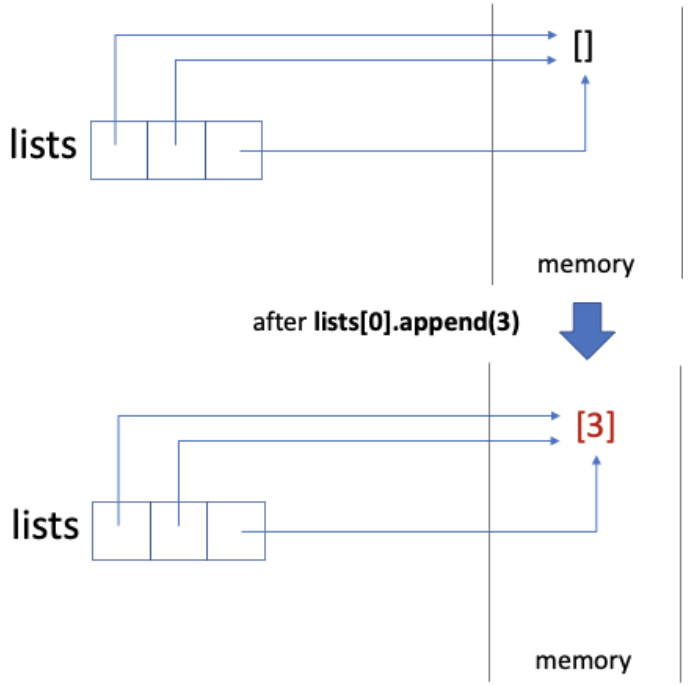
```
my_dict = {}
my_list = [1,2,3]

my_dict[my_list] = "Is it possible?"
```

Run Code | Visualize

Preview: Shallow Copy of a List

- `Repetition` returns a concatenated list of shallow copies of a list:

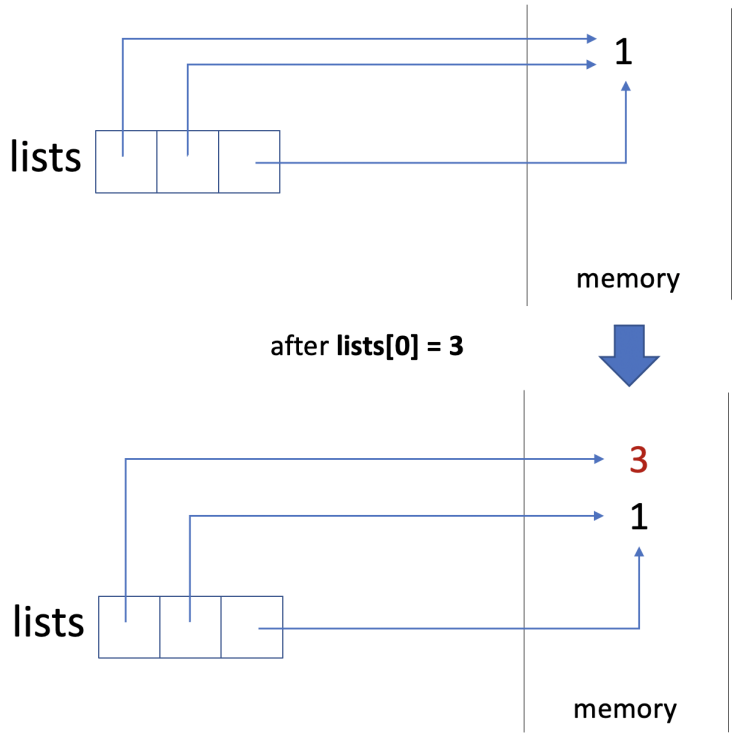
```
>>> lists = [[]] * 3
# 3 shallow copies of [[]] are concatenated!
>>> lists
[[], [], []]
>>> lists[0].append(3)
>>> lists
[[3], [3], [3]]
>>> lists[1].append(4)
>>> lists
[[3, 4], [3, 4], [3, 4]]
```



Preview: Shallow Copy of a List

- `Repetition` returns a shallow copy of the list:

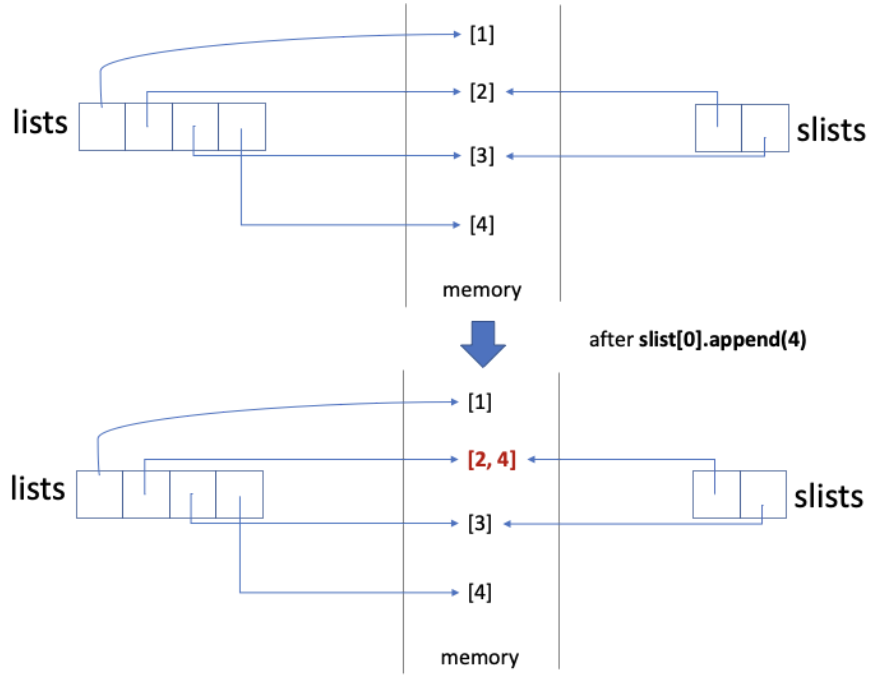
```
>>> lists = [1] * 3
>>> lists
[1, 1, 1]
>>> lists[0] = 3
>>> lists
[3, 1, 1]
>>> lists[1]=4
>>> lists
[3, 4, 1]
```



Preview: Shallow Copy of a List

- `Slicing` returns a shallow copy of the list:

```
>>> lists = [[1], [2], [3], [4]]
>>> slists = lists[1:3]
>>> slists
[[2], [3]]
>>> slists[0].append(4)
>>> slists
[[2, 4], [3]]
>>> lists
[[1], [2, 4], [3], [4]]
>>>
```



오늘의 실습 - 연습문제

문제 설명

2차원 행렬(`N*M`)의 크기를 결정하는 10 이하의 자연수 `N,M`과 **회전 횟수** `R`이 주어질 때, **정해진 규칙에 맞게 원본 행렬을 생성하여 시계 방향으로 90도씩 `R`번 회전한 결과를 출력하는 프로그램**을 작성해주세요.

- 원본 행렬을 생성하는 규칙은 주어진 `N,M`에 대해 n번째 row는 n의 양의 배수를 크기 순대로 `M`개 나열하되 각 값을 10으로 나눈 나머지입니다.(n=1,2,..N)

Input

- 첫 줄에 10 이하의 자연수 `N, M`으로 공백으로 구분하여 입력받습니다.
- 두 번째 줄에 자연수 `R`을 입력받습니다.

Output

- 원본 행렬의 **회전한 결과인 행렬**의(`X*Y` 크기라고 가정) **각 원소는 공백으로 구분하여 총 X줄로 출력**됩니다.

오늘의 실습 - 연습문제

입출력 예시

Input #1

```
2 4
3
```

Output #1

```
4 8
3 6
2 4
1 2
```

Input #2

```
4 7
1
```

Output #2

```
4 3 2 1
8 6 4 2
2 9 6 3
6 2 8 4
0 5 0 5
4 8 2 6
8 1 4 7
```

Input #3

```
3 3
4
```

Output #3

```
1 2 3
2 4 6
3 6 9
```