AI전문가과정 - Python Review

Practice Session #3

Structed Types (1/2)

Structured Data Type

다양한 자료형

- 프로그래밍 언어에서는 자료를 한 개 이상으로 담는 자료형이 존재합니다.
- Python에서는 기본적으로 많이 사용하는 자료형은 다음과 같습니다.
  - str : 문자열, `"hello world"`
  - list : 리스트, `[1, 2, 3]`
  - tuple : 튜플, `(1, 2, 3)`
  - dict : 딕셔너리, `{1:'one', 2:'two'}`
  - set : 셋(집합), `{1, 2, 3}`
- string을 제외하고는 Container라고도 부릅니다.

Structured Data Type

포인트!

- 왜 사용하는가?
- 어떻게 처음에 만드는가?
- 어떻게 원소에 접근하는가?
- 어떻게 원소를 추가하는가?
- 어떻게 원소를 삭제하는가?
- 어떠한 연산과 메서드를 지원하는가?

꼭 코드를 실행해보며 익히시기 바랍니다.

String

문자열과 인덱스

- 문자열은 기본 자료형이며, *문자의 연속*으로 이루어진 자료이기도 합니다.
- 문자열의 특정 원소를 사용하려면 **몇 번째 원소**인지 알아야 합니다.
  - 이를 문자열의 **인덱스(index)** 라고 합니다.
- 인덱스는 `0`부터 시작하며 `전체길이-1`이 마지막 인덱스입니다.

```
01234567890
hello world
```

String

- 문자열에서 특정 문자를 접근하기 위해서는 `[]`를 사용합니다.
  - 전체길이보다 큰 수가 들어가면 에러가 발생합니다.

```
print('hello world'[0])
print('hello world'[8])
```

Run Code Visualize

- 음수를 넣으면 역순으로 접근할 수 있습니다.

```
print('hello world'[-1])
print('hello world'[-10])
```

Run Code Visualize

String

문자열과 슬라이싱

- 문자열은 수정은 불가능합니다. 다만 문자열의 부분 문자열을 사용할 수는 있습니다.
- 슬라이싱(slicing)라고 하며, `[a:b]`의 형태로 사용합니다.
  - 인덱스 a이상 b 미만의 영역을 선택하며, a>=b인 경우 빈 리스트를 반환합니다.

```
s = "hello world"
print(s[1:])
print(s[:5])
print(s[4:9])
```

Run Code Visualize

- 음수 인덱스도 슬라이싱에 사용가능합니다.

```
s = "hello world"
print(s[-5:])
print(s[2:-2])
```

Run Code Visualize

String

문자열과 메서드

- Python은 각 자료형별로 다양한 기능을 메서드로 제공합니다.
- `find`는 부분문자열을 찾습니다.

```
s = '안녕하세요,TA입니다. '
print(s.find('안녕'))
```

Run Code Visualize

- `split` 은 문자열을 기준에 따라 문자열을 나눕니다.

```
s = 'hi,TA,bye,TA'
print(s.split(','))
print('1 2 3'.split())
```

Run Code Visualize

String

- `strip` 은 문자열 앞/뒤의 여백을 제거합니다.

```
s = ' 앞뒤에 여백이 있는 문자열 '
print('|'+s.strip()+'|')
print('|'+s.lstrip()+'|')
print('|'+s.rstrip()+'|')
```

Run Code Visualize

- 앞뒤의 특정 문자들을 지울수도 있습니다.

```
s = 'ah..컴개실..hu'
print(s.strip('ahu.'))
```

Run Code Visualize

List

리스트의 선언과 원소 접근

- 리스트는 자료를 순서대로 담는 자료형입니다.
- `[]`로 여러 자료를 담아서 만들 수 있습니다.
- 순서가 보장되는 자료형입니다.
- 문자열과 같이 index와 slicing을 사용할 수 있습니다.

```
lst = [0, 1, 2, 3, 'hello', 'world']
print(type(lst))
print(lst[0])
print(lst[-2:])
print(lst[2:-2])
```

Run Code | Visualize

List

리스트의 원소 수정

- 리스트의 원소의 공간은 하나하나가 변수처럼 작용합니다.
- =를 사용하여 리스트의 특정 원소를 바꿀 수 있습니다.

```
lst = [0, 1, 2, 3, 4, 5, 6]
lst[0] = 10
print(lst)
```

Run Code Visualize

List

리스트의 원소 추가

- 리스트에 원소를 추가하는 방법은 3가지입니다.
- `append`, `extend`, `+`연산자

```
lst = [0]
lst.append(1)
print(lst)
lst.extend([2, 3])
print(lst)
lst = lst + [4, 5]
print(lst)
```

Run Code Visualize

List

리스트의 원소 삭제

- 원소의 삭제는 pop을 사용하면 됩니다.

```
lst = ['one', 'two', 'three', 'four']
p1 = lst.pop()
print(lst, p1)
p2 = lst.pop(1)
print(lst, p2)
```

Run Code Visualize

Tuple

튜플의 선언과 원소 접근

- 튜플은 자료를 묶어서 사용하는 자료형입니다.
- `()`로 선언합니다.
- 리스트와 거의 같지만, 튜플은 원소를 바꿀 수 없습니다. (추가도 불가능)
- `[]`를 사용하여 접근가능하며, 슬라이싱도 가능합니다.

```
tp = (0, 1, 2, 3, 'hello')
print(type(tp))
print(tp[0])
print(tp[-1])
print(tp[1:])
```

Run Code | Visualize

Tuple

튜플의 연산

- + 연산과 * 연산을 제공합니다.

```
print((1, 2)+(2, 3))
print((1, 2)*3)
```

Run Code | Visualize

Dictionary

딕셔너리의 선언과 접근

- 딕셔너리는 사전 자료형입니다. `dict()`또는 `{}`로 선언합니다.
- 딕셔너리는 순서가 존재하긴 하나 순서로 사용하기 보다는 `key`를 인덱스처럼 사용하여 `value`를 가져옵니다.
- 접근을 할 때는 `[]`를 사용합니다.

```
dct = {1:'one', 2:'two', 3:'three', 'one':1}
print(type(dct))
print(dct[1])
print(dct['one'])
```

Run Code Visualize

Dictionary

- `get` 메서드를 사용하여 접근할 수도 있습니다.

```
dct = {1:'one', 2:'two', 3:'three'}
print(dct.get(1))
print(dct.get(4, 'NONE'))
```

Run Code Visualize

Dictionary

딕셔너리 원소 추가와 수정

- `[]`에 `=` 대입연산을 사용 수정 및 추가할 수 있습니다.

```
dct = {1:'one', 2:'two', 3:'three'}
dct[1] = '하나'
print(dct)
dct[4] = 'four'
print(dct)
```

Run Code Visualize

Dictionary

- `update`를 사용하면 여러 key-value를 업데이트할 수 있습니다.

```
dct = {1:'one', 2:'two', 3:'three'}
dct.update({4:'four', 5:'five'})
print(dct)
```

Run Code Visualize

Dictionary

딕셔너리 원소 삭제

- 원소의 삭제는 `pop(key)`를 하여 원소를 제거할 수 있습니다.

```
dct = {1:'one', 2:'two', 3:'three'}
var = dct.pop(1)
print(dct, var)
```

Run Code Visualize

Dictionary

딕셔너리의 key와 value

- `keys()`와 `values()`를 사용하여 따로 뽑아낼 수 있습니다.

```
dct = {1:'one', 2:'two', 3:'three'}
print(dct.keys())
print(dct.values())
```

Run Code | Visualize

Set

셋의 선언

- 셋(집합) 자료형은 집합을 표현할 수 있는 자료형입니다.
- dict와 마찬가지로 `{}`로 만들고, `set()`을 사용할 수도 있습니다.
- 수학에서 사용하는 집합의 특징을 가지고 있습니다.
  - 중복된 원소는 허용하지 않습니다.
  - 순서가 보장되지 않습니다.
- set은 특정 원소에 접근이 불가능합니다.

```
st = {1, 1, 2, 2, 3, 3, 3, 4}
print(st)
st2 = {3, 2, 100, 4}
print(st2)
print(type(st))
```

Run Code Visualize

Set

셋의 원소 추가

- `add`를 사용하여 원소를 하나 추가할 수 있습니다.

```
st = {1, 2, 3}
st.add(4)
print(st)
```

Run Code Visualize

- 여러 개를 동시에 추가하는 `update`가 있습니다.

```
st = {1, 2, 3}
st.update({4, 5, 6})
print(st)
```

Run Code Visualize

Set

셋의 원소 삭제

- `remove`를 사용하면 내부 원소를 제거할 수 있습니다.

```
st = {1, 2, 3}
st.remove(1)
print(st)
```

Run Code Visualize

- 유사하게 `discard`를 사용할 수 있습니다. 없는 원소를 삭제해도 에러가 발생하지 않습니다.

```
st = {1, 2, 3}
st.discard(4)
print(st)
```

Run Code Visualize

Set

셋의 연산

- 집합의 다양한 연산을 제공합니다.
- 다음은 합집합, 교집합, 차집합입니다.

```
st1 = {0, 1, 2, 3}
st2 = {2, 3, 4, 5}
print(st1|st2)
print(st1&st2)
print(st1-st2)
```

Run Code Visualize

Set

- 메서드를 사용하여 위의 연산을 진행할 수도 있습니다.

```
st1 = {0, 1, 2, 3}
st2 = {2, 3, 4, 5}
print(st1.union(st2))
print(st1.intersection(st2))
print(st1.difference(st2))
```

Run Code Visualize

자료형 공통

내장함수

- container 자료형에서는 대표적으로 4가지 내장함수를 사용할 수 있습니다.
  - `len` : 길이, `min` : 최솟값, `max` : 최댓값, `sum` : 합
- 단, `len()`을 제외하고는, 자료형 간의 비교가 가능해야 합니다.
- 대표로 `list`에 적용해보겠습니다.

```
lst = [1, 2, 3]
print(len(lst))
print(min(lst))
print(max(lst))
print(sum(lst))
```

Run Code Visualize

자료형 공통

in과 not in

- 원소의 유무를 확인하기 위해 `in`과 `not in` 연산자를 사용할 수 있습니다.

```
st = {'one', 'two', 'three'}
print('one' in st)
print('two' in st)
```

Run Code Visualize

자료형 공통

for문 사용하기

- for문을 사용하면 해당 자료형의 자료를 모두 살펴볼 수 있습니다.

```
s = 'hello'
for i in s:
    print(i)
```

Run Code Visualize

자료형 공통

```
lst = [1, 2, 3]
for i in lst:
    print(i)
```

Run Code Visualize

```
tp = ('q', 'w', 'e')
for i in tp:
    print(i)
```

Run Code Visualize

자료형 공통

- 딕셔너리는 key를 기준으로 반복하기에 값을 구하기 위해서는 테크닉이 필요합니다.

```
dct = {1:'one', 2:'two', 3:'three', 4:'four'}
for i in dct:
    print(i)
    print(dct[i])
```

Run Code Visualize

- 값만 따로 뽑는 법도 있습니다.

```
dct = {1:'one', 2:'two', 3:'three', 4:'four'}
for i in dct.values():
    print(i)
```

Run Code Visualize

자료형 공통

- 셋은 순서를 보장하지 않으니 주의합시다.

```
for i in {4, 3, 2, 100}:
    print(i)
```

Run Code Visualize

자료형 공통

	String	List	Tuple	Dictionary	Set
Component	a single character	object	object	object	object
Mutable	X	O	X	O	O
Ordered	O	O	O	O	X
Duplicated	O	O	O	동일 Key 불가	X
Constructor	'', "", """""", ''''''', str()	[] or list()	() or tuple()	{} or dict()	{} or set()

AI전문가과정 - Python Review

자료형 공통

연습문제 및 실습문제

hci lab
SEOUL NATIONAL UNIVERSITY

연습문제

첫 줄에 문자열 `S`을 입력받습니다. 두 번째 줄에 숫자 `N`를 입력 받습니다.

문자열에서 앞/뒤에서 각각 `N`개를 제거한 문장을 출력하시오.

입출력 예시

Input Example #1

```
AWESOME DCCP
3
```

Input Example #2

```
BTS!BTS!BTS!
4
```

Input Example #3

```
TAcarrotTA
2
```

Output Example #1

```
SOME D
```

Output Example #2

```
BTS!
```

Output Example #3

```
carrot
```

실습문제 #1

첫 줄에 자연수 N을 입력받습니다. 다음 N줄에 이름이 들어옵니다.

겹치지 않는 이름의 개수는 몇 개인지 구하시오. 이름의 대소문자가 다르면 다른 이름입니다.

입출력 예시

Input Example #1

```
2
Alice
Bob
```

Output Example #1

```
2
```

Input Example #2

```
4
Alice
Bob
Bob
bob
```

Output Example #2

```
3
```

Input Example #3

```
5
Alice
Alice
Alice
Alice
Alice
```

Output Example #3

```
1
```

실습문제 #2

구성된 암호문 s가 입력으로 들어옵니다. 암호문 s에서 특정 알파벳을 다음 문자열로 변경하려고 합니다.

original change

original	change
a	ta
b	buy
c	carrot
d	dccp

표에 없는 알파벳은 공백(' ')으로 변경합니다. 암호문을 해독해주세요!

입출력 예시

Input Example #1

```
a
```

Input Example #2

```
cxcxc
```

Input Example #3

```
dqawbecx
```

Output Example #1

```
ta
```

Output Example #2

```
carrot carrot carrot
```

Output Example #3

```
dccp ta buy carrot
```