파이썬 기초 개념 - 2

Kyungsik Han

본 영상에서 다룰 내용

CODING ✖

- 다른 언어 대비 파이썬의 특징
- 문자열 함수
- 파이썬 배열 자료구조
  - List
  - Tuple
  - Dictionary
  - List and dictionary comprehension
  - Set

C, Python, Java

CODING

C

```
#include “studio.h”
int main(){
    printf(“hello\n”);
}
```

Python

```
print(“hello”)
```

파이썬 에서는 ; 가 없음

Java

```
public class Hi {
    public static void main(String [] args){
        System.out.println(“hello”);
    }
}
```

C, Python, Java

C

```
#include “studio.h”
int main(){
    int x = 3;
    int y = 4;
    printf(“%s”\n, x+y)
}
```

Python

```
x = 3
y = 4
print(x+y)
```

파이썬 에서는 type이 없음

파이썬 데이터 타입

- **Numeric:** integers, float, complex
- **Sequence:** list, tuple, range
- **Binary:** byte, bytearray
- **True/False:** bool
- **Text:** string

문자열 함수

문자열 함수

CODING

```
>>> word = 'hello'
>>> word.lower()
```

```
hello
```

```
>>> word.upper()
```

```
HELLO
```

```
>>> '1' + '2'
```

```
'12'
```

```
>>> '12' * 2
```

```
'1212'
```

```
>>> 'Hi' + ' there'
```

```
'Hi there'
```

```
>>> '1'*2 + '2'*3
```

```
'11222'
```

문자열 함수


```
>>> s = '  Extras \n'
>>> s.strip()
```

```
'Extras'
```

```
>>> s = '***10***'
>>> s.strip('*')
```

```
'10'
```

```
>>> s = 'Let\'s split the words'
>>> s.split(' ')
```

```
["Let's", 'split', 'the', 'words']
```

```
>>> s = 'Jane,Doe,Cars,5'
>>> s.split(',')
```

```
['Jane', 'Doe', 'Cars', '5']
```

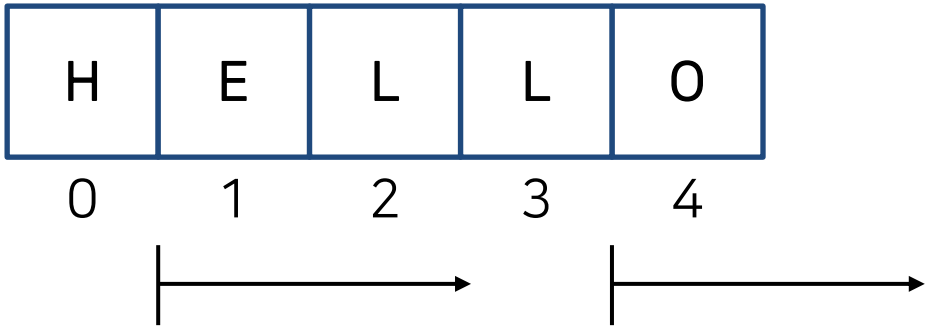
문자열 함수 slicing

```
>>> word = 'Hello'
>>> word[1:3]
```

'el'

```
>>> word[4:7]
```

'o'



문자열 함수 slicing

```
>>> word = 'Hello'
>>> word[1:3]
```

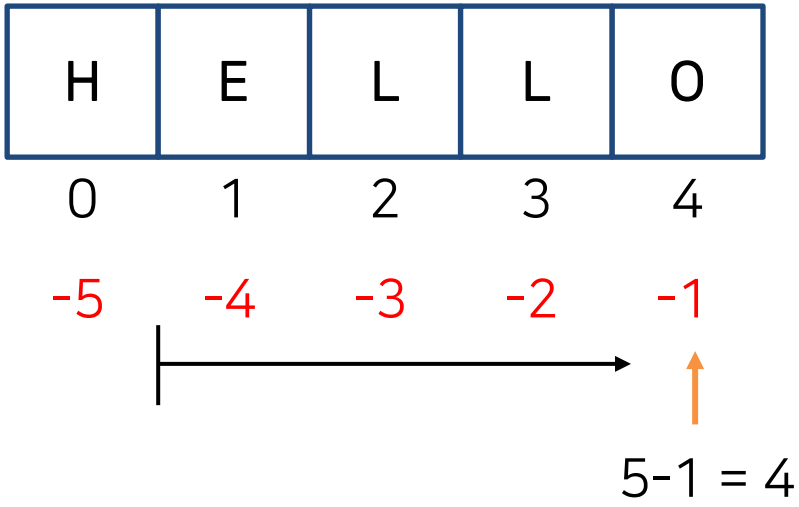
'el'

```
>>> word[4:7]
```

'o'

```
>>> word[-4:-1]
```

'ell'



문자열 함수

CODING

```
>>> word = 'Hello'
>>> 'HE' in word
```

```
False
```

```
>>> 'He' in word
```

```
True
```

```
>>> word.find('el')
```

```
1
```

```
>>> word = '1234'
>>> int(word)
```

```
1234
```

```
>>> float(word)
```

```
1234.0
```

```
>>> word = 'Hi'
>>> int(word)
```

<Error>

문자열 함수

CODING

```
>>> statement = 'We love {} {}.'
>>> statement.format('data','analysis')
```

'We love data analysis.'

```
>>> statement = 'We love {0} {1}.'
>>> statement.format('data','analysis')
```

'We love data analysis.'

```
>>> statement = 'We love {1} {0}.'
>>> statement.format('analysis','data')
```

'We love data analysis.'

문자열

CODING

문자열에 특정 문자가 있는지 확인하기

```
msg = 'My name is 홍길동'
if 'is' in msg:
    print('msg에는 is가 있습니다')
else:
    print('msg에는 is가 없습니다')
```

문자열에서 좌우 공백 제거하기

```
txt = ' 양쪽에 공백이 있는 문자열입니다. '
ret1 = txt.lstrip()
ret2 = txt.rstrip()
ret3 = txt.strip()
print('<' + ret1 + '>')
print('<' + ret2 + '>')
print('<' + ret3 + '>')
```

<양쪽에 공백이 있는 문자열입니다 >
< 양쪽에 공백이 있는 문자열입니다>
<양쪽에 공백이 있는 문자열입니다>

리스트 (List)

리스트 basics

CODING

```
>>> list = [11,22,33]
>>> list
```

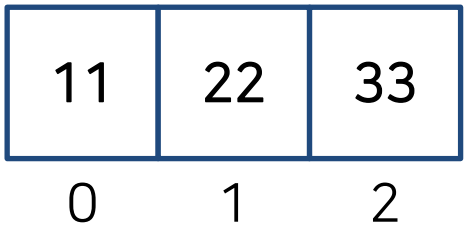
```
[11, 22, 33]
```

```
>>> list[1]
```

```
22
```

```
>>> list[3]
```

Error-index out of range



리스트 iteration


```
>>> list = [11,22,33]
>>> for i in list:
        print(i)
```

```
11
22
33
```

```
>>> for i in range(0,len(list)):
        print(list[i])
```

```
11
22
33
```

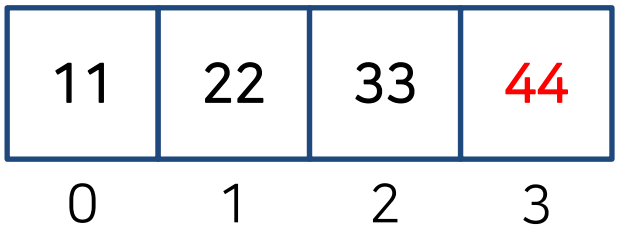
리스트 append


```
>>> list = [11,22,33]
```

```
>>> list.append(44)
```

```
>>> list
```

```
[11, 22, 33, 44]
```



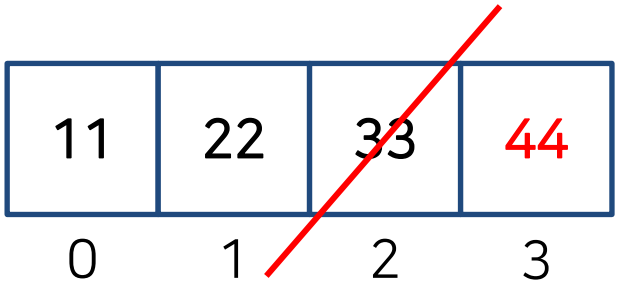
리스트 pop


```
>>> list = [11,22,33,44]
>>> list.pop(2)
```

```
33
```

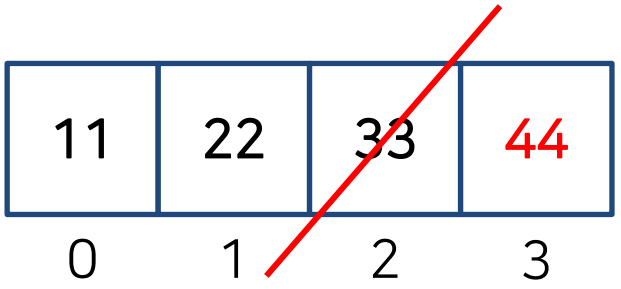
```
>>> list
```

```
[11, 22, 44]
```



리스트 remove

```
>>> list = [11,22,33,44]
>>> list.remove(33)
```




```
>>> list
```

```
[11, 22, 44]
```

리스트 extend, append

CODING

```
>>> list = [1,2,3]
>>> list2 = [4,5,6]
```

```
>>> list.extend(list2)
>>> list
```

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

```
>>> list = [1,2,3]
>>> list2 = [4,5,6]
>>> list.append(list2)
>>> list
```

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

리스트 zip


```
>>> list = [1,2,3]
>>> list2 = [4,5,6]
>>> for x, y in zip(list, list2):
        print(x, ", ", y)
```

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

시퀀스 자료 인덱싱 이해하기 – (1)

CODING

Sequence 자료 indexing

```
>>> strdata = 'Time is money!!'
>>> listdata = [1,2,[1,2,3]]
>>> print(strdata[5])
i
>>> print(strdata[-2])
!
>>> print(listdata[0])
1
>>> print(listdata[-1])
[1,2,3]
>>> print(listdata[2][-1])
3
```

T	i	m	e		i	s		m	o	n	e	y	!	!
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

1	2	[1,2,3]
0	1	2
-3	-2	-1

시퀀스 자료 인덱싱 이해하기 – (2)

CODING

Sequence 자료 indexing

```
>>> strdata = 'Time is money!!'
>>> print(strdata[1:5])
ime
>>> print(strdata[:7])
Time is
>>> print(strdata[9:])
oney!!
>>> print(strdata[:-3])
Time is mone
>>> print(strdata[-3:])
y!!
>>> print(strdata[:])
Time is money!!
>>> print(strdata[::2])
Tm smny!
```

T	i	m	e		i	s		m	o	n	e	y	!	!
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

튜플 (Tuple)

튜플 (Tuple)


```
>>> tuple1 = ('Honda','Civic',4,2017)
>>> tuple1
```

```
('Honda', 'Civic', 4, 2017)
```

```
>>> tuple1[1]
```

```
'Civic'
```

```
>>> len(tuple1)
```

```
4
```

튜플 (Tuple)


```
>>> tuple1 = ('Honda','Civic',4,2017)
>>> for i in tuple1:
…         print(i)
```

```
Honda
Civic
4
2017
```

튜플 (Tuple)

튜플 값은 변경이 불가

```
>>> tuple1 = ('Honda','Civic',4,2017)
>>> tuple1[3]=2018
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

딕셔너리 (Dictionary)

딕셔너리 (Dictionary)

CODING

Key	Value
'A123'	'홍길동'
'A241'	'김철수'
'A392'	'박영희'

Key	Value
'C123'	['홍길동','김철수','박영희']
'C241'	['이홍이','길동이','한사람']

Key	Value
('Ghostbuster', 2016)	5.4
('Ghostbuster', 1984)	7.8
('Cars', 2006)	7.1

딕셔너리 (Dictionary) - add value

CODING

```
>>> dict = {('Ghostbusters', 2016):5.4,
            ('Ghostbusters', 1984):7.8}
>>> dict[('Cars', 2006)] = 7.1
```

```
>>> dict
```

{('Ghostbusters', 2016):5.4,
('Ghostbusters', 1984):7.8,
('Cars', 2006):7.1}

딕셔너리 (Dictionary) - basic

CODING

```
>>> dict = {('Ghostbusters', 2016):5.4,
            ('Ghostbusters', 1984):7.8}
```

```
>>> dict[('Ghostbusters', 2016)]
```

5.4

```
>>> len(dict)
```

2

딕셔너리 (Dictionary) – get value

CODING

```
>>> dict = {('Ghostbusters', 2016):5.4,
            ('Ghostbusters', 1984):7.8,
            ('Cars', 2006):7.1}
>>> x = dict[('Cars', 2006)]
>>> x
```

7.1

```
>>> x = dict.get(('Cars', 2006))
>>> x
```

7.1

딕셔너리 (Dictionary) – delete value

CODING

```
>>> dict = {('Ghostbusters', 2016):5.4,
            ('Ghostbusters', 1984):7.8,
            ('Cars', 2006):7.1}
>>> dict.pop(('Ghostbusters', 2016))
>>> dict
```

{('Ghostbusters', 1984):7.8,
('Cars', 2006):7.1}

```
>>> del dict[('Cars',2006)]
```

{('Ghostbusters', 1984):7.8}

딕셔너리 (Dictionary) – iteration (1)

CODING

```
>>> dict = {('Ghostbusters', 2016):5.4,
            ('Ghostbusters', 1984):7.8,
            ('Cars', 2006):7.1}
>>> for i in dict:
        print(i)
```

('Ghostbusters', 2016)
('Ghostbusters', 1984)
('Cars', 2006)

딕셔너리 (Dictionary) – iteration (1)


```
>>> dict = {('Ghostbusters', 2016):5.4,
            ('Ghostbusters', 1984):7.8,
            ('Cars', 2006):7.1}
>>> for i in dict:
        print(i[0], i[1])
```

'Ghostbusters', 2016
'Ghostbusters', 1984
'Cars', 2006

딕셔너리 (Dictionary) – iteration (2)

CODING

```
>>> dict = {('Ghostbusters', 2016):5.4,
            ('Ghostbusters', 1984):7.8,
            ('Cars', 2006):7.1}
>>> for key, value in dict.items():
        print(key, ":", value)
```

('Ghostbusters', 2016):5.4
('Ghostbusters', 1984):7.8
('Cars', 2006):2.1

List and Dictionary Comprehension

List comprehension

CODING

[출력표현식 **for** 요소 **in** 입력sequence [**if** 조건식]]

```
>>> list = [i**2 for i in range(1,11)]
>>> list
```

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

List comprehension


```
>>> list = [_____________________________________]
>>> list
```

[0, 1, 2, 3, 4, 5]

```
>>> list = [i for i in range(0,6)]
```

List comprehension


```
>>> list = [_____________________________________]
>>> list
```

[0, 1, 0, 1, 0, 1, 0, 1, 0, 1]

```
>>> list = [i%2 for i in range(0,10)]
```

List comprehension


```
>>> import random
>>> list = [random.randint(0,5) for i in
            range(0,10)]
>>> list
```

[4, 3, 4, 2, 1, 5, 0, 1, 2, 3]

Dictionary comprehension


```
>>> dict = {i: i**2 for i in range(1,11)}
>>> dict
```

{1:1, 2:4, 3:9, 4:16, 5:25, 6:36, 7:49, 8:64, 9:81, 10:100}

Dictionary comprehension


```
>>> dict = {i: chr(i) for i in range(65,91)}
>>> dict
```

{65: 'A', 66: 'B', 67: 'C', 68: 'D', 69: 'E', 70: 'F', 71: 'G', 72: 'H', 73: 'I', 74: 'J', 75: 'K', 76: 'L', 77: 'M', 78: 'N', 79: 'O', 80: 'P', 81: 'Q', 82: 'R', 83: 'S', 84: 'T', 85: 'U', 86: 'V', 87: 'W', 88: 'X', 89: 'Y', 90: 'Z'}

Sets

Sets

- Unordered
- Unique (no duplicates)
- Support set operations (e.g., union, intersecti
  on)

Set Basics – add


```
>>> colors = set(['blue', 'green', 'red'])
>>> colors
```

{'blue', 'green', 'red'}

```
>>> colors.add('yellow')
>>> colors
```

{'blue', 'green', 'red', 'yellow'}

```
>>> colors.add('blue')
>>> colors
```

{'blue', 'green', 'red', 'yellow'}

Set Basics – discard


```
>>> colors = set(['blue', 'green', 'red'])
>>> colors
```

{'blue', 'green', 'red'}

```
>>> colors.discard('green')
>>> colors
```

{'blue', 'red'}

Set Basics – union & intersection (1)

CODING

```
>>> colors_1 = set(['blue', 'green', 'red'])
>>> colors_2 = set(['blue', 'yellow'])
>>> either = colors_1.union(colors_2)
>>> either
```

{'blue', 'green', 'red', 'yellow'}

```
>>> colors_1 = set(['blue', 'green', 'red'])
>>> colors_2 = set(['blue', 'yellow'])
>>> both = colors_1.intersection(colors_2)
>>> both
```

{'blue'}

Set Basics – union & intersection (2)

CODING

```
>>> colors_1 = set(['blue', 'green', 'red'])
>>> colors_2 = set(['blue', 'yellow'])
>>> either = colors_1 | colors_2
>>> either
```

{'blue', 'green', 'red', 'yellow'}

```
>>> colors_1 = set(['blue', 'green', 'red'])
>>> colors_2 = set(['blue', 'yellow'])
>>> both = colors_1 & colors_2
>>> both
```

{'blue'}

다음 영상에서 배울 내용

CODING

- 파이썬 기초개념 3
  - Numpy

수고하셨습니다