인공지능 기초 프로그래밍

Python 기초 (자료형2)

Daegu Campus of
KOREA POLYTECHNIC

List, Tuple, Set, etc.

파이썬 프로그래밍의 기초, 자료형 II

- 리스트 자료형

  - **리스트(List)란?**

    - 자료형의 집합을 표현할 수 있는 자료형

```
>>> odd = [1, 3, 5, 7, 9]
```

    - 숫자와 문자열만으로 프로그래밍을 하기엔 부족한 점이 많음
      - 예) 1부터 10까지의 숫자 중 홀수 모음인 집합 {1, 3, 5, 7, 9}는 숫자나 문자열로 표현 불가능
      - 리스트로 해결 가능!

파이썬 프로그래밍의 기초, 자료형 II

- 리스트 자료형

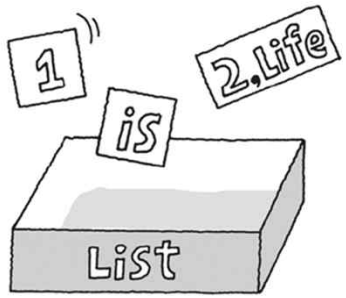
▪ 리스트 사용법

- 대괄호([ ])로 감싸고 각 요솟값은 쉼표(,)로 구분

```
리스트명 = [요소1, 요소2, 요소3, …]
```

- 리스트 안에 어떠한 자료형도 포함 가능

```
>>> a = []
>>> b = [1, 2, 3]
>>> c = ['Life', 'is', 'too', 'short']
>>> d = [1, 2, 'Life', 'is']
>>> e = [1, 2, ['Life', 'is']]
```



파이썬 프로그래밍의 기초, 자료형 II

- 리스트 자료형
  - **리스트 인덱싱**
    - 문자열과 같이 인덱싱 적용 가능

```
>>> a = [1, 2, 3]
>>> a
[1, 2, 3]
```

    - 파이썬은 숫자를 0부터 세기 때문에 a[0]이 리스트 a의 첫 번째 요소

```
>>> a[0]
1
```

    - a[-1]은 리스트 a의 마지막 요솟값

```
>>> a[-1]
3
```

    - 요솟값 간의 덧셈

```
>>> a[0] + a[2]  ◀── 1 + 3
4
```

파이썬 프로그래밍의 기초, 자료형 II

- 리스트 자료형

▪ 리스트 인덱싱

▪ 리스트 내에 리스트가 있는 경우

```
>>> a = [1, 2, 3, ['a', 'b', 'c']]
```

▪ a[-1]은 마지막 요솟값인 리스트 ['a', 'b', 'c'] 반환

```
>>> a[0]
1
>>> a[-1]
['a', 'b', 'c']
```

▪ 리스트 a에 포함된 ['a', 'b', 'c'] 리스트에서 'a' 값을 인덱싱을 사용해 반환할 방법은?

```
>>> a[-1][0]
'a'
```

▪ a[-1]로 리스트 ['a', 'b', 'c']에 접근하고, [0]으로 요소 'a"에 접근

파이썬 프로그래밍의 기초, 자료형 II

- 리스트 자료형
  - **리스트 슬라이싱**
    - 문자열과 같이 슬라이싱 적용 가능

```
>>> a = [1, 2, 3, 4, 5]
>>> a[0:2]
[1, 2]
```

```
>>> a = [1, 2, 3, 4, 5]
>>> b = a[:2] ◀── 처음부터 a[1]까지
>>> c = a[2:] ◀── a[2]부터 마지막까지
>>> b
[1, 2]
>>> c
[3, 4, 5]
```

파이썬 프로그래밍의 기초, 자료형 II

- 리스트 자료형
  - **리스트 연산하기**
    - **더하기(+)**
      - + 기호는 2개의 리스트를 합치는 기능
      - 문자열에서 "abc" + "def" = "abcdef"가 되는 것과 같은 의미

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> a + b
[1, 2, 3, 4, 5, 6]
```

    - **반복하기(*)**
      - * 기호는 리스트의 반복을 의미
      - 문자열에서 "abc" * 3 = "abcabcabc"가 되는 것과 같은 의미

```
>>> a = [1, 2, 3]
>>> a * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

    - **리스트 길이 구하기**
      - len( ) 함수 사용
      - 문자열, 리스트 외에 앞으로 배울 튜플과 딕셔너리에서도 사용 가능한 내장 함수

```
>>> a = [1, 2, 3]
>>> len(a)
3
```

자료형 II - Practice #1 (list)

```
# list
list1 = [1, 2, 3, 0.1]
print(f'{list1}')
list2 = [30, 3, ["study", "bed", "game"], "1층", "3억"]
print(list2)
list3 = []
list3.append("1")
list3.append(32)
print(list3, type(list3[0]), type(list3[1]))

# list indexing/slicing
print(f'Start index of the List=%d, first data of a list3 = {list3[0]}' % 0)
print(list2[-3])
print("1+2=%d" % (list1[0] + list1[1]))
str2 = f'My house area is {list2[0]}, there are {len(list2[-3])} rooms and \n' \
       f'I like the {list2[-3][2]} room!\n' \
       f'{list2[-2:]} are shown in web site!'
print(str2)
list1 = [1, 2, 3]
list2 = [4, 5, 6]
list3 = list1 + list2
print(f'list 3 length is {len(list3)}: {list3}')
```

파이썬 프로그래밍의 기초, 자료형 II

- 리스트 수정/삭제, 리스트 관련 함수
  - **리스트의 수정과 삭제**
    - **리스트에서 값 수정하기**

```
>>> a = [1, 2, 3]
>>> a[2] = 4
>>> a
[1, 2, 4]
```

    - **리스트 요소 삭제하기**
      - del 키워드 사용

del 객체

```
>>> a = [1, 2, 3]
>>> del a[1]
>>> a
[1, 3]
```

```
>>> a = [1, 2, 3, 4, 5]
>>> del a[2:]
>>> a
[1, 2]
```

※ 슬라이싱 기법 활용 가능

파이썬 프로그래밍의 기초, 자료형 II

- 리스트 수정/삭제, 리스트 관련 함수
  - **리스트 관련 함수**
    - **append( )**
      - 리스트의 맨 마지막에 요소 추가

```
>>> a = [1, 2, 3]
>>> a.append(4) ◀── 리스트의 맨 마지막에 4를 추가
>>> a
[1, 2, 3, 4]
```

      - 어떤 자료형도 추가 가능

```
>>> a.append([5,6]) ◀── 리스트의 맨 마지막에 [5,6]을 추가
>>> a
[1, 2, 3, 4, [5, 6]]
```

    - **sort( )**
      - 리스트의 요소를 순서대로 정렬

```
>>> a = [1, 4, 3, 2]
>>> a.sort()
>>> a
[1, 2, 3, 4]
```

      - 문자의 경우 알파벳 순서로 정렬 가능

파이썬 프로그래밍의 기초, 자료형 II

- 리스트 수정/삭제, 리스트 관련 함수
  - **리스트 관련 함수**
    - **reverse( )**
      - 리스트를 역순으로 뒤집어 줌
      - 요소를 역순으로 정렬하는 것이 아닌, 현재의 리스트 그대로 뒤집음

```
>>> a = ['a', 'c', 'b']
>>> a.reverse()
>>> a
['b', 'c', 'a']
```

    - **index( )**
      - 요소를 검색하여 위치 값 반환

```
>>> a = [1,2,3]
>>> a.index(3)  ◀── 3은 리스트 a의 세 번째(a[2]) 요소
2
```

      - 값이 존재하지 않으면, 값 오류 발생

```
>>> a.index(0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>   ── 오류 메시지
ValueError: 0 is not in list
```

파이썬 프로그래밍의 기초, 자료형 II

- 리스트 수정/삭제, 리스트 관련 함수

▪ 리스트 관련 함수

▪ insert( )

- 리스트에 요소 삽입
- insert(a, b)
  - a번째 위치에 b를 삽입하는 함수

```
>>> a = [1, 2, 3]
>>> a.insert(0, 4) ◀── a[0] 위치에 4 삽입
[4, 1, 2, 3]
```

▪ remove( )

- remove(x)
  - 리스트에서 첫 번째로 나오는 x를 삭제

```
>>> a = [1, 2, 3, 1, 2, 3]
>>> a.remove(3)
[1, 2, 1, 2, 3]
```

- 값이 여러 개인 경우 첫 번째 것만 삭제

```
>>> a.remove(3)
[1, 2, 1, 2]
```

파이썬 프로그래밍의 기초, 자료형 II

- 리스트 수정/삭제, 리스트 관련 함수

▪ 리스트 관련 함수

- **pop( )**
  - 리스트의 맨 마지막 요소를 돌려주고 해당 요소 삭제
  - pop(x)
    - 리스트의 x번째 요소를 돌려주고 해당 요소 삭제

```
>>> a = [1, 2, 3]
>>> a.pop()
3
>>> a
[1, 2]
```

```
>>> a = [1, 2, 3]
>>> a.pop(1)
2
>>> a
[1, 3]
```

- **count( )**
  - 리스트에 포함된 요소의 개수 반환
  - count(x)
    - 리스트 안에 x가 몇 개 있는지 조사하여 그 개수를 돌려주는 함수

```
>>> a = [1, 2, 3, 1]
>>> a.count(1)
2
```


파이썬 프로그래밍의 기초, 자료형 II

- 리스트 수정/삭제, 리스트 관련 함수

▪ 리스트 관련 함수

- **extend( )**
  - 리스트에 리스트를 더하는 함수
  - extend(x)
    - x에는 리스트만 올 수 있음

a.extend([4, 5])

=

a += [4, 5]

```
>>> a = [1, 2, 3]
>>> a.extend([4, 5])
>>> a
[1, 2, 3, 4, 5]
>>> b = [6, 7]
>>> a.extend(b)
>>> a
[1, 2, 3, 4, 5, 6, 7]
```

자료형 II - Practice #2 (list)

```
# list add/delete
list_a = [1, "2", 4]
print(f'{list_a}')
list_a[1] = 4
print(f'{list_a}')
list_a = list_a + [5, 6]
print(list_a)
del list_a[2]
print(list_a)

str_1 = "123"
print(str_1[0], str_1[1], str_1[2])
# str_1[2] = "3"
# str_1 = str_1[:2]+"4"
# print(str_1)

# list append / sort
test_list = [1, 2, 3]
test_list.append([4, 8, 6, 7, 9])
test_list.append("string")
```

파이썬 프로그래밍의 기초, 자료형 II

- 튜플 자료형

▪ 튜플(Tuple)이란?

▪ 리스트와 유사한 자료형

리스트	튜플
[ ]로 둘러쌈	( )로 둘러쌈
생성 / 삭제 / 수정 가능	값 변경 불가능

```
>>> t1 = ()
>>> t2 = (1,)
>>> t3 = (1, 2, 3)
>>> t4 = 1, 2, 3
>>> t5 = ('a', 'b', ('ab', 'cd'))
```

- 튜플은 1개의 요소만을 가질 때는 요소 뒤에 콤마(,)를 만드시 붙여야 함 (예) t2 = (1,)
- 괄호( )를 생략해도 무방함 (예) t4 = 1, 2, 3
- 프로그램이 실행되는 동안 값을 유지해야 한다면 튜플을, 수시로 값을 변경해야 하면 리스트 사용

파이썬 프로그래밍의 기초, 자료형 II

- 튜플 자료형

▪ 튜플의 요솟값을 지울 수 있을까?

- 튜플의 요솟값은 한 번 정하면 지우거나 변경할 수 없음!

```
>>> t1 = (1, 2, 'a', 'b')
>>> del t1[0] ◀── 튜플 t1의 첫 번째 요소를 지우려고 시도
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object doesn't support item deletion
```

형 오류(Type Error) 발생

```
>>> t1 = (1, 2, 'a', 'b')
>>> t1[0] = 'c' ◀── 튜플 t1의 첫 번째 요솟값을 변경하려고 시도
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

형 오류 발생

파이썬 프로그래밍의 기초, 자료형 II

- 튜플 자료형

▪ 튜플 다루기

▪ 인덱싱하기

```
>>> t1 = (1, 2, 'a', 'b')
>>> t1[0]
1
>>> t1[3]
'b'
```

▪ 슬라이싱하기

```
>>> t1 = (1, 2, 'a', 'b')
>>> t1[1:] ◀── t1[1]부터 끝까지
(2, 'a', 'b')
```

▪ 튜플 더하기와 곱하기

```
>>> t2 = (3, 4)
>>> t1 + t2
(1, 2, 'a', 'b', 3, 4)
```

```
>>> t2 * 3
(3, 4, 3, 4, 3, 4)
```

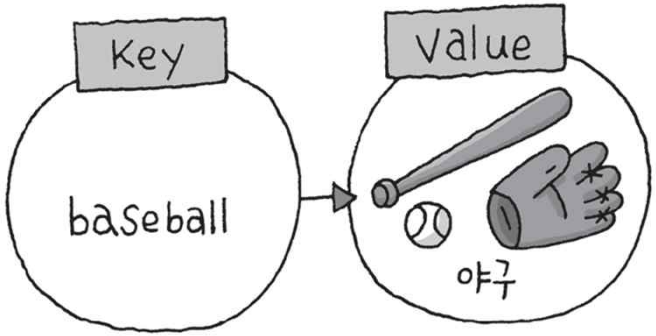
▪ 튜플 길이 구하기

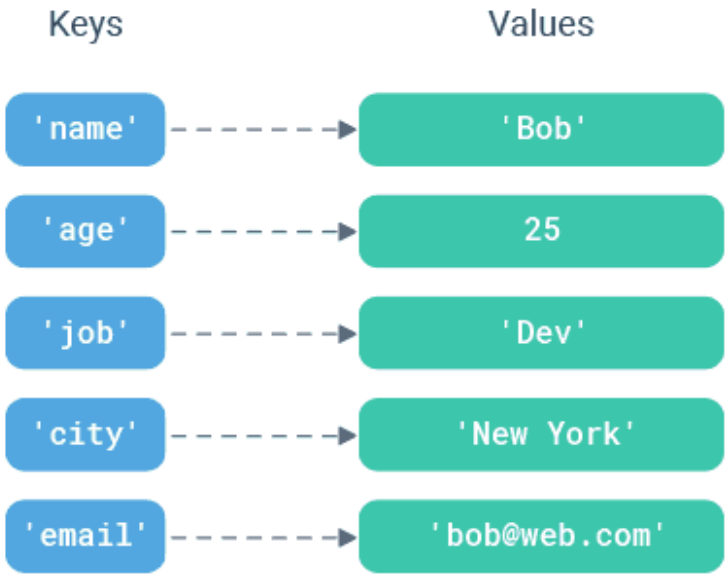
- len( ) 함수

```
>>> t1 = (1, 2, 'a', 'b')
>>> len(t1)
4
```

파이썬 프로그래밍의 기초, 자료형 III

- 딕셔너리(Dictionary)란?
  - 대응 관계를 나타내는 자료형
  - 연관 배열(Associative aray) 또는 해시(Hash)
  - Key와 Value를 한 쌍으로 갖는 자료형
  - 순차적으로 해당 요솟값을 구하지 않고, Key를 통해 Value를 바로 얻는 특징





파이썬 프로그래밍의 기초, 자료형 III

- 딕셔너리(Dictionary)

  - **딕셔너리의 모습**

```
{Key1:Value1, Key2:Value2, Key3:Value3, ···}
```

  - Key와 Value의 쌍 여러 개 (Key : Value)

  - { }로 둘러싸임
  - 각 요소는 쉼표(,)로 구분됨

```
>>> dic = {'name':'pey', 'phone':'0119993323', 'birth': '1118'}
```

```
>>> a = {1: 'hi'}
```

```
>>> a = {'a': [1,2,3]}
```

파이썬 프로그래밍의 기초, 자료형 III

- 딕셔너리(Dictionary)

▪ 딕셔너리 쌍 추가, 삭제하기

▪ 딕셔너리 쌍 추가

```
>>> a = {1: 'a'}
>>> a[2] = 'b'  ◀── {2:'b'} 쌍 추가
>>> a
{1: 'a', 2: 'b'}
```

▪ 딕셔너리 요소 삭제

```
>>> del a[1]  ◀── key가 1인 key:value 쌍 삭제
>>> a
{'name': 'pey', 3: [1, 2, 3], 2: 'b'}
```

▪ 딕셔너리에서 Key 사용해 Value 얻기

```
>>> grade = {'pey': 10, 'julliet': 99}
>>> grade['pey']  ◀── Key가 'pey'인 딕셔너리의 Value를 반환
10
>>> grade['julliet']  ◀── Key가 'julliet'인 딕셔너리의 Value를 반환
99
```

- 리스트나 튜플, 문자열은 요솟값 접근 시 **인덱싱**이나 **슬라이싱** 기법을 사용

- 딕셔너리는 **Key**를 사용해 **Value** 접근

주의 사항:
- 동일한 key를 사용할 경우, 1개를 제외한 나머지 key:value 값은 모두 무시
- Key에는 변할 수 있는 자료형 사용하면 Error 발생

Practice #1 (Dictionary)

```python
# Practice 1
a = {1: 'a', 2: 'b'}
a[3] = 'c'
print(a)
a['name'] = ["Shin", "Park", "Kim"]
print(a)
del a[1] # Dictionary a에서 key가 1인 key:value 쌍을 삭제한다.
print(a)
a = {"Dept": ["AI-Engineering", "Smart Electronic"], "StudentNum": [22, 60]}
print(a['Dept'], a['StudentNum'])
a = {'1': 'a', '2': 'b', '3': 'c'}
# Key를 중복해서 사용하면 하나만 남기고 나머지는 무시
a['1'] = 'aa'
a['1'] = 'aaa'
print(a)

# Error Case
a = {
#     ['name', 'age']: (["Shin", "Kim", "Park"], [22, 23, 25])
    ('name', 'age'): (["Shin", "Kim", "Park"], [22, 23, 25])
}
print(a)
```

파이썬 프로그래밍의 기초, 자료형 III

- 집합자료형(Set)

  - **집합(Set)이란?**

    - 집합에 관련된 것을 쉽게 처리하기 위해 만든 자료형
    - 파이썬 2.3부터 지원

    - set 키워드를 사용하여 생성
      - set( )에 리스트를 입력하여 생성

```
>>> s1 = set([1,2,3])
>>> s1
{1, 2, 3}
```

set( )에 문자열을 입력하여 생성

```
>>> s2 = set("Hello")
>>> s2
{'e', 'H', 'l', 'o'}
```

파이썬 프로그래밍의 기초, 자료형 III

- 집합자료형(Set)

▪ 집합의 특징

1) 중복을 허용하지 않음
2) 순서가 없음(Unordered)

```
>>> s2 = set("Hello")
>>> s2
{'e', 'H', 'l', 'o'}
```

- 리스트나 튜플은 순서가 있기 때문에 인덱싱을 통해 자료형의 값을 얻지만 set 자료형은 순서가 없기 때문에 인덱싱 사용 불가 (딕셔너리와 유사)

- 인덱싱 사용을 원할 경우 리스트나 튜플로 변환 필요
  - list( ), tuple( ) 함수 사용

```
>>> t1 = tuple(s1) ◀── 튜플로 변환
>>> t1
(1, 2, 3)
>>> t1[0]
1
```

```
>>> s1 = set([1,2,3])
>>> l1 = list(s1) ◀── 리스트로 변환
>>> l1
[1, 2, 3]
>>> l1[0]
1
```

파이썬 프로그래밍의 기초, 자료형 III

- 집합자료형(Set)

▪ 교집합, 합집합, 차집합 구하기

▪ set 자료형을 유용하게 사용할 수 있음

```
>>> s1 = set([1, 2, 3, 4, 5, 6])
>>> s2 = set([4, 5, 6, 7, 8, 9])
```

▪ 교집합

▪ '&' 기호나 intersection( ) 함수 사용

```
>>> s1 & s2
{4, 5, 6}
```

```
>>> s1.intersection(s2)
{4, 5, 6}
```

▪ 합집합

▪ '|' 기호나 union( ) 함수 사용

```
>>> s1 | s2
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

```
>>> s1.union(s2)
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

▪ 차집합

▪ '-' 기호나 difference( ) 함수 사용

```
>>> s1 - s2
{1, 2, 3}
>>> s2 - s1
{8, 9, 7}
```

```
>>> s1.difference(s2)
{1, 2, 3}
>>> s2.difference(s1)
{8, 9, 7}
```

파이썬 프로그래밍의 기초, 자료형 III

- 집합자료형(Set)

▪ 집합 관련 함수

▪ add( )

- 이미 만들어진 set 자료형에 값 추가

```
>>> s1 = set([1, 2, 3])
>>> s1.add(4)
>>> s1
{1, 2, 3, 4}
```

▪ update( )

- 여러 개의 값을 한꺼번에 추가

```
>>> s1 = set([1, 2, 3])
>>> s1.update([4, 5, 6])
>>> s1
{1, 2, 3, 4, 5, 6}
```

▪ remove( )

- 특정 값을 제거

```
>>> s1 = set([1, 2, 3])
>>> s1.remove(2)
>>> s1
{1, 3}
```


Practice #3 (Set/Set Functions)

```python
# 집합 자료형 사용방법
s1 = set([3, 1, 2, 4]) # set() 괄호 안에 list 형태로 작성하거나
s2 = set("Hello") # 문자열을 입력하여 만들 수 있음
print(s1, s2) # s2의 경우, 중복되는 것이 빠져있는 것을 확인 할 수 있음

# Point: 중복 허용하지 않으며, 순서가 없음. ==> 인덱싱 등 사용 불가능 ==> 사용하고 싶으면 형변환
s1_tuple = tuple(s1)
# set 출력 값을 바탕으로 형 변환
print(type(s1_tuple), s1_tuple, s1_tuple[0], s1_tuple[1])

# Set Function (교집합, 합집합, etc.)
s1 = set([1, 2, 3, 4, 5])
s2 = ([2, 3, 5])
print(f'교집합={s1.intersection(s2)}, 합집합={s1.union(s2)}, 차집합={s1.difference(s2)}')

# 관련함수
# set에 값 하나 추가
s1 = set([1, 3, 4])
s1.add("5")
print(s1)
```

파이썬 프로그래밍의 기초, 자료형 III

- Bool 자료형

  - **불(Bool)이란?**

    - 참(True)과 거짓(False)을 나타내는 자료형

```
>>> a = True
>>> b = False
```

    - type( ) 함수를 사용하여 자료형 확인

```
>>> type(a)
<class 'bool'>
>>> type(b)
<class 'bool'>
```

    - 조건문의 반환 값으로도 사용됨

```
>>> 1 == 1
True
```

```
>>> 2 > 1
True
```

```
>>> 2 < 1
False
```

파이썬 프로그래밍의 기초, 자료형 III

- Bool 자료형
  - **자료형의 참과 거짓**

자료형	값	참 or 거짓
문자열	"python"	참
	""	거짓
리스트	[1,2,3]	참
	[]	거짓
튜플	()	거짓
딕셔너리	{}	거짓
숫자형	0이 아닌 숫자	참
	0	거짓
	None	거짓

파이썬 프로그래밍의 기초, 자료형 III

- Bool 자료형

▪ 불 연산

- bool( ) 함수
  - 자료형의 참/거짓을 식별하는 내장 함수

- 예제 1: 문자열의 참/거짓
  - 'python' 문자열은 빈 문자열이 아니므로 bool 연산의 결과로 True 반환

```
>>> bool('python')
True
```

  - '' 문자열은 빈 문자열이므로 bool 연산의 결과로 False 반환

```
>>> bool('')
False
```

- 예제 2: 리스트, 숫자의 참/거짓

```
>>> bool([1,2,3])
True
>>> bool([])
False
>>> bool(0)
False
>>> bool(3)
True
```

Practice #4 (Bool)

```python
# bool 자료형 사용
a = True
b = False
print(type(a), type(b), f'1과 1은 같나요?{1 == 1}, 1과 2는 같나요?{1 == 2}'
                        f'1은 2보다 작나요?{1 < 2}')

# str_t = ""가 되면 False가 되어 While문에서 빠져 나옴.
str_t = "Hello"
idx = 0
while str_t:
    print(idx, len(str_t), str_t)
    idx += 1
    str_t = str_t[:len(str_t) - idx]

# 해당 표를 조금 더 쉽게?
if 1:
    print("True")

if []:
    print("True")
else:
    print("False")
```

파이썬 프로그래밍의 기초, 자료형 III

- 변수 (자료형의 값을 저장하는 공간)
  - **변수란?**
    - 파이썬에서 사용하는 변수는 객체를 가리키는 것이라고 할 수 있음
      - 객체 = 자료형과 같은 것을 의미하는 말

```
>>> a = 1
>>> b = "python"
>>> c = [1,2,3]
```

    - a, b, c를 변수라고 함

    - 변수 선언 방법
      - =(assignment) 기호 사용

```
변수 이름 = 변수에 저장할 값
```

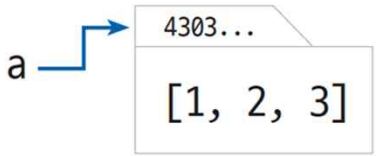

파이썬 프로그래밍의 기초, 자료형 III

- 변수 (자료형의 값을 저장하는 공간)
  - **변수란?**
    - 변수의 예시

```
>>> a = [1, 2, 3]
```

      - [1, 2, 3] 값을 가지는 리스트 자료형(객체)이 자동으로 메모리에 생성됨
      - 변수 a는 [1, 2, 3] 리스트가 저장된 메모리의 주소를 가리킴
        - id( ) 함수를 사용하여 메모리 주소 확인 (변수가 가리키고 있는 객체의 메모리 주소 값을 반환)

```
>>> a = [1, 2, 3]
>>> id(a)
4303029896
```



파이썬 프로그래밍의 기초, 자료형 III

- 변수 (자료형의 값을 저장하는 공간)

▪ 변수를 만드는 여러 가지 방법

▪ 튜플

```
>>> a, b = ('python', 'life')
```

- 괄호 생략 가능

```
>>> (a, b) = 'python', 'life'
```

▪ 리스트

```
>>> [a,b] = ['python', 'life']
```

▪ 변수 값 바꾸기

```
>>> a = 3
>>> b = 5
>>> a, b = b, a ◀── a와 b의 값을 바꿈
>>> a
5
>>> b
3
```

Practice #5 (Variable)

```python
# 복사
a = [1, 2, 3]
b = a[:] # 리스트 슬라이싱을 사용하면 새로운 객체를 생성하고 b는 새로 생성된 객체의 주소를 참조

c = a.copy()
print(id(a), id(b), id(c), c)

# 변수 만드는 여러가지 방법
a, b = ("coding", "life")
(a, b) = ("coding", "life")
[a, b] = ['coding', 'life']
print(type(a), type(b), a, b)
a = ("coding", "life")
print(type(a))
a = "coding", "life"
print(type(a))

# Swap
a, b = 3, 10
print(a, b)
a, b = b, a
print(a, b)
```

</> source code

```c
#include <stdio.h>

int main(void) {
    int a = 10;
    int b = 3;
    int tmp = 0;
    printf("a, b = %d, %d\n", a, b);
    tmp = a;
    a = b;
    b = tmp;
    printf("a, b = %d, %d\n", a, b);
    return 0;
}
```

Thank you

Q&A

www.kopo.ac.kr
jsshin7@kopo.ac.kr

