인공지능 기초 프로그래밍

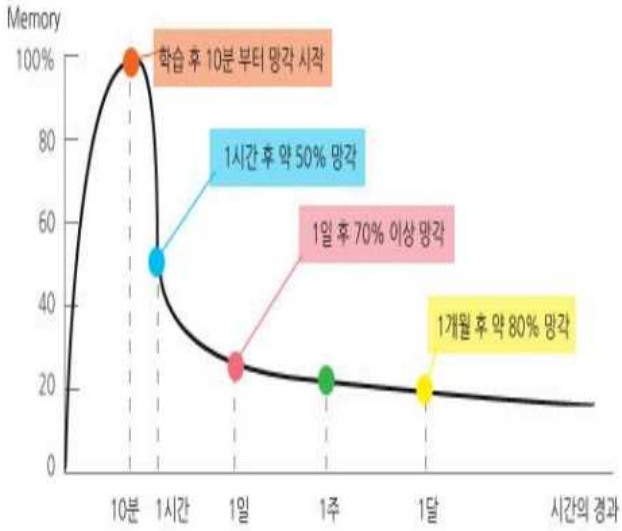
Python 기초 (자료형2)

Daegu Campus of
KOREA POLYTECHNIC

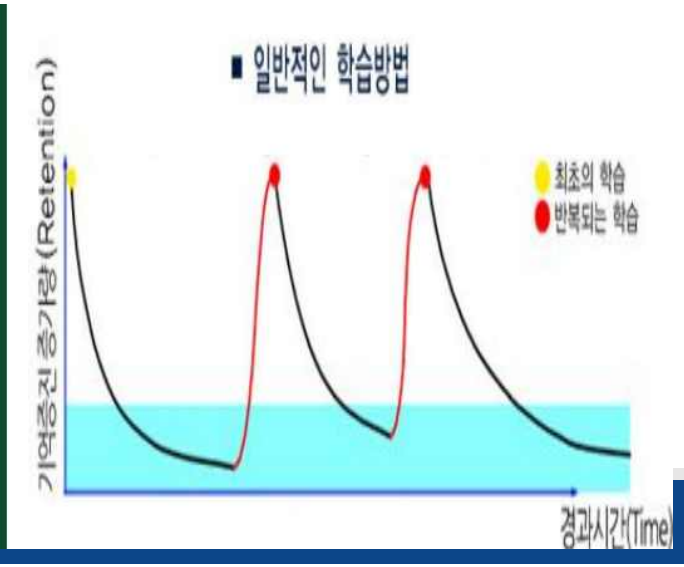
수업 효율 [말로 설명하고, 반복 학습하기]

수업 효율

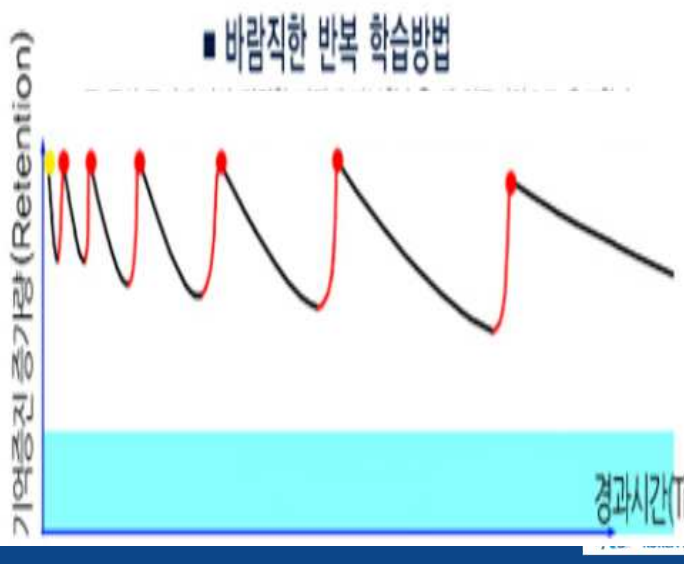
에빙하우스의 망각곡선



일반적인 학습



반복학습



파이썬 프로그래밍의 기초, 자료형 (Recap)

▪ 문자열 인덱싱

- **인덱싱(Indexing)**
  - '가리킨다'는 의미
  - 파이썬은 0부터 숫자를 셈
  - a[번호]
    - 문자열 안의 특정 값 뽑아냄
    - 마이너스(-)
      - 문자열 뒤부터 셈

L	i	f	e		i	s		t	o	o		s	h	o	r	t	,		Y	o	u		n	e	e	d		P	y	t	h	o	n
0										1										2										3			
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3

```
>>> a = "Life is too short, You need Python"
>>> a[0]
'L'
>>> a[12]
's'
>>> a[-1]
'n'
```

```
a[0]: 'L', a[1]: 'i', a[2]: 'f', a[3]: 'e', a[4]: ' ', ···
```

파이썬 프로그래밍의 기초, 자료형 (Recap)

▪ 문자열 슬라이싱

▪ 슬라이싱(Slicing)

- '잘라낸다'는 의미
- a[시작 번호:끝 번호]
  - 시작 번호부터 끝 번호까지의 문자를 뽑아냄
  - 끝 번호에 해당하는 것은 포함하지 않음

L	i	f	e		i	s		t	o	o		s	h	o	r	t	,		Y	o	u		n	e	e	d		P	y	t	h	o	n
0										1										2											3			
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3

```
>>> a = "Life is too short, You need Python"
>>> a[0:4]
'Life'
```

```
>>> a = "20010331Rainy"
>>> date = a[:8]
>>> weather = a[8:]
>>> date
'20010331'
>>> weather
'Rainy'
```

Practice #4 (문자열 연산, 인덱싱, 슬라이싱) (Recap)

```
# String Operation
str1 = "Hi Everyone"
str2 = "My name is james"
str3 = "**********"# Comment Block
print(f'{str3*10}\n{str1+str2},\n'
      f'This Block is comment block\n'
      f'{str3*10} ')

# indexing/Slicing
lenStr2 = len(str2)
# j --> "J"
print(str2[-5])
#str2[-5] = "J" # Error
#str2 = str2[:-5]+"J"+str2[lenStr2-4:]
print(str2)

str2 = "20210308Sunny"
print('Today weather is ' + '\"' + str2[8:] + '\"')
```

▪ 문자열 포매팅

▪ 포매팅(Formatting)

1. 숫자 바로 대입
   - 문자열 포맷 코드 %d

```
>>> "I eat %d apples." % 3
'I eat 3 apples.'
```

```
>>> number = 3
>>> "I eat %d apples." % number
'I eat 3 apples.'
```

2. 문자열 바로 대입
   - 문자열 포맷 코드 %s

```
>>> "I eat %s apples." % "five"
'I eat five apples.'
```


파이썬 프로그래밍의 기초, 자료형

▪ 문자열 포매팅

▪ 문자열 포맷 코드

코드	설명
%s	문자열(String)
%c	문자 1개(Character)
%d	정수(Integer)
%f	부동 소수(Floating-point)
%o	8진수
%x	16진수
%%	Literal % (문자 '%' 자체)

```
>>> number = 10
>>> day = "three"
>>> "I ate %d apples. so I was sick for %s days." % (number, day)
'I ate 10 apples. so I was sick for three days.'
```

```
>>> "I have %s apples" % 3
'I have 3 apples'
>>> "rate is %s" % 3.234
'rate is 3.234'
```

▪ 문자열 포매팅

▪ 문자열 포맷 코드 활용법

1. 정렬과 공백

- %s를 숫자와 함께 사용하면, 공백과 정렬 표현 가능

```
>>> "%10s" % "hi"
'        hi' ◀── hi가 오른쪽 정렬됨
```


```
>>> "%-10sjane" % 'hi'
'hi        jane' ◀── hi가 왼쪽 정렬됨
```

▪ 문자열 포매팅

▪ 문자열 포맷 코드 활용법

2. 소수점 표현하기

- %f를 숫자와 함께 사용하면, 소수점 뒤에 나올 숫자의 개수 조절 및 정렬 가능

```
>>> "%0.4f" % 3.42134234
'3.4213'
```

```
>>> "%10.4f" % 3.42134234
'    3.4213'
```

▪ 문자열 포매팅

recommendation

▪ f 문자열 포매팅

- 파이썬 3.6 버전부터 f 문자열 포매팅 기능 제공
- 문자열 앞에 f 접두사를 붙이면, f 문자열 포매팅 기능 사용 가능

```
>>> name = '홍길동'
>>> age = 30
>>> f'나의 이름은 {name}입니다. 나이는 {age}입니다.'
'나의 이름은 홍길동입니다. 나이는 30입니다.'
```

```
>>> age = 30
>>> f'나는 내년이면 {age+1}살이 된다.'
'나는 내년이면 31살이 된다.'
```


Practice #5 (문자열 Formating)

```
number = 10
day = "three"
str1 = "I ate %d apples.\nSo I was sick for %s days." % (number, day)
print(str1)
str1 = f"I ate {number} apples.\nSo I was sick for {day} days."
print(str1)

str1 = "%4s\n%4s\n%4s\n%4s" % ("a", "ab", "abc", "abcd")
print(str1)

str1 = "%-10sjane" % ("hi")
print(str1, len(str1))

print(f"{3.42134234:0.4f}")
str2 = f"{3.42134234:10.3f}"
print(str2, len(str2))
```

▪ 문자열 관련 함수

- 문자열 자료형이 가진 내장 함수
- **count( )**
  - 문자 개수 세는 함수

```
>>> a = "hobby"
>>> a.count('b')
2
```

- **find( )**
  - 찾는 문자열이 처음 나온 위치 반환
  - 없으면 -1 반환

```
>>> a = "Python is the best choice"
>>> a.find('b')
14 ◀── 문자열에서 b가 처음 나온 위치
>>> a.find('k')
-1
```


▪ 문자열 관련 함수

▪ index( )

- find와 마찬가지로,
  찾는 문자열이 처음 나온 위치 반환
- 단, 찾는 문자열이 없으면 오류 발생

```
>>> a = "Life is too short"
>>> a.index('t')
8
>>> a.index('k')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: substring not found
```

k가 없기 때문에 오류 발생

▪ join( )

- 문자열 삽입

```
>>> ",".join('abcd')
'a,b,c,d'
```

▪ upper( )

- 소문자를 대문자로 변환

```
>>> a = "hi"
>>> a.upper()
'HI'
```


▪ 문자열 관련 함수

- **lower( )**
  - 대문자를 소문자로 변환

```
>>> a = "HI"
>>> a.lower()
'hi'
```

- **lstrip( )**
  - 가장 왼쪽에 있는 연속된 공백 삭제

```
>>> a = " hi "
>>> a.lstrip()
'hi '
```

- **rstrip( )**
  - 가장 오른쪽에 있는 연속된 공백 삭제

```
>>> a= " hi "
>>> a.rstrip()
' hi'
```

- **strip( )**
  - 양쪽에 있는 연속된 공백 삭제

```
>>> a = " hi "
>>> a.strip()
'hi'
```


▪ 문자열 관련 함수

▪ replace( )

- replace(바뀌게 될 문자열, 바꿀 문자열)
- 문자열 안의 특정 값을 다른 값으로 치환

```
>>> a = "Life is too short"
>>> a.replace("Life", "Your leg")
'Your leg is too short'
```

▪ split( )

- 공백 또는 특정 문자열을 구분자로 해서 문자열 분리
  분리된 문자열은 리스트로 반환됨

```
>>> a = "Life is too short"
>>> a.split() ◀── 공백을 기준으로 문자열 나눔
['Life', 'is', 'too', 'short']
>>> b = "a:b:c:d"
>>> b.split(':') ◀── : 기호를 기준으로 문자열 나눔
['a', 'b', 'c', 'd']
```


Practice #6 (문자열 관련 함수)

```
str1 = """
Spread out before him that April day was the largest flotilla Communist-ruled China
 48 ships, dozens of fighter jets, more than 10,000 military personnel.
"""

cnt_a = str1.count("ships")
print(cnt_a)
print(str1.find("jets"), str1.find("h"), str1.index("h"), str1.find("korea"))
print(str1.upper())
print(str1.lower())

str1 = "12345"
print(",".join(str1))

str_list = "Life is too short"
print(str_list.replace("short", "long"))
print(str_list)

str_list = str_list.split()
print(str_list, type(str_list))

str_list = "    ".join(str_list)
print(str_list, type(str_list))
```

```
str_txt = "vehicle 0 0 50 50 vehicle 50 50 250 250"
```

Vehicle x y w h가 반복되는 데이터가 있을 경우,
해당 데이터에서 넓이가 100이상인 vehicle인 경우에는
Vehicle->truck으로 변경
- Vehicle만 있고, 두 개의 라벨링 데이터만 존재

vehicle 10 10 50 50 Truck 50 50 250 250

C dll 연동: Hello World

- C:\windows\SysWOW64\msvcrt.dll (MS Visual C Runtime DLL)-> 현재 작업 directory에 복사

- dynamic-link library(동적 링크 라이브러리)

```
from ctypes import cdll
libc = cdll.LoadLibrary('msvcrt.dll')
libc.printf(b'hello world!\n')

listTmp = [1, 2, 3, 4, 5, 6, 7]
print(f'{len(listTmp)}')

for idx in listTmp:
    libc.printf(b"%d ", idx)
```


List, Tuple,

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

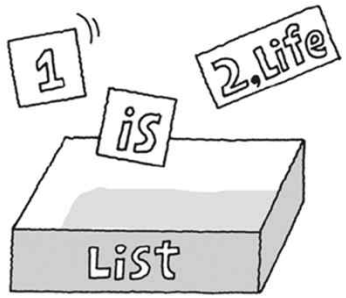
▪ 리스트 사용법

- 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)

```python
# 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
```

Thank you

Q&A

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

