파이썬프로그래밍의 기초
자료형

3주차 실습

1. 숫자형(Number)

- 숫자 형태로 이루어진 자료형

항목	사용 예
정수	123, -345, 0
실수	123.45, -1234.5, 3.4e10
복소수	1 + 2j, -3j
8진수	0o34, 0o25
16진수	0x2A, 0xFF

1. 숫자형

- 정수형(Integer)

```
>>> a = 123
>>> a = -178
>>> a = 0
```

- 실수형(Floating-point)

```
>>> a = 1.2
>>> a = -3.45
```

```
>>> a = 4.24E10
>>> a = 4.24e-10
```

- 8진수(Octal)

```
>>> a = 0o177
```

- 16진수(Hexadecimal)

```
>>> a = 0x8ff
>>> b = 0xABC
```

1. 숫자형

- 복소수(Complex number)
  - 복소수.real

```
>>> a = 1+2j
>>> a.real
1.0
```

  - 복소수.imag

```
>>> a = 1+2j
>>> a.imag
2.0
```

  - 복소수.conjugate

```
>>> a = 1+2j
>>> a.conjugate()
(1-2j)
```

  - abs(복소수)

```
>>> a = 1+2j
>>> abs(a)
2.2360679774997898
```

1. 숫자형

- 숫자형을 활용하기 위한 연산자
  - 사칙연산

```
>>> a=3; b=4
>>> a+b
7
>>> a*b
12
>>> a/b
0.75
```

  - % 연산자

```
>>> 7 % 3
1
>>> 3 % 7
3
```

  - ** 연산자

```
>>> a = 3
>>> b = 4
>>> a ** b
81
```

  - // 연산자

```
>>> 7 / 4
1.75
```

```
>>> 7 // 4
1
```

2. 문자열 자료형

- 문자열 만드는 방법

1. 큰따옴표로 양쪽 둘러싸기
2. 작은따옴표로 양쪽 둘러싸기
3. 큰따옴표 3개를 연속으로 써서 양쪽 둘러싸기
4. 작은따옴표 3개를 연속으로 써서 양쪽 둘러싸기

```
"Hello World"
```

```
'Python is fun'
```

```
"""Life is too short, You need python"""
```

```
'''Life is too short, You need python'''
```

2. 문자열 자료형

1. 문자열에 작은따옴표 (') 포함시키기

```
>>> food = "Python's favorite food is perl"
```

2. 문자열에 큰따옴표 (") 포함시키기

```
>>> say = '"Python is very easy." he says.'
```

3. \(백슬래시)를 이용해서 작은따옴표(')와 큰따옴표(")를 문자열에 포함시키기

```
>>> food = 'Python\'s favorite food is perl'
>>> say = "\"Python is very easy.\" he says."
```

2. 문자열 자료형

- 여러 줄인 문자열을 변수에 대입하고 싶을 때
  1) 줄을 바꾸기 위한 이스케이프 코드 \n 삽입하기

```
>>> multiline = "Life is too short\nYou need python"
```

  2) 연속된 작은따옴표 3개(''') 또는 큰따옴표 3개(""") 이용

```
>>> multiline='''
... Life is too short
... You need python
... '''
```

```
>>> multiline="""
... Life is too short
... You need python
... """
```

```
>>> print(multiline) )
Life is too short
You need python
```

2. 문자열 자료형

- 문자열 연산하기
  1. 문자열 더해서 연결하기(Concatenation)

```
>>> head = "Python"
>>> tail = " is fun!"
>>> head + tail
'Python is fun!'
```

  2. 문자열 곱하기

```
>>> a = "python"
>>> a * 2
'pythonpython'
```

2. 문자열 자료형

- 문자열 연산하기

3. 문자열 곱하기 응용

```
print("=" * 50)
print("My Program")
print("=" * 50)
```

```
==================================================
My Program
==================================================
```

2. 문자열 자료형

- 문자열 포매팅 (Formatting)
  1. 숫자 바로 대입

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

  2. 문자열 바로 대입

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

코드	설명
%s	문자열 (String)
%c	문자 1개(character)
%d	정수 (Integer)
%f	부동소수 (floating-point)
%o	8진수
%x	16진수
%%	Literal % (문자 `%` 자체)

2. 문자열 자료형

- 문자열 포매팅 (Formatting)

  3. 숫자 값을 나타내는 변수로 대입

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

  4. 2개 이상의 값 넣기

```
>>> 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.'
```

2. 문자열 자료형

- 문자열 관련 함수들
  - 문자 개수 세기(count)

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

  - 위치 알려주기1(find)

```
>>> a = "Python is best choice"
>>> a.find('b')
10
>>> a.find('k')
-1
```

  - 위치 알려주기2(index)

```
>>> 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
```

2. 문자열 자료형

- 문자열 관련 함수들
  - 문자열 삽입(join)

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

  - 소문자를 대문자로 바꾸기(upper)

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

  - 대문자를 소문자로 바꾸기(lower)

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

2. 문자열 자료형

- 문자열 관련 함수들
  - 왼쪽 공백 지우기(lstrip)

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

  - 오른쪽 공백 지우기(rstrip)

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

  - 양쪽 공백 지우기(strip)

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

2. 문자열 자료형

- 문자열 관련 함수들
  - 문자열 바꾸기(replace)

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

  - 문자열 나누기(split)

```
>>> a = "Life is too short"
>>> a.split()
['Life', 'is', 'too', 'short']
>>> a = "a:b:c:d"
>>> a.split(':')
['a', 'b', 'c', 'd']
```

3. 리스트 자료형

리스트명 = [요소1,요소2,요소3,...]

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

3. 리스트 자료형

- 리스트의 인덱싱

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

```
>>> a[0]
1
>>> a[-1]
['a', 'b', 'c']
>>> a[3]
['a', 'b', 'c']
```

```
>>> a[-1][0]
'a'
>>> a[-1][1]
'b'
>>> a[-1][2]
'c'
```

3. 리스트 자료형

- 리스트의 슬라이싱

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

문자열의 슬라이싱

```
>>> a = "12345"
>>> a[0:2]
'12'
```

3. 리스트 자료형

- 리스트의 수정, 변경과 삭제
  1. **리스트에서 하나의 값 수정하기**

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

  2. **리스트에서 연속된 범위의 값 수정하기**

```
>>> a[1:2]
[2]
>>> a[1:2] = ['a', 'b', 'c']
>>> a
[1, 'a', 'b', 'c', 4]
```

3. 리스트 자료형

- 리스트의 수정, 변경과 삭제

3. [ ] 사용해 리스트 요소 삭제하기

```
>>> a[1:3] = [ ]
>>> a
[1, 'c', 4]
```

4. del 함수 사용해 리스트 요소 삭제하기

```
>>> a
[1, 'c', 4]
>>> del a[1]
>>> a
[1, 4]
```

4. 튜플 자료형

- **리스트**는 **[**과 **]**으로 둘러싸지만 **튜플**은 **(**과 **)**으로 둘러싼다.
- **리스트는 그 값의 생성, 삭제, 수정이 가능하지만 튜플은 그 값을 바꿀 수 없다.**

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

4. 튜플 자료형

1. 튜플 요소값 삭제 시 오류

```
>>> t1 = (1, 2, 'a', 'b')
>>> del t1[0]
Traceback (innermost last):
File "", line 1, in ?del t1[0]
TypeError: object doesn't support item deletion
```

2. 튜플 요소값 변경 시 오류

```
>>> t1 = (1, 2, 'a', 'b')
>>> t1[0] = 'c'
Traceback (innermost last):
File "", line 1, in ?t1[0] = 'c'
TypeError: object doesn't support item assignment
```

5. 딕셔너리 자료형

{Key1:Value1, Key2:Value2, Key3:Value3 ...}

- Key와 Value의 쌍 여러 개가 {과 }로 둘러싸여 있다.
- 각각의 요소는 Key : Value 형태로 이루어져 있고 쉼표(,)로 구분되어 있다.
- ※ Key에는 변하지 않는 값을 사용하고, Value에는 변하는 값과 변하지 않는 값 모두 사용할 수 있다.

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

5. 딕셔너리 자료형

1. 딕셔너리 쌍 추가하기

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

2. 딕셔너리 요소 삭제하기

```
>>> del a[1]
>>> a
{2: 'b', 3: [1, 2, 3], 'name': 'pey'}
```

5. 딕셔너리 자료형

- 딕셔너리를 사용하는 방법

```
{"김연아":"피겨스케이팅", "류현진":"야구", "박지성":"축구", "귀도":"파이썬"}
```

  - 딕셔너리에서 Key 사용해 Value 얻기

```
>>> grade = {'pey': 10, 'julliet': 99}
>>> grade['pey']
10
>>> grade['julliet']
99
```

5. 딕셔너리 자료형

- 딕셔너리 관련 함수들
  - Key 리스트 만들기(keys)

```
>>> a = {'name': 'pey', 'phone': '0119993323', 'birth': '1118'}
>>> list(a.keys())
['phone', 'name', 'birth']
```

- dict_keys 객체는 리스트를 사용하는 것과 차이가 없지만, 리스트 고유의 함수인 append, insert, pop, remove, sort 등의 함수를 수행할 수는 없다.

```
>>> for k in a.keys():
...     print(k)
...
phone
birth
name
```

5. 딕셔너리 자료형

- dict_keys 객체를 리스트로 변환

```
>>> list(a.keys())
['phone', 'birth', 'name']
```

- Value 리스트 만들기(values)

```
>>> list(a.values())
['0119993323', 'pey', '1118']
```

- Key: Value 쌍 모두 지우기(clear)

```
>>> a.clear()
>>> a
{}
```

5. 딕셔너리 자료형

- Key로 Value얻기(get)

```
>>> a = {'name':'pey', 'phone':'0119993323', 'birth': '1118'}
>>> a.get('name')
'pey'
>>> a.get('phone')
'0119993323'
```

딕셔너리에 Key 값이 없는 경우

```
>>> a.get('nokey')
>>> a['nokey']
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
KeyError: 'nokey'
```

-> 디폴트값을 지정한 경우

```
>>> a.get('foo', 'bar')
'bar'
```

5. 딕셔너리 자료형

- 해당 Key가 딕셔너리 안에 있는지 조사하기(in)

```
>>> a = {'name':'pey', 'phone':'0119993323', 'birth': '1118'}
>>> 'name' in a
True
>>> 'email' in a
False
```

6. 집합 자료형

- 집합에 관련된 것들을 쉽게 처리하기 위해 만들어진 자료형이다.

```
>>> s1={1,2,3}
>>> s1
{1, 2, 3}
>>> s2=set([1,2,3])
>>> s2
{1, 2, 3}
```

6. 집합 자료형

- 중복을 허용하지 않는다.
- 순서가 없다(Unordered).

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

※ 중복을 허용하지 않는 set의 특징은 자료형의 중복을 제거하기 위한 필터 역할로 종종 사용.

6. 집합 자료형

- 집합 자료형 활용하는 방법
  - 교집합

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

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

  - 합집합

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

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

  - 차집합

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

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

6. 집합 자료형

- 집합 자료형 관련 함수들
  - 값 1개 추가하기(add)

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

  - 값 여러 개 추가하기(update)

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

  - 특정 값 제거하기(remove)

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

과제

- 과제1. 한 인기 식당의 방문자수는 다음 표와 같습니다. 03/08의 방문자수를 리스트의 시작으로 visitor라는 이름의 리스트를 만드세요.

날짜	방문자수
03/08	488
03/09	500
03/10	501
03/11	461
03/12	474

과제

- 과제2. [과제1]에서 만든 ‘visitor’를 이용하여 가장 방문자수가 많았던 날의 방문자수를 화면에 출력하세요. (힌트: 최댓값을 찾는 함수로 max( ) 함수를 사용.)

과제

- 과제3. [과제1]에서 만든 ‘visitor’를 이용하여 가장 방문자수가 적었던 날의 방문자수를 화면에 출력하세요. (힌트: 최댓값을 찾는 함수로 min( ) 함수를 사용.)

과제

- 과제4. [과제1]에서 만든 ‘visitor’를 이용해서 방문자수가 가장 높았던 날의 방문자수의 차이를 화면에 출력하세요.

과제

- 과제5. [과제1]에서 만든 리스트를 이용해서 “방문자 수 :” 문자열과 함께 3월9일 , 3월 10일의 방문자수만 화면에 출력하세요.

<결과 출력 예시>

```
방문자 수 :  501 461
```

과제

- 과제6. [과제1]의 표를 사용해서 날짜를 딕셔너리의 key값으로, 방문자수를 딕셔너리의 value 값으로 하여 ‘visitor2’라는 딕셔너리를 생성하세요.

과제

- 과제7. [과제6]에서 생성한 'visitor2' 딕셔너리를 사용해서 3월 12일의 방문자수를 출력하세요.

과제 제출 방법

- 과제캡쳐 후 워드or한글파일에 첨부/정리하여 제출
- 파일형식 : [과제번호2]_이름(조이름)_학번
  - 제출 형식 어길 시 감점처리
- 제출 : dbcyy1@gmail.com로 제출
- 제출기간 : 3월22일 화요일 23시59분까지

- **첨부파일 넣었는지, 메일 반송되지 않았는지 꼭 확인하세요!**