02장 변수와 자료형

2.1 변수

- 일반적으로 프로그램은 연산을 수행하며, 이러한 연산을 하기 위해서는 값들을 임시로 혹은 영구적으로 어딘가에 저장해 둬야 하는데, 이러한 값들을 저장하는 저장소를 변수라고 한다.
- 예를 들면 코드가 a = 3이라고 하면 3이라는 값을 가지는 정수 자료형(객체)이 자동으로 메모리에 생성된다. a는 변수의 이름이며, 3이라는 정수형 객체가 저장된 메모리 위치를 가리키게 된다. 즉, 변수 a는 객체가 저장된 메모리의 위치를 가리키는 레퍼런스(Reference)라고도 할 수 있다.
- 변수명은 문자, 숫자, 밑줄(_)을 표함할 수 있으나 숫자는 가장 처음에 나올 수 없다. 또한 대소문자를 구분하여 사용한다. 마지막으로 예약어는 변수명으로 사용할 수 없다.
  - 가능한 예: a, b, friends, grunt, var3, _attr, is4later
  - 잘못된 예: 2_Variable, and, as, def

```
# 변수란?
friend = 1
Friend = 10
print(friend)
print(Friend)

# 변수를 만드는 여러 가지 방법
a, b = ('python', 'life')
(a, b) = 'python', 'life'
a = b = 'python'
a = 3
b = 5
a, b = b, a
print(a)
print(b)

# 메모리에 생성된 변수 없애기
a = 3
b = 3
del(a)
del(b)

# 리스트를 변수에 넣고 복사하고자 할 때
a = [1,2,3]
b = a
a[1] = 4
print(a)
print(b)

# [:] 이용
a = [1,2,3]
b = a[:]  # a 리스트 전체를 복사하여 b에 대입
a[1] = 4
print(a)
print(b)

# copy 모듈 이용
from copy import copy
b = copy(a)
print(b)

print(b is a)
# False, b와 a가 서로 다른 객체임을 알 수 있다.
```

2.2 숫자 자료형

2.2.1 숫자 자료형은 어떻게 만들고 사용할까?

```
# 숫자형은 어떻게 만들고 사용할까?
# 정수형
a = 123
a = -178
a = 0

# 실수형
a = 1.2
a = -3.45
a = 4.24E10
a = 4.24E-10

# 8진수와 16진수
a = 0o177
a = 0x8ff
```

2.2.2 숫자 자료형을 활용하기 위한 연산자

```
# 숫자형을 활용하기 위한 연산자
# 사칙연산
a = 3
b = 4
print(a+b)
print(a*b)
print(a/b)

# x의 y제곱을 나타내는 ** 연산자
print(a**b)

# 나눗셈 후 나머지를 반환하는 % 연산자
print(7%3)
print(3%7)

# 나눗셈 후 소수점 아랫자리를 버리는 // 연산자
print(7/4)
print(7//4)
```

2.3 문자열 자료형

2.3.1 문자열은 어떻게 만들고 사용할까?

- 문자열은 단일인용부호(')나 다중인용부호(")로 묶어서 표현한다.
- 좀 더 쉽게 다량의 문자열을 넣기 위해 """ 위해 '''을 사용한다.
- 문자열 내에서 줄바꿈이나 탭 등의 특수문자를 어떻게 표현할 수 있을까? 이스케이프 코드를 사용한다. 이스케이프 코드란 프로그래밍할 때 사용할 수 있도록 미리 정의해 둔 '문자 조합

'이다.

코드	의미
\n	개행(줄바꿈)
\t	탭
\r	캐리지 반환
\0	널(Null)
\\	문자 '\'
\'	단일 인용부호(')
\"	이중 인용부호(")

```
# 문자열은 어떻게 만들고 사용할까?
"Hello World"
'Python is fun'
"""Life is too short,
You need python"""
'''Life is too short,
You need python'''

# 문자열에 따옴표 포함시키기
food = "Python's favorite food is perl"
say = '"Python is very easy." he says.'
food = 'Python\'s favorite food is perl'
say = "\"Python is very easy.\" he says."

# 여러 줄인 문자열을 변수에 대입하고 싶을 때
multiline = "Life is too short\nYou need python"
multiline = '''
Life is too short
You need python
'''
print(multiline)
print("\t탭\n다음줄")
```

2.3.2 문자열 연산하기

```
# 문자열 연산하기
head = "Python"
tail = " is fun!"
print(head + tail)
print(head * 2)
print("=" * 50)
print("My Program")
print("=" * 50)
```

2.3.3 문자열 인덱싱과 슬라이싱

```
# 문자열 인덱싱과 슬라이싱
a = "Life is too short, You need Python"
print(a[0])
# L
print(a[12])
# s
print(a[-1])
#'n'
```

```
a = "Life is too short, You need Python"
print(a[0:4])
# 'Life'
a = "20010311Rainy"
date = a[:8]
weather = a[8:]
print(date)
# '20010331'
print(weather)
# 'Rainy'
```

2.3.4 문자열 포매팅

```
# 문자열 포매팅
print("I eat %d apples." % 3)
#'I eat 3 apples.'
number = 10
day = "three"
print("I ate %d apples. so I was sick for %s days." % (number, day))
#'I ate 10 apples. so I was sick for three days.'

# 정렬과 공백
print("%10s" % "hi")
#'        hi'
print("%-10sjane." % 'hi')
#'hi        jane.'

# 소수점 표현
print("%0.4f" % 3.42134234)
#'3.4213'
print("%10.4f" % 3.42134234)
#'    3.4213'

# 고급 문자열 포매팅
print("I eat {0} apples".format(3))
#I eat 3 apples

print("I eat {0} apples".format("three"))
#I eat three apples

number = 3
print("I eat {0} apples".format(number))
#I eat 3 apples

# 2개 이상의 값 넣기
number = 10
day = "three"
print("I ate {0} apples, so I was sick for {1} days.".format(number, day))
#I ate 3 apples, so I was sick for three days.
```

2.3.5 문자열 관련 함수들

```
# 문자열 관련 함수들
# 문자 개수 세기(count)
a = "hobby"
```

```
print(a.count('b'))
#2

# 위치 알려주기(find)
a = "Python is best choice"
print(a.find('b'))
#10
print(a.find('k'))
#-1

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

# 문자열 삽입(join)
a= ","
print(a.join('abcd'))
#'a,b,c,d'

a = "hi"
print(a.upper())
#'HI'
a = "HI"
print(a.lower())
#'hi'

# 양쪽 공백 지우기(strip)
a = " hi "
print(a.strip())
#'hi'

# 문자열 바꾸기(replace)
a = "Life is too short"
print(a.replace("Life", "Your leg"))
#'Your leg is too short'

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

2.3.6 유니코드

- 운영체제에서 문자열 표현 방식을 인코딩(incoding)이라고 한다. 각 언어권의 운영체제마다 서로 다른 인코딩을 사용하게 되면 타 언어권의 문자열을 제대로 표현할 수 없다.
- 세계의 모든 문자 코드를 일관된 방식으로 표현할 수 있는 체계를 만들자는 취지에서 시작된 게 바로 유니코드(unicode)이다.
- 콘솔 입력이나 파일은 유니코드 포인트로 동작하는 것이 아니라 인코딩된 바이트 열로 동작한다. 따라서 파이썬이 콘솔로 입력된 코드나 스크립트 코드를 읽을 때는 이 코드가 어떤 방식으로 인코딩되어 있는지 알고 있어야 한다. 콘솔 입력이면 환경 변수 PYTHONIOENCODING으로 지정하고, 스크립트(파일)이면 첫 줄에 다음처럼 인코딩 설정 정보를 넣어 주면 된다.

```
#-*- coding: utf-8 -*-

print(type('가'))
str1 = '가'.encode('utf-8')
print(type(str1))
print(str1)
print(str1.decode('utf-8'))

# chr(), ord()는 어떤 문자의 유니코드 값을 알고 싶거나 혹은 반대로 유니코드 값을 문자로 변환할 때에 사용한다.
print(ord('가')
print(chr(44032))
```

2.4 리스트 자료형

- 리스트는 값의 나열이다. 순서가 존재하고 여러 종류의 값을 담을 수 있으며, []로 묶어서 정의한다.
- 문자열과 마찬가지로 0부터 시작하는 인덱스가 있으며, 슬라이싱도 가능하다.

```
# 리스트는 어떻게 만들고 사용할까?

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

# 리스트의 인덱싱과 슬라이싱
# 리스트의 인덱싱
a = [1, 2, 3]
print(a)
# [1, 2, 3]
print(a[0])
#1
print(a[0] + a[2])
#4
print(a[-1])
#3
a = [1,2,3,['a','b','c']]
print(a[0])
#1
print(a[-1])
#['a','b','c']
print(a[3])
#['a','b','c']
print(a[-1][0])
#'a'

# 리스트의 슬라이싱
a = [1,2,3,4,5]
print(a[0:2])
#[1,2]
b = a[:2]
c = a[2:]
print(b)
#[1,2]
print(c)
#[3,4,5]
```

```
a = "12345"
print(a[0:2])
#'12'

# 리스트 연산자
a = [1,2,3]
b = [4,5,6]
print(a+b)
print(a*3)
a[2]=4
print(a)
a[1:2] = ['a','b','c']
print(a)
a[1:3] = []
print(a)
#[1,'c',4]
del a[1]
print(a)
#[1,4]

a = [1, 2, 3]
a[2] + "h1"  # 형 오류(TypeError)가 발생한다.
str(a[2]) + "hi"  # str()은 정수나 실수를 문자열의 형태로 바꾸어 주는 파이썬의 내장 함수이다.

# 리스트 관련 함수들
a = [1,2,3]
a.append(4)
print(a)
#[1,2,3,4]
a.append([5,6])
print(a)

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

a = [1,4,3,2]
a.sort()
print(a)
a = ['a','c','b']
a.sort()
print(a)
a = [1,2,3,1]
print(a.count(1))
#2
```

2.5 세트 자료형

2.5.1 세트는 어떻게 만들까?

- 세트(set)는 수학시간에 배운 집합과 동일하다. 세트는 리스트와 마찬가지로 값의 모임이며, {}로 묶어서 정의한다.
- set에는 다음과 같은 2가지 큰 특징이 있다.
  - 중복을 허용하지 않는다.
  - 순서가 없다(Unordered). -> 인덱싱으로 값을 얻을 수 없다.

```
# 세트(집합) 자료형은 어떻게 만들까?
s1 = set([1,2,3])
print(s1)
# {1, 2, 3}

s2 = set("Hello")
print(s2)
# {'e', 'l', 'o', 'H'}

# 집합 자료형의 특징
# set 자료형에 저장된 값을 인덱싱으로 접근하려면 리스트나 튜플로 변환한 후 해야 한다.
s1 = set([1, 2, 3])
l1 = list(s1)  # 리스트로 변환
print(l1)
# [1, 2, 3]
print(l1[0])
# 1

t1 = tuple(s1)  # 튜플로 변환
print(t1)
# (1, 2, 3)
print(t1[0])
# 1

# 집합 자료형 활용하는 방법
s1 = set([1, 2, 3, 4, 5, 6])
s2 = set([4, 5, 6, 7, 8, 9])

# 교집합
print(s1 & s2)
# {4, 5, 6}
print(s1.intersection(s2))
# {4, 5, 6}

# 합집합
print(s1 | s2)
# {1, 2, 3, 4, 5, 6, 7, 8, 9}
print(s1.union(s2)
# {1, 2, 3, 4, 5, 6, 7, 8, 9}

# 차집합
print(s1 - s2)
# {1, 2, 3}
print(s2 - s1)
# {8, 9, 7}
print(s1.difference(s2))
print(s2.difference(s1))
```

2.5.2 세트 관련 함수들

```
# 세트 관련 함수들
# 값 1개 추가하기(add)
s1 = set([1,2,3])
s1.add(4)
print(s1)

# 값 여러개 추가하기(update)
s1 = set([1,2,3])
s1.update([4,5,6])
```

```
print(s1)

# 특정 값 제거하기(remove)
s1 = set([1,2,3])
s1.remove(2)
print(s1)
```

2.6 튜플 자료형

2.6.1 튜플은 어떻게 만들까?

- 튜플(tuple)은 몇 가지 점을 제외하곤 리스트와 거의 비슷하며 리스트와 다른 점은 다음과 같다.
  - 리스트는 [과]으로 둘러싸지만 튜플은 (과)으로 둘러싼다.
  - 리스트는 그 값의 생성, 삭제, 수정이 가능하지만 튜플은 그 값을 바꿀 수 없다.

```
# 튜플은 어떻게 만들까?
t1 = ()
t2 = (1,)   # 단지 1개의 요소만을 가질 때는 요소 뒤에 콤마(,)를 반드시 붙여야 한다.
t3 = (1,2,3)
t4 = 1,2,3  # 괄호()를 생략해도 무방하다.
print(t4)
#(1, 2, 3)
t5 = ('a','b',('ab','cd'))

a = set((1,2,3))
print(a)
#[1,2,3]
type(a)
#set
b = list(a)
#[1,2,3]
type(b)
#list
c = tuple(b)
c
#(1,2,3)
type(c)
#tuple

# 튜플의 요소값을 지우거나 변경하려고 하면 어떻게 될까?
t1 = (1,2,'a','b')
#del t1[0]
#TypeError: 'tuple' object doesn't support item deletion

t1 = (1,2,'a','b')
#t1[0] = 'c'
#TypeError: 'tuple' object does not support item assignment
```

2.6.2 튜플의 인덱싱과 슬라이싱, 더하기(+)와 곱하기(*)

```
# 튜플의 인덱싱과 슬라이싱, 더하기(+)와 곱하기(*)
t1 = (1,2,'a','b')
print(t1[0])
```

```
print(t1[3])
print(t1[1:])
t2 = (3,4)
print(t1+t2)
print(t2*3)
```

2.7 딕셔너리 자료형

2.7.1 딕셔너리는 어떻게 만들까?

■ 딕셔너리는 키와 값의 쌍으로 구성된 자료구조이다.

```
# 딕셔너리는 어떻게 만들까?
dic = {'name':'pey','phone':'0119992323','birth':'1118'}
print(dic)
a = {'a':[1,2,3]}
print(a)

d = dict(a=1, b=3, c=5)
print(d)
print(type(d))
```

2.7.2 딕셔너리 쌍 추가, 삭제하기

```
# 딕셔너리 쌍 추가, 삭제하기
a = {1:'a'}
a[2] = 'b'
print(a)
a['name'] = 'pey'
print(a)
a[3] = [1,2,3]
print(a)
del a[1]
print(a)
```

2.7.3 딕셔너리를 사용하는 방법

■ 딕셔너리의 Key로 쓸 수 있느냐 없느냐는 Key가 변하는 값인지 변하지 않는 값인지에 달려 있다. 리스트는 그 값이 변할 수 있기 때문에 Key로 쓸 수 없다.

```
# 딕셔너리를 사용하는 방법
grade = {'pey':10, 'juliet':99}
print(grade['pey'])
print(grade['juliet'])

# 딕셔너리 만들 때 주의할 사항
a = {[1,2]:'hi'}  # 리스트를 key로 사용할 수 없다.
```

2.7.4 딕셔너리 관련 함수들

■ 딕셔너리를 자유자재로 사용하기 위해 딕셔너리가 자체적으로 가지고 있는 관련 함수들을 사용한다.

```
# 딕셔너리 관련 함수들
# Key 리스트 만들기(keys)
a = {'name':'pey', 'phone':'011099924942', 'birth':'1118'}
a = a.keys()
print(a)
print(list(a))

a = {'name':'pey', 'phone':'011099924942', 'birth':'1118'}
for k in a.keys():
    print(k)

# Value 리스트 만들기(values)
print(a.values())

# Key, Value 쌍 얻기(items)
print(a.items())

# Key, Value 쌍 모두 지우기(clear)
a.clear()
print(a)

# Key로 Value 얻기(get)
a = {'name':'pey', 'phone':'011099924942', 'birth':'1118'}
print(a.get('name'))
print(a.get('phone'))

# 해당 Key가 딕셔너리 안에 있는지 조사하기(in)
a = {'name':'pey', 'phone':'011099924942', 'birth':'1118'}
print('name' in a)
print('email' in a)
```

2.8 부울 자료형

■ 부울(bool)은 참과 거짓을 나타내는 자료형으로, 사용 가능한 값은 True와 False이다.

```
isRight = False
type(isRight)
1 < 2
1 != 2
1 == 2
True and False
True & True
True or False
False | False
not True
bool(0)
bool(-1)
bool( ' test ' )
bool(None)
```

2.9 얕은 복사와 깊은 복사

■ 파이썬에서 모든 변수는 해당 값이 저장된 객체의 주소를 가지고 있다.

```
a = [1, 2, 3]
b = a  # 얕은 복사(shallow copy), 해당 객체의 주소값을 복사(참조)한다.
a[0] = 38
a
b
id(a), id(b)  # id() 함수는 객체의 고유한 값인 아이디를 반환한다.

# 리스트의 경우, a와 b가 같은 객체를 공유하지 않게 하려면 다음과 같이 강제로 복사하면 된다.
a = [1, 2, 3]
b = a[:]  # 깊은 복사(deep copy), 해당 값을 복사한 객체를 만들고 그 주소을 복사한다.
id(a), id(b)
a[0] = 38
a
b

# copy 모듈 이용
from copy import copy
a = [1, 2, 3]
b = copy(a)  # 깊은 복사(deep copy), b = a[:]
a[0] = 38
a
b
```

2.10 요약

■ 변수: 변화는 값의 저장공간
■ 자료형: 변수에 저장되는 자료의 형태
■ 자료형의 종류
- 숫자: 정수, 실수, 8진수/16진수
- 문자열: "", '', """ """", ''' '''
- 리스트: [], 인덱싱으로 순서가 있다. 중복 O
- 세트: {}, 인덱스가 없어 순서가 없다. 중복 X
- 튜플: (), 리스트구조와 같으나 추가/삭제가 불가능
- 딕셔너리: {key:value}, 인덱스가 없어 순서가 없다. key는 중복X, valeu는 중복O
- 부울: True/False

[과제] 연습문제

Q1 홍길동씨의 주민등록번호는 881120-1068234이다. 홍길동씨의 주민등록번호를 연월일(YYYYMMDD) 부분과 그 뒤의 숫자 부분으로 나누어 출력해 보자.

```
# Q3
pin = '881120-1068234'
yyyymmdd =
num =
print(              )
print(              )
```

```
# 결과값
881120
1068234
```

Q2 주민등록번호 뒷자리의 맨 첫 번째 숫자는 성별을 나타낸다. 주민등록번호에서 성별을 나타내는 숫자를 출력해 보자.

```
# Q4
pin = "881120-1068234"
print(       )

# 결과값
1
```

Q3 [1,3,5,4,2]라는 리스트를 [5,4,3,2,1]로 만들어 보자.

```
# Q6
a = [1,3,5,4,2]
a.
a.
print(             )

# 결과값
# [5, 4, 3, 2, 1]
```

Q4 ['Life','is','too','short']라는 리스트를 Life is too short 라는 문자열로 만들어 출력해 보자.

```
# Q7
a = ['Life','is','too','short']
result =
print(         )
# 결과값
# Life is too short
```

Q5 (1, 2, 3) 튜플에 값 4 를 추가하여 (1, 2, 3, 4)를 만들어 출력해 보자.

```
a = (1, 2, 3}
a =
print(a)  # (1, 2, 3, 4)
```

Q6 다음과 같은 딕셔너리가 a 가 있다.

```
a = dict()
print(a)
{}

다음중 오류가 발생하는 경우를 고르고, 그 이유를 설명해 보자.
```

```
(1) a['name'] = 'python'
(2) a[('a',)] = 'python'
(3) a[[1]] = 'python'
(4) a[250] = 'python'
```

Q7 딕셔너리 a 에서 'B'에 해당되는 값을 추출해 보자.

```
a = {'A':90, 'B':80, 'C':70}
result =
print(a)       # {'A':90, 'C':70}
print(result)  # 80 출력
```

Q8 a 리스트에서 중복 숫자를 제거해 보자.

```
a = [1, 1, 1, 2, 2, 3, 3, 3, 4, 4, 5]
aSet =
b =
print(b)  # [1, 2, 3, 4, 5] 출력
```

Q9 파이썬은 다음처럼 동일한 값에 여러 개의 변수를 선언할 수 있다. 다음과 같이 a, b 변수를 선언한 후 a 의 첫 번째 요소값을 변경하면 b 값은 어떻게 될까? 그리고 이런 결과가 나오는 이유에 대해 설명해 보자.

```
a = b = [1, 2, 3]
a[1] = 4
print(a)
print(b)
```