2018년도 가을학기 SCSC 알고리즘 실습 2주차 부록

Data types, Flow control, Function, Class

문현수, munhyunsu@cs-cnu.org

Thursday September 13, 2018

1 개요

많은 프로그래밍 언어는 자료형, 함수, 그리고 클래스라는 개념을 채용하고 있다. 이중 자료형은 사용자가 원하는 동작을 하도록 컴퓨터에게 '명시적으로' 정보를 주는 것이며 함수와 클래스는 개발자가 중복된 코드를 하지 않도록 도와준다. 새로운 언어를 이용해 개발을 시작하기에 앞서서 자료형의 종류와 함수, 클래스의 지원 여부를 파악해야 머릿속에 있는 생각을 좀 더 쉽게 표현할 수 있다. 따라서 우리는 알고리즘 실습 2주차 학습으로 '자료형, 함수, 그리고 클래스'를 다룬다.

프로그래머는 각종 공식 문서와 친해져야 한다. JAVA(`https://docs.oracle.com/javase/8/docs/api/`)와 같은 프로그래밍 언어뿐 아니라 Android(`https://developer.android.com/docs/`), Hadoop(`https://hadoop.apache.org/docs/current/`)과 같은 OS, 프레임워크에 대한 공식 문서가 존재한다. 시중에 출판된 책은 독자가 이해하기 편하게 정리되어 있지만 출판과정의 한계로 최신 정보가 누락되기도 한다. 하지만 공식 문서는 언어, OS, 프레임워크 개발자들이 직접 작성하는 문서이기 때문에 최신 정보와 활용법이 포함된다. 이번 실습 2주차에서 다룰 '자료형, 함수, 그리고 클래스'에 대한 더 깊고 자세한 내용이 궁금하다면 Python 3 공식 문서(`https://docs.python.org/3/`)를 참고해야한다. 특히 Python 3 Standard Library는 앞으로의 실습(2주차 이후)에서도 계속 언급할 것이므로 즐겨찾기에 추가해두고 참고하면 유용할 것이다.

2 Data types

C, JAVA와 같은 프로그래밍 언어와 마찬가지로 Python 3에도 다양한 자료형이 있다. Standard Library에서 Python 3의 모든 자료형을 확인할 수 있다(2018-09-13 기준 4장). 지원하는 자료형 중에서 자주 사용하는 것은 50%가 되지 않으므로 우리는 자주 사용하는 것만 다룬다.

2.1 Basic Data Types

Python 3 Standard Library에는 Basic data types라는 분류가 없지만 이해의 편의를 위해 Basic data types와 Advanced data types로 구분하여 설명한다. Basic data types에는 참/거짓을 나타내는 Boolean, 문자열을 나타내는 String, 숫자를 나타내는 Numeric이 있다. 각각의 data types마다 사용할 수 있는 연산자, 멤버 함수가 정해져있다.

Boolean types은 참을 나타내는 'True'와 거짓을 나타내는 'False' 2개의 값 중 하나를 가진다. 첫번째 문자가 대문자이므로 다른 언어들에서 표현하는 방법과 헷갈리지 않도록 주의한다. 만약 PyCharm과 같은 IDE를 사용한다면 굵은 글씨(Bold)로 표현되어 제대로 입력했는지 시각적으로 확인할 수 있다. Boolean types에는 3가지 연산자가 있다.

- True **and** True == True
- True **and** False == False
- False **and** False == False
- True **or** True == True
- True **or** False == True
- False **or** False == False
- **not** True == False
- **not** False == True

String types는 문자열을 나타낸다. C나 JAVA와 다르게 Python 3에서는 '문자'형은 없이(Char) '문자열'만 주로 사용한다. Python 3에서 문자열 표현 방법이 3가지나 있으므로 아래 예제를 통하여 확인한다.

Listing 1: Python 3 문자열 예제

```
1  text1 = ' 홑따옴표로 ␣ 묶으면 ␣ 문자열 '
2  text2 = " 쌍따옴표로 ␣ 묶어도 ␣' 문자열 '"
3  text3 = ''' 홑따옴표 3 개로 묶어도
4  "문"
5  "자"
6  "열" '''
7  text4 = """ 쌍따옴표 3 개로 묶어도
8  '문' '자' '열' """
9
10 print(text1)
11 print(text2)
12 print(text3)
```

```
13 print(text4)
```

Numetic types은 정수형을 나타내는 'int'형과 소수형을 나타내는 'float'형이 있다. `int()`와 `float()` 명령어로 자료형 변환이 가능하다. `input()`은 사용자에게 표준 입력을 기다리는 명령어로 사용자 입력값을 '문자열' 형태로 반환한다. 아래 예제는 자료형을 확인하는 방법과 소수형, 정수형으로 변환하는 예제다.

Listing 2: Python 3 숫자 예제

```
1  user_num = input(' 숫자를 ␣  입력하세요  .␣')
2  print(' 입력한 ␣ 값의 ␣ 자료형 :␣{0}'.format(type(user_num)))
3
4  float_num = float(user_num)
5  print(' 변환된 ␣ 값 :␣{0}'.format(float_num))
6  print(' 변환된 ␣ 값의 ␣ 자료형 :␣{0}'.format(type(float_num)))
7
8  int_num = int(float_num)
9  print(' 변환된 ␣ 값 :␣{0}'.format(int_num))
10 print(' 변환된 ␣ 값의 ␣ 자료형 :␣{0}'.format(type(int_num)))
```

2.2 Advanced Data Types

프로그래밍을 할 때 자료를 이해하기 쉽고, 사용하기 편하게 저장해두어야 효율적이다. 따라서 자료를 구조적으로 저장하기 위하여 Sequence, Set, Mapping types를 만들었다. 각 자료형 중에서 `list()`, `dict()`가 굉장히 많이 사용되므로 꼭 기억해두어야 한다.

Sequence types는 순서가 있으며 중복도 허용되는 자료형들을 말한다. 대표적으로 `list()`, `tuple()`, `range()`가 있다. 이중에서 우리가 가장 많이 활용하는 것은 `list()`이므로 아래 예제를 통하여 사용법을 익힌다.

Listing 3: Python 3 list() 예제

```
1  var_list = list()
2  print( var_list )
3  var_list .append(' 문자열 ')
4  print( var_list )
5  var_list .append(913)
6  print( var_list )
7  var_list .append(' 문자열 ')
8  print( var_list )
9  var_list .append(True)
10 print( var_list )
11 var_list .pop(0)
```

```
12 print( var_list )
13  var_list .remove(True)
14 print( var_list )
```

Set types는 순서도, 중복도 허용하지 않는 자료형이다. 아래 예제를 통하여 익혀보고 `list()`와 비교하여 기억해두어야 한다.

Listing 4: Python 3 list() 예제

```
1  var_set  = set()
2  print(var_set)
3  var_set .add(' 문자열  ')
4  print(var_set)
5  var_set .add(913)
6  print(var_set)
7  var_set .add(' 문자열  ')
8  print(var_set)
9  var_set .add(True)
10 print(var_set)
11 var_set .remove(True)
12 print(var_set)
```

Mapping types는 Python 3의 가장 유용한 자료형으로 Key-Value 형태로 자료를 저장한다. 자료를 불러오거나 저장할 때 Key를 지정한다. Key는 Basic data types과 몇몇 Advanced data types이 가능하다. 아래 예제로 다루어본다.

Listing 5: Python 3 list() 예제

```
1  var_dict  = dict()
2  var_dict ['Key'] = 'Value'
3  print(var_dict)
4  var_dict [9]  = 13
5  print(var_dict)
6  var_dict [(4,  2,  1)]  = {'키 - 튜플 ',  ' 밸류 -셋'}
7  print(var_dict)
8  var_dict .pop('Key')
9  print(var_dict)
```

3 Flow control

Python 3 소스코드는 위에서 아래로 읽고 실행한다. 하지만 몇몇 조건에서는 기록되어있는 소스코드를 건너뛰거나 소스코드 구간을 반복할 필요가 있다. 이렇게 소스코드 실행 흐름을 제어하는 것을 'Flow control'이라고 한다. Flow control을

위한 키워드는 `if`, `while`, `for` 등이 있다. 본래 Flow control은 많은 시간과 지면을 할애하여 설명하여야 하지만 우리가 학습하는 알고리즘은 기본적인 프로그래밍을 배웠다고 간주하기 때문에 간단한 예제만을 보인다.

Listing 6: Python 3 if 예제

```
num = 43
if num % 2 == 0:
    print(' 짝수 ')
elif num % 2 == 1:
    print(' 홀수 ')
else:
    print(' 소수 ')
```

Listing 7: Python 3 while 예제

```
num = 59
while num > 0:
    print(' 현재 ␣ 숫자 ␣{0}!␣2 로 ␣ 나눕니다 .'.format(num))
    num = num // 2
```

Listing 8: Python 3 for 예제

```
for index in range(0, 10):
    print('range() 를 ␣ 활용한 ␣index!␣{0}'.format(index))

var_list = [' 이렇게 ', ' 활용할 ', ' 수 ', ' 있음 !', ' 편리함 ']
for var in var_list :
    print(var)
```

4 Function and Class

프로그래밍이 발전하며 코드 중복을 없애기 위한 다양한 개념이 등장했다. 그 중 가장 대표적인 것이 함수(Function)와 클래스(Class)다. Flow control과 마찬가지로 기초 예제를 통하여 학습한다.

Listing 9: Python 3 Function 예제

```
def say_hello ():
    print(' 안녕 !! ')

say_hello ()
```

Listing 10: Python 3 Class 예제

```
1  class Greeter(object):
2      def __init__(self, lan):
3          self.lan = lan
4
5      def say_hello(self):
6          if self.lan == 'en':
7              print('Hello!')
8          elif self.lan == 'kr':
9              print(' 안녕 !')
10         else:
11             print('Hi')
12
13 greeter = Greeter('en')
14 greeter.say_hello()
```

5 과제

이것은 부록이므로 과제가 없다.

5.1 조교 연락처

- 문현수
- munhyunsu@cs-cnu.org
- 공학5호관 633호 데이터네트워크연구실
- 이메일, 연구실 방문