# 2018년도 가을학기 SCSC 알고리즘 실습 1주차

문현수, munhyunsu@cs-cnu.org

Thursday September 6, 2018

## 1 개요

2018년 많은 유명 개발자들이 프로그래머가 배워야할 프로그래밍언어 1순위로 Python을 선택하고 있다. Python은 1991년 귀도 반 로섬(Guido van Rossum)이 발표한 언어로 C나 JAVA와 다르게 인터프리터형 언어이다. Python의 핵심 철학은 아래와 같으며 이는 20개의 철학 중 5개만 선별한 것이다(Python에서 `import this`를 통해 철학을 모두 볼 수 있다.).

- 아름다운 것이 추한 것 보다 낫다.
- 명시적인 것이 암시적인 것보다 낫다.
- 단순함이 복잡함보다 낫다.
- 복잡함이 난해한 것보다 낫다.
- 가독성은 중요하다.

위의 철학을 기반으로 개발된 Python은 특유의 생산성 덕분에 사업, 과학, 공학 분야에서 많이 사용되기 시작하였다. 특히 Python기반의 NumPy, Jupyter Notebook, tensorflow가 데이터 분석을 위한 필수적인 도구로 자리잡았다. 이러한 도구를 제대로 활용하기 위해서는 문제 해결을 위한 알고리즘 이해가 필요하다. 언어의 단순함 때문에 Python은 알고리즘을 학습하고 활용하는데에 있어서 본질에 집중할 수 있게 해준다.

앞으로 우리는 Python을 활용하여 알고리즘을 학습한다. 자료구조를 활용한 간단한 문제 해결부터 Graph, Tree, Sort, Search에 대한 대표적인 알고리즘을 배운다. 또한 Dynamic Programming과 같은 빠른 문제 해결을 위한 방법도 다룬다.

이번 실습 1주차에서는 알고리즘 학습을 위한 Python 개발환경을 세팅하고, 프로그래밍 결과가 정확한지 테스트하는 방법을 배운다. 실습을 통하여 학습자는 Python 인터프리터와 코딩에 대하여 이해하게 된다. 또한, 실습 및 과제로 개발한 프로그램의 Output을 테스트하는 방법을 학습한다.

# 2 Python 개발환경

Python은 Python 홈페이지(`https://www.python.org/`)에서 다운받을 수 있다. 주로 Windows 64bit 설치 파일을 다운로드하면 되지만 학습자 각자의 OS 환경에 따라서 다를 수 있다.

## 2.1 환경 변수(Environment variable)

우리는 어떠한 물건을 찾을 때 품목에 따라서 방문하는 장소가 다르다. 예를 들어, 책을 찾을 때에는 도서관이나 서점을 방문하며 컴퓨터/노트북을 살 때에는 전자매장을 방문한다. 컴퓨터도 마찬가지로 명령어, 파일 등을 찾을 때에 방문하는 장소가 정해져있다. 이러한 장소를 '환경 변수'라고 부른다. 예를 들어, Windows에 'notepad를 실행'이란 명령을 보내면 Windows는 PATH라는 환경 변수가 가르키고 있는 장소에서 찾아본다.

Python을 설치할 때에는 환경 변수를 설정하는 것이 편리하다. Python 환경 변수를 설정해두면 Windows와 같은 OS에 'python 실행'이라고 명령을 내렸을 때 에러를 표시하지 않고 정상적으로 실행할 것이다. Windows에서는 '제어판-시스템'에서 환경 변수를 설정할 수 있으며 Linux와 같은 환경에서는 PATH 변수를 설정해주면 된다. 환경 변수 중 PATH는 명령어를 찾는 장소를 가르킨다.

수동으로 환경 변수를 설정하는 것이 복잡하거나 어렵다면 Python을 설치할 때 설치 프로그램에게 환경 변수를 설정해달라 부탁할 수 있다. 특히, Windows Python 설치 프로그램을 실행시키면 첫화면에 'Add Python3.x to PATH'라는 옵션이 보일 것이다. 이 옵션은 설치 프로그램이 Python을 PATH라는 환경 변수에 추가하도록 지시한다.

## 2.2 IDE(Integrated Development Environment)

프로그래밍은 메모장과 같은 텍스트 에디터에서도 할 수 있지만 통합 개발 환경(IDE)을 활용하면 편리하다. 우리가 문서를 편집할 때에는 한글(hwp)이나 워드(doc)를 사용하는 것처럼 IDE는 프로그래밍을 할 때 사용한다. IDE를 사용하면 하나의 프로그램에서 코딩과 실행, 디버깅을 모두 할 수 있으며 여러개의 소스코드를 프로젝트(Project)라는 개념으로 쉽게 다룰 수 있다. 따라서 우리는 Python 코딩을 도와줄 IDE를 설치한다.

Python을 위한 대표적인 IDE는 PyCharm, Atom, Jupyter Notebook이 있다. 알고리즘을 학습하는 목적에 가장 알맞은 IDE는 PyCharm이다. PyCharm은 JetBrain이라는 회사에서 개발, 유지보수하는 프로그램으로 유료버전(Professional)과 무료버전(Community)이 있다. 우리는 유료버전의 기능이 필요없으므로 무료버전을 설치한다. PyCharm 설치 프로그램은 PyCharm 홈페이지(`https://www.jetbrains.com/pycharm/`)에서 다운로드 받을 수 있다.

PyCharm을 설치한 후 테마와 같은(요즈음 많은 프로그래머는 Dacular 테마를 사용한다.) 몇가지 설정을 하고나면 프로젝트를 생성하는 화면이 나온다. 프로젝트

생성 메뉴가 있는 화면이 보이면 제대로 설치가 된 것이다.

### 2.3 프로젝트(Project)와 소스코드(Source code)

대부분의 프로그램은 하나 이상의 소스코드가 모여 구성된다. 이 때 하나의 프로그램을 위하여 여러개의 소스코드를 모아둔 것을 프로젝트라고 부른다. 따라서 소스코드를 생성하기에 앞서 프로젝트를 생성해야한다. PyCharm에서 'Create Project'를 눌러 프로젝트를 생성하면 소스코드가 없는 빈 프로젝트가 보일 것이다.

PyCharm 왼편의 Navigator에서 프로젝트 구조를 볼 수 있다. 생성한 프로젝트를 마우스 오른쪽키로 누르면 'New-Python File'로 Python 소스코드를 생성할 수 있다. 원하는 이름을 입력해도 되지만 이후 실습을 위하여 greeting을 입력한다. 우리는 이 소스코드를 중심으로 이번 실습을 진행할 것이다.

## 3 unittest

현대 프로그램 개발 문화의 중심에는 TDD(Test Driven Development)가 있다. TDD는 프로그램을 개발할 때에 '테스트 코드'를 작성해두고 이 테스트 코드를 '통과'하는 함수나 클래스를 만드는 패러다임을 말한다. 테스트를 통과한다는 것은 함수나 클래스가 '의도한 동작'을 한다는 것이므로 통과한 함수나 클래스를 활용한 프로그램도 원하는대로 동작함을 의미한다. 거대화되고 있는 현대 프로그램에서 TDD는 오류없는 프로그램 개발을 위한 필수 요소가 되었다.

TDD로 개발하기 위해서는 대상 함수나 클래스를 테스트하는 'unittest'가 필요하다. JAVA에서는 JUnit이 대표적이며 Python에서도 unittest라는 라이브러리(도구)가 제공된다. unittest는 함수나 클래스를 호출하며 Input에 대한 Output을 확인하고 그것이 기대한 값과 같은지 점검한다. 테스트가 끝나면 성공한 테스트와 실패한 테스트를 개발자에게 보여준다. 알고리즘을 학습하는 우리는 테스트코드를 기반으로 자신의 코드가 '정상 동작하는지' 확인할 것이다.

Listing 1: Python unittest 결과 예제

```
test_func_a (__main__.TestFunc) ... ok
test_func_b (__main__.TestFunc) ... ok

----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK
```

위의 결과는 func_a와 func_b라는 함수 2개를 unittest한 결과이다. 만일 테스트가 통과하지 못하면 failed 메시지가 출력되며 왜 통과하지 못했는지 기대값과 결과값을 보여줄 것이다.

## 3.1 Greeting Program

우리는 알고리즘 학습의 첫번째 실습으로서 unittest를 통한 개발을 해볼 것이다. 대부분의 프로그래머는 처음 코딩을 배울 때 컴퓨터를 통해 세상에게 인사하고 있다. 우리도 다양한 언어로 세상에게 인사하는 프로그램을 개발한다. 이번 실습에서 제작한 프로그램(greeting.py)의 기능은 아래와 같다.

1. hello_en() 함수를 호출하면 'Hello, World!'라고 반환한다.
2. hello_kr() 함수를 호출하면 '안녕하세요!'라고 반환한다.

이 프로그램은 사용자의 언어에 따라서 `hello_en()` 혹은 `hello_kr()`을 호출한다. 프로그램을 통해 인사말을 반환받으면 `print()`와 같은 명령을 통해 출력할 수 있을 것이다. 지금 우리는 출력은 고려하지 않고 반환되는 인사말에 집중한다.

## 3.2 Greeting Program Test

앞으로 개발한 greeting 프로그램이 정상적으로 동작하는지 확인할 수 있는 테스트 코드를 구성한다. greeting 프로그램이 가지고 있어야할 함수는 `hello_en()`과 `hello_kr()`이므로 이것을 모두 테스트 한다. 두개의 함수 모두 입력값(인자)은 없으며 함수 별로 요구되는 출력값이 다르므로 2번의 테스트를 거친다.

Listing 2: Greeting 프로그램 테스트코드

```
1  import unittest
2
3  import greeting
4
5  class GreetingTest(unittest.TestCase):
6      def test_greeting_en(self):
7          self.assertEqual('Hello, World!',
8                           greeting.hello_en())
9
10     def test_greeting_kr(self):
11         self.assertEqual('  안녕하세요  !',
12                          greeting.hello_kr())
13
14
15 if __name__ == '__main__':
16     unittest.main(verbosity=2)
```

위의 테스트 코드는 greeting.py 안에 구현된 hello_en()함수와 hello_kr()함수의 반환값을 테스트한다. 코드를 입력할 때 :이후에는 공백 4번으로 코드 블락(Code block)을 지정하는 것에 유의해야한다. 예를 들어, 4번 라인 끝에 붙은 :

는 GreetingTest클래스가 4번 라인부터 12번 라인(공백이 없음)까지의 내용으로 구현됨을 표시한다. greeting.py를 구현하지 않고 테스트 코드를 실행시키면 당연하게도 테스트를 통과하지 못 한다.

Listing 3: greeting.py에 아무런 내용이 없는 상태로 테스트한 결과

```
...
AttributeError: module 'greeting' has no attribute '
    hello_en'
...
AttributeError: module 'greeting' has no attribute '
    hello_kr'
...
Ran 2 tests in 0.003s

FAILED (errors=2)
```

위의 오류는 'hello_en()'과 'hello_kr()'이 greeting.py에 없다는 이야기다. greeting.py에 아무런 코드도 입력하지 않았으니 테스트코드는 해당 함수 찾을 수 없다. 이처럼 unittest기반 프로그램 구현은 테스트코드를 우선 작성한 후 목표 프로그램을 제작함으로써 이루어진다.

### 3.3 Python Function return

테스트코드가 준비되었으니 우리가 목표로하는 프로그램을 구현하려면 함수를 만들어야 한다. 프로그램의 목표가'hello_en()함수와 hello_kr()함수를 호출했을 경우 각각의 인사말이 반환될 것'이므로 2개의 함수를 만든다.

Listing 4: greeting.py 함수

```
1 def hello_en():
2     pass
3
4
5 def hello_kr():
6     pass
```

Python에서 def 키워드는 함수/멤버함수를 정의할 때 사용한다. 위의 코드를 구현한 후 테스트를 수행시켜보면 아까와는 다른 오류가 나타난다. 오류 내용에서 알 수 있듯 함수를 수행시켰으나 기대한 값과 실제 결과값이 다르다. 지금까지 구현한 버전의 함수에서 `pass`는 아무런 연산을 하지 않고 넘어가는 것을 의미한다. 따라서 반환값이 없기 때문에 '무'를 의미하는 `None`이 반환되었다.

Listing 5: greeting.py에 함수를 정의만하고 테스트한 결과

```
Ran 2 tests in 0.003s

FAILED (errors=2)
 ...
None != Hello, World!

Expected :Hello, World!
Actual   :None
 ...
None !=  안녕하세요 !

Expected :  안녕하세요 !
Actual   :None
 ...
```

return을 이용하여 함수의 반환값을 지정할 수 있다. hello_en()의 반환값은 'Hello, World!'이므로 2번 라인처럼 코딩할 수 있다. 이제 테스트코드는 원하는 반환값을 받을 수 있을 것이다.

Listing 6: greeting.py hello_en() 함수

```
1 def hello_en():
2     return 'Hello, World!'
```

구현한 후 테스트 다시 수행해보면 hello_en()에 대한 테스트는 통과된다.

Listing 7: greeting.py에 hello_en()을 제대로 구현한 후 실행한 결과

```
...
Ran 2 tests in 0.003s

FAILED (failures=1)
...
```

# 4 과제

실습에서 우리는 hello_en() 함수만 구현하였다. 이번 실습의 과제는 아직 구현하지 않은 hello_kr() 함수를 구현하는 것이다. 또한 작성되어 있는 테스트 코드를 수정하여 새로운 언어(일본어, 독일어, 중국어 등)의 인삿말에 대한 테스트, 구현도 하길 바란다.

## 4.1 과제 목표

- hello_kr() 구현

- 새로운 인사말에 대한 테스트 코드 수정 및 greeting.py 구현

### 4.2 제출 관련

- 마감 날짜: 2018. 09. 12. 18:00 (00시가 아님!!)
- 딜레이: 1일당 20% 감점
- 제출 방법: 과목 사이버캠퍼스
- 제출 형식: 과제 리포트 PDF(HWP, DOC 받지 않음!), 소스코드(구현한 .py 만 추가할 것)
- 리포트에 포함해야하는 내용: 목표, 목표를 위해 알아야하는 것, 해결 방법, 결과, (선택)느낀점 or전달할 말
- 제출 파일 제목: AL_201550320_문현수_01.pdf, AL_201550320_문현수_01.zip(파일명 준수!)

### 4.3 조교 연락처


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