ILE3-003 Python 기초 #1

Python

Python의 특징

Platform 독립적인 언어

Windows / Linux / MacOS 등 OS에 종속되지 않음

> C/C++의 경우 Cross Compiler를 사용하여 각 Platform에 맞는 build를 만들지 않는 이상, 해당 Platform에서만 작동

인터프리터 언어

Compile 과정이 필요 없음

> Python
> `$ python main.py`

> C++
> `$ g++ main.cpp`
> `$ ./a.out`

실시간으로 대화형으로 개발 가능

인터프리터 언어의 특징

```
$ python
Python 3.8.4 (tags/v3.8.4:dfa645a, Jul 13 2020, 16:30:28) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 'hello world'
>>> print(a)
'hello world'
```

변수 타입 선언이 암시적

> python
> `a = 0`

> C++
> `int a = 0;`

메모리 자동 관리

C/C++과 같이 `new`, `delete`를 Gabage Collector가 자동으로 처리 해 줌

코드 블럭을 Space로 관리

C/C++, C#, JAVA 등의 언어는 중괄호를 이용하여 코드 블럭을 관리하나, Python은 Space로만 구분
C, C++, Java

```
int foo(int x)
{
    ...
    return y;
```

```
}

```

Python

```python
def foo(x):
    ...
    return y
```

변수 선언

변수의 Type은 우측 값에 따라 암시적으로 설정 됨

```python
a = 1        #정수
b = 1.2      #소수
c = 1+2j     #복소수
d = 'A'      #문자열
e = 'AAAAAA' #문자열
```

TypeCasting

자료형의 변환이 필요 할 때

```python
a = '1'         #정수 문자열
b = '123.456'   #소수 문자열
c = int(a)      #정수로 변환
d = float(b)    #소수로 변환
e = int(b)      #소수를 정수로 변환 시도 -> ValueError 발생
f = float(a)    #정수 문자열을 소수로 변환은 가능
g = str(c)      #정수를 문자열로 변환 (1 -> '1')
h = str(d)      #소수를 문자열로 변환 (123.456 -> '123.456')
```

Console을 통한 입출력

입력

사용자가 Enter를 입력할 때까지의 내용을 `a`에 저장

```python
a = input()  #문자열 형식으로 입력 받음
```

출력

```python
print('HELLO WORLD')
print('HELLO' + ' ' + 'WORLD')
```

조건문

C/C++과 유사하나 `if` - `elif` - `else` 형태로 구성 됨
각 조건의 마지막에는 `:`을 입력

```python
a = 0
if a == 1:
    print('a is 1')
elif a == 2:
    print('a is 2')

```

```
else:
    print('a is not 1,2')
```

반복문

While

C/C++과 동일한 `while` 구문을 사용하며, 동일하게 `break`문으로 탈출 가능

```
num = 0
while num < 5:
    print(num)
    num = num + 1
    if num >= 2:
        break;
```

For

C/C++의 일반적인 For문(초기 구문, 조건 구문, 증가 구문)과는 다른 형태를 지니므로 사용 시 조심 할 것
`foreach`문과 유사

```
for i in range(0, 2):   # range(0,2) -> 0부터 시작하여 2개의 값을 1씩 증가하여 반환 (0, 1)
    print(i)
```

파일 입출력

Text Mode / Binary Mode

OS 별로 문자열 표현 방법 차이가 존재

```
Linux:   abcdefghijk\n
Windows: abcdefghijk\r\n
```

- Binary Mode
  Memory에 있는 데이터 그대로 기록
- Text Mode
  줄바꿈 문자 및 파일 끝에 대한 변환 OS에 따라 수행 후 기록

File Open

C/C++의 `fopen`과 유사

```
f = open('test.txt', 'rt')
# f = open('test.txt', mode='rt', encoding='utf-8')
f.close()
```

> File Open시| Option
> [OPEN_MODE][TEXT/BINARY_MODE]

Read

기본적으로 `readline()`, `readlines()` 함수를 제공하여 Text File의 읽기를 쉽게 수행할 수 있음

Example File: test.txt

```
Hello, World
안녕하세요?
```

```
>>> f = open('test.txt', 'rt')
>>> read = f.readline()
>>> print(read)
Hello, World
>>> read = f.readline()
>>> print(read)
안녕하세요?
>>> f.close()
```

```
>>> f = open('test.txt', 'rt')
>>> read = f.readlines()
>>> print(read)
['Hello, World\n', '안녕하세요?']
>>> f.close()
```

Write

`write()` 함수를 이용하여 Text File의 쓰기를 수행
`write()` 수행 시 줄바꿈 문자는 포함 되지 않음

> write의 경우 문자열을 입력으로 받기 때문에 다른 Type의 변수는 숫자로 변경을 수행하여야 함

```
f = open('test.txt', 'w')
f.write('안녕하세요!')
f.write('Hello World')
f.close()
```

> 위 Python Code의 출력파일의 내용
> `안녕하세요!Hello World`

```
f = open('test.txt', 'w')
f.write('안녕하세요!\n')
f.write('Hello World')
f.close()
```

> 위 Python Code의 출력파일의 내용
> `안녕하세요!`
> `Hello World`

과제

입력받은 데이터를 읽어 아래의 값을 확인하여 아래의 형태로 'result.txt'로 출력

- 30000 이하의 값 개수
- 최소값
- 최대값
- 평균

Input Example

```
17425
10383
62468
44527
23798
```

```
5297
80990
97429
66495
30803
88697
22001
64016
84255
54334
92095
...
```

Output Example

```
30000 이하의 값: 243개
최소값: 12
최대값: 99234
평균: 3453.234643
```