April 7th, 2020
Programming Practice 2

Practice 1

Note

✓ How to evaluate submissions

Until April 17th, it is a trial period for a new teaching method, so the submitted program will not be evaluated. Evaluation methods after April 21st will be posted by April 21st.

✓ How to attend the class “Programming Practice 2” until April 28th

The opening period of manaba+R is about one day for the reasons described in the “Submission deadline”.

Please start studying at 16:20 on April 7th according to this document. It is not for learning from the next day (April 8th).

✓ Submission deadline

Traditionally, students should submit their programs during class hours. However, real-time questions and answers are difficult because face-to-face classes are not held until May 1st. It is expected that questions and answers will take a long time.

The total response time is one day in order to answer questions as follows:

① Email the question to Ogawa (ogawa@is.ritsumei.ac.jp)
Please email the question by 16:00 on April 8th, taking into account the response time to the question.
Do not use “Threads” to send the question.

② The organized questions and their answers will be posted on “Treads” of “Programming Practices 2” course, manaba+R.

In this way, appropriate information can be transmitted to everyone.

The data structures and algorithms learned in the class “Data Structures and Algorithms” are given in Python. However, it is necessary to use the knowledge to create programs in Java. Therefore, a program written in Python will be rewritten in Java in today’s Practice.

1. An example of rewriting

A program written in Python that measures the processing time of the algorithm in which max value of array “data” is obtained in Fig. 1. This program runs the algorithm 100,000 times and displays the contents of “data”, the result and its execution time shown in Fig. 2.

See “Library Reference” on the web page “https://docs.python.org/3/” for details of the Python program.

```
from time import time
data = [2, 3, 4, 5, 6, 3, 5, 3, 6, 5, 8, 9, 4]

def get_max(S):
    n = len(S)
    A = 0
    for i in range(n):
        if S[i] > A:
            A = S[i]
    return A

if __name__ == "__main__":
    start_time = time()
    for i in range(100000):
        result = get_max(data)
    end_time = time()
    elapsed = end_time - start_time
    print('data is', data)
    print('The maximum value is', result)
    print('The execution time is', elapsed, 'seconds')
```

Fig. 1 Program to obtain max value written in Python.

```
data is [2, 3, 4, 5, 6, 3, 5, 3, 6, 5, 8, 9, 4]
The maximum value is 9
The execution time is 0.13971209526062012 seconds
```

Fig. 2 Result of program shown in Fig. 1.

When creating a Java program, we need to consider the following:

(1) The program is saved in the file named same as the class name.
(2) The main method is necessary to execute the program. In the main method, it should be limited to creating a instance of its own class and calling other methods. It is better not to write the main processing.
(3) All variables must be declared in advance with their data type.
(4) The semicolon ";" is required to indicate the end of the instruction.

Fig. 3 shows a program written in Java which performs the same processing as the program of Fig. 1. The result of this Java program is shown in Fig. 4.

See the web page
https://docs.oracle.com/javase/10/docs/api/index.html?overview-summary.html
for details of the Java program.

```
/*****************************************************************
 * April 7th, 2020
 * Algorithm of obtaining max of data
 * by Ogawa
 *****************************************************************/
public class GetMax {
    int[] data = {2, 3, 4, 5, 6, 3, 5, 3, 6, 5, 8, 9, 4};
    int result;

    public static void main(String[] args) {
        GetMax gm = new GetMax();
        gm.measure();
    }

    public void measure() {
        long start, end, elapsed;

        start = System.currentTimeMillis();
        for(int i = 0; i < 100000; i++) {
            result = get_max(data);
        }
        end = System.currentTimeMillis();
        elapsed = end - start;

        System.out.print("data is [");
        for(int i =0; i < data.length - 1; i++) {
            System.out.print(data[i] + ", ");
        }
        System.out.println(data[data.length - 1] + "]");
```

Fig. 3 Program to obtain max value written in Java.

```
        System.out.println("The maximum value is " + result);
        System.out.println("The execution time is " + elapsed + "
millisecond");
    }

    public int get_max(int[] data) {
        int max = 0;

        for(int i = 0; i < data.length; i++) {
            if(data[i] > max) {
                max = data[i];
            }
        }
        return max;
    }
}
```

Fig. 3 continue.

```
data is [2, 3, 4, 5, 6, 3, 5, 3, 6, 5, 8, 9, 4]
The maximum value is 9
The execution time is 5 millisecond
```

Fig. 4 Result of program shown in Fig.3.

2. Practice

The prefix averages are represented by a sequence A such that $A[j]$ which is the average of elements $S[0]$, ..., $S[j]$, for $j = 0, .., n - 1$, that is,

$$A[j] = \frac{\sum_{i=0}^{j} S[i]}{j+1}$$

Fig. 5 shows an example of a Python program that calculates the prefix average. The result of this python program is shown in Fig. 6.

```
from time import time
data = [2, 5, 4, 5, 6, 9, 5, 3, 6, 5, 8, 3, 4]

def prefix_average(S):
    n = len(S)
    A = [0] * n
    total = 0
    for j in range(n):
        total += S[j]
        A[j] = total / (j + 1)
    return A

if __name__ == "__main__":
    start_time = time()
    for i in range(100000):
        result = prefix_average(data)
    end_time = time()
    elapsed = end_time - start_time
    print('The execution time is', elapsed, 'seconds')
    for i in range(len(result)):
        print('result[', i, '] is', result[i])
```

Fig. 5 Program of the prefix average written in Python.

```
The execution time is 0.2257688045501709 seconds
result[ 0 ] is 2.0
result[ 1 ] is 3.5
result[ 2 ] is 3.6666666666666665
result[ 3 ] is 4.0
result[ 4 ] is 4.4
result[ 5 ] is 5.166666666666667
result[ 6 ] is 5.142857142857143
result[ 7 ] is 4.875
result[ 8 ] is 5.0
result[ 9 ] is 5.0
result[ 10 ] is 5.2727272727272725
result[ 11 ] is 5.083333333333333
result[ 12 ] is 5.0
```

Fig. 6 Result of program shown in Fig. 5.

Create a Java program (file name is “Prefix_Average.java”) to execute algorithm for computing prefix averages described as the prefix_average (in Fig. 5) 100,000 times and output the execution time and processing result. (Refer source the program in Fig. 3)

Use data as follows:

```
int[] data = {2, 5, 4, 5, 6, 9, 5, 3, 6, 5, 8, 3, 4};
```

The expected output is shown in Fig. 7.

```
The execution time is 11 millisecond
result[0] is 2.0
result[1] is 3.5
result[2] is 3.6666666666666665
result[3] is 4.0
result[4] is 4.4
result[5] is 5.166666666666667
result[6] is 5.142857142857143
result[7] is 4.875
result[8] is 5.0
result[9] is 5.0
result[10] is 5.2727272727272725
result[11] is 5.083333333333333
result[12] is 5.0
```

Fig. 7 The expected output of “Prefix_Average.java”.

Submit the program “Prefix_Average.java” to Title “Practice 1” in Assignments page of “Programming Practice 2” course, manaba+R until 18:00 on April 8th, 2020.

The program should include the following information in the first part.

- ✓ Contents of the program
- ✓ Submission date
- ✓ Program creator