The Transactions of the Korea Information Processing Society
TKIPS Vol.13, No.9, pp.437~443
eISSN: 3022-7011
https://doi.org/10.3745/TKIPS.2024.13.9.437

Implementation of a Coding Style Checking System in an Online Judge System

Yeonghun Kim† · Junseok Cheon†† · Gyun Woo†††

ABSTRACT

Adhering to coding style guidelines is crucial for both companies and developers as it improves code readability and reduces the costs associated with testing and maintenance. However, teaching coding style in programming courses poses challenges. Setting up an environment for learning coding styles is hard, and there are no predefined coding style rules for beginners. From the learners' perspective, since adherence to coding styles does not affect their grades, they do not feel a strong need to learn them. This paper introduces a coding style checking system for an online evaluation system. The proposed system is implemented to check and evaluate coding styles in C, Java, and Python. Additionally, we applied 234 out of the 1,023 rules provided by the language-specific tools, which is 23.08%, allowing for the application of coding style rules according to the course progression. Moreover, we motivated learners to improve their coding style by adding quality scores to their basic scores. After introducing the coding style education system, the number of students scoring over 25 points on their initial submissions increased by 149.47%, from 18 students in the first week to 44 students in the sixth week. Learners used the coding style checking system to learn how to apply coding style rules and subsequently implemented their code in adherence to the specified coding styles.

Keywords : Online Judge, Auto Judge System, Coding Style, Programming Education

온라인 평가 시스템에서 코딩 스타일 검사 시스템 구현

김 영 훈† · 천 준 석†† · 우 균†††

요 약

코딩 스타일을 준수하는 것은 코드의 가독성이 좋아지고, 테스트 및 유지 보수에 필요한 비용을 줄일 수 있어 기업과 개발자 모두에게 중요하다. 하지만 프로그래밍 언어 수업에서 코딩 스타일을 교육하는 데 어려움이 있다. 왜냐하면 코딩 스타일 학습을 위한 환경 구축도 힘들고, 초보자를 위한 코딩 스타일 규칙이 따로 정의되어 있지 않다. 학습자 측에서는 코딩 스타일을 준수하지 않더라도 학점에 반영되지 않으므로, 학습의 필요성이 별로 와닿지 않는다. 본 논문에서는 온라인 평가 시스템을 위한 코딩 스타일 검사 시스템을 소개한다. 제안 시스템은 C와 Java, Python 코딩 스타일을 검사하고 평가할 수 있도록 구현하였다. 또한, 언어별 도구에서 제공하는 규칙 1,023개 중 23.08%인 234개를 반영하여 수업 진도에 따라 코딩 스타일 규칙을 적용할 수 있게 하였으며, 기본 점수에 품질 점수를 추가로 부여하여 학습자들의 코딩 스타일 학습 동기를 부여하였다. 코딩 스타일 교육 시스템을 도입한 후 최초 제출 시 점수가 1주 차에서 25점 이상을 받은 학생이 18명이었지만, 6주 차에서 44명으로 149.47% 증가했다. 학습자는 코딩 스타일 검사 시스템을 이용하여 코딩 스타일 규칙을 적용하는 방법을 학습하고, 이후 코드를 구현할 때 제시한 코딩 스타일을 준수하여 코드를 구현할 수 있었다.

키워드 : 온라인 저지, 자동 채점 시스템, 코딩 스타일, 프로그래밍 교육

1. 서 론

프로그램을 구현할 때 코딩 스타일을 준수하는 것은 매우 중요하다. 코딩 스타일을 준수하면 코드의 가독성이 좋아지고, 다른 사람이 구현한 코드를 쉽게 이해할 수 있어서 프로그램의 테스트 및 유지 보수 비용을 줄일 수 있다[1-4]. 그러므로 코딩 스타일을 준수하는 것은 기업과 개발자 모두에게 중요하다.

하지만 프로그래밍 언어 수업에서 코딩 스타일을 교육하기에는 여러 어려움이 있다. 첫 번째로 코딩 스타일 학습을 위한 환경을 구축하기 어렵다. 왜냐하면 언어별 코딩 스타일 검사 도구가 많고, 도구마다 지향하는 목적이 달라서 제공하는 검사 규칙이 다르다. 그러므로 교육에서 사용할 검사 도구와 규

※ 이 과제는 부산대학교 기본연구지원사업(2년)에 의하여 연구되었음.
※ 이 논문은 2024년 ACK 2024의 일반논문으로 "Pylint를 이용한 Python 코딩 규칙 검사 시스템"의 제목으로 발표된 논문을 확장한 것임.
† 준 회 원 : 부산대학교 컴퓨터공학전공 석사
†† 준 회 원 : 부산대학교 컴퓨터공학전공 박사과정
††† 종신회원 : 부산대학교 정보컴퓨터공학부 교수
Manuscript Received : July 12, 2024
Accepted : August 21, 2024
* Corresponding Author : Gyun Woo(woogyun@pusan.ac.kr)

칙을 통일하고 어느 환경에서라도 같은 검사 결과를 제공해야 한다.

두 번째 어려움은 교육에 사용할 코딩 스타일 규칙을 선정하는 문제이다. 대부분의 프로그래밍 언어는 표준 코딩 스타일을 따로 지정해 주지 않는다. 표준 코딩 스타일이 존재한다고 하더라도 이러한 코딩 스타일 규칙을 어떤 것부터 시작하여 교육해야 하는지 판단하기 힘든 경우가 많다. 코딩 스타일의 중요성에 비해 스타일 교육 자체는 가볍게 여겨지고 있다.

세 번째 어려움은 학생들에게 코딩 스타일을 학습하기 위한 동기를 부여하기 어렵다는 점이다. 프로그래밍 언어 학습자는 코딩 스타일보다 코드의 효율성을 중시하는 경향이 있다. 제대로 동작하는 코드가 작성되어야 스타일도 생각할 여유가 있으므로 이는 매우 자연스러운 일이다. 그러나 이 과정에서 코딩 스타일을 지키지 않는 습관이 굳어져 향후 코딩 스타일을 수정하는 데 많은 시간이 소요된다[5]. 이러한 문제점을 해결하기 위해 코딩 스타일을 점수화하여 학점에 반영함으로써 학습자들의 코딩 스타일 학습 동기를 부여할 필요가 있다[6, 7].

본 논문은 이러한 문제점을 해결하기 위해 온라인 평가 시스템을 위한 코딩 스타일 검사 시스템을 제안한다. 프로그래밍 수업에서 많이 사용되는 C와 Java, Python에서 수업 진행에 따라 코딩 스타일 규칙을 선정한다. 또한, 코딩 스타일 학습 동기를 높이기 위해 품질 점수식을 구성하여 학습자가 제출한 코딩 스타일을 정량적으로 평가한다.

본 논문의 구성은 다음과 같다. 2장에서 코딩 스타일과 코딩 스타일 검사 도구, 온라인 평가 시스템에 대해 알아본 후, 3장과 4장에서 논문에 제안하는 시스템의 설계 및 구현을 설명하고, 5장에서 제안 시스템을 실제 Python 수업에서 사용한 결과를 알아본다. 6장에서 제안 시스템의 한계점을 살펴보고, 7장에서 결론짓는다.

2. 관련 연구

2.1 코딩 스타일

코딩 스타일은 프로그래밍 코드의 가독성과 일관성, 유지보수성을 높이기 위해 사용하는 규칙이다. 언어별로 여러 코딩 스타일이 있지만, 공통으로 적용되는 요소들이 있다. 들여쓰기 및 공백은 코드의 구조를 보기 좋게 하도록 스타일마다 일정한 들여쓰기와 공백을 제안하며, 일반적으로 공백 4칸 또는 8칸을 사용한다. 명명 규칙(naming rules)은 변수, 함수, 클래스 등의 명칭을 일관성 있게 이름 짓는 방법으로, 뱀체(snake case)와 낙타체(camel case), 파스칼체(pascal case) 등 다양한 방법이 있다. 코드 배치는 코드 블록 간의 간격, 한 줄에 들어가는 문자 수, 중괄호 위치 등을 정의한다. 특히, 여는 중괄호의 위치가 함수 및 제어 구조와 같은 줄에 위치하는지, 새로운 줄에 위치하는지를 정하는 규칙이 포함된다.

2.2 코딩 스타일 검사 도구

1) C의 코딩 스타일 검사 도구

C에서 사용할 수 있는 코딩 스타일 검사 도구 중에서 Cppcheck는 주로 버그나 문법 오류, 코딩 스타일 문제를 식별하는 데 목적을 두고 있다[8]. 또한, 검사 규칙 설정하고 심각도에 따라 분류하여 정보를 제공한다. PC-lint는 다양한 코딩 스타일 규칙을 제공하지만, 심각도나 규칙을 설정할 수 있는 기능을 제공하지 않는다[9]. 본 논문에는 C의 코딩 스타일 검사 도구로, 검사할 규칙을 관리자가 원하는 대로 설정할 수 있는 Cppcheck를 사용한다.

2) Java의 코딩 스타일 검사 도구

Java에서 사용할 수 있는 코딩 스타일 검사 도구로는 PMD와 Checkstyle이 있다[10, 11]. PMD는 코드 스멜(code smell)이나 최적화되지 않은 코드, 코딩 스타일 문제를 검사할 수 있다. 반면에, Checkstyle은 주로 코딩 스타일의 문제를 식별하는 데 초점을 맞춘 도구이다. 본 논문에는 코딩 스타일도 검사할 수 있으면서 코드 스멜 등 종합적인 검사 기능을 제공하는 PMD를 사용한다.

3) Python의 코딩 스타일 검사 도구

Python에서 사용할 수 있는 코딩 스타일 검사 도구로는 Pylint, Pyflakes가 있다[12, 13]. Pylint는 PEP-8 기준으로 코딩 스타일을 준수 여부와 더불어 코드 스멜도 검사한다. 또한, 코드의 복잡성, 코드 중복 등 다양한 검사 규칙을 제공한다. 반면, Pyflakes는 주로 코딩 스타일보다는 오류가 발생할 수 있는 코드를 식별하는 데 초점을 맞춘 도구이다. 본 논문에는 다양한 검사 규칙을 제공하는 Pylint를 사용한다.

2.3 온라인 평가 시스템

온라인 평가 시스템은 준비된 입력 데이터와 채점 데이터를 이용하여 학습자의 코드를 평가할 수 있는 시스템이다. 학습자가 풀이 코드를 제출하면 준비된 입력 데이터로 동작한 결과와 채점 데이터를 비교하여 평가한다. 이러한 온라인 평가 시스템에는 DOMjudge, HUSTOJ, neoESPA 등이 있다[14-16].

그중에서 neoESPA는 2015년에 개발된 온라인 평가 시스템으로 C, C++, Java, Python, Haskell을 지원한다. neoESPA는 특정 시간 내에 풀이 코드를 제출해야 하는 시간제한 기능과 제출할 수 있는 횟수를 제한하는 기능을 제공한다. 우리는 neoESPA에 코딩 스타일 검사 기능을 추가하여 학습자가 코딩 스타일을 배울 수 있도록 하였다.

온라인 평가 시스템과 더불어 고려해야 할 것은, 코딩 테스트 관련 플랫폼과 온라인 코딩 시스템이다. 코딩 테스트를 지원하기 위해 문제 은행 방식으로 문제를 제공하고 이를 온라인으로 테스트해 주는 시스템으로, 백준 온라인 저지와 리트코드(leetcode), 프로그래머스(programmers), 코드시그널(co-

designal) 등이 있다[17-20]. 또한, 온라인으로 코딩 환경을 제공해 주는 시스템으로 구름IDE나 라이브코드(LiveCode), 원컴파일러(OneCompiler) 등이 있다[21-23]. 이들 시스템도 컴파일 및 실행 환경을 제공하며, 특히 코딩 테스트 지원 플랫폼은 자동 채점까지 수행해 준다. 그러나 이들 중 어떤 시스템도 코딩 스타일까지 검사하여 제공하는 시스템은 없다.

3. 시스템 설계

3.1 시스템 설계

이 논문에서 제안하는 시스템은 Fig. 1과 같다. 이 구조는 기존 연구에서 제안했던 구조를 발전시킨 형태이다[24]. 기존 연구에서 추가된 부분은 Pylint뿐만 아니라 C와 Java에도 스타일 규칙 검사 도구를 적용한 점이다. neoESPA에서 채점에 사용할 언어를 설정하면 해당 언어에 맞는 스타일 검사 도구가 코드의 코딩 스타일을 검사한다. 즉, 언어를 C로 설정하면 Cppcheck가 동작하고 Java로 설정하면 PMD, Python으로 설정하면 Pylint가 동작한다.

코딩 스타일 검사 도구의 출력이 다르지만, 필요한 데이터를 추출할 수 있게 개발하였다. 언어별 검사 도구는 기본적으로 검사 규칙과 메시지, 라인 정보가 포함되어 있다. 따라서 코딩 스타일 검사 모듈에서 각 도구에 맞는 구문 분석 모듈을 이용해 데이터를 추출하도록 수정하였다. 또한, 언어별 규칙 검사 설정을 생성하고, 코딩 스타일 검사 모듈에서 도구별 동작을 수행하는 코드를 추가하였다.

3.2 코딩 스타일 규칙 선정

프로그래밍 초보자에게는 도구에서 제공하는 코딩 스타일을 모두 채택하여 학습자에게 제공하는 것이 적합하지 않다. 코딩 스타일 검사 도구의 출력 메시지는 프로그래밍 언어에 익숙한 개발자를 대상으로 작성되어 있다. 즉, 코딩 스타일 오류로 아직 배우지 않은 내용이 포함되어 있으면 학습자는 코드를 어떻게 수정해야 하는지 알기 힘들다.

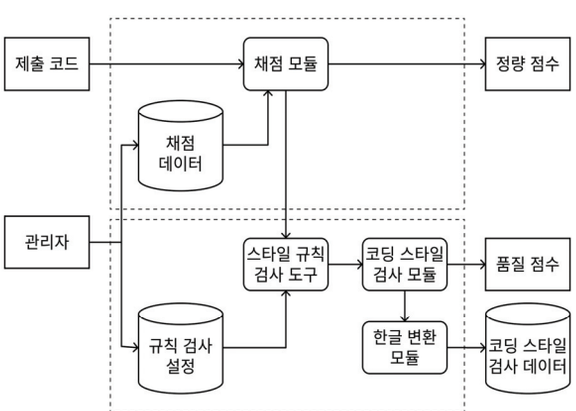

Fig. 1. The Overview of the Proposed Approach

Table 1. Analysis of Coding Style Rules Selected per Tool

도구 이름	전체 규칙 수	선별 규칙 수	반영 비율(%)
Cppcheck	310	77	24.84
PMD	284	43	15.14
Pylint	420	114	27.14
계	1,014	234	23.08

예로, Pylint의 consider-using-dict-comprehension 규칙은 Python의 사전(dictionary) 조건제시법을 학습하지 않으면 학습자가 설명을 읽더라도 코드를 어떻게 수정하는지 알기 어렵다. 또한, 변수나 클래스, 함수 등 이름을 명명할 때 적용하는 낙타체, 뱀체를 모르면 올바른 명명 규칙을 적용하기 어렵다. 따라서 학습자에게 학습 진도에 따른 검사 규칙을 별도로 구성하여 제시할 필요가 있다.

수업 진도에 맞는 코딩 스타일 규칙을 선정하기 위해 수업 진도별 과제에서 학생들이 제출한 코드를 이용하였다. 언어별 학생들의 코드는 부산대학교에서 2017년도 2학기 Java 수업에서 제출된 1,980개 코드와 2023년도 1학기 Python 수업 4,864개 코드, 2023년도 2학기 C 수업 3,040개 코드이다. 제출된 코드는 수업에서 학생들은 매주 학습한 내용을 바탕으로 1시간 이내에 해결하는 문제와 24시간 이내에 해결하는 두 가지 문제를 푼 코드이다. 학생들이 문제별로 최대 10번의 풀이 코드를 제출할 수 있어 하나의 문제에 여러 가지 풀이를 제출할 수 있다.

Table 1은 선정된 규칙 수를 나타낸다. 수업 진도별로 학생들의 풀이 코드를 코딩 스타일 검사 도구로 검사한 결과를 수집하였다. Cppcheck는 310개의 규칙 중 77개를 선정하여 24.84%를 반영한다. 또한, PMD는 284개 규칙 중 43개인 15.14%, Pylint는 420개 규칙 중 114개로 27.14%를 반영한다. 학생들의 프로그래밍 언어를 배울 때 필요한 코딩 스타일 규칙은 전체 규칙 중 23.08% 정도만 필요한 것을 알 수 있다.

3.3 품질 점수

코딩 스타일은 프로그래밍 언어를 학습할 때부터 체화하는 것이 중요하다. 코딩 스타일 학습 동기를 부여하기 위해 제출한 코드의 코딩 스타일 점수를 학점에 반영한다. 이에 따라 학습자는 코딩 스타일 규칙을 준수한 코드를 작성하게 되고 수업 진도가 진행됨에 따라 높은 품질의 코드를 작성하는 습관을 지닐 수 있다.

Equation 1은 품질 점수 산출식이다. 산출식에서 S는 채점 점수, W는 채점 점수 반영 비율, P는 감점 반영 비율, n은 검출된 오류 개수로, 반영 비율 W와 P는 0보다 크고, 1보다 작은 임의의 값으로 설정한다. 기존 연구의 품질 점수식은 n을 계산할 때 코딩 스타일 검사 도구의 규칙 부류를 나누고, 부류별 가중치와 검출 횟수의 곱으로 산정했다. 하지만 도구별 부류를 나누는 기준이 다르고, 부류가 명확히 나뉘지 않기 때문에 가중치를 부여하기 어려운 문제가 있었다. 개선된 품

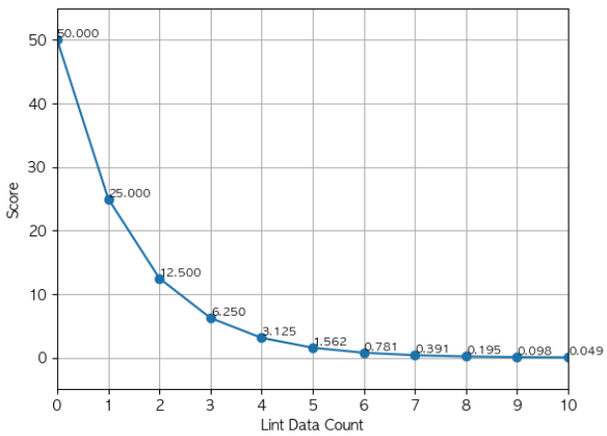

Fig. 2. Quality Score by Error Count

질 계산 식은 부류를 나누지 않고, 코딩 스타일 검사 도구의 검출 개수 n만 고려한다.

$$Q(n) = \begin{cases} 0, & S < 0 \text{ 이면} \\ S \times W \times P^{n}, & S \geq 0 \text{ 이면} \end{cases} \quad (1)$$

Fig. 2는 W와 P가 50%일 때 오류 개수 별 품질 점수를 나타낸다. W가 50%이기 때문에 품질 점수의 최대 점수는 50점이 된다. 또한, 검출 개수가 하나라도 있는 경우 품질 점수가 차감되는 비중이 크다. 이는 적당한 점수만 받고 넘어가려고 하는 것을 방지하기 위함이다.

4. 시스템 구현

4.1 검사 메시지 한글 변환

Fig. 3은 한글 출력을 위해 구현한 JSON 파일의 일부를 나타낸다. 코딩 스타일 검사 도구의 출력 메시지는 두 가지 유형이 있다. 하나는 “missing-function-docstring”과 같이 같이 규칙에 따라 고정된 메시지를 출력하는 유형이며, 다른 하나는 “invalid-name”과 같이 코드 내 변수나 낙타체, 뱀체와 같은 규칙을 포함하여 메시지가 출력되는 유형이다.

“invalid-name”과 같이 출력 메시지에 변수가 포함된 경우, 해당 문장을 그대로 한국어로 번역하면 영어와 변수를 구별하기 어려워 학습자가 이해하기 힘든 문장이 될 수 있다. 따라서 출력 메시지에서 변수는 번역하지 않는 것이 중요하다. 이를 위해 정규표현식을 사용하여 출력 메시지에서 변수를 추출한 후, 변수 외의 다른 부분을 한국어로 번역하여 문장을 완성하였다.

4.2 코딩 스타일 규칙 적용

프로그래밍 언어 수업의 교육과정에 따라 코딩 스타일 규칙을 선정하였다. 하지만 현장에 따라 원활한 교육이 진행되

```
"invalid-name": {
    "eng": "(.+) name (.+) doesn't conform to (.+)",
    "kor": "%s 이름 %s이(가) %s에 부합하지 않습니다."
},
"missing-function-docstring": {
    "eng": "Missing function or method docstring",
    "kor": "함수 또는 메소드 문서화 문자열이 누락되었습니다"
},
```

Fig. 3. Partial JSON File for Korean Translation

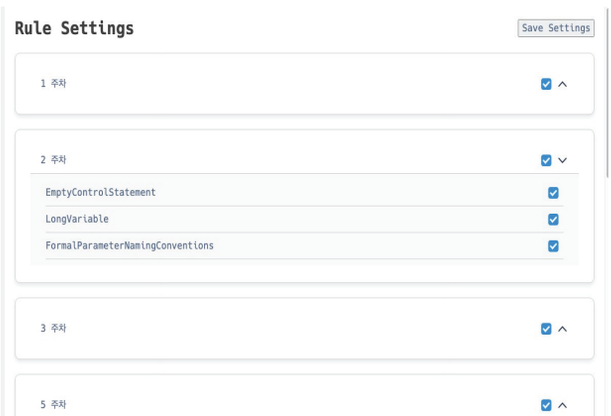

Fig. 4. Rule Setting by Course Progress

지 않을 수 있다. 그러므로 진행된 수업에 따라 검사할 규칙을 추가하거나 제거할 수 있어야 한다.

Fig. 4는 수업 진도별 규칙 설정 페이지를 나타낸다. 초기 세팅은 모든 규칙이 비활성화되어 있다. 진도별 문제를 올릴 때 관리자가 수업 주차의 오른쪽에 있는 체크박스를 이용하여 주차 내의 규칙을 모두 추가하거나 제거할 수 있다. 또한, 개별 규칙을 적용할 때 주차 내 각 규칙을 추가 및 제거할 수 있도록 기능을 구현하였다. 이를 이용해서 편리하게 스타일 검사 도구의 규칙을 설정할 수 있다.

4.3 코드 스타일 뷰어

코드 스타일 뷰어는 온라인 평가 시스템에서 코드 스타일을 확인할 수 있는 기능이다. 온라인 평가 시스템에서는 기존 채점 데이터뿐만이 아니라 코딩 스타일 검사 규칙도 관리하고 있다. 이 코딩 검사 규칙과 학습자 코드를 비교하여 학습자가 잘못 작성한 부분을 알려준다.

Fig. 5는 코드 스타일 뷰어 예시이다. 규칙을 위반하면 빨간색 동그라미와 빨간색 밑줄로 표시되므로 쉽게 확인할 수 있다. 또한, 동그라미나 밑줄을 클릭하면 규칙과 영어 및 한글 메시지가 있는 메시지 박스가 나타난다. 만약 설명을 읽어도 잘 이해되지 않는다면 참고 링크 버튼을 클릭하여 도구의 해당 규칙을 설명한 홈페이지로 이동할 수 있다. 하지만 해당 페이지는 도구에서 제공하는 페이지이기 때문에 영어로 설명이 제공된다.

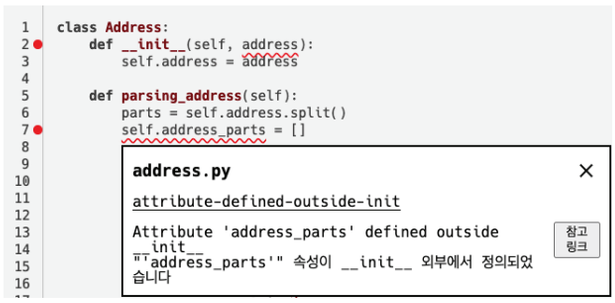

Fig. 5. Example of Code Style Viewer

5. 실험 및 평가

5.1 품질 점수 실험

품질 점수에 따른 학생들의 코딩 스타일을 준수하는 것이 습관화되는 것을 확인하기 위해 프로그래밍 언어 수업을 수강하는 학생을 대상으로 실험하였다. 실험은 부산대학교의 2024년도 1학기 Python 수업을 듣는 학생들을 대상으로 진행하였다. 학습자가 제출한 코드의 최초 품질 점수와 최종 품질 점수를 비교하고, 주차가 진행될수록 최초 점수가 향상되는지 알아본다.

Fig. 6과 Fig. 7은 수업 진도별 문제의 최초 점수 분포와 최종 점수 분포를 나타낸다. 품질 점수를 도입한 수업의 첫 주차의 점수는 Fig. 6A와 Fig. 7A를 비교할 수 있다. 최초 제출 시 0점이었던 30명의 학습자가 코딩 스타일을 수정하여 최종 제출에는 50점을 받은 것을 알 수 있다. 또한, Fig. 6B에서 15명이 품질 점수를 50점으로 받았으나 Fig. 7B에서 61명으로 이전에 비해 306.67% 증가했다. 3주 차와 4주 차에는 품질 점수가 이전보다 평균적으로 낮다. 이는 어려운 문제가 출제되어 코딩 스타일을 수정하는 기회가 적어 완전히 해결하지 못한 것으로 판단된다. 이후 5주 차와 6주 차에서 최초 제출 시 낮은 품질 점수를 받은 학생들이 원활하게 코딩 스타일을 수정한 것을 알 수 있다.

수업이 진행될수록 코딩 스타일을 준수하는 것이 습관이 된 것을 확인 할 수 있다. Fig. 6A에서 25점 이상을 받은 학생이 18명이었지만, Fig. 6F에서 44명으로 149.47% 증가한 것을 알 수 있다. 초기 의도한 대로 학습자가 일정 점수만 받고 수정하기를 포기하지 않았으며, 구현된 품질 점수 산출식은 높은 품질 점수를 받기 위해 지속해서 노력하도록 유도하는데 효과적임을 알 수 있다.

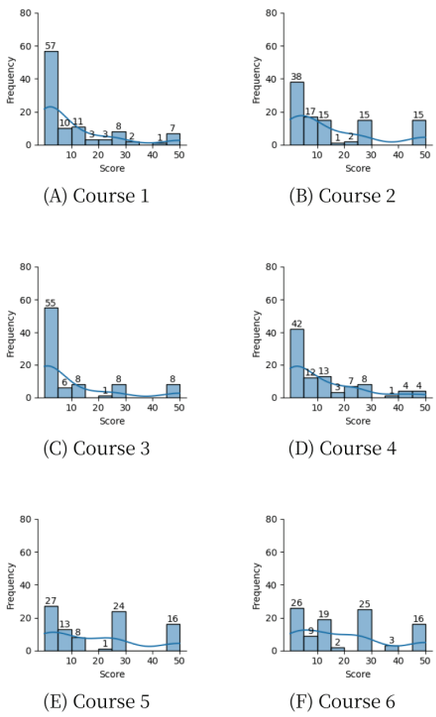

Fig. 6. Distribution of Initial Scores by Course Progress

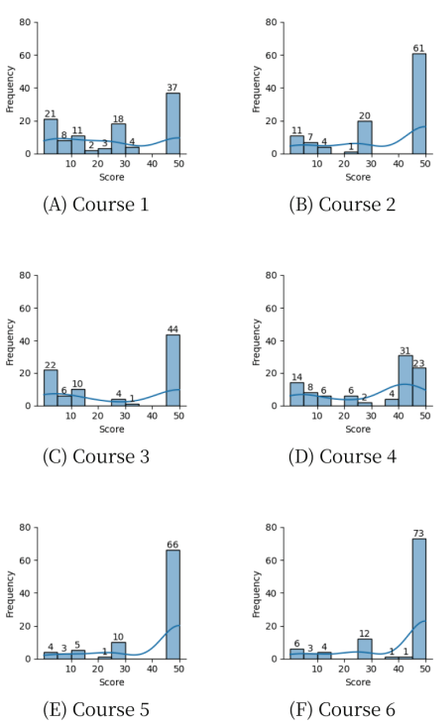

Fig. 7. Distribution of Final Scores by Course Progress

6. 토 의

학습자는 소개한 시스템에서 수업 진도별 코딩 스타일 규칙으로 어떤 규칙이 적용되는지 알기 힘들다. 초기 의도는 자신이 구현한 코드를 규칙에 맞게 수정함으로 방법을 코딩 스타일을 익힐 수 있도록 계획했다. 하지만, 실험에서는 최대 10번의 제출 횟수가 정해져 있어 수업의 내용이나 문제가 어렵게 출제되었을 때 학습자가 문제를 푸는 데 집중하는 것으로 나타났다. 코딩 스타일은 그 이후의 순서가 되었다. 이에 따라, 학생들이 코딩 스타일에 대한 학습을 충분히 하지 못했다.

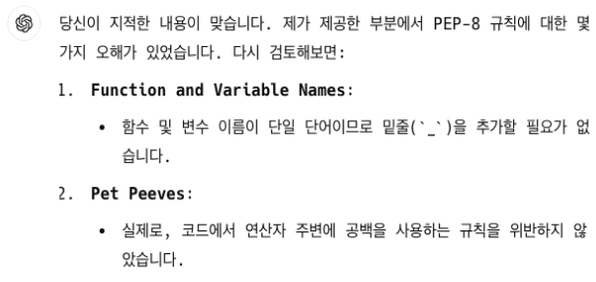

Fig. 8. Incorrect Explanation in chatGPT

이를 해결하기 위해, 최대 제출 횟수를 늘릴 수도 있지만 이는 온라인 평가 시스템의 사용 의도와 반대된다. 그러므로, 최대 10번 이내에 문제도 풀고, 코딩 스타일도 학습하기 위해 해당 주차에 적용하는 코딩 스타일 규칙을 미리 알려주는 것이 필요한 것으로 생각된다. 이를 통해 학생들이 코드를 제출하기 전 규칙을 확인하고 코드를 수정하면 제출 횟수의 구애를 받지 않고 코딩 스타일을 학습할 수 있을 것으로 기대된다. 향후 코드 구현이 완료되지 않았더라도 코딩 스타일에 대한 학습이 가능하도록 추가 연구 및 개발이 필요하다.

또한, 학습자가 이해하는 데 필요한 상세한 설명이 부족하다. 코딩 스타일 규칙을 잘 이해하지 못했을 때 확인하는 참고 링크는 영어로 설명되어 있어 학습자가 이해하기 힘들 수 있다. 그러므로, 낙타체나 뱀체와 처음 보는 용어와 같이 학습하지 않은 규칙에 대해서 추가적인 예제와 같이 설명을 추가하고 한국어 메시지에서 설명이 부족한 경우 이를 고려한 추가 문구를 추가하는 등의 방법을 추가 연구 및 개발이 필요하다.

마지막으로, 생성형 AI를 사용하면 학생들이 스스로 학습하는 데 있어 자세한 설명과 직접적인 코드 수정을 제공하여 이점을 얻을 수 있다. 그러나 모든 코드에 대해 정확한 답변을 제공하지 못할 수 있다. Fig. 8은 생성형 AI 중 하나인 chatGPT가 부정확한 정보를 설명한 예시를 보여준다. 학생이 제출한 Python 코드에서 PEP-8 규칙을 위배한 부분을 질의했으나, chatGPT가 지적한 세 가지 문제점 중 두 가지는 실제로 규칙을 위반하지 않았음에도 코드에 문제가 있다고 설명했다.

이러한 점을 고려할 때, 온라인 평가 시스템의 평가를 위해 사용하기에는 어려움이 있다. 다만, 코드 스타일 뷰어에서 부족한 설명을 보완하기 위해 학생들에게 잘못된 답변이 제공될 수 있음을 충분히 인지시킨 후 생성형 AI를 활용할 수 있도록 개발 및 구현한다면, 학생들의 학습을 도울 수 있을 것으로 생각된다.

7. 결 론

이 논문에서는 온라인 평가에서 코딩 스타일 교육을 위한 시스템을 소개하였다. 기존 연구에서 Python만 지원하던 것을 C, Java까지 지원하도록 시스템을 확장하였다. 확장 시스템은 온라인 평가 시스템에 탑재되어 코딩 스타일 검사를 제공하므로 학습자는 장소나 운영체제, 특정 도구에 구애받지 않고 같은 검사 서비스를 받을 수 있다.

또한, 이 논문에서는 처음 프로그래밍 언어를 배우는 학습자를 위해 코딩 스타일 규칙 집합을 선정하여 제시하였다. 학습자를 위한 코딩 스타일 규칙을 선정하기 위해 이전 수업에서 제출된 코드를 이용하여 전체 규칙 중 23.08%를 수업 진도별 규칙으로 선정하였다. 현재는 C, Java, Python에 대해 코딩 스타일 규칙 집합이 제시되었는데, 같은 방식을 이용하면 다른 언어로 확장할 수 있다.

마지막으로, 품질 점수로 학습자의 코딩 스타일을 점수화하여 이를 학점에 반영할 수 있는 기반을 마련하였다. 코딩 스타일을 교육한다고 하더라도 이에 대한 보상이 없으면 학생들은 어떤 동기부여도 받을 수 없다. 따라서 이 논문에서는 코딩 스타일을 점수화할 방법을 제시하였으며, 이를 코딩 스타일 점수로 도입하였다. 그 결과, 학생들의 코딩 스타일 점수가 대폭 향상됨을 확인할 수 있었다.

코딩 스타일 점수를 도입하여 Python 기초 프로그래밍 교과에 적용한 결과, 학습자의 코딩 스타일을 준수 습관이 체화된 것을 확인할 수 있었다. 최초 제출 시 점수가 1주 차에서 25점 이상을 받은 학생이 18명이었지만, 6주 차에서 44명으로 149.47% 증가했다. 이는 코딩 스타일 검사 시스템이 코딩 스타일 교육에 효과적임을 말해준다. 학생들은 코딩 스타일 규칙을 코드에 반영하면서 연습하고, 이후 과제에서 코딩 스타일 규칙을 준수하며 코드를 작성하려고 노력하였다. 이는 제안한 코딩 스타일 검사 시스템이 실제 코딩 스타일 교육에 도움이 된다는 사실을 시사한다.

향후 연구로서, 학습자가 코딩 스타일에 대해 더 잘 이해할 수 있도록 추가 연구할 계획이다. 학습자가 풀이 코드를 제출해야 확인할 수 있는 규칙을 정식 제출 전에 확인할 수 있도록 해, 코드를 완벽하게 구현하기 전에 규칙을 미리 파악할 수 있도록 할 수 있다. 또한, 학습자가 코딩 스타일 규칙을 잘 이해하지 못했을 때 초보자 수준에서 이해를 돕는 방법도 추가로 연구할 계획이다.

References

[1] X. Fang, “Using a coding standard to improve program quality,” in *Proceedings Second Asia-Pacific Conference on Quality Software*, Hong Kong, pp.73-78, 2001.

[2] B. Boehm and V. B. Basili, “Software defect reduction top 10 list,” *Software engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research*, Vol.34, No.1, pp.75-78, 2007.

[3] R. E. Berry and B. A. E. Meekings, “A style analysis of C programs,” *Communications of the ACM*, Vol.28, No.1, pp.80-88, 1985.

[4] T. D. LaToza, G. Venolia, and R. DeLine, “Maintaining mental models: a study of developer work habits,” in *Pro-*

ceedings of the 28th International Conference on Software Engineering, Shanghai, pp.492-501, 2006.
[5] Y. Prokop, O. Trofymenko, and O. Zadereyko "Developing students' code style skills," in *2023 IEEE 18th International Conference on Computer Science and Information Technologies (CSIT)*, Lviv, pp.1-4, 2023.
[6] T. C. Wooten, "Factors influencing student learning in introductory accounting classes: A comparison of traditional and nontraditional students," *Issues in Accounting Education*, Vol.13, No.2, pp.357-373, 1998.
[7] I. S. Seol and H. J. Kim, "Impact of a participative grading policy on students' effort for learning: A preliminary investigation," *Journal of Social Sciences*, Vol.14, No.2, pp.263-276, 2008.
[8] danmar, Cppcheck [Internet], https://cppcheck.sourceforge.io, last visited on July 4.
[9] Vector Imformatik GmbH, PC-lint Plus [Internet], https://pclintplus.com/pc-lint-plus, last visited on July 4.
[10] PMD, PMD [Internet], https://pmd.github.io, last visited on July 4.
[11] Checkstyle, Checkstyle [Internet], https://checkstyle.sourceforge.io, last visited on July 4.
[12] pylint-dev, Pylint [Internet], https://github.com/pylint-dev/pylint, last visited on July 4.
[13] PyCQA, Pyflakes [Internet], https://github.com/PyCQA/pyflakes, last visited on July 4.
[14] J. Eldering, N. Gerritsen, K. Johnson, T. Kinkhorst, M. Pluijmaekers, M. Vasseur and T. Werth, DOMjudge [Internet], https://www.domjudge.org, last visited on July 4.
[15] zhblue, HUSTOJ [Internet], https://github.com/zhblue/hustoj, last visited on July 4.
[16] X. Liu, Y. Kim, H. Cho, and G. Woo, "NeoESPA-A new evaluation system for programming assignments," in *Proceedings of the 14th International Conference on Electronics, Information, and Communication (ICEIC 2015)*, Singapore, 2015.
[17] B. J. Choi, Baekjoon Online Judge [Internet], https://www.acmicpc.net, last visited on July 4.
[18] LeetCode, LeetCode [Internet], https://leetcode.com, last visited on July 4.
[19] programmers, programmers [Internet], https://programmers.co.kr, last visited on July 4.
[20] CodeSignal, CodeSignal [Internet], https://codesignal.com, last visited on July 4.
[21] goorm, goormide [Internet]. https://ide.goorm.io, last visited on July 4.
[22] LiveCode, LiveCode [Internet], https://livecode.com, last visited on July 4.
[23] OneCompiler, OneCompiler [Internet], https://onecompiler.com, last visited on July 4.

김 영 훈

https://orcid.org/0009-0009-7615-9660
e-mail : yonghun83@pusan.ac.kr
2013년 영산대학교 사이버보안학과(학사)
2020년 ~ 현 재 부산대학교
정보컴퓨터공학전공(석사)
관심분야 : Programming Language & Web Programming

천 준 석

https://orcid.org/0009-0009-0447-2033
e-mail : jscheon@pusan.ac.kr
2011년 동서대학교 컴퓨터공학과(학사)
2013년 부산대학교
전자전기컴퓨터공학과(석사)
2013년 ~ 현 재 부산대학교
정보컴퓨터공학전공 박사과정
관심분야 : Functional Programming Language, Proof Assistant

우 균

https://orcid.org/0009-0006-0469-3723
e-mail : woogyun@pusan.ac.kr
1991년 한국과학기술원 전산학(학사)
1993년 한국과학기술원 전산학(석사)
2000년 한국과학기술원 전산학(박사)
2000년 ~ 2004년 동아대학교
컴퓨터공학과 조교수
2004년 ~ 현 재 부산대학교 정보컴퓨터공학부 교수
관심분야 : Programming Language Analysis, Design, and Implementation, Program Similarity Comparison, Automatic Coding Style Checking, Program Visualization, Software Testing, Functional Programming, Object-Oriented Programming, Web and Cloud Computing