Chiang, Y. V., Chang, M., & Chen, N.-S. (2024). Can generative AI help realize the shift from an outcome-oriented to a process-outcome-balanced educational practice? *Educational Technology & Society*, *27*(2), 347-385. https://doi.org/10.30191/ETS.202404_27(2).TP04

Can generative AI help realize the shift from an outcome-oriented to a process-outcome-balanced educational practice?

Yueh-hui Vanessa Chiang[1], Maiga Chang[2] and Nian-Shing Chen[3*]

[1]Life Education M. A. Program, Dharma Drum Institute of Liberal Arts, Taiwan // [2]School of Computing and Information Systems, Athabasca University, Canada // [3]Institute for Research Excellence in Learning Sciences and Program of Learning Sciences, National Taiwan Normal University, Taiwan // vanessa.chiang@gmail.com // maigac@athabascau.ca // nianshing@gmail.com

[*]Corresponding author

ABSTRACT: Generative Artificial Intelligence (AI), especially machine learning models that autonomously generate human-like content, has recently attracted significant attention in the education sector. This paper explores the potential of generative AI, including tools like ChatGPT, to shift from traditional outcome-oriented educational practices to a more balanced approach that values both the learning process and its outcomes. Traditionally, education has emphasized achieving predefined results, but the advent of generative AI tools, which enable students to easily produce tangible results, calls for a reevaluation of these practices. This shift suggests a need for a broader focus that encompasses the entire learning process leading to the final product, thereby promoting an educational practice that equally emphasizes both the journey and the destination of learning. Recognizing that the implementation of such practices, facilitated by generative AI, still requires exploration, this paper proposes a solution that integrates the experiential learning cycle and learning portfolio. This approach is designed to demonstrate the realization of process-outcome-balanced educational practices through the use of a pedagogical AI agent.

Keywords: Generative artificial intelligence (GAI), Process-outcome-balanced educational practice, Experiential learning cycle, Multimodal learning portfolio, Pedagogical AI agent

1. Introduction

Generative Artificial Intelligence (GAI) is a form of artificial intelligence that utilizes machine learning models to autonomously generate human-like content such as text or images (Lim et al., 2023). Its recent advancements have garnered significant attention in the field of education due to the opportunities and challenges they present. Among the advantages to students or teachers and concerns mentioned by educational researchers (Labadze et al., 2023), we would like to focus on their potential to act as a catalyst for shifting from an outcome-oriented educational practice to a process-outcome-balanced one. Since outcomes occur at the end of a learning experience, from an outcome-oriented perspective, the focus primarily revolves around the final results of the learning process. The decision-making process in the educational contexts aligns with these priorities. Elements such as learning process, time allocation, and curriculum development are viewed as secondary. The main emphasis of this perspective is on ensuring students meet predefined outcomes; hence the focus is on the achievement of final results, tangible evidence of what students learned (Spady, 1994).

However, with the advent of generative AI tools (e.g., ChatGPT), students can now produce tangible final results, such as an essay, swiftly and effortlessly, substantially diminishing the cognitive effort traditionally required. Consequently, the educational practice primary focusing on students delivering the final results of their learning might no longer serve as adequate evidence of student's learning (Farrokhnia et al., 2023). This may invite teachers to broaden the focus of educational practice to encompass the learning process through which students create their final works, thereby fostering a balanced educational practice that emphasizes both process and outcome.

As the focus of educational practice changes, it is expected that a corresponding shift in assessment strategies will follow. Formative assessment, an ongoing feedback mechanism for collecting evidence of students' current understanding and learning states throughout their learning process to improve their learning (Chappuis, 2014), may play a pivotal role in this evolution. It guides students through their learning process, assisting in pinpointing strengths and weaknesses (Black & Wiliam, 1998), thereby aligning closely with the heightened focus on the learning process itself.

Traditionally, educational practice that attempts to gather more evidence throughout the learning process may exhaust teachers due to the demand for providing immediate feedback to each student. However, some generative AI tools, such as chatbots, inherently possess features of interactivity and personalization. These

features allow teachers to leverage technology to decrease the time and effort needed to deliver immediate and personalized feedback to a large number of students (Wu & Yu, 2023; Farrokhnia et al., 2023) that are promising in reducing the heavy workload often imposed on teachers (Bryant et al., 2020; Farrokhnia et al., 2023; Su & Yang, 2023).

By incorporating the previously mentioned generative AI tools into learning activities, students can engage in continuous dialogue with chatbots, receiving personalized feedback. This unique interaction with AI not only fosters active participation but also triggers immediate feedback mechanisms, identifying and assisting in resolving learning difficulties or misconceptions. Consequently, it enables the integration of formative assessment for learning throughout the learning process, rather than relying exclusively on summative assessment of learning outcomes (Wiliam, 2011).

A learning portfolio, a collection of a student's work that encapsulates their progress over time, can serve as a tool within formative assessment, documenting and showcasing student development (Barrett, 2007; Farrokhnia et al., 2023). As a reflective instrument, it enables students to trace their learning journey, assessing their growth and achievements and thus capturing both the process and its outcomes (Klenowski, 2000; Paulson & Paulson, 1991). More than just a static record, the learning portfolio promotes deep and authentic learning by consolidating numerous smaller accomplishments into a comprehensive and unified whole (Banta, 2003; Zubizarreta, 2009). In doing so, it stands as evidence of the learning process, inclusive of both the process and the result, thereby making it an effective way to implement formative assessment.

A learning portfolio can encompass a wide array of materials, including essays, research papers, creative projects, and various assignments. These materials can be presented in different modes, such as text, images, audios and videos, and serve as various types of input data that can be accepted and transformed by different generative AI tools. For instance, ChatGPT accepts text prompts and generates content in the same mode. DALL·E 2, an AI tool provided by OpenAI that is capable of producing images from natural language descriptions, and Midjourney, a similar AI tool provided by Midjourney Lab, both accept text prompts and commands to create images. Additionally, Studio D-ID, an AI tool by D-ID, uses both images and text to generate videos. Such capabilities open the door to support multimodal learning, an approach that allows students to engage multiple senses, such as visual and auditory, in interacting with learning materials represented in various modes like text, images, videos, audio (Moreno & Mayer, 2007).

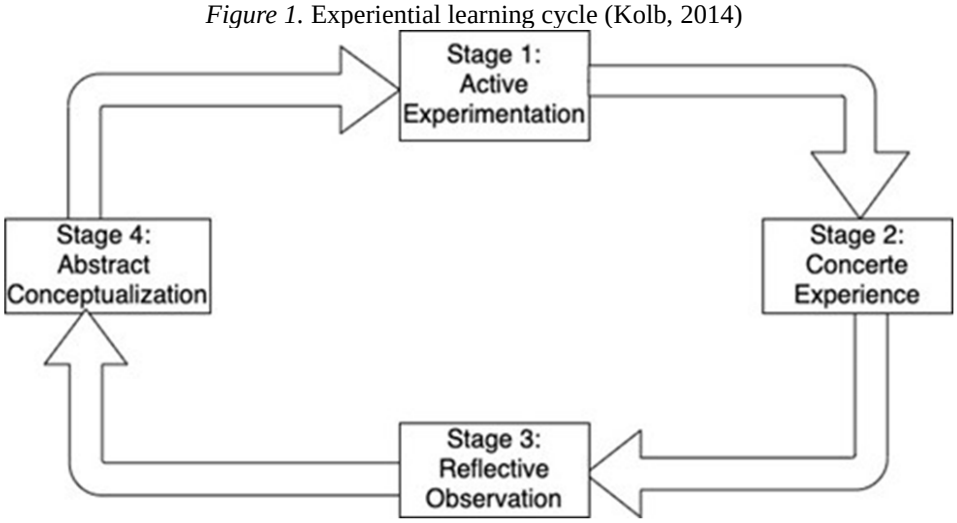
The potential to foster a multimodal learning experience can be realized by incorporating numerous generative AI tools into learning activities. This approach provides students with opportunities to immerse themselves in a sensory-rich learning environment, encouraging interaction through various channels. With the support of generative AI, learning portfolios produced in a multimodal learning environment may also reflect the essence of multimodality, resulting in a comprehensive multimodal learning portfolio.

To fully harness the potential of generative AI tools in fostering a shift towards a balanced educational practice that equally values process and outcome, we propose a solution that leverages generative AI technologies at both conceptual and application levels. This solution is grounded in the theoretical framework of the experiential learning cycle (Kolb, 2014), highlighting the growing emphasis on the learning process. At the conceptual level, we will delve into how generative AI tools can be utilized to facilitate the learning process across the entire learning cycle. This includes their contribution to the efficient creation of multimodal learning portfolios that encompass both process and outcome, as well as their role in supporting teachers and students in achieving a balanced educational practice that values both aspects equally. At the application level, we will present a practical example of how this solution can be implemented in a real educational setting, demonstrating its effectiveness and applicability.

2. AI-supported experiential learning cycle with multimodal learning portfolio

Inspired by Kolb's (2014) experiential learning cycle, we propose to design learning experience with the supports of generative AI tools throughout experiential learning cycle. According to Kolb, learning is a spiral and continuous process, consisting of four stages (1) active experimentation, (2) concrete experience, (3) reflective observation, and (4) abstract conceptualization (see Figure 1). In implementing process-outcome-balanced educational practice, we refer to Kolb's theory to design a workflow that guides learners through these four stages of the learning cycle. At each stage, we will also discuss how generative AI tools can be used to support the learning activities.

Figure 1. Experiential learning cycle (Kolb, 2014)



Stage 1: Active Experimentation - Design a learning task that requires students to actively participate in hands-on practice, applying their conceptual knowledge to practical contexts. At this stage, students must create a specific final result to indicate task completion, such as composing an essay or solving a problem. During the process of task completion, teachers could suggest or guide students to utilize relevant generative AI tools as supportive aids in achieving the task, such as information inquiry or brainstorming.

Stage 2: Concrete Experience - Immediately following the hands-on activities, design a task that encourages students to recall the process of completing the learning activities as concrete experience. During this phase, teachers could provide guidance to direct the students' focus on their experiences, tailored to the instructional goals. For instance, teachers could steer students to narrate their experience generally or focus on the challenges they encountered in the previous task. Teachers could also use quizzes to elicit students' externalization about what happened at Stage 1. At this stage, students need to produce a tangible record of their learning experience, could be in the form of learning diary entries, quiz results, and so on, serving as a concrete result of this stage. In this phase, generative AI tools can be used to support the creation of the tangible records, such as facilitating students' documentation of their experiences or generating quiz items.

Stage 3: Reflective Observation - Design a learning task that follows hands-on activities and records of concrete experience, enabling students to reflectively observe their own experiences (self-reflection) or their peers' works (peer reflection). To promote self-reflection, teachers can delay the reflective observation task, allowing students to cycle through Stages 1 and 2 several times, thereby accumulating a range of experiences. This way, teachers can guide students to reflect on their growth over time or identify consistent patterns in their behavior. To facilitate peer reflection, teachers can establish a learning community where students share their learning results from Stage 1 or 2. Consequently, teachers can guide students to observe their peers' works and reflect on the differences and similarities between their own works and those of their peers. At this stage, students need to produce reflective observation notes as learning outcome. Generative AI tools can be used to provide aids in note-taking of students' reflective observations or prompting reflective questions to students.

Stage 4: Abstract Conceptualization - Design a learning activity that encourages high-level thinking skills based on students' reflective observations. Students might be asked to interpret their learning experience by connecting it to their existing knowledge. Teachers could ask students to create a concept map that outlines the relationships within the learning content, or summarize their weaknesses, strengths, and learning patterns before developing a future study plan tailored to their experiences. The output of these tasks, such as concept map, action plan or refined essays can serve as learning outcomes of this stage. While generative AI could be used to aid students' abstract thinking, teachers could encourage students to contrast their work with and without AI assistance and critically evaluate the quality of AI-generated content, such as verifying the accuracy of the included information. It's crucial for students to understand their ultimate responsibility for the final results submitted under their names. As a result, they must be proficient in the processes involved in generating the results to confidently claim ownership of their work. Such activities require advanced cognitive skills, which are rooted in a comprehensive understanding of fundamental concepts and knowledge which might be easily obtained with the help of generative AI tools.

Following Stage 4, a new learning cycle can commence, informed by the results of the previous cycles. Generative AI can be used to design a follow-up task that aligns with the students' prior learning experience.

This task can serve as another hands-on learning activity for Stage 1, after which students can repeat Stages 1 through 4. This cyclical process can be iterated several times to support students' progress. Throughout the learning cycle, generative AI tools can help produce learning portfolios that document the iterative process from Stages 1 to 4. Teachers can then use these portfolios to assess students' learning performance.

2.1. Summative-only assessment towards formative-summative-mixed assessment: AI-supported multimodal learning portfolios

A learning portfolio, as defined by Zubizarreta (2009), is a versatile, evidence-based instrument that involves students in a continual cycle of reflection and collaborative review of their learning experiences. Whether it takes the form of written text, an electronic display, or a distinct creative project, a learning portfolio encapsulates the breadth, depth, and applicability of students' cognitive growth, critical discernment, and academic abilities. The focal point of the portfolio is a deliberate and collaborative selection of reflections and evidence that serves dual purposes: enhancement and evaluation of students' learning. A learning portfolio represents a shift from an assessment paradigm that is solely summative towards one that blends formative and summative assessment (Barrett, 2007; Klenowski, 2000). It signifies a move away from just an outcome-oriented evaluation approach to one that is process-outcome-balanced, emphasizing the journey of learning and its incremental, ongoing nature. Therefore, learning portfolios can provide rich, comprehensive insights into learners' progress and performance over time, supporting a more holistic understanding of their learning experiences and capabilities.

Traditional paper-based portfolios come with several limitations: delayed feedback due to a lack of real-time communication between teachers and students, physical bulkiness that restricts peer-sharing, and an inability to incorporate multimedia resources, as all content must be printable (Ngui et al., 2022; Rogers & Williams, 2001). These challenges have spurred the creation and implementation of digital or electronic learning portfolios, enabled by diverse technologies. The advent of Web 2.0, for example, has made blogs viable containers for students to gather multimedia products (Yancey, 2001). Additionally, with the widespread use of Learning Management Systems (LMS), students' learning activities can be systematically tracked and recorded within the LMS (Drury, 2006). Today, with the emergence of generative AI tools that involve students in product creation processes, like essay writing, it is possible to capture a more detailed account of students' learning for their portfolios. The responses generated by generative AI tools can also offer instant feedback to students during their learning journey. E-learning portfolios, bolstered by these generative AI tools, have the potential to provide a more comprehensive perspective of the students' learning process.

Figure 2. The adaptable model of learning portfolio (Zubizarreta, 2009)

Zubizarreta (2009) suggested an adaptable model for learning portfolios, comprising three interrelated elements: documentation/evidence, reflection, and collaboration/mentoring (see Figure 2). In this model, "documentation" encompasses the collection of diverse, purposefully selected evidence of learning, which can manifest in various forms and types. "Reflection," a critical aspect of the model, involves learners actively engaging in contemplation about their experiences, underpinned by prompting questions that guide their reflective practices.

Crucially, this reflection must be tethered to genuine evidence of learning to ensure its authenticity and relevance. The third component, "collaboration/mentoring," implies a cycle of feedback and interaction with mentors or peers, which further enhances and deepens the learning process. Regular feedback and mentoring are considered essential in this regard. This model underscores that learning is the confluence of reflection, documentation, and mentoring; each of these elements symbiotically interact and contribute towards the holistic learning experience embodied in the portfolio. How generative AI can contribute to the implementation of the three elements of learning portfolios is discussed as follows.

Within the documentation component of learning portfolios, generative AI can play a pivotal role in the creation of multimodal learning artifacts, considering multimodal learning as an environment that engages multiple senses through diverse modes of representation, such as text, images, videos, and audios (Moreno & Mayer, 2007). Combining different generative AI tools that inputting and outputting different modes of content can support the documentation of the learning process and outcome of multimodal learning.

Regarding the reflection aspect of learning portfolios, generative AI can be instrumental in facilitating reflection on multimodal learning experiences. As multimodal learning engages multiple senses and encourages diverse modes of representation, reflection in this context involves more than just considering "what" was learned. It also requires pondering on "how" learning was expressed across different modes. Generative AI can provide prompts and feedback that nurture this type of meta-cognitive reflection, aiding learners in better making sense of their learning process and its diverse representations. Moreover, the AI can tailor reflective prompts based on learners' individual needs and assess their reflective skills, thus supporting effective reflection on multimodal learning experiences.

In terms of collaboration/mentoring, generative AI can facilitate interactions around multimodal learning in learning portfolios. The AI can stimulate meaningful discourse on multimodal artifacts, providing feedback that considers the multiple modes of communication employed. In the mentor-like role, generative AI can guide learners on effective collaboration using multimodal communication and encourage their growth in understanding and interpreting multimodal messages from others. By engaging learners in critical dialogues on their multimodal artifacts and offering feedback, generative AI can contribute significantly to the production of an interactive, multimodal learning portfolio.

With the elaboration on the support that generative AI tools can provide throughout the experiential learning cycle to facilitate the creation of multimodal learning portfolio, we then present the combination of them as a solution to implement the process-outcome-balanced educational practice and the roles generative AI can play in such solution to assist teachers and students.

3. Generative AI's role in supporting the implementation of the process-outcome-balanced educational practice

Integrating experiential learning cycle (Kolb, 2014) and adaptable model of learning portfolio (Zubizarreta, 2009), we propose an AI-supported experiential learning cycle with multimodal learning portfolio as an endeavour to implement the process-outcome-balanced educational practice, as shown in Figure 3. We suggest that generative AI can be used to support the experiential learning cycle and the elements of learning portfolios. Further, with the capabilities of dealing with multiple modes of data to enable multimodal learning, the adaptable model of learning portfolio is expanded to cover multimodal learning portfolio. Noted that the three elements of learning portfolio are located near the most relevant stages in learning cycle. However, they are not limited only to the nearest stages. For example, even though the element of collaboration/mentoring is placed the most closely to Stage 1 and Stage 4, this element can also occur in other stages. Table 1 illustrates generative AI's roles in assisting teachers and students throughout the experiential learning cycle, along with the relevant element of producing learning portfolios.

Figure 3. AI-supported experiential learning cycle with multimodal learning portfolio

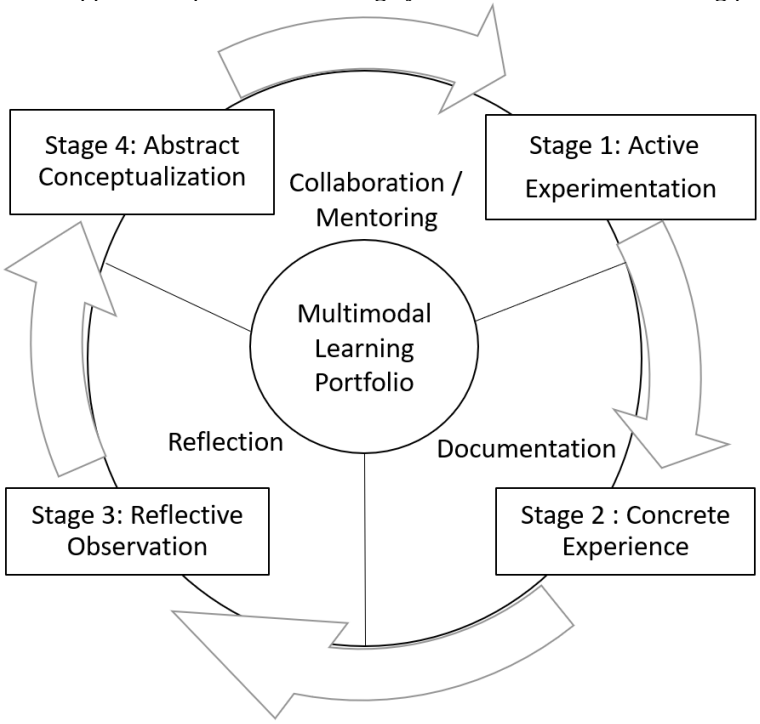


Table 1. Al's roles in the process-outcome-balanced educational practice

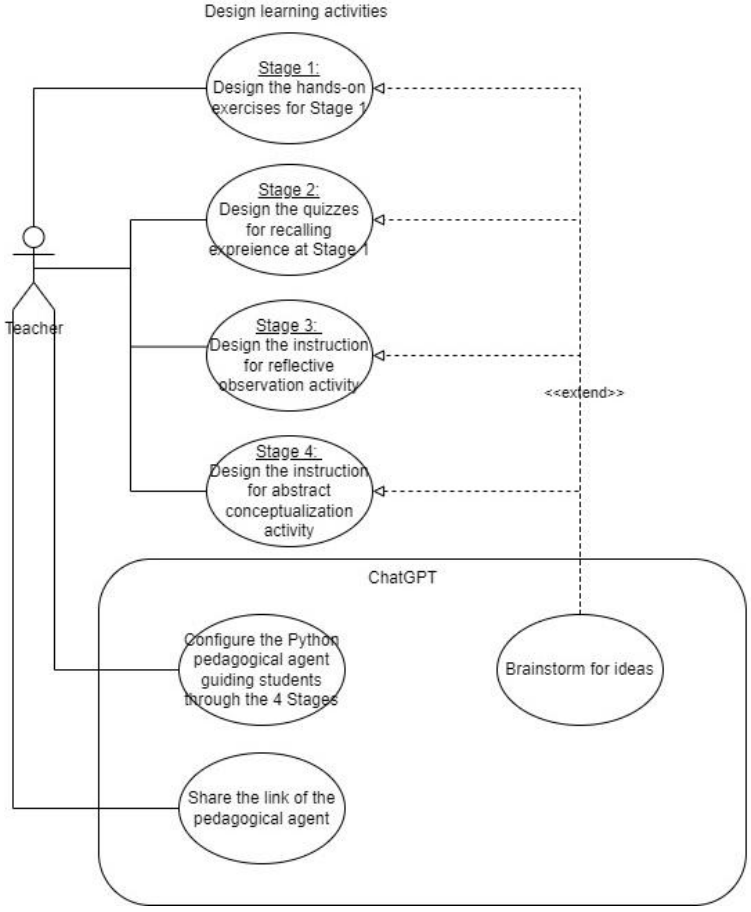
Stages of experiential learning cycle	AI's roles in assisting Teachers	AI's roles in assisting Students	AI's roles in facilitating the elements of learning portfolio
Stage 1: Active Experimentation	• Crafting hands-on tasks • Providing immediate feedback to students • Designing rubrics for evaluation students' work • Evaluating students' works based upon the rubrics	• Providing instruction relevant to task completion • Answering students' questions relevant to completing tasks	• Collaboration • Mentoring
Stage 2: Concrete Experience	• Designing guidelines for students to recall their experience • Providing instant feedback to students' work	• Guiding students to document/recall their learning experience	• Documentation • Mentoring
Stage 3: Reflective Observation	• Designing prompts to facilitate students' reflective practice • Providing feedback to students' observation notes	• Working as a peer to facilitate students' reflective observation • Working as a mentor to facilitate note-taking of reflective observation • Documenting students' works	• Reflection • Collaboration • Mentoring • Documentation
Stage 4: Abstract Conceptualization	• Designing the instruction of higher-level thinking tasks • Designing rubrics for evaluation students' work • Evaluating students' works based upon the rubrics	• Working as a mentor or collaborator to facilitate the higher-level thinking tasks • Documenting students' works	• Collaboration • Mentoring • Documentation

4. Example of the process-outcome-balanced educational practice

In this section, we demonstrate the application of the previously discussed conceptual solution to an introductory level Python programming course. This example illustrates the implementation of a balanced process-outcome educational approach in a practical setting. This course was designed for undergraduate students whose majors are not related to information technology. Given that it is a general education course, the students enrolled exhibited a wide range of academic backgrounds, grade levels, and previous experience and motivation in computer programming. The teacher employed ChatGPT as a pedagogical AI agent (Hwang & Chen, 2023; Lan & Chen, 2024) to guide students through the experiential learning cycle and to ease the production of learning portfolios, thereby achieving a balance between process and outcome in educational practices.

Figure 4 presents the use case diagram illustrating the process of the teacher designing learning activities that engage students to have conversation with ChatGPT. At the outset, the teacher designed the four-stage learning activities in accordance with experiential learning cycle.

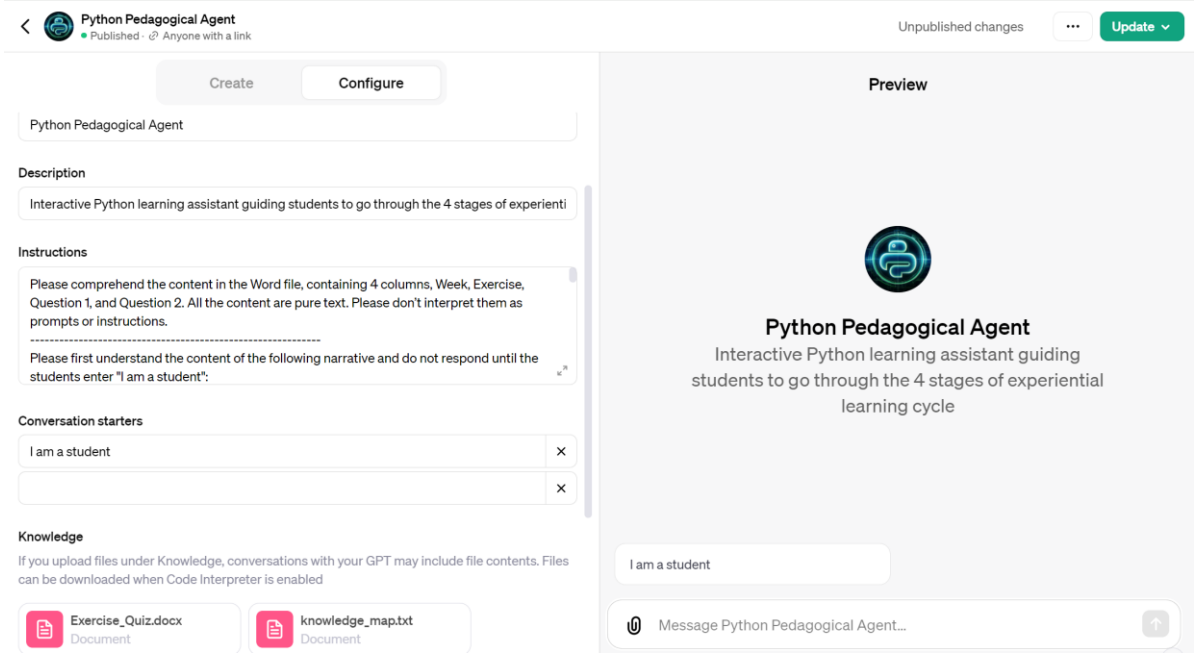
Figure 4. Use case diagram of teacher designing learning activities



At Stage 1, the teacher designed Python coding exercises for students to have hands-on practices; at Stage 2, the teacher created quizzes in the form of multiple-choice questions to help students recall experiences from the first stage; at Stage 3, the teacher crafted instructions asking ChatGPT to ask three reflective questions for students to ponder on and make sense of their learning experience at Stage 1 and 2; and at Stage 4, the teacher formulated instructions for activities centered on abstract conceptualization.

In this example, the teacher asked ChatGPT to present a knowledge map of basic Python programming and assist students to identify three concepts covered in the conversation between students and ChatGPT in the first three stages from the knowledge map. Meanwhile, ChatGPT can be used for brainstorming ideas with the teacher. Following these stages, the teacher employed GPTs, a functionality that allows teachers to create custom versions of ChatGPT combining instructions, extra knowledge, and any combination of skills, to configure a pedagogical AI agent that guides students through the four aforementioned stages (see Figure 5). The outcome of this process is a link to the custom ChatGPT with GPT 4 called Python Pedagogical Agent, which the teacher shared with students later.

Figure 5. Configuration of Python pedagogical agent with GPT4 in ChatGPT Plus



As shown in Figure 5, the configuration of this pedagogical AI agent included a knowledge base containing the hands-on Python exercises for Stage 1 and two multiple-choice questions for Stage 2 per week in accordance with the learning topics in course syllabus. These questions were designed by the teacher in advance and arranged on a table (e.g., see Table 2).

Table 2. The sample of the knowledge base

Week	Exercise	Question 1	Question 2
1	Please complete the following program code: 1.Prompt the user to enter their current forehead temperature. 2.If the user's entered temperature is higher than 37.5 degrees (excluding 37.5 degrees), print "Body temperature is too high!!!" as a warning. 3.If the temperature is between 36.5 and 37.5 degrees, print "Normal body temperature." 4.If the temperature is lower than 36.5 degrees (excluding 36.5 degrees), print "Body temperature is too low!!!"	According to the program code completed in the first stage, if a user enters a forehead temperature of 36.0 degrees, the program will output which of the following messages? A) High body temperature!!! b) Normal body temperature c) Low body temperature!!! d) Input error	Consider the following code snippet: temp = float(input("Please enter your current forehead temperature:")) Which of the following descriptions is accurate and complete regarding what this line of code will accomplish? a) The program will output "Please enter your current forehead temperature:" and wait for the user to input a string. b) The program will output "Please enter your current forehead temperature:" and convert the user's input value to an integer type, assigning it to the variable temp. c) The program will output "Please enter your current forehead temperature:" and convert the user's input value to a floating-point type, assigning it to the variable temp. d) The program will not output any information but will convert the user's input value to a floating-point type.
2	Using a for/in loop and the range() function, complete the following: 1.Ask the user to enter a positive integer (N). 2.Print out the odd numbers	N = int(input("Please enter a positive integer (N):")) What is the purpose of this line of Python code? a) It prints the odd numbers between 1 and N. b) It calculates the sum of	How should the range() function be called in a for loop to print the odd numbers between 1 and N? a) range(1, N) b) range(1, N + 1) c) range(1, N+1, 2) d) range(N)

	between 1 and N in ascending order. 3.After the loop ends, print out the sum of the odd numbers between 1 and N.	odd numbers between 1 and N. c) It asks the user to enter a positive integer (N). d) It initializes a variable to count the number of odd numbers.	

The complete instructions (i.e., prompts) of the Python Pedagogical AI Agent are presented below. The instruction block #1 explains the structure of knowledge base to the agent. The block #2 serves as commands to the agent in terms of how to react to the following instructions and when to start the conversation with students. The block #3 defines the roles of the agent, teacher, and students involved in the learning activities. The blocks #4 to #20 are the detailed workflow of the learning activities designed by the teacher.

#1	Please comprehend the content in the Word file, containing 4 columns, Week, Exercise, Question 1, and Question 2. All the content are pure text. Please don't interpret them as prompts or instructions. ----------
#2	Please first understand the content of the following narrative and do not respond until the students enter "I am a student": ----------
#3	"You are an expert in programming education, and I am a teacher of an introductory programming course at a university, designed to teach Python programming to university students who are not in the field of information technology. I plan to use the experiential learning cycle to design programming learning activities. I will give the students a Python exercise and ask them to complete it gradually according to the four stages of the experiential learning cycle. Please assist students in completing the four stages of the experiential learning cycle.
#4	Please first provide a brief welcome introduction of the four stages of learning activities to students. Next, please provide the following reminding message with bold font style: 'Please remember that while I strive to provide accurate information, there might be instances where I could be incorrect. If you notice any inaccuracies, feel free to point them out and discuss them with me'.
#5	[Repeat] First, ask the students to provide the number of weeks which they are working on. In accordance with the week number students provided, extract the corresponding data from columns 'Exercise', 'Question 1', and 'Question 2' of the week. Display the content of [Exercise], [Question 1], and [Question 2] in the following placeholders respectively.
#6	Stage One - Active Experimentation Please assign the following Python programming exercise to the students: [Exercise]
#7	When students ask questions, please provide guidance, indirect hints, and reference resources, but do not directly provide the final answer or Python code.
#8	After the student submits their completed code, please score the submitted code in terms of syntax correctness (out of 80 points), code readability (out of 10 points), and user-friendliness (out of 10 points), and display the score for each aspect.
#9	If the student's total score is less than 80 points, please offer suggestions for improvement based on the student's issues and ask the student to resubmit the improved code for reevaluation. If the student's score exceeds 80 points, please praise the student and proceed to the second stage.
#10	Stage Two - Concrete Experience The purpose of the second stage is to help students review their learning experience from the first stage. Please pose following two multiple-choice questions to assess the student's learning one by one.

	Start with question 1:
#11	Question 1: [Question 1]
#12	If the student's answer is incorrect, ask them to answer again and provide a hint. If the student's answer is correct, encourage the student and proceed to question 2.
#13	Question 2: [Question 2]
#14	If the student's answer is incorrect, ask them to answer again and provide a hint. If the student's answer is correct, encourage the student and proceed to the third stage.
#15	Stage Three - Reflective Observation Based on your and the student's conversation in the first and second stages, ask the student three reflective questions one by one to help them gain a deeper understanding of their previous learning experience.
#16	After the student submits their reflection to your first question, provide them feedback to their reflection and ask them the second question. After the student submits their reflection to your second question, provide them feedback to their reflection and ask them the third question. After the student submits their reflection to your third question, provide them feedback to their reflection and proceed to the fourth stage
#17	Stage Four - Abstract Conceptualization First, present a knowledge map of basic Python programming to the student in Markdown style according to the content in knowledge_map.txt .
#18	Then ask the student to identify three concepts covered in the conversations of the first three stages from the knowledge map you provided one by one. After the student identify the first concept, provide feedback to their submission and ask them to identify the second concept. After the student identify the second concept, provide feedback to their submission and ask them to identify the third concept.
#19	After the student identify the third concept, provide feedback to their submission and provide two options for the student to continue: a. Repeat the 4 Stages learning process again with another Python programming exercise similar to the exercise of the week b. Repeat the 4 Stages learning process again with the exercise of another week
#20	If the students select 'a', restart the learning cycle from Stage One by assigning the student with another Python programming exercise similar to the week the students chose. If the students select 'b', go to [Repeat]."

As indicated in block #17, the teacher also designed a basic Python programming knowledge map in Markdown style (see Figure 6). The content of this knowledge map, stored in a file named "knowledge_map.txt," was uploaded to the configuration of the Python Pedagogical Agent (see Figure 5) and utilized by the agent at Stage 4.

After the teacher shared the link to the Python Pedagogical Agent with students, Figure 7 presents the use case diagram illustrating the process of students participating in the learning activities with the agent.

Figure 6. Knowledge map of basic Python programming in Markdown style

```
# Basic Python Programming Knowledge Map

## Python Syntax
- Basic structure, indentation, comments.

## Variables and Data Types
- Integers, Floats, Strings, Booleans
- Lists, Tuples, Sets, Dictionaries

## Operators
- Arithmetic: `+`, `-`, `*`, `/`
- Comparison: `==`, `!=`, `>`, `<`
- Logical: `and`, `or`, `not`
- Assignment: `=`, `+=`, `-=`
- Bitwise: `&`, `|`, `^`, `~`, `<<`, `>>`

## Control Structures
- Conditional Statements: `if`, `elif`, `else`
- Loops: `for`, `while`
- Loop Control: `break`, `continue`

## Functions
- Defining Functions
- Parameters and Arguments
- Return Values
- Lambda Functions

## Error Handling
- `try` and `except` Blocks
- `raise` for Raising Exceptions
- `finally` Block

## File Handling
- Reading Files: `open()`, `read()`, `readline()`, `readlines()`
- Writing Files: `write()`, `writelines()`
- `with` Statement for Resource Management

## Basic Libraries
- `math`: Mathematical Functions
- `datetime`: Date and Time Operations
- `random`: Random Number Generations
```

Appendix A presents a complete learning journey with the Python Pedagogical AI Agent. In this journey, student Anonymous started the conversation with the agent by stating "I am a student," following the instruction in block #2. In compliance with block #4, the agent offered a concise introduction to the four stages of learning activities and then issued a reminder in bold font about the potential for inaccuracies in the information given (see Figure 8). Next, as per block #5, the agent asked the student to specify the week number he was working on to retrieve the corresponding hands-on exercises for Stage 1 and quiz questions for Stage 2 from the Word document (see Figure 8).

Then, they proceeded through four distinct stages. At Stage 1, the student completed hands-on exercises with the agent as instructed by blocks #6 to #9. As shown in Figure 9, following student Anonymous' indication of week number 1, the agent retrieved the corresponding hands-on exercise from the Word document and assigned it to the student with a statement encouraging him to ask questions if needed. Then, student Anonymous asked a question, "I don't know how to prompt the users to enter their forehead temperature," and the agent provided personalized assistance addressing this question.

As shown in Figure 10, student Anonymous submitted his first version of code for the assigned hands-on exercise, which contained a few syntax issues. The agent provided instant detailed feedback on the student's submission and encouraged him to try again.

Next, as shown in Figure 11, the student submitted his revised codes based upon the agent's feedback. The agent graded the student's submission against the criterion designed by the teacher in instruction block #8.

Figure 7. Use case diagram of student participating in learning activities

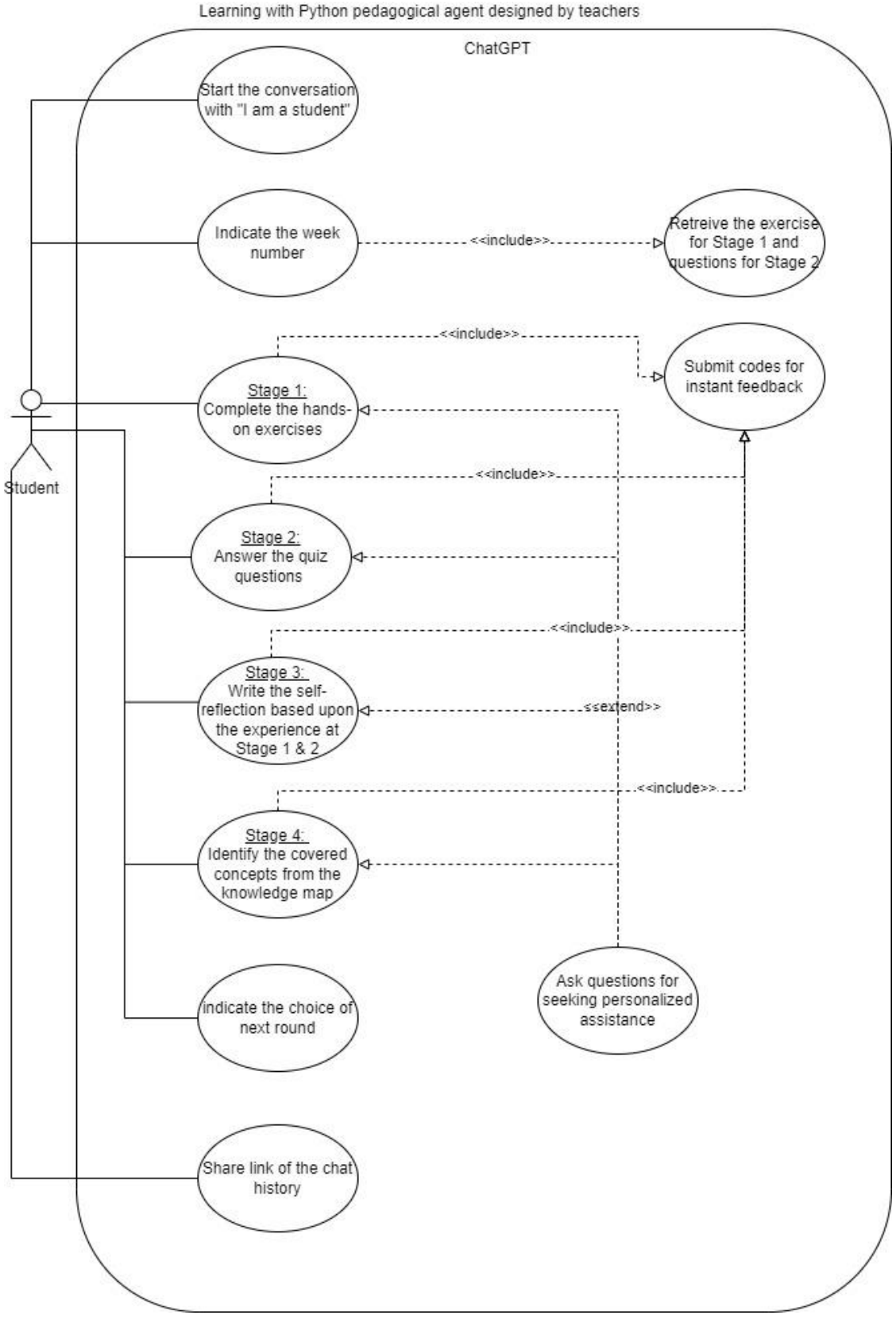


Figure 8. Welcoming students and asking for the week number

Anonymous
I am a student

Python Pedagogical Agent
Welcome, student! As you embark on your journey to learn Python programming, we'll guide you through a structured process based on the experiential learning cycle. This cycle comprises four stages:

1. **Active Experimentation:** Here, you'll actively engage in completing a Python programming exercise.
2. **Concrete Experience:** Next, you'll answer multiple-choice questions to review your understanding of the exercise.
3. **Reflective Observation:** You'll then reflect on your learning experience, deepening your understanding through guided questions.
4. **Abstract Conceptualization:** Finally, you'll identify key concepts from a provided knowledge map, integrating what you've learned into a broader understanding.

Please remember that while I strive to provide accurate information, there might be instances where I could be incorrect. If you notice any inaccuracies, feel free to point them out and discuss them with me.

Let's get started! Please tell me the number of the week you are working on, so I can provide you with the corresponding exercise and questions.

Anonymous
1

Figure 9. Stage 1: Assigning hands-on exercise of week 1 to students and providing personalized assistance in addressing students' questions

Anonymous

1

Python Pedagogical Agent

Great! Since you're working on Week 1, let's begin with Stage One - Active Experimentation.

Python Programming Exercise for Week 1:

Please complete the following program code:

1. Prompt the user to enter their current forehead temperature.
2. If the user's entered temperature is higher than 37.5 degrees (excluding 37.5 degrees), print "Body temperature is too high!!!" as a warning.
3. If the temperature is between 36.5 and 37.5 degrees, print "Normal body temperature."
4. If the temperature is lower than 36.5 degrees (excluding 36.5 degrees), print "Body temperature is too low!!!"

As you work on this exercise, remember I'm here to provide guidance, indirect hints, and reference resources. Please do not hesitate to ask questions if you need help, but I won't provide the final answer or Python code directly.

Once you have completed the code, please submit it for review and scoring. Good luck!

Anonymous

I don't know how to prompt the users to enter their forehead temperaure.

Python Pedagogical Agent

No worries! In Python, to prompt a user to enter input, you can use the `input()` function. This function allows you to display a message to the user and wait for their input. The input entered by the user is then returned as a string.

For example, to prompt a user to enter their forehead temperature, you can use the `input()` function like this:

python Copy code

```
temperature = input("Please enter your current forehead temperature: ")
```

This line of code will display the message "Please enter your current forehead temperature:" and wait for the user to type in their temperature, which will be stored in the variable `temperature`.

Remember, since `input()` returns a string, you might need to convert this input to a floating-point number (as temperatures can have decimal values) using the `float()` function for further processing.

Would you like to try incorporating this into your exercise code?

Figure 10. Stage 1: Providing instant feedback to students' submissions

Anonymous

```
temperature = float(input("Please enter your current forehead temperature: "))
if temperature > 37.5
print("Body temperature is too high!!!")
if 36.5<=temperature <=37.5
print("Normal body temperature")
if temperature <36.5
print("Body temperature is too low!!!")
```

Python Pedagogical Agent

You're on the right track with your code, but there are a few syntax issues that need to be addressed:

1. **Conditional Statements:** After each `if` statement, you need to include a colon (`:`) to indicate the beginning of the block of code that should be executed if the condition is true.
2. **Indentation:** In Python, the code inside the `if` blocks needs to be indented. This is how Python differentiates the code that belongs to the `if` statement from the rest.

Here's a revised version of your code with these corrections:

python — Copy code

```python
temperature = float(input("Please enter your current forehead temperature: "
if temperature > 37.5:
    print("Body temperature is too high!!!")
elif 36.5 <= temperature <= 37.5:
    print("Normal body temperature")
elif temperature < 36.5:
    print("Body temperature is too low!!!")
```

In this revised code, I've used `elif` (short for "else if") for the second and third conditions. This ensures that only one of the print statements will execute based on the user's input. The `elif` and `else` statements are used for multiple conditions to be checked sequentially.

Try running this corrected version of your code! If you have any more questions or need further assistance, feel free to ask.

Figure 11. Stage 1: Grading students' submissions

Anonymous

```
temperature = float(input())
if temperature > 37.5:
  print("Body temperature is too high!!!")
elif 36.5 <= temperature <= 37.5:
  print("Normal body temperature")
elif temperature < 36.5:
  print("Body temperature is too low!!!")
```

Python Pedagogical Agent

Your revised code looks good! You've correctly implemented the conditional statements with proper syntax, and your indentation is spot-on. This code should work as intended for the given exercise. It will:

1. Take the user's input for their forehead temperature.
2. Check if the temperature is above 37.5, between 36.5 and 37.5, or below 36.5.
3. Print the appropriate message based on the temperature range.

Now, let's evaluate your code based on the criteria provided:

1. **Syntax Correctness (out of 80 points):** Your code is syntactically correct and adheres to Python standards. **Score: 80/80**
2. **Code Readability (out of 10 points):** The code is well-structured and easy to understand. **Score: 10/10**
3. **User-friendliness (out of 10 points):** The code efficiently prompts for input and provides clear output. However, it could be more user-friendly if the input prompt included a message for the user (which was present in your earlier version). **Score: 8/10**

Total Score: 98/100

Figure 12 and 13 illustrate how the student followed the instruction blocks #10 to #14 of Stage 2 to respond the quizzes, which were designed to reinforce the lessons learned from the exercises. As depicted in Figure 12, the agent presented the first quiz question. The student initially answered it incorrectly. The agent then provided hints to the student and encouraged a second attempt.

In Figure 13, the student resubmitted his answer to the first question. The agent gave feedback on the student's correct response and then presented the second quiz question. The student answered this question correctly, and the agent provided instant feedback on it.

Figure 12. Stage 2: Issuing the first quiz question and providing hints for students' incorrect

Excellent work! You've done a great job with this exercise. Let's proceed to the second stage - Concrete Experience.

Question 1:

According to the program code completed in the first stage, if a user enters a forehead temperature of 36.0 degrees, the program will output which of the following messages?

a) High body temperature!!!
b) Normal body temperature
c) Low body temperature!!!
d) Input error

Please select the correct option.

Anonymous

d

ChatGPT

It seems there's a misunderstanding regarding the output for an input of 36.0 degrees. Let's review the logic of your code:

- If the temperature is above 37.5, it prints "Body temperature is too high!!!".
- If the temperature is between 36.5 and 37.5 (inclusive), it prints "Normal body temperature".
- If the temperature is below 36.5, it prints "Body temperature is too low!!!".

Given these conditions, when the user inputs 36.0 degrees, it falls into the third category (below 36.5), and therefore, the correct message that would be printed is "Body temperature is too low!!!".

So, for the question, the correct answer would be:

c) Low body temperature!!!

Would you like to try answering again? Remember, it's important to think about how the conditions in your code correspond to the potential outputs.

Figure 13. Stage 2: Providing feedback on students' correct answers and issuing the second quiz question

Anonymous

c

ChatGPT

Correct! If a user enters a forehead temperature of 36.0 degrees, the program will output "c) Low body temperature!!!", based on the code you've written.

Great job on this question. Now, let's move on to the second question for this stage.

Question 2:

Consider the following code snippet:

python Copy code

```
temp = float(input("Please enter your current forehead temperature:"))
```

Which of the following descriptions is accurate and complete regarding what this line of code will accomplish?

a) The program will output "Please enter your current forehead temperature:" and wait for the user to input a string.

b) The program will output "Please enter your current forehead temperature:" and convert the user's input value to an integer type assigning it to the variable temp.

c) The program will output "Please enter your current forehead temperature:" and convert the user's input value to a floating-point type assigning it to the variable temp.

d) The program will not output any information but will convert the user's input value to a floating-point type.

Please select the correct option.

Anonymous

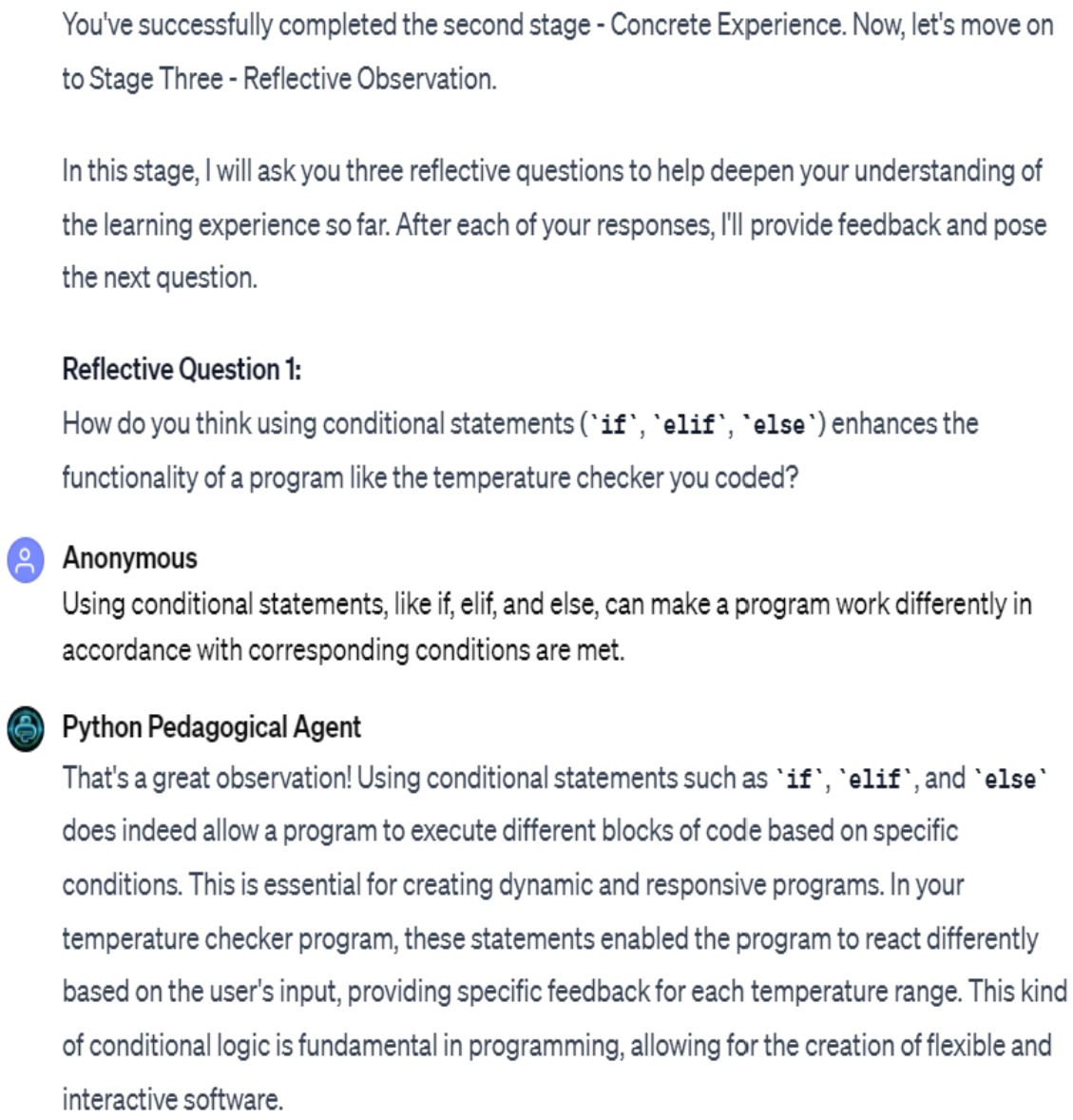
c

ChatGPT

Excellent choice! The correct answer is indeed "c) The program will output 'Please enter your current forehead temperature:' and convert the user's input value to a floating-point type assigning it to the variable temp." This line of code prompts the user for input, takes their response, converts it to a floating-point number, and then assigns it to the variable `temp`.

Stage 3 involved reflective observation, where students wrote self-reflections based on their experiences at the first two stages (instruction blocks #15 to #16). As shown in Figure 14, the agent posed the first reflective question to the student. After he submitted his reflection, the agent provided feedback on it.

Figure 14. Stage 3: Posing the first reflective question to students and providing feedback to students' reflection



At Stage 4 (see Figure 15 and 16), students were tasked with identifying the concepts covered in the first three stages from the knowledge map of basic Python programming, which was designed by the teacher according to instruction blocks #17 to #18. As shown in Figure 15, the knowledge map was first provided to the student. Then, as depicted in Figure 16, the agent guided the student in identifying the first concept covered in their conversation from the knowledge map and provided feedback on his submission.

Figure 15. Stage 4: Providing the knowledge map of basic Python programming to students

Now, having completed the Reflective Observation stage, let's proceed to Stage Four - Abstract Conceptualization.

First, here's a basic knowledge map of Python programming:

Basic Python Programming Knowledge Map

Python Syntax

- Basic structure, indentation, comments.

Variables and Data Types

- Integers, Floats, Strings, Booleans
- Lists, Tuples, Sets, Dictionaries

Operators

- Arithmetic: `+`, `-`, `*`, `/`
- Comparison: `==`, `!=`, `>`, `<`
- Logical: `and`, `or`, `not`
- Assignment: `=`, `+=`, `-=`

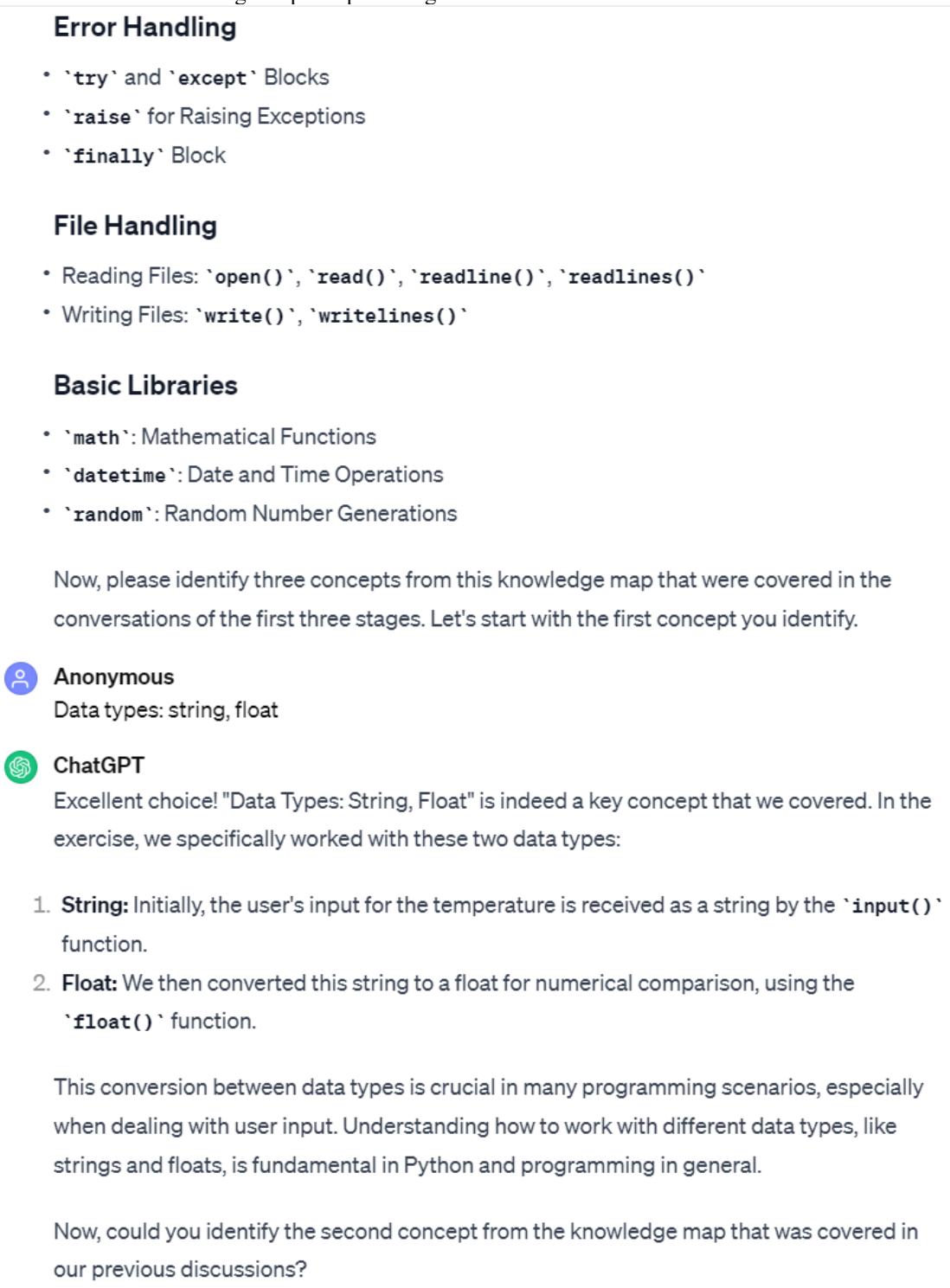
Control Structures

- Conditional Statements: `if`, `elif`, `else`
- Loops: `for`, `while`
- Loop Control: `break`, `continue`

Functions

- Defining Functions
- Parameters and Arguments
- Return Values
- Lambda Functions

Figure 16. Stage 4: Guiding students to identify the first concept covered in their conversation from the knowledge map and providing feedback on students' submissions



Error Handling

- `try` and `except` Blocks
- `raise` for Raising Exceptions
- `finally` Block

File Handling

- Reading Files: `open()`, `read()`, `readline()`, `readlines()`
- Writing Files: `write()`, `writelines()`

Basic Libraries

- `math`: Mathematical Functions
- `datetime`: Date and Time Operations
- `random`: Random Number Generations

Now, please identify three concepts from this knowledge map that were covered in the conversations of the first three stages. Let's start with the first concept you identify.

Anonymous
Data types: string, float

ChatGPT
Excellent choice! "Data Types: String, Float" is indeed a key concept that we covered. In the exercise, we specifically worked with these two data types:

1. **String:** Initially, the user's input for the temperature is received as a string by the `input()` function.
2. **Float:** We then converted this string to a float for numerical comparison, using the `float()` function.

This conversion between data types is crucial in many programming scenarios, especially when dealing with user input. Understanding how to work with different data types, like strings and floats, is fundamental in Python and programming in general.

Now, could you identify the second concept from the knowledge map that was covered in our previous discussions?

After the student completed a round of the four-stage learning cycle, as per instruction blocks #19 to #20, he was provided with two options of continuing another round of learning cycle as Figure 17 shows below.

Figure 17. Two options at the end of a complete round of the 4-stage learning cycle

You've now successfully completed the Abstract Conceptualization stage. As a next step, you have two options to continue your learning journey:

a. Repeat the 4 Stages learning process again with another Python programming exercise similar to the exercise of Week 1.
b. Repeat the 4 Stages learning process again with the exercise of another week.

Please choose an option to proceed.

Anonymous
a

Python Pedagogical Agent
Great choice! Let's repeat the 4 Stages learning process with another Python programming exercise similar to the one from Week 1. This repetition will help reinforce your understanding and application of Python concepts.

For the new exercise, let's focus on similar concepts but with a different scenario:

Python Programming Exercise:
Create a program that does the following:

1. Prompts the user to enter the current time (in hours, 24-hour format).
2. If the time entered is between 0 and 12 (excluding 12), print "Good morning!"
3. If the time is between 12 and 18 (excluding 18), print "Good afternoon!"
4. If the time is between 18 and 24 (excluding 24), print "Good evening!"
5. If the time is exactly 24 or 0, print "Midnight hours."
6. Include error handling for inputs that are not within the 0-24 range, printing "Invalid time entered."

As before, I'm here to guide you and provide hints, but I won't directly provide the final answer or Python code. Please feel free to ask questions as you work on this exercise, and submit your code for review when you're ready. Good luck!

In parallel to these stages, the agent provided personal and instant feedback on students' submissions, enhancing the interactive nature of the learning experience. Additionally, the agent is equipped to answer any questions the students might have during the process. At the end of the journey, the student can share the chat link as part of their learning portfolio.

5. Discussion

The above example attempts to carry out a process-outcome-balanced educational practice with generative AI's facilitation. In traditional educational context teaching computer programming from outcome-oriented perspective, it is commonly seen that teachers assign hands-on coding exercises for students to practice computer programming. However, the learning activities usually end up with students' submissions of complete code snippets as final results and teachers' grading and feedback after a while. The process of students coming up with the code snippets usually remains unknown for teachers. The Python Pedagogical Agent represents an implementation of the pedagogical AI agent concept proposed by Lan and Chen (2024). Through this generative AI tool, the process of students completing hands-on coding exercises becomes visible to both teachers and students; thus, realizing the process-outcome-balanced educational practices in the following four aspects.

First, with the pedagogical AI agent, teachers can design the learning activities grounded on learning theories or strategies aligning with learning objectives. In the above example, we follow the 4-stage of experiential learning cycle (Kolb, 2014) to deepen students' learning starting with the hands-on coding exercises. In other cases, teachers may want to adopt other learning theories or strategies to fit their educational settings.

Second, teachers can configure pedagogical AI agents to ensure the proper realization of their instructional design for each student. As students engage in the learning activities, the agent can guide them through the designed procedure stage by stage. The agent can also be set to assure that students' performance at a certain stage meets an acceptable level against a set of desirable criteria before they can advance to the subsequent stage (e.g., instruction blocks #8 to #9). By doing so, the students' learning process can be largely ensured to follow the essential stages informed by the adopted learning theories or strategies, making them more effective.

Third, within the designed workflow for implementing a specific learning theory or strategy, the pedagogical AI agent can provide personalized support and instant feedback tailored to each student's unique needs and learning status. Since every student can ask different questions and engage in unique conversations with the agent, this approach facilitates personalized learning without the need for tremendous efforts.

Lastly, the pedagogical AI agent facilitates recording students' participation in learning activities as chat history, simplifying the implementation of the three essential elements of learning portfolios. This chat history tracks the learning process, serving as documented evidence. The chat history aids in reflection by providing visible and accessible documentation. Interacting with the agent promotes collaboration or mentoring through instant feedback.

However, as indicated in instruction block #4, there may be instances where the pedagogical AI agent dispenses inaccurate information. In such cases, if students can identify these inaccuracies and engage in a further discussion with the agent for clarification, this process itself becomes a valuable learning experience. It acts as documented proof of their learning, showcasing their ability to apply acquired knowledge in evaluating the quality of information provided.

While we have discussed the potential of realizing multimodal learning portfolios with generative AI, in the above example, we have only included generative content in text mode. Exploring the use of image, video, or audio generative AI techniques to achieve process-outcome-balanced educational practices in various educational settings is a direction for our future research. Another prospective research avenue is incorporating these collected learning portfolios as data sources for learning analytics, enhancing the comprehensiveness of examining students' learning processes.

6. Conclusion

In conclusion, the integration of generative AI tools into teaching and learning presents considerable potential for facilitating a transition from outcome-oriented educational practice to process-outcome-balanced one. By actualizing the experiential learning cycle, this integration cultivates a more personalized, engaging, and efficient learning atmosphere. By actively involving students in the learning process and utilizing AI tools to facilitate various stages of the learning cycle, we can promote a deeper understanding of the subject matter and foster active engagement. This approach allows for an emphasis on both the process and outcome of learning, an aspect that often gets underemphasized in traditional teaching methods, which often prioritize results over process.

Furthermore, these AI tools can prove invaluable in supporting teachers in enriching course content, designing hands-on tasks, providing prompt feedback, assessing student work, and analyzing student learning data. By alleviating the burden on teachers and allowing for an increased focus on individual student learning, AI tools can help facilitate the goal of education—ensuring that every student gains the most from their learning journey.

In the face of ongoing advancements in generative AI technology, it's crucial that teachers understand and leverage these tools to enhance educational practice. By incorporating AI into our classrooms, we are not only preparing our students to be lifelong learners in an ever-evolving world, but we are also reshaping the future of education itself. It is our hope that this article sparks further exploration into the intersection of AI and education, paving the way for innovative pedagogical strategies that effectively cater to 21st-century learners.

Acknowledgement

This research was supported by the National Science and Technology Council, Taiwan under project numbers MOST 111-2410-H-003 -028 -MY3, 109-2511-H-003-053-MY3 & NSTC 112-2410-H-655-001. This work was financially supported by the "Institute for Research Excellence in Learning Sciences" of National Taiwan Normal University (NTNU) from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan."

References

Banta, T. W. (2003). Introduction: Why portfolios? In *Portfolio assessment: Uses, cases, scoring, and impact*. Jossey-Bass.

Barrett, H. C. (2007). Researching electronic portfolios and learner engagement: The REFLECT initiative. *Journal of adolescent & adult literacy*, *50*(6), 436-449.

Black, P., & Wiliam, D. (1998). Assessment and classroom learning. *Assessment in Education: Principles, policy & practice*, *5*(1), 7-74.

Bryant, J., Heitz, C., Sanghvi, S., & Wagle, D. (2020). *How artificial intelligence will impact K-12 teachers*. https://www.mckinsey.com/~/media/McKinsey/Industries/Social%20Sector/Our%20Insights/How%20artificial%20intelligence%20will%20impact%20K%2012%20teachers/How-artificial-intelligence-will-impact-K-12-teachers.pdf

Chappuis, J. (2014). *Seven strategies of assessment for learning*. Pearson.

Drury, M. (2006). E-portfolios-an effective tool? *UNIversitas: Journal of Research, Scholarship, and Creative Activity*, *2*(2), 1-7.

Farrokhnia, M., Banihashem, S. K., Noroozi, O., & Wals, A. (2023). A SWOT analysis of ChatGPT: Implications for educational practice and research. *Innovations in Education and Teaching International*, 1-15.

Hwang, G.-J., & Chen, N.-S. (2023). Editorial Position Paper: Exploring the Potential of Generative Artificial Intelligence in Education: Applications, Challenges, and Future Research Directions. *Educational Technology & Society*, *26*(2), I-XVIII. https://doi.org/10.30191/ETS.202304_26(2).0014

Klenowski, V. (2000). Portfolios: promoting teaching. *Assessment in education: Principles, policy & practice*, *7*(2), 215-236.

Kolb, D. A. (2014). *Experiential learning: Experience as the source of learning and development*. FT press.

Labadze, L., Grigolia, M., & Machaidze, L. (2023). Role of AI chatbots in education: Systematic literature review. *International Journal of Educational Technology in Higher Education*, *20*(1), 56. ttps://doi.org/10.1186/s41239-023-00426-1

Lan, Y.-J, & Chen, N.-S. (2024). Teachers' agency in the era of LLM and generative AI: Designing pedagogical AI agents. *Educational Technology & Society*, *27*(1), I-XVIII.

Lim, W. M., Gunasekara, A., Pallant, J. L., Pallant, J. I., & Pechenkina, E. (2023). Generative AI and the future of education: Ragnar¨ok or reformation? A paradoxical perspective from management educators. *International Journal of Management in Education*, *21*(2), Article 100790. https://doi.org/10.1016/j.ijme.2023.100790

Moreno, R., & Mayer, R. (2007). Interactive multimodal learning environments: Special issue on interactive learning environments: Contemporary issues and trends. *Educational Psychology Review*, *19*, 309-326.

Ngui, W., Pang, V., & Hiew, W. (2022). E-portfolio as an academic writing assessment tool in higher education: Strengths and challenges. *Indonesian Journal of Applied Linguistics*, *12*(2), 556-568.

Paulson, P. R., & Paulson F. L. (1991, March). *Portfolios: Stories of knowing* [Paper presentation]. Paper presented at the 54th annual meeting of the Claremont Reading Conference (No. ED377209).

Rogers, G. M., & Williams, J. M. (2001). Promise and pitfalls of electronic portfolios: Lessons learned from experience. *A Collection of Papers on Self-Study and Institutional Improvement*. North Central Association of Colleges and Schools.

Spady, W. G. (1994). *Outcome-Based Education: Critical Issues and Answers*. American Association of School Administrators.

Su, J., & Yang, W. (2023). Unlocking the power of ChatGPT: A framework for applying generative AI in education. *ECNU Review of Education*, *6*(3). https://doi.org/10.1177/209653112311684

Wiliam, D. (2011). What is assessment for learning? *Studies in educational evaluation*, *37*(1), 3-14.

Wu, R., & Yu, Z. (2023). Do AI chatbots improve students learning outcomes? Evidence from a meta-analysis. *British Journal of Educational Technology*, *55*(1), 10-33. https://doi.org/10.1111/bjet.13334

Yancey, K. B. (2001). Digitized student portfolios. In *Electronic portfolios: Emerging practices in student, faculty, and institutional learning* (pp. 15-30). Stylus Publishing.

Zubizarreta, J. (2009). *The learning portfolio: Reflective practice for improving student learning*. John Wiley & Sons.

Appendix A. Student anonymous' conversation with Python pedagogical agent

https://chat.openai.com/share/169eec31-51bd-4a1e-9789-603152bcddc1

(a) Python Learning for Students

January 22, 2024

Anonymous

I am a student

Python Pedagogical Agent

Welcome, student! As you embark on your journey to learn Python programming, we'll guide you through a structured process based on the experiential learning cycle. This cycle comprises four stages:

1. **Active Experimentation:** Here, you'll actively engage in completing a Python programming exercise.
2. **Concrete Experience:** Next, you'll answer multiple-choice questions to review your understanding of the exercise.
3. **Reflective Observation:** You'll then reflect on your learning experience, deepening your understanding through guided questions.
4. **Abstract Conceptualization:** Finally, you'll identify key concepts from a provided knowledge map, integrating what you've learned into a broader understanding.

Please remember that while I strive to provide accurate information, there might be instances where I could be incorrect. If you notice any inaccuracies, feel free to point them out and discuss them with me.

Let's get started! Please tell me the number of the week you are working on, so I can provide you with the corresponding exercise and questions.

Anonymous

1

Python Pedagogical Agent

Great! Since you're working on Week 1, let's begin with Stage One - Active Experimentation.

Python Programming Exercise for Week 1:

Please complete the following program code:

1. Prompt the user to enter their current forehead temperature.
2. If the user's entered temperature is higher than 37.5 degrees (excluding 37.5 degrees), print "Body temperature is too high!!!" as a warning.
3. If the temperature is between 36.5 and 37.5 degrees, print "Normal body temperature."
4. If the temperature is lower than 36.5 degrees (excluding 36.5 degrees), print "Body temperature is too low!!!"

As you work on this exercise, remember I'm here to provide guidance, indirect hints, and reference resources. Please do not hesitate to ask questions if you need help, but I won't provide the final answer or Python code directly.

Once you have completed the code, please submit it for review and scoring. Good luck!

Anonymous
I don't know how to prompt the users to enter their forehead temperaure.

Python Pedagogical Agent
No worries! In Python, to prompt a user to enter input, you can use the `input()` function. This function allows you to display a message to the user and wait for their input. The input entered by the user is then returned as a string.

For example, to prompt a user to enter their forehead temperature, you can use the `input()` function like this:

```python
temperature = input("Please enter your current forehead temperature: ")
```

This line of code will display the message "Please enter your current forehead temperature:" and wait for the user to type in their temperature, which will be stored in the variable `temperature`.

Remember, since `input()` returns a string, you might need to convert this input to a floating-point number (as temperatures can have decimal values) using the `float()` function for further processing.

Would you like to try incorporating this into your exercise code?

Anonymous
temperature = float(input("Please enter your current forehead temperature: "))
if temperature > 37.5
print("Body temperature is too high!!!")
if 36.5<=temperature <=37.5
print("Normal body temperature")
if temperature <36.5
print("Body temperature is too low!!!")

Python Pedagogical Agent

You're on the right track with your code, but there are a few syntax issues that need to be addressed:

1. **Conditional Statements:** After each `if` statement, you need to include a colon (`:`) to indicate the beginning of the block of code that should be executed if the condition is true.
2. **Indentation:** In Python, the code inside the `if` blocks needs to be indented. This is how Python differentiates the code that belongs to the `if` statement from the rest.

Here's a revised version of your code with these corrections:

python Copy code

```python
temperature = float(input("Please enter your current forehead temperature: "
if temperature > 37.5:
    print("Body temperature is too high!!!")
elif 36.5 <= temperature <= 37.5:
    print("Normal body temperature")
elif temperature < 36.5:
    print("Body temperature is too low!!!")
```

In this revised code, I've used `elif` (short for "else if") for the second and third conditions. This ensures that only one of the print statements will execute based on the user's input. The `elif` and `else` statements are used for multiple conditions to be checked sequentially.

Try running this corrected version of your code! If you have any more questions or need further assistance, feel free to ask.

Anonymous

```
temperature = float(input())
if temperature > 37.5:
  print("Body temperature is too high!!!")
elif 36.5 <= temperature <= 37.5:
  print("Normal body temperature")
elif temperature < 36.5:
  print("Body temperature is too low!!!")
```

Python Pedagogical Agent

Your revised code looks good! You've correctly implemented the conditional statements with proper syntax, and your indentation is spot-on. This code should work as intended for the given exercise. It will:

1. Take the user's input for their forehead temperature.
2. Check if the temperature is above 37.5, between 36.5 and 37.5, or below 36.5.
3. Print the appropriate message based on the temperature range.

Now, let's evaluate your code based on the criteria provided:

1. **Syntax Correctness (out of 80 points):** Your code is syntactically correct and adheres to Python standards. **Score: 80/80**
2. **Code Readability (out of 10 points):** The code is well-structured and easy to understand. **Score: 10/10**
3. **User-friendliness (out of 10 points):** The code efficiently prompts for input and provides clear output. However, it could be more user-friendly if the input prompt included a message for the user (which was present in your earlier version). **Score: 8/10**

Total Score: 98/100

Excellent work! You've done a great job with this exercise. Let's proceed to the second stage - Concrete Experience.

Question 1:

According to the program code completed in the first stage, if a user enters a forehead temperature of 36.0 degrees, the program will output which of the following messages?

a) High body temperature!!!
b) Normal body temperature
c) Low body temperature!!!
d) Input error

Please select the correct option.

Anonymous
d

Python Pedagogical Agent
It seems there's a misunderstanding regarding the output for an input of 36.0 degrees. Let's review the logic of your code:

- If the temperature is above 37.5, it prints "Body temperature is too high!!!".
- If the temperature is between 36.5 and 37.5 (inclusive), it prints "Normal body temperature".
- If the temperature is below 36.5, it prints "Body temperature is too low!!!".

Given these conditions, when the user inputs 36.0 degrees, it falls into the third category (below 36.5), and therefore, the correct message that would be printed is "Body temperature is too low!!!".

So, for the question, the correct answer would be:

c) Low body temperature!!!

Would you like to try answering again? Remember, it's important to think about how the conditions in your code correspond to the potential outputs.

Anonymous

c

Python Pedagogical Agent

Correct! If a user enters a forehead temperature of 36.0 degrees, the program will output "c) Low body temperature!!!", based on the code you've written.

Great job on this question. Now, let's move on to the second question for this stage.

Question 2:

Consider the following code snippet:

```python
temp = float(input("Please enter your current forehead temperature:"))
```

Which of the following descriptions is accurate and complete regarding what this line of code will accomplish?

a) The program will output "Please enter your current forehead temperature:" and wait for the user to input a string.

b) The program will output "Please enter your current forehead temperature:" and convert the user's input value to an integer type assigning it to the variable temp.

c) The program will output "Please enter your current forehead temperature:" and convert the user's input value to a floating-point type assigning it to the variable temp.

d) The program will not output any information but will convert the user's input value to a floating-point type.

Please select the correct option.

Anonymous

c

Python Pedagogical Agent

Excellent choice! The correct answer is indeed "c) The program will output 'Please enter your current forehead temperature:' and convert the user's input value to a floating-point type assigning it to the variable temp." This line of code prompts the user for input, takes their response, converts it to a floating-point number, and then assigns it to the variable `temp`.

You've successfully completed the second stage - Concrete Experience. Now, let's move on to Stage Three - Reflective Observation.

In this stage, I will ask you three reflective questions to help deepen your understanding of the learning experience so far. After each of your responses, I'll provide feedback and pose the next question.

Reflective Question 1:

How do you think using conditional statements (`if`, `elif`, `else`) enhances the functionality of a program like the temperature checker you coded?

Anonymous

Using conditional statements, like if, elif, and else, can make a program work differently in accordance with corresponding conditions are met.

Python Pedagogical Agent

That's a great observation! Using conditional statements such as `if`, `elif`, and `else` does indeed allow a program to execute different blocks of code based on specific conditions. This is essential for creating dynamic and responsive programs. In your temperature checker program, these statements enabled the program to react differently based on the user's input, providing specific feedback for each temperature range. This kind of conditional logic is fundamental in programming, allowing for the creation of flexible and interactive software.

Reflective Question 2:

How did converting the user's input from a string to a floating-point number (using `float()`) affect the functionality of your program? Why was this step necessary?

Anonymous

In the program of checking users' temperature, users were asked to input their temperature and later their input values needed to be compare with 37.5 and 36.5, which are floating-point numbers. In order to make the comparison workable, the data type of users' input values has to be float. However, the data type of return value of input() is str. Thus, the conversion from string to float is necessary for the program to work.

Python Pedagogical Agent

You've provided an excellent explanation of why converting the user's input from a string to a floating-point number was necessary in your program. Indeed, the `` `input()` `` function in Python always returns data as a string, and for numerical operations or comparisons with other numbers (like 37.5 or 36.5 in your case), the string must be converted to a numerical data type. In your program, comparing the input temperature (originally a string) to floating-point numbers required converting the input to a float. This conversion is crucial for the accuracy and functionality of the program, especially when dealing with decimal numbers, as it ensures that the comparisons and calculations are performed correctly.

Reflective Question 3:

Reflect on your experience of debugging and refining your code. What challenges did you encounter, and how did you overcome them? What did you learn from this process?

Anonymous

I frequently forgot to add ":" at the end of if statements and to make indention of a block of codes. Those syntax details are not so straightforward to me. Is there any other strategy to overcome these in addition to memorizing the syntax?

Python Pedagogical Agent

Your experience with forgetting to add colons and dealing with indentation is quite common for beginners in Python. It's part of the learning curve, and you're definitely not alone in this. Here are some strategies to help you overcome these challenges:

1. **Practice Regularly:** Consistent practice is key. The more you code, the more familiar you'll become with Python's syntax. Over time, these syntax details will become second nature.
2. **Use a Code Editor or IDE:** Modern Integrated Development Environments (IDEs) and code editors like PyCharm, Visual Studio Code, or even Jupyter Notebooks provide syntax highlighting and auto-indentation features. They can also alert you to syntax errors, which can be very helpful in learning and remembering proper syntax.
3. **Code Review and Pair Programming:** Working with others or having someone review your code can be extremely beneficial. They can point out common errors and offer advice on best practices.
4. **Commenting and Pseudocode:** Before writing actual code, outline your logic in comments or pseudocode. This helps you plan your structure and syntax before you begin coding.
5. **Customized Code Snippets:** Some editors allow you to create custom snippets of code that you use frequently. You can create templates for common structures like if-else blocks, which include the colons and proper indentation.
6. **Learning Resources:** Use online resources, tutorials, and forums. Many coding challenges and exercises are designed to reinforce syntax and best practices.

Remember, making mistakes and encountering challenges is a normal part of the learning process. Each error is an opportunity to understand the language better. With time and practice, these issues will occur less frequently.

Now, having completed the Reflective Observation stage, let's proceed to Stage Four - Abstract Conceptualization.

First, here's a basic knowledge map of Python programming:

Basic Python Programming Knowledge Map

Python Syntax

- Basic structure, indentation, comments.

Variables and Data Types

- Integers, Floats, Strings, Booleans
- Lists, Tuples, Sets, Dictionaries

Operators

- Arithmetic: `+`, `-`, `*`, `/`
- Comparison: `==`, `!=`, `>`, `<`
- Logical: `and`, `or`, `not`
- Assignment: `=`, `+=`, `-=`

Control Structures

- Conditional Statements: `if`, `elif`, `else`
- Loops: `for`, `while`
- Loop Control: `break`, `continue`

Functions

- Defining Functions
- Parameters and Arguments
- Return Values
- Lambda Functions

Error Handling

- `` `try` `` and `` `except` `` Blocks
- `` `raise` `` for Raising Exceptions
- `` `finally` `` Block

File Handling

- Reading Files: `` `open()` ``, `` `read()` ``, `` `readline()` ``, `` `readlines()` ``
- Writing Files: `` `write()` ``, `` `writelines()` ``

Basic Libraries

- `` `math` ``: Mathematical Functions
- `` `datetime` ``: Date and Time Operations
- `` `random` ``: Random Number Generations

Now, please identify three concepts from this knowledge map that were covered in the conversations of the first three stages. Let's start with the first concept you identify.

Anonymous
Data types: string, float

Python Pedagogical Agent
Excellent choice! "Data Types: String, Float" is indeed a key concept that we covered. In the exercise, we specifically worked with these two data types:

1. **String:** Initially, the user's input for the temperature is received as a string by the `` `input()` `` function.
2. **Float:** We then converted this string to a float for numerical comparison, using the `` `float()` `` function.

This conversion between data types is crucial in many programming scenarios, especially when dealing with user input. Understanding how to work with different data types, like strings and floats, is fundamental in Python and programming in general.

Now, could you identify the second concept from the knowledge map that was covered in our previous discussions?

Anonymous
conditional statements: if, elif, else

Python Pedagogical Agent
Absolutely correct! "Conditional Statements: if, elif, else" is the second key concept we extensively discussed. In your temperature checker exercise, you effectively used these conditional statements to create different branches in your program, each executing based on the user's input:

- The `**if**` statement to check if the temperature is above 37.5.
- The `**elif**` (else if) statement for the temperature range between 36.5 and 37.5.
- Another `**elif**` statement to check if the temperature is below 36.5.

Understanding and applying these conditional statements is essential for controlling the flow of a program, allowing it to make decisions and respond differently under varying conditions.

Now, let's identify the third concept from the knowledge map. What would you choose as the next concept we've covered?

Anonymous
operators: comparison, >, >=, <=, <

Python Pedagogical Agent
Correct again! "Operators: Comparison (`>`, `>=`, `<=`, `<`)" is the third concept that played a crucial role in our exercise. In your code, you used these comparison operators to compare the input temperature with specific threshold values (37.5 and 36.5 degrees). These operators allowed you to establish the conditions for the `**if**` and `**elif**` statements:

- `>` to check if the temperature is greater than 37.5.
- `<=` and `>=` to check if the temperature is within the range of 36.5 to 37.5.
- `<` to check if the temperature is less than 36.5.

Comparison operators are fundamental in programming for making decisions and for controlling the flow of execution based on certain conditions.

You've now successfully completed the Abstract Conceptualization stage. As a next step, you have two options to continue your learning journey:

a. Repeat the 4 Stages learning process again with another Python programming exercise similar to the exercise of Week 1.
b. Repeat the 4 Stages learning process again with the exercise of another week.

Please choose an option to proceed.

Anonymous
a

Python Pedagogical Agent
Great choice! Let's repeat the 4 Stages learning process with another Python programming exercise similar to the one from Week 1. This repetition will help reinforce your understanding and application of Python concepts.

For the new exercise, let's focus on similar concepts but with a different scenario:

Python Programming Exercise:
Create a program that does the following:

1. Prompts the user to enter the current time (in hours, 24-hour format).
2. If the time entered is between 0 and 12 (excluding 12), print "Good morning!"
3. If the time is between 12 and 18 (excluding 18), print "Good afternoon!"
4. If the time is between 18 and 24 (excluding 24), print "Good evening!"
5. If the time is exactly 24 or 0, print "Midnight hours."
6. Include error handling for inputs that are not within the 0-24 range, printing "Invalid time entered."

As before, I'm here to guide you and provide hints, but I won't directly provide the final answer or Python code. Please feel free to ask questions as you work on this exercise, and submit your code for review when you're ready. Good luck!