Name(s)_____Madison/Wendy/Benji_____ Period ___D__ Date _____

Practice PT Overview and Rubric - Python

Overview

You will submit this project and write responses to the reflection questions in the style of the AP® Create Performance Task. The document below has been constructed to mimic the AP Create Performance Task. Some but not all of the language is pulled directly from the AP document. Some of the prompts have been modified slightly or simply omitted for clarity and to better fit the *Python Programming* project.

Programming Requirements

Process

The process of creating your program includes individual and collaborative work. Individual work means some portions of the design, development, and implementation of your program must be completed independently. Collaboration can take different forms and can occur at different times in the program development process.

You will be required to respond to prompts about your collaboration, as well as to identify the portions of your program that were created independently. The following are examples of different forms of collaboration:

- **A.** Collaboration can take the form of brainstorming and sharing ideas before the process of writing code begins. Partners can then choose to work together or independently at selected times during the programming process.
- **B.** Collaboration can take the form of working together to develop an idea, beginning the programming process together, and then working independently to add different features to the collaboratively-developed portion of the program.
- **C.** Collaboration can involve Pair Programming, in which one partner “drives” (enters code) while the other “navigates” (recommends and reviews code entered by driver), with the partners changing roles after designated time intervals.
- **D.** Collaboration can involve each partner developing pieces of the program and combining those pieces during the development process.
- **E.** Collaboration can blend any or all of the above techniques and may include an iterative process in which one or more of these techniques, or other collaboration techniques, are employed several times in the program design, development, and testing phases.

Program

Your program must demonstrate a variety of capabilities and implement several different language features that, when combined, produce a result that cannot be easily accomplished without computing tools and techniques. **You will be required to respond to prompts about the program development process and your program code, including questions about the abstractions you used. The program must demonstrate:**

- Use of several effectively integrated programming elements from the programming language you are using
- Use and creation of abstractions to manage the complexity of your program (e.g., functions/procedures; abstractions provided by the programming language; APIs)

Submission Requirements

1. Group Planning Document

As a group, you will submit a **single copy** of your group planning document. Make sure that all group members' names are listed on the document.

2. Program Code

When you have completed your program, submit it to your teacher by sharing via Google or GitHub. Submitting indicates to the teacher that your project is ready to be reviewed.

Your final program **code should:**

- contain the functions you wrote and the functions you got from your teammates
- make calls to both the functions you wrote and your teammates' functions in your program.

Note: It is OK if you had to alter your teammates' code slightly to make it work in your program, or to improve its functionality.

3. Individual Written Responses

After completing your project, respond to each of the following reflection questions. **Your response to any one prompt must not exceed 300 words.**

a. Provide an overview of the purpose of your program and how your program code works. Describe the most important program features, rather than providing a line-by-line summary of the program code.

b. Describe the most difficult programming problem you encountered while writing your individual code. What was the difficulty? Explain how you resolved it.

c. Identify an abstraction used in your program and explain how it helped manage the complexity of your program.

d. Explain in detail points in your development process where collaboration was used.
    - Describe the form of collaboration you used. Refer to Process section A-E in your description.
    - Explain how this collaboration affected your program development. Cite specific examples from the collaboration, such as how the group worked together to arrive at solutions, or feedback that you gave and received.

Rubric - Python

Component	1	2	3	Score
Group Planning Document				
Project Design	The description and/or sketch/digital image of the design are simplistic and lacking in details. Not enough information is given to realistically build a program from.	The description and/or sketch/digital image are limited in details. While it might be possible to program from the design, there are too many details missing for the programming task to be easy.	The description and/or sketch/digital image are rich in details. A programmer would have few questions and find it easy to work from this design.	
Top-Down Design	The image has not been broken into logical components, or most components have not been assigned a high-level function with a descriptive name.	Most aspects of the image have been broken into components; however, the components are not distinct or logical and do not represent top-down design, or the functions are poorly named.	The image has been broken into logical components that represent top-down design; each component has been assigned a high-level function with a descriptive name.	
Task Assignments	Tasks have not been evenly divided among team members and/or tasks have not been prioritized.	Tasks have been divided among members but the assignments and prioritizations do not reflect a realistic estimate of the time constraints or anticipate problems that might arise.	Tasks have been prioritized and evenly divided among members with considerations made for timing or problems that might arise.	
Program Code				
Functions and Abstraction	The program does not make use of the high-level functions agreed upon by the team.	The program makes limited or inconsistent use of levels of abstraction and the functions created by the team members.	Appropriate levels of abstraction are expressed in code using the high-level functions created by the team. (Modifications of functions are allowed.)	
Functions with Parameters	The program does not feature a function with a parameter.	Program contains a function with a parameter that is used in a limited or trivial way.	Program contains at least one function with a parameter to control behavior in a meaningful way.	
Loops	The program does not feature a loop.	A loop is used in a limited or trivial way to repeatedly execute portions of code.	A loop is used in a meaningful way to repeatedly execute portions of code.	

Functionality	There is a weak connection between the output of your function(s) and the function descriptions agreed upon by the group.	There is a moderate connection between the output of your function(s) and the function descriptions agreed upon by the group.	There is a clear and obvious connection between the output of your function(s) and the function descriptions agreed upon by the group.	
Final Program Incorporates Work from Teammates	Final program does not make use of code written by other teammates.	Final program uses code written by some of the other teammates.	Final program uses code written by each of the teammates. (Note: Students may make alterations to their teammates' code as needed to function correctly in the final program.)	
Individual Written Responses				
a. Program Overview	The connection between the program and its purpose is unclear. Or it is unclear how the program features connect to the purpose.	There is a logical connection between the program and its purpose. Or the purpose of the program is weakly supported by the features identified.	There is a compelling connection between the program and its stated purpose, supported by details of the important features identified.	
b. Difficulties Encountered	The response generally describes the development of the program without clearly describing any specific problems.	The response describes the developmental steps of the program, but it includes little or no information about how any problems were addressed.	The response fully describes the development details that enable the reader to understand the the difficulties that were encountered and how they were resolved.	
c. Abstraction	The explanation of how the selected code illustrates abstraction is incorrect or incomplete.	The explanation of how the selected code illustrates abstraction is mostly complete but lacks a clear explanation of how it helps manage the complexity of the program.	The explanation of how the selected code illustrates abstraction is well-supported by details and clearly describes how it helps manage the complexity of the program.	
d. Collaboration	The response describes a non-collaborative process, or an ineffective collaborative process. The explanation does not describe how the process affected the program development.	The response cites some effective collaboration, but it is unclear how the collaboration affected the program development or any feedback was provided or incorporated.	The response describes effective collaboration and cites specific examples of how collaboration impacted the program development. Examples of providing and incorporating feedback are included.	