Teacher: Jessica Novillo Argudo, Jing Xue, Richard Parker
Unit Plan: Introduction to Python
Topic of the Lesson: Summative Assessment / Building a Quiz App
Grade and Content: 10th - 12th / Computer Science
Timing/Pace: 3 class periods (45 min each)

Learning Objective:
Students will be able to demonstrate mastery of the previous lessons by building 3-4 different Quiz Apps.

NYS Standards:

9-12.CT.2 Collect and evaluate data from multiple sources for use in a computational artifact.

9-12.CT.5 Modify a function or procedure in a program to perform its computation in a different way over the same inputs, while preserving the result of the overall program.

9-12.CT.8 Develop a program that effectively uses control structures in order to create a computer program for practical intent, personal expression, or to address a societal issue.

9-12.CT.9 Systematically test and refine programs using a range of test cases, based on anticipating common errors and user behavior

Lesson Abstract:

Students have been learning how to program in Python using variables, conditionals, expressions and loops. In this final lesson, students will demonstrate mastery of the skills/ concepts learned by building a quiz app. Students are familiar with the quiz app from Snap and because this unit is meant to help transition students from block-based programming to text-based programming, this assessment is appropriate for gauging student mastery of this introductory unit.

Content-specific vocabulary/ concepts:

- variables
- conditionals
- lists
- loops
- boolean expressions
- casting
- user input
- sequencing
- concatenation
- print statements

Materials/Resources:

- Internet connection
- laptops
- Replit
- Nearpod
- smart board

Formative Assessments:

- journal/ programming log
- Nearpod collaborative board and polls

Summative Assessment:

Quiz App program

- quiz_app_basic
- quiz_app_intermediate
- quiz_app_advanced
- quiz_app_challenge

Warm-up/ Mini lesson:

(Day 1) Students will load their Quiz App Snap programs from earlier in the year and into their programming logs write out an approximation of the algorithm for their quiz program. Students will describe what the quiz app is doing.

Activity / Sequence of Lesson:

Class will discuss elements of the Snap Quiz App program. Students will view a Google slide presentation that summarizes some of the previous lessons by looking at how Snap blocks translate to Python. After viewing and discussing the presentation, students will be asked to open their Repls and create a Python file called quiz_app_basic.

Days 1-3: Teacher will have the class join a Nearpod. The Nearpod collaborative board will be present on the smart board and students will be instructed to post any clarifying questions to the board to be addressed by the teacher during class. At the end of the class, students will answer a poll question on how much they completed.

Day 1-2: Students will be instructed to complete a **basic quiz app** with the following guidelines:

- In Replit, create a new file called quiz_app_basic.py
- create a list of lists. Each inner list should have two elements - the question and the answer. students are to have at least 4-5 sets of questions and answers.
- create a variable called score and set it to 0
- set that list to a variable name that is descriptive of the sets of questions and answers
- create a variable for user input
- create a for loop that iterates through the questions and checks user input to determine whether the answer is correct or not
- if the answer is not correct, the program should let the user know
- if the answer is correct, the score should increase by one
- The program should report the score after all the questions have been answered
- students are to complete their daily programming log/ journal

Day 2: Students will be instructed to complete a **intermediate quiz app** with the following guidelines:

- In Replit, create a new file called quiz_app_intermediate.py
- You may copy and paste the previous quiz app into this version
- Instead of changing/setting the score by one for a correct response, develop and implement logic that will change or set the score based on a grading scale of 0-100.
- Create a variable that will track the question number and should be printed for each question when asked (Question 1: How are you? Question 2: What time is it?...)
- Develop the program so that if the user gets the question incorrect the program will inform the user of the correct answer.
- The program should report the score after all the questions have been answered

Day 3: Students will be instructed to complete a **advanced quiz app and quiz app challenge** with the following guidelines:

- In Replit, create a new file called quiz_app_advanced.py
- You may copy and paste the previous quiz app into this version
- Create a variable to keep track of the user's attempts
- Give the user 2-3 attempts at answering the question
- After the second or third attempt, inform the user of the correct answer
- The program should report the score after all the questions have been answered

Day 3 Challenge 1: If students have completed all three versions of the quiz app, students will complete the following challenges by adding functionality to their program with the following guidelines:

- Use the advanced quiz app to add functionality, no need for a new file
- Amend your question and answer data structure by either adding a list as a third element to each sublist or amend the second item of each sublist by inserting a list of “acceptable responses”
- If the user responds to the current question with any of the items in the answer list, the program should recognize it as a correct response
- Maintain the functionality of the intermediate and advance quiz apps by reporting the correct answer if the user gets it wrong after 2-3 attempts

Day 3 Challenge 2: If students have completed all three versions of the quiz app and the 1st challenge, students will directed to complete the 2nd challenge with the following guidelines:

- Continue to use the advanced quiz app to add functionality, no need for a new file
- Develop your own functionality to the quiz, no guidelines for what you may do. This challenge is so that you can be creative and decide what you would like to add. For example, you could develop code that gave the user a hint after an incorrect response.

Summary / Next Steps / Exit Slip:

- Students will submit quiz_app_basic and the teacher will use the rubric checklist to either give or not give an exit ticket for the next version of the quiz app. All students will have to receive checks for all 9 rubric categories to move onto the next quiz app.
- Students who complete all 3 quiz apps will be given challenges to further improve the quiz app.
- The goal is for students to complete as many, if not all of the quiz apps and challenges.

Quiz App Scoring and Rubric Checklist

No submission = 0

Approaching Quiz App (not complete) = 40-60

Quiz App Basic = 65 - 75

Quiz App Intermediate = 75 - 85

Quiz App Advanced = 85 - 95

Quiz App Challenge = 100

Students will receive checks for each skill category satisfactorily demonstrated

RUBRIC CHECKLIST	Quiz Apps >>>>>	BASIC	INTERMEDIATE	ADVANCED	CHALLENGES 1-2
variables	• Student is able to create and implement variables in the program				
functions	• Student is able to create and define a function				
conditionals	• Student is able to create an if /if else statement with proper syntax, indentation and case to be tested using operators and/or boolean expressions				
loops	• Student is able to create a for/ while loop with proper syntax, indentation and for while loop - a proper exit condition				

lists	• Student is able to make a list and fill it with values such as another list(sublist) for this program to function as intended.				
concatenation	• Student is able to demonstrate concatenation by casting variables and using +				
user input	• Student is able to take input from the user and implement it properly in the program for it to function as intended.				
sequencing	• Student is able to sequence the program in a way that allows the program to function as intended.				
print statements	• Student is able to execute print statements for the program to function as intended.				