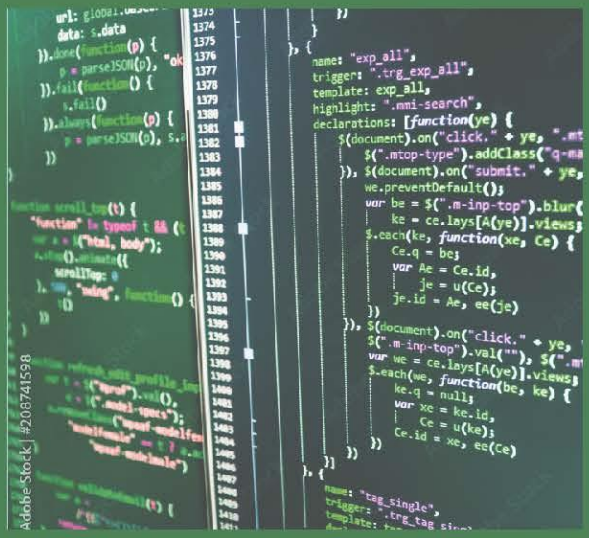
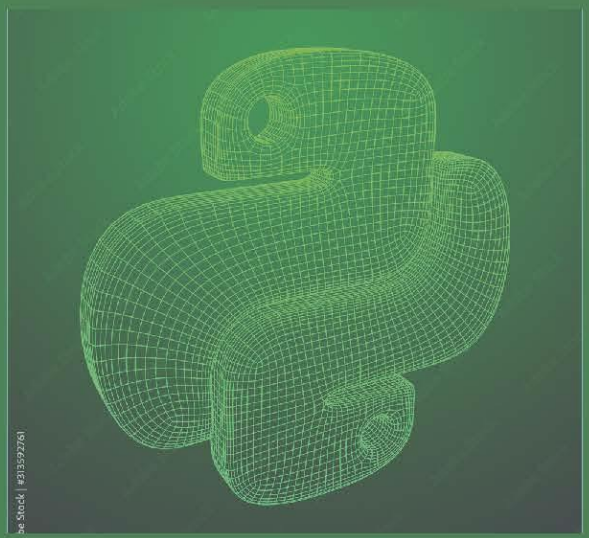
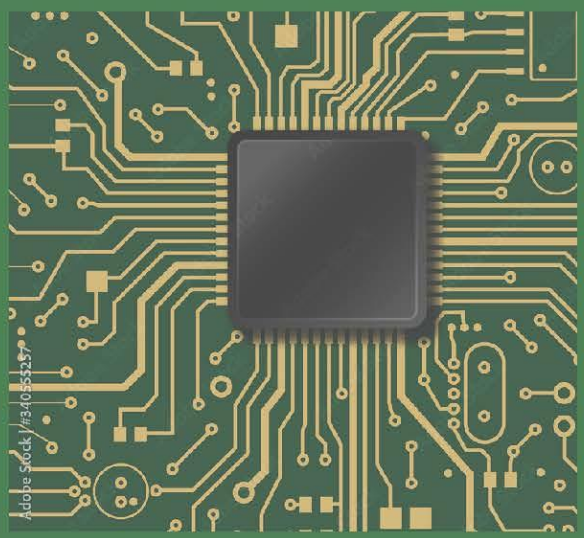
Python Programming I

COMPUTER SCIENCE AND INFORMATION TECHNOLOGY | CAREER AND TECHNICAL EDUCATION

BP14







NORTH CAROLINA
State Board of Education
Department of Public Instruction

Learning that works for North Carolina
CTE

Table of Contents

INTRODUCTION

The BP14 Python Programming I Curriculum Guide was developed to assist teachers in preparing students to meet the North Carolina State Board of Education's guiding vision: "Every public-school student in North Carolina will be empowered to accept academic challenges, prepared to pursue their chosen path after graduating high school, and encouraged to become lifelong learners with the capacity to engage in a globally-collaborative society."

BP14 Python Programming I is rigorous and relevant, is based on state and national content standards, and engages technology to teach today's generation of students. Business and industry representatives reviewed the standards and provided input on the content for this course as one that helps to prepare for high-skill, high-wage, or in demand occupational opportunities.

The CTE Course Management System includes the course standards, course information, and career pathway. Equipment and technical guides can be found on NCCTE Admin. The following essential standards are covered in this course:

- 1.00: Understand ethics, security, and history of computer programming.
- 2.00: Apply problem-solving tools to design programming solutions.
- 3.00: Apply basic data types and operators.
- 4.00: Apply input and output operations.
- 5.00: Analyze code segment functionality by performing troubleshooting and handling errors.
- 6.00: Apply user-defined functions.
- 7.00: Apply decision-making and loop statements.
- 8.00: Apply advanced data types to store and manipulate data.
- 9.00: Apply operations using modules and tools.

Aligned to the course standards and each objective, this guide contains unpacked content, resources, and instructional activities. These materials are samples that may be augmented, modified, or replaced for the teaching and learning experiences.

DISCLAIMER STATEMENT

Contributions of many individuals and from many written resources have collectively made this curriculum guide possible. The major authors do not claim or guarantee that its contents will eliminate acts of malpractice or negligence. The responsibility to adhere to safety standards and best professional practices is the duty of the practitioners, teachers, students, and/or others who apply the contents of this document. This guide was developed with federal Strengthening Career and Technical Education for the 21st Century Act of 2018 funds.

ACKNOWLEDGEMENTS

North Carolina Computer Science and Information Technology Education would like to thank the following educators who assisted with the development of this course:

- Ashley Faulkner, Harnett County Schools
- Andrew Fox, Wake County Public Schools System
- Donald Mace, Charlotte-Mecklenburg Schools
- Judi Pfeffer, North Carolina Virtual Public Schools

A special thank you goes to the following business and industry representatives who assisted in the development of the course content:

- Sterling Beane Jr., Sr. Account Executive, US Education Microsoft Corporation
- Mike W Maddock, VP, Certiport Volume Licensing Business—Americas

Career and Technical Education State Staff
Kim Forbes, CSIT Education Consultant
Mary Jane Thomas, State Adviser, NC FBLA
Jerianne Taylor, State Advisor, NC TSA
Paul Heidepriem, Executive Director, SkillsUSA NC
Robert Van Dyke, Section Chief for CTE Curriculum and Assessment
Linda Lay, Section Chief for Credentials and Support Services
Nancy Cross, Section Chief for CTE Reporting and Analysis
Lee O'Neal, Section Chief for CTE Regional Services
Angela LeMay, Assistant Director for Curriculum, Credentials, and Support Services
Marty Tobey, Assistant Director for CTE Regional Services and Reporting
Trey Michael, CTE State Director

Career and Technical Education
North Carolina Department of Public Instruction
6307 Mail Services Center
Raleigh, NC 27699-6307

CTSO INTEGRATION

The aligned co-curricular Career and Technical Student Organizations (CTSOs) for BM21 Introduction to Data Science are the 1) Future Business Leaders of America (FBLA), 2) SkillsUSA, and the 3) Technology Student Association (TSA).

FBLA inspires and prepares students to become community-minded business leaders in a global society through relevant career preparation and leadership experiences. For questions pertaining to this course and supporting students' access to CSIT experiences, email CTEFBLA@dpi.nc.gov

SkillsUSA empowers its members to become world-class workers, leaders, and responsible American citizens. SkillsUSA improves the quality of our nation's future skilled workforce through the development of Framework skills that include personal, workplace and technical skills grounded in academics. For questions pertaining to this course and supporting students' access to CSIT experiences, email CTESkillsUSANC@dpi.nc.gov

TSA enhances personal development, leadership, and career opportunities in STEM, whereby members apply and integrate these concepts through intra-curricular activities, competitions, and related programs. For questions pertaining to this course and supporting students' access to CSIT experiences, email CTETSA@dpi.nc.gov.

PROOF OF LEARNING OVERVIEW

A Proof of Learning (POL) is a tool to assess student achievement and mastery of the course standards. CTE uses a POL as an indicator of program quality in the state Perkins plan. A CTE course POL can be a CTE State Assessment, a Performance-based Measurement, or an industry-aligned credential.

Performance-based Measurement (PBM)
The POL for this course will be a Performance-based Measurement (PBM). A PBM measures students' ability to use higher order thinking skills and knowledge learned to demonstrate attainment of the course standards. The Performance-based Measurement Guide will be at the end of this Curriculum Guide.

CURRICULUM VENDOR INFORMATION

Curriculum may be accessed via the Microsoft Imagine Academy member login site. You can access the Microsoft Imagine Academy content by following these steps:

1. Open Microsoft Edge browser in a private window and go to https://member.imagineacademy.microsoft.com/
2. Click Sign In and select Work or school account to log in with your O365 account (should that not work, please use Personal account)
3. If your school is not already a member, enter the following Member ID and Program Key:

Membership ID	*Program Key*
5300040698	B95F1549-35B1-4844-8EE8-20EEE9036411

4. To get started with Microsoft Imagine Academy, refer to the login video and other quick trainings at Microsoft Imagine Academy Quick Start Site.

For questions, please refer to the Microsoft Imagine Academy Support Forum.

INSTRUCTIONAL MATERIALS

All course curricula including presentations, assignments, projects, quiz questions, etc. can be found in the DPI Canvas course: BP14 Python Programming I (NCDPI 2022). The course was created so teachers could import it into their own blank Canvas course and have a ready-made course.

All items listed under each Unpacked Content heading are the activities referenced in the BP14 Python Programming I Canvas course that will be facilitated or implemented in the classroom.

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**1.00**	**2%**	**B2**	**Understand ethics, security, and history of computer programming.**
INDICATOR:	**1.01**	**1%**	**Explain the ethics, security, and history of computer science.**	
ESSENTIAL QUESTIONS:	1. Discuss the evolution of computer hardware and programming languages. 2. Why would we study ethics in computer programming?			

UNPACKED INSTRUCTIONAL CONTENT

Course content is found in the DPI Canvas shell in Module 1.

A. Understand the history of computer programming.
   a. History of computer programming notes
   b. History of computer programming discussion
   c. Alternate to discussion: History of computer science assignment
   d. History of computer programming quiz

B. Understand ethics and security in computer programming.
   a. Ethics and security in computer programming notes
   b. Ethics and security in computer programming discussion
   c. Alternate to discussion: ethics and security assignment
   d. Ethics and security in computer programming activity
   e. Ethics and security in computer programming assignment
   f. History of computer programming quiz

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**1.00**	**2%**	**B2**	**Understand ethics, security and history of computer programming.**
INDICATOR:	**1.02**	**1%**	**Summarize the history and development of Python and its place among other programming languages.**	
ESSENTIAL QUESTIONS:	1. How is Python different from other programming languages? 2. How have programming languages evolved over time?			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 1.**				

A. Understand the history of Python.
    a. History of Python assignment

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**2.00**	**2%**	**B3**	**Apply problem solving tools to design programming solutions.**
INDICATOR:	**2.01**	**1%**	**Utilize algorithms and pseudo code in terms of their efficiency, correctness, and clarity.**	
ESSENTIAL QUESTIONS:	How do algorithms and pseudo code work together to create better programs?			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 2.**				

A. Understand the programming process.
    a. Programming process notes
    b. Programming process assignment
    c. Programming process quiz

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**2.00**	**2%**	**B3**	**Apply problem- solving tools to design programming solutions.**
INDICATOR:	**2.02**	**1%**	**Build code segments that use algorithms.**	
ESSENTIAL QUESTIONS:	1. How would you recognize an algorithm if you saw one?			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 2.**				

A. Understand problem-solving tools.
    a. Problem-solving and design tools notes
    b. Problem-solving and design tools discussion
    c. Alternate to discussion: problem-solving and design tools assignment
    d. Problem-solving and design tools quiz

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**3.00**	**11%**	**B3**	**Apply basic data types and operators.**
INDICATOR:	**3.01**	**3%**	**Develop to cast basic data types (string, integer, float, and Boolean).**	
ESSENTIAL QUESTIONS:	2. How can I identify the different data types? 3. Compare and contrast the different data types.			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 3.**				

A. Understand basic data types and cast to them.
    a. Error journal
    b. Module 3 slides
    c. U1 Module 1 Activity 1.2 Data types and variables
    d. Quiz U1 M1.2
    e. U1 Module 1 Activity 1.3 type () function
    f. Quiz U1 M1.3

COURSE:	BP14 Python Programming I			
ESSENTIAL STANDARD:	3.00	11%	B3	Apply basic data types and operators.
INDICATOR:	3.02	2%	Develop descriptive, correctly named variables that represent multiple types of data.	
ESSENTIAL QUESTIONS:	1. Explain the importance of naming variables for the data they hold.			
UNPACKED INSTRUCTIONAL CONTENT Course content is found in the DPI Canvas shell in Module 3.				

A. Understand descriptive correctly named variables.
    a. Error journal
    b. Module 3 slides
    c. U1 Module 1 Activity 1.2 Data types and variables
    d. Quiz U1 M1.2

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**3.00**	**11%**	**B3**	**Apply basic data types and operators.**
INDICATOR:	**3.03**	**2%**	**Choose the appropriate operator to achieve the intended result.**	
ESSENTIAL QUESTIONS:	2. Describe the fundamental Boolean operators. 3. Employ combined comparisons to control program flow.			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 3.**				

A. Understand mathematic operators and when to use them.
    a. Error journal
    b. Module 3 slides
    c. U1 Module 1 Activity 1.4 Addition with strings and integers and errors
    d. Quiz U1 M1.4
    e. U1 Module 1 Activity 1.8 Quote display and Boolean
    f. Quiz U1 M1.8

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**3.00**	**11%**	**B3**	**Apply basic data types and operators.**
INDICATOR:	**3.04**	**2%**	**Organize the sequence of execution based on operator precedence.**	
ESSENTIAL QUESTIONS:	1. Explain the importance of operator precedence in obtaining the desired result when coding.			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 3.**				

A. Understand operator precedence and why it matters.
    a. Error journal
    b. Module 3 slides
    c. U1 Module 1 Activity 1.4 Addition with strings and integers and errors
    d. Quiz U1 M1.4
B. Understand the Boolean data type and how to display quotation marks.
    a. Error journal
    b. U1 Module 1 Activity 1.8 Quote display and Boolean
    c. Quiz U1 M1.8

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**3.00**	**11%**	**B3**	**Apply basic data types and operators.**
INDICATOR:	**3.05**	**2%**	**Construct comments and docstrings to clarify code segments**	
ESSENTIAL QUESTIONS:	2. Describe why and when programmers should use comments and docstrings when coding. 3. Why are using comments considered a part of good programming technique?			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 3.**				

A. Understand the importance of commenting and docstrings to programming.
    a. Error journal
    b. Module 3 slides
    c. U1 Module 1 Activity 1.1 Print and comments
    d. Quiz U1 M1.1

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**4.00**	**12%**	**B3**	**Apply input and output operations.**
INDICATOR:	**4.01**	**4%**	**Identify code segments that perform input and output operations.**	
ESSENTIAL QUESTIONS:	1. Identify code segments that perform input operations. 2. Identify code segments that perform output operations.			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 4.**				

A. Understand code segments that perform output operations.
    a. Error journal
    b. Module 4 slides
    c. U1 Module 1 Activity 1.1 Print and comments
    d. Quiz U1 M1.1
B. Understand code segments that perform input operations.
    a. Error journal
    b. U1 Module 1 Activity 1.6 input() function
    c. Quiz U1 M1.6

COURSE:	BP14 Python Programming I			
ESSENTIAL STANDARD:	4.00	12%	B3	Apply input and output operations.
INDICATOR:	4.02	4%	Modify output of strings using escape sequences and commas with print() statements.	
ESSENTIAL QUESTIONS:	1. Explain how to effectively determine when to use escape sequences with print() statements. 2. Explain the difference between string addition and using commas with print() statements.			
UNPACKED INSTRUCTIONAL CONTENT Course content is found in the DPI Canvas shell in Module 4.				

A. Understand how to modify strings using the comma formatting technique.
   a. Error journal
   b. Module 4 slides
   c. U1 Module 1 Activity 1.7 Print formatting
   d. Quiz U1 M1.7
   e. U1 Module 1 Activity 1.8 Quote display and Boolean
   f. Quiz U1 M1.8
   g. U1 Module 1 Activity 1.9 String formatting and the “in” keyword.
   h. Quiz U1 M1.9
   i. U1 Module 1 Project Practice A
   j. U1 Module 1. Project Practice B
   k. U1 Module 1 Project

B. Understand how to modify strings using escape sequences formatting technique.
   a. Error journal
   b. Module 7 slides
   c. U1 Module 4 Activity 4.2 Formatting with escape sequences
   d. Quiz U1 M4.2

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**4.00**	**12%**	**B3**	**Apply input and output operations.**
INDICATOR:	**4.03**	**4%**	**Modify output of numbers using fstring and string format function.**	
ESSENTIAL QUESTIONS:	1. Compare and contrast the benefits of using fstring over string format.			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 7.**				

A. Understand how to modify strings using fstring and string formatting techniques.
    a. Error journal
    b. Module 4 slides
    c. U3 Module 2 Activity 2.4 Old and new Python formatting
    d. Quiz U1 M4.2

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**5.00**	**12%**	**B4**	**Analyze code segment functionality by performing troubleshooting and handling errors.**
INDICATOR:	**5.01**	**4%**	**Inspect code segments to correct code segments that have errors.**	
ESSENTIAL QUESTIONS:	2. How would being able to inspect code segments to correct errors help make better programmers?			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 3.**				

A. Understand how to identify the different types of errors in Python.
    a. Error journal
    b. Module 3 slides
    c. U1 Module 1 Activity 1.4 Addition with strings and integers and errors
    d. Quiz U1 M1.4

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**5.00**	**12%**	**B4**	**Analyze code segment functionality by performing troubleshooting and handling errors.**
INDICATOR:	**5.02**	**4%**	**Create code segments that handle exceptions.**	
ESSENTIAL QUESTIONS:	1. Differentiate between exception types and recognize their meaning.			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 4.**				

A. Understand how to code for errors so it does not break your code.
    a. Error journal
    b. Module 4 slides
    c. U3 Module 3 Activity 3.1 Try…except…else…finally
    d. Quiz U3 M3.1

COURSE:	BP14 Python Programming I			
ESSENTIAL STANDARD:	**5.00**	**12%**	**B4**	**Analyze code segment functionality by performing troubleshooting and handling errors.**
INDICATOR:	**5.03**	**4%**	**Analyze troubleshooting techniques.**	
ESSENTIAL QUESTIONS:	1. Evaluate the different debugging techniques and when to use them.			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 4.**				

A. Understand when to use the different debugging techniques.
    a. Error journal
    b. Module 4 slides
    c. U3 Module 3 Activity 3.1 Try…except…else…finally
    d. Quiz U3 M3.1

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**6.00**	**20%**	**B3**	**Apply user-defined functions.**
INDICATOR:	**6.01**	**6%**	**Apply functions to reduce code complexity and reuse code segments.**	
ESSENTIAL QUESTIONS:	1. Explain the benefits of creating and using functions in code.			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 5.**				

A. Understand how to apply functions to make coding easier.
    a. Error journal
    b. Module 5 slides
    c. U1 Module 2 Activity 2.1 Simple functions
    d. Quiz U1 M2.1

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**6.00**	**20%**	**B3**	**Apply user-defined functions.**
INDICATOR:	**6.02**	**6%**	**Apply functions with and without parameters that do or do not return values.**	
ESSENTIAL QUESTIONS:	1. Explain returns and how do you use them in functions.			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 5.**				

A. Understand how to create parameters in a function.
    a. Error journal
    b. Module 5sSlides
    c. U1 Module 2 Activity 2.2 Function return and parameters
    d. Quiz U1 M2.2

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**6.00**	**20%**	**B3**	**Apply user-defined functions.**
INDICATOR:	**6.03**	**4%**	**Apply default parameter values in functions effectively.**	
ESSENTIAL QUESTIONS:	1. Explain parameters and how they are used in functions.			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 5.**				

A. Understand how to use default parameters in a function.
   a. Error journal
   b. Module 5 slides
   c. U1 Module 2 Activity 2.2 Function return and parameters
   d. Quiz U1 M2.2
   e. U1 Module 2 Activity 2.3 Sequence
   f. Quiz U1 M2.3
   g. Module 3 slides
   h. U1 Module 1 Activity 1.1 Print and comments
   i. Quiz U1 M1.1
   j. U1 Module 1 Activity 1.6 input() function
   k. Quiz U1 M1.6
   l. Unit 1 Project Practice Module 2
   m. U1 Module 2 Project

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**6.00**	**20%**	**B3**	**Apply user-defined functions.**
INDICATOR:	**6.04**	**4%**	**Choose between local and global scope.**	
ESSENTIAL QUESTIONS:	1. Compare and contrast variable scope and how to identify local or global.			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 5.**				

A. Understand variable scope.
    a. Error journal
    b. Module 5 slides
    c. U3 Module 4 Activity 4.2 Variable scope
    d. Quiz U3 M4.2

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**7.00**	**20%**	**B3**	**Apply decision-making and loop statements.**
INDICATOR:	**7.01**	**10%**	**Build code segments that use decision and branching statements to control flow and logic.**	
ESSENTIAL QUESTIONS:	1. Explain how using if statements improve code performance.			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 6.**				

A. Understand conditional statements.
   a. Error journal
   b. Module 6 slides
   c. U1 Module 3 Activity 3.1 Boolean strings
   d. Quiz U1 M3.1
   e. U1 Module 3 Activity 3.2 Comparison operators
   f. Quiz U1 M3.2
   g. U1 Module 3 Activity 3.3 String comparison
   h. Quiz U1 M3.3
   i. U1 Module 3 Activity 3.4 Conditionals, if…elif…else, casting
   j. Quiz U1 M3.4
   k. U1 Module 3 Activity 3.5 Conditionals, type and mathematics extended
   l. Quiz U1 M3.5
   m. U1 Module 3 Project Practice A
   n. U1 Module 3 Project Practice B
   o. U1 Module 3 Project

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**7.00**	**20%**	**B3**	**Apply decision- making and loop statements.**
INDICATOR:	**7.02**	**10%**	**Build loops to control flow and/or perform iteration.**	
ESSENTIAL QUESTIONS:	1. Explain how using loops improve code performance.			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 7.**				

A. Understand looping and how to control data flow.
   a. Error journal
   b. Module 7slides
   c. U1 Module 4 Activity 4.1 Nested conditionals
   d. Quiz U1 M4.1
   e. U1 Module 4 Activity 4.3 While loops and incrementing
   f. Quiz U1 M4.3
   g. U1 Module 4 Activity 4.4 While loops Boolean comparison
   h. Quiz U1 M4.4
   i. U1 Module 4 Project Practice A
   j. U1 Module 4 Project Practice B
   k. U1 Module 4 Project
   l. Module 9 slides
   m. U2 Module 3 Activity 3.1 For loops and list iteration
   n. Quiz U2 Module 3.1

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**8.00**	**15%**	**B3**	**Apply advanced data types to store and manipulate data.**
INDICATOR:	**8.01**	**6%**	**Modify strings using indexing and string methods to compare.**	
ESSENTIAL QUESTIONS:	1. Identify an index and how it is used. 2. Explain how a string is indexed.			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 8.**				

A. Understand how to modify string using indexing and string methods to compare.
   a. Error journal
   b. Module 8 slides
   c. U2 Module 1 Activity 1.1 String sequences
   d. Quiz U2 M1.1
   e. U2 Module 1 Activity 1.2 Index slicing
   f. Quiz U2 M1.2
   g. U2 Module 1 Activity 1.3 Iterating strings
   h. Quiz U2 M1.3
   i. U2 Module 1 Activity 1.4 String methods
   j. Quiz U2 M1.4
   k. U2 Module 1 Project practice
   l. U2 Module 1 Project

COURSE:	BP14 Python Programming I			
ESSENTIAL STANDARD:	8.00	15%	B3	Apply advanced data types to store and manipulate data.
INDICATOR:	8.02	4%	Develop lists and tuples to store data.	
ESSENTIAL QUESTIONS:	1. Identify ways to create lists and access items in a list. 2. Explain the importance of iterating through lists using for with in.			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 9 and Module 10.**				

A. Understand how to create lists to store data.
   a. Error journal
   b. Module 9 slides
   c. U2 Module 2 Activity 2.1 List sequences
   d. Quiz U2 M2.1
   e. U2 Module 2 Activity 2.2 List append
   f. Quiz U2 M2.2
   g. U2 Module 2 Activity 2.3 List insert
   h. Quiz U2 M2.3
   i. U2 Module 2 Activity 2.4 List delete, pop, and remove
   j. Quiz U2 M2.4
   k. U2 Module 2 Project practice
   l. U2 Module 2 Project
   m. U2 Module 3 Activity 3.1 For loop and list iteration
   n. Quiz U2 Module 3.1
   o. U2 Module 3 Activity 3.2 Range iteration
   p. Quiz U2 Module 3.2
   q. U2 Module 3 Activity 3.3 List .extend(), .reverse(), and .sort() methods
   r. Quiz U3 Module 3.3
   s. U2 Module 3 Project practice
   t. U2 Module 3 Project

B. Understand how to create tuples to store data.
   a. Error journal
   b. Module 10 slides
   c. U3 Module 3 Activity 3.3 Tuples
   d. Quiz U3 M3.3
   e. U3 Module 3 Project practice
   f. U3 Module 3 Project

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**8.00**	**15%**	**B3**	**Apply advanced data types to store and manipulate data.**
INDICATOR:	**8.03**	**5%**	**Build lists and tuples with conditionals and loops to sort, filter, transform, and manipulate data.**	
ESSENTIAL QUESTIONS:	1. Explain how to write nested conditionals and how it will help you code better.			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 9 and Module 10.**				

A. Understand how to create tuples to store data.
   a. Error journal
   b. Module 10 slides
   c. U3 Module 2 Activity 2.1 Comparison operators
   d. Quiz U3 M2.1
   e. U3 Module 2 Activity 2.2 Advanced loop
   f. Quiz U3 M2.2
   g. U3 Module 2 Activity 2.3 Containment “is” and “in”
   h. Quiz U3 M2.3
   i. U3 Module 2 Project practice
   j. U3 Module 2 Project

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**9.00**	**6%**	**B3**	**Apply operations using modules and tools.**
INDICATOR:	**9.01**	**3%**	**Select appropriate built-in modules to design and perform basic operations.**	
ESSENTIAL QUESTIONS:	1. How do you import different Python modules?			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 9 and Module 11.**				

A. Understand how to use built-in modules to improve your basic code.
    a. Error journal
    b. Module 9 slides
    c. U2 Module 3 Activity 3.4 Between strings and lists
    d. Quiz U3 Module 3.4
    e. Module 11 Slides
    f. U3 Module 1 Activity 1.1 Python modules
    g. Quiz U3 Modules 1.1
    h. U3 Module 1 Activity 1.2 Date, time modules
    i. Quiz U3 Modules 1.2

COURSE:	**BP14 Python Programming I**			
ESSENTIAL STANDARD:	**9.00**	**6%**	**B3**	**Apply operations using modules and tools.**
INDICATOR:	**9.02**	**3%**	**Select appropriate built-in modules to design and perform complex computing problems.**	
ESSENTIAL QUESTIONS:	How do you compute mathematical expressions using functions from the math module?			
UNPACKED INSTRUCTIONAL CONTENT **Course content is found in the DPI Canvas shell in Module 9 and Module 11.**				

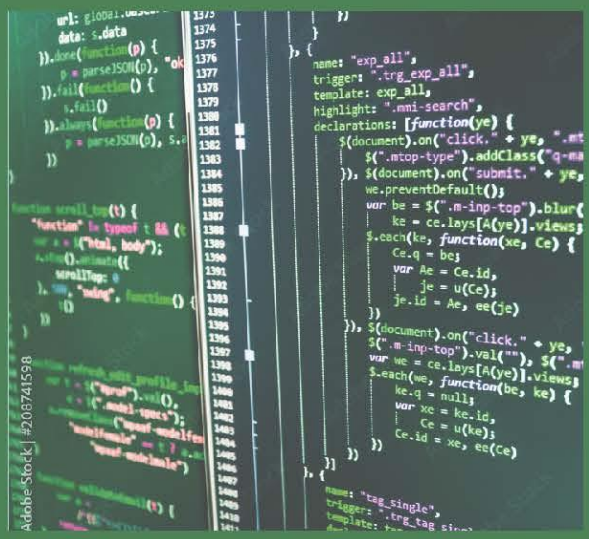
A. Understand how to use built-in modules to improve your complex code.
    a. Error journal
    b. Module 11 slides
    c. U3 Module 1 Activity 1.3 Timedelta modules
    d. Quiz U3 Modules 1.3
    e. U3 Module 1 Project practice
    f. U3 Module 1 Project

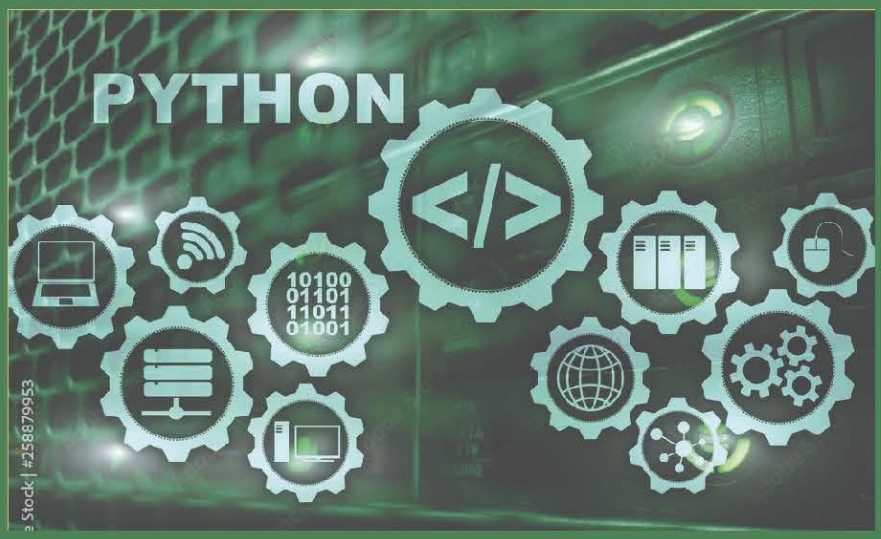
Python Programming I

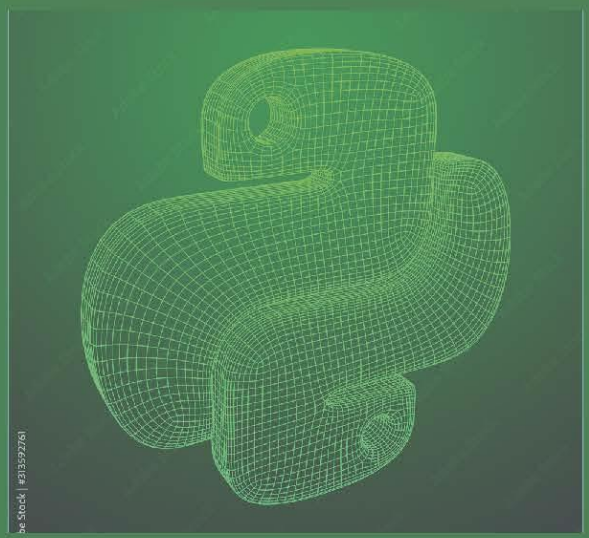
Performance-based Measurement Guide

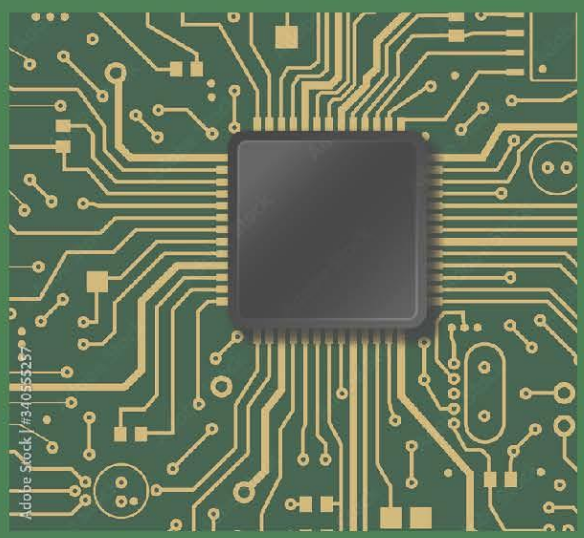
COMPUTER SCIENCE AND INFORMATION TECHNOLOGY | CAREER AND TECHNICAL EDUCATION

BP14









NORTH CAROLINA
State Board of Education
Department of Public Instruction

Learning that works for North Carolina
CTE

PBM Overview

Performance-based Measurements (PBMs), when used as the proof of learning (POL) for a course, give teachers a tool to evaluate student achievement and mastery of course concepts. A PBM measures students' ability to apply the skills and knowledge learned from course standards. Typically, the task challenges students to use their higher order thinking skills to create a product or complete a process (Chun, 2010). PBM's look and perform differently for each course based upon the recommendations of the subject-matter experts for each course.

COURSE PROFICIENCY
Proficiency for this course is a weighted score of 70% or above or 70 points or more.

COURSE STANDARDS AND PBM ALIGNMENT
The table below indicates how components of the Performance-based Measurement align with the corresponding course standards.

Corresponding Course Standard	Performance-based Measurement Component
1.00: Understand ethics, security, and history of computer programming. 2.00: Apply problem-solving tools to design programming solutions. 3.00: Apply basic data types and operators. 4.00: Apply input and output operations. 5.00: Analyze code segment functionality by performing troubleshooting and handling errors. 6.00: Apply user-defined functions. 7.00: Apply decision-making and loop statements. 8.00: Apply advanced data types to store and manipulate data. 9.00: Apply operations using modules and tools.	Project Code (70%)
	Documentation (10%)
	Flowchart (10%)
	Error Journal Reflection (10%)

PBM COMPONENTS
Students will create an original, independent project that exhibits the use of skills they learned throughout the course to show their knowledge of one or more concepts. Students are also asked to document their project using an independent project framework, which emphasizes metacognitive development and code analysis.

During this project, students will:

- code a unique, original program.
- document the details of their program code by adding #comments and docstrings identifying and explaining the code which includes declaring and initializing various variables, and at least one of each of the following: a conditional, a loop, and a function.
- create a flowchart illustrating the program's process, identifying essential steps and simultaneously the overall picture of the process; and

- complete a thorough course error journal, reflecting on errors encountered and the process used to handle them.

Students will design and create a final project that solves a problem or serves a purpose.

Students are asked to propose their own original, independent project, and are expected to complete it within the time allotted.

The expectation is that students will work steadily on their independent project for several sessions, while testing and troubleshooting their program code.

Review the following rubrics for the project code, documentation, flowchart, and error journal to clearly explain the expectations of the PBM. The PBM is a chance for students to use all the skills learned throughout the course to create something that is original, and that solves a problem or serves a purpose.

PROJECT IDEAS

- Create a program/app that helps somebody by solving a problem.
- Create a game.

TEACHER GUIDELINES

Time Frame

Depending on the class time allotted (semester or yearlong), the error journal, project proposal, and initial flowchart should be due in a specific time frame. It is suggested for a semester-long class that parts of the final project be due throughout the semester (project proposal, initial flowchart, and draft code) and that at least one week at the end be allowed to finish coding, documenting, and submitting all components.

The following is a suggested timeline.

Prior to the last week of the course:

- Error Journal–through course, beginning in the first programming module,
- Project proposal submitted for approval–must include how all required structures will be included in final project,
- Initial flowchart of the project,
- Initial draft of code (at least 50% coded/documented).

Last week of the course:

- Final project code and documentation,
- Flowchart of the final, completed project,
- Completed Error Journal.

STUDENT INSTRUCTIONS FOR COMPLETING PBM

You are asked to create an original, independent project that demonstrates the use of the skills you have been taught in the course. You are also asked to document your learning process of building your project using an independent project framework, which emphasizes metacognitive development and process-oriented work.

During this project, you will:

- Code a unique, original program demonstrating the skills you have been taught in the course,
- Document the details of your program code through the use of header comments, #comments and docstrings,
- Create a flowchart illustrating the program's process,
- Reflect on errors encountered throughout the course by keeping a journal of errors and the process followed to handle them.

PBM COMPONENTS

- 70% Project code
- 10% Documentation and Reflection
- 10% Flowchart
- 10% Error Journal

PROJECT IDEAS

You will design and make a final project that solves a problem or serves a purpose. You are asked to propose your own independent project and are expected to complete it within the time allotted. The expectation is that you are working steadily on your independent project for several sessions, while testing and troubleshooting your program code.

You may create something that solves a problem or serves a purpose, or you may create a game.

Review the following Scoring Rubrics for the project code, documentation, and reflection to clearly understand the expectations of the PBM. The PBM is a chance for you to use all the skills learned throughout the course to create something that is original, and that solves a problem or serves a purpose.

Project Code Rubric

Expectations: You are responsible creating a working program with the following program structures:

PROJECT CODE RUBRIC (70%)					
Evaluative Criteria	**4 Points**	**3 Points**	**2 Points**	**0 Points**	**Student Score**
Naming Conventions and Program Readability (3.02)	No errors, code is clean, understandable, and well-organized. All variables and controls are properly and descriptively named.	The code is easy to read. Issues with consistent indentation, use of white space, variable naming, or general organization.	The code is readable only by someone who knows what it is supposed to be doing or naming conventions are not followed.	The code is poorly organized and very difficult to read.	
Basic data types and operators (3.01, 3.03, 3.04)	Program includes use of each basic data type and various basic operators.	Program includes use of 3/4 of basic data types and various basic operators.	Program includes use of 2/4 of the basic data types and various basic operators.	Program includes 0 or 1/4 basic data types and few basic operators.	
Input (4.01)	At least 2 input statements with properly formatted prompts have been implemented.	At least 1 input statement with a properly formatted prompt or 2 input statements with improperly formatted prompts have been implemented.	Only one input statement with improperly formatted prompt has been implemented .	No input statement has been used in the program.	
Output (4.01, 4.02, 4.03)	All output is pleasingly displayed. Program shows use of at least 1 escape sequence, commas, and at least 1 formatted print statement–old or new formatting may be used.	Output is mostly pleasingly displayed. Program missing 1 of the following: use of at least 1 escape sequence, commas, and at least 1 formatted print statement–old or new formatting may be used.	All output is not pleasingly displayed. Program missing 2 of the following: use of at least 1 escape sequence, commas, and at least 1 formatted print statement–old or new formatting may be used.	Print formatting is not evident.	
Error handling (5.01, 5.02, 5.03)	Exception/error handling used where all user input occurs –all possible errors are handled.	Exception/error handling used where most user input occurs, or most possible errors are handled.	Exception/error handling used where some user input occurs, or some possible errors are handled.	Errors are not handled or not handled correctly.	
Password (1.01)	Program is password protected–user must enter any password that is alphanumeric to run the program. Loop and string test is used.	Program is password protected–user must enter any password that is alphanumeric to run the program. Loop or string test is not correctly used.	Program is password protected–user must enter any password that is alphanumeric to run the program. Loop and string test is not correctly used.	Program is not password protected.	
Modules (9.01, 9.02)	At least 2 different modules are imported and at	At least 1 module is imported and at least 1 object is	Modules are present but are not imported or used correctly.	No use of modules is present in the program.	

	least 1 object is created and used and/or displayed.	created and used and/or displayed.			
Evaluative Criteria	**10 Points**	**6 Points**	**4 Points**	**0 Points**	**Student Score**
Functions (6.01, 6.02, 6.03)	Code very effectively demonstrates the use of modularization. At least 2 functions must be defined and called with parameters, return values.	Code effectively demonstrates the use of modularization. At least 1 function must be defined and called with parameters, return values.	Code somewhat demonstrates the use of modularization. Function is missing parameters or return values. .	Code does not include a function with parameters and return values.	
Conditional Structures (7.01)	At least 1 complex (uses AND, OR, NOT) and 1 basic conditional statement (if…elif…else) is correctly used to make a decision based on information stored in variables.	A basic conditional is used to make a decision based on information stored in variables. The complex conditional statement does not correctly use a logical operator (AND, OR, NOT) in the Boolean expression.	Only 1 conditional is created and does not use information stored in variables to make a decision or does not correctly use a logical operator (AND, OR, NOT) in the Boolean expression.	No conditional statement is present.	
Loops (7.02)	Both for loops (to process data from a list or to construct data types) and while loops (to control when a loop stops based on a condition.) are used correctly in the program.	Either for loops (to process data from a list or to construct data types) and while loops (to control when a loop stops based on a condition.) are used correctly in the program.	Both for loops (to process data from a list or to construct data types) and while loops (to control when a loop stops based on a condition.) are used but incorrectly in the program.	No loops are present in the program.	
Advanced Data Types (8.01, 8.02, 8.03	Lists are used to store data, must include at least 1 use of parallel lists, and must show use of the following list methods: access, append, insert, and remove.	Lists are used to store data, missing at least 1 use of parallel lists or doesn't show use of all the following list methods: access, append, insert, and remove.	Lists are used to store data, missing use of parallel lists and doesn't show use of all the following list methods: access, append, insert, and remove.	No lists are present in the program.	

Evaluative Criteria	**2 Points**	**0 Points**	**Student Score**
Search (8.01)	At least one search using the 'in' keyword correctly is present–May show with strings, lists, tuples, or Boolean operators.	No search is present in the program.	
		Total Project Code (70 possible)	

Documentation and Reflection Rubric

Expectation: You are responsible for documenting your code for this project. Your documentation will include header comments, #comments, and docstrings. (3.05)

Header comments (4%)—belong as the top code cell of the program (may be multiple #comments or a docstring format) and include name, date and a short (~100 word) reflection that summarizes the progress of learning, reflecting on debugging methods and the overall design process.

Include the following topics, proofread your work and use complete sentences to address the following:

- Describe something that surprised you as you worked on your project.
- Describe any challenges you had this week.
- Did anyone help you this week? Who and how?
- Choose an adjective that describes how you are feeling about your project. Explain why you chose this word.
- If you had more time to work on this project, what would you add?

#comments (3%)—Comments are used to increase the readability and understandability of your code. All important sections and structures should be preceded by a #comment explaining what your code is doing in a particular section or structure. Examples may include:

defining variables

conditional statement that checks if age is greater than 18, displaying output dependent upon the result.

docstrings (3%)—all functions require a docstring comments after the def statement that describe the function and its components and include the following:

```
"""

summary line – what is the function doing.

args:

    descriptions of parameters

returns:

    descriptions of return values

"""
```

DOCUMENTATION AND REFLECTION RUBRIC (10%)						
Evaluative Criteria	**4 Points**	**3 Points**	**2 Points**	**1 Point**	**0 Points**	**Student Score**
Header Comments	Header comments are proofread and completely address all 5 topics.	Header comments are not well proofread and/or only address 4/5 of the topics.	Header comments are not well proofread and/or only address 3/5 of the topics.	Header comments are not well proofread and/or only address 2/5 of the topics.	Header comments are missing.	

Evaluative Criteria	**3 Points**	**2 Points**	**0 Points**	**Student Score**
#comments	Descriptive #comments are visible for all major sections and structures.	#comments are visible for some major sections and structures but they are not adequately descriptive.	#comments are missing and/or not descriptive for most major sections and structures.	
docstrings	Complete docstrings are included for all functions.	50% complete docstrings are included for some functions.	Incomplete docstrings are included for some functions.	
Total Documentation Reflection Points (10 possible)				

Flowchart Rubric

Expectation: The final flowchart is expected to use the correct shapes and flowlines and be a visual representation of your final program.

FLOWCHART RUBRIC (10%)					
Evaluative Criteria	**3 Points**	**2 Points**	**1 Point**	**0 Points**	**Student Score**
Flowchart Shapes	Flowchart uses the correct shapes for the correct structures.	Flowchart mostly uses the correct shapes for the correct structures.	Flowchart uses some of the correct shapes for the correct structures.	No shapes used.	
Flowchart Sequence	Flowchart uses flowlines to show the correct sequence and labeling for this project.	Flowchart mostly uses flowlines to show the correct sequence and labeling for this project.	Flowchart somewhat uses flowlines to show the correct sequence and labeling for this project.	Flowlines do not show sequence.	
Evaluative Criteria	**4 Points**	**3 Points**	**2 Points**	**0 Points**	**Student Score**
Flowchart Representation	Flowchart is representative of final code.	Flowchart is mostly representative of final code.	Flowchart is somewhat representative of final code.	Flowchart is not representative of the final code.	
Total Flowchart Points (10 possible)					

Error Journal Rubric

Expectation: The final Error Journal is completed for all modules and the final project

ERROR JOURNAL RUBRIC (10%)					
Evaluative Criteria	**10 Points**	**5 Points**	**1 Point**	**0 Points**	**Student Score**
Error Journal	Error Journal is complete for all modules and the final project.	Error Journal is missing 1–4 modules or the final project.	Error Journal is missing 5–9 modules or the final project.	Error Journal is mostly incomplete or not submitted.	
Total Error Journal Points (10 possible)					

Final Project Submission

You will submit the project code with documentation, final flowchart, and error journal utilizing the rubrics provided. You must earn at least 70 of the possible 100 total points to meet proficiency for this course.

BP14 Python Programming I—PBM Checklist	(Student Name)		
PROJECT CODE			
Student included or demonstrated knowledge of…	**Date Completed**	**Points Earned**	**Teacher Initial**
1. Naming Conventions/Program Readability			
2. Basic Data Types and Operators			
3. Input			
4. Output			
5. Error Handling			
6. Password			
7. Modules			
8. Functions			
9. Conditional Statements			
10. Loops			
11. Advanced Data Types			
12. Search			
Total Project Code Points (70%)			
DOCUMENTATION AND REFLECTION			
Student included or demonstrated knowledge of…	**Date Completed**	**Points Earned**	**Teacher Initial**
1. Header Comments with Reflection			
2. #comments			
3. docstrings			
Total Documentation Points (10%)			
FLOWCHART			
Student included or demonstrated knowledge of…	**Date Completed**	**Points Earned**	**Teacher Initial**
1. Flowchart Shapes			
2. Flowchart Sequence			
3. Flowchart Representation			
Total Flowchart Points (10%)			
ERROR JOURNAL			
1. Error Journal Completed			
Total Error Journal Points (10%)			
Total Final Project Points			

PBM Accountability Form

Directions: Complete this form, or the form provided by your district, to verify that students enrolled in this course have met or not met specific criteria to earn proficiency on the Performance-based Measurement (PBM). Refer to the Performance-based Measurement Guide for this course for details on course proficiency.

It is highly recommended that teachers keep a copy of all student records to verify PBM performance.

Course Number	Course Name
School/District Name	**Teacher Name**
Class Period	**Semester and Year**

Proficiency Status				
	Student Name	**Student ID**	**Met**	**Not Met**
1			☐	☐
2			☐	☐
3			☐	☐
4			☐	☐
5			☐	☐
6			☐	☐
7			☐	☐
8			☐	☐
9			☐	☐
10			☐	☐
11			☐	☐
12			☐	☐
13			☐	☐

Proficiency Status				
	Student Name	**Student ID**	**Met**	**Not Met**
14			☐	☐
15			☐	☐
16			☐	☐
17			☐	☐
18			☐	☐
19			☐	☐
20			☐	☐
21			☐	☐
22			☐	☐
23			☐	☐
24			☐	☐
25			☐	☐
26			☐	☐
27			☐	☐
28			☐	☐
29			☐	☐
30			☐	☐

I verify that the students listed have met/not met the proficiency as stated in the Performance-based Measurement Guide for this course. I have reported course proficiency in NCCTE Admin, as documented on this form, for the purpose of CTE reporting and accountability (state and federal).

Teacher Signature _____ Date _____

The Curriculum and Instructional Management Coordinator (CIMC) maintains the original form for five years. The teacher should retain a copy for their records.