Basics of Python

Lesson Title: Basics of Python
Subject: Technology
Grade Level: 9-12

This lesson aims for students to grasp the fundamental concepts of Python programming and to understand how to effectively use Python with various libraries.

Introduction/Hook

The lesson will begin by asking students to consider where they encounter technology and code in their daily lives, prompting them to think about apps, websites, or even smart devices. The teacher will then introduce Python as one of the most popular and versatile programming languages, underpinning many of these technologies. A short, engaging video clip (approximately 2-3 minutes) showcasing diverse applications of Python, such as in artificial intelligence, web development, and data science, will be played to capture student interest and illustrate Python's real-world relevance. Following the video, students will be asked what they noticed or found interesting, leading into a discussion about the power of programming. An interactive game, perhaps a quick online quiz or matching activity about common programming terms, will serve as a light introduction to the coding environment, preparing students for the hands-on portion of the lesson.

Instructional Activities

The instructional activities will start with a clear definition of Python, highlighting its readability, high-level nature, and versatility. The teacher will conduct a live coding demonstration using an online integrated development environment (IDE) for ease of access and immediate practice. The first basic concept introduced will be the 'print()' function, demonstrating how to display text on the screen with a "Hello World" example. Following this, the concept of variables will be explained, covering how to store different types of data such as integers, floating-point numbers, and strings. Students will learn about declaring variables and assigning values. Simple arithmetic operations using variables will be demonstrated, including addition, subtraction, multiplication, and division. The 'input()' function will then be introduced, showing students how to accept user input and store it in a variable.

Next, the lesson will transition to Python libraries. The teacher will explain what libraries are: collections of pre-written code that extend Python's functionality for specific tasks. The process of importing a library will be shown using the 'import' keyword. Two simple and highly illustrative libraries will be introduced: the

'random' library for generating random numbers and the 'math' library for mathematical functions. Demonstrations will include using 'random.randint()' to simulate a dice roll and 'math.sqrt()' to calculate a square root. Students will be encouraged to follow along with the live coding demonstration in their own online IDEs, immediately applying each new concept as it is introduced. Interactive games may be used periodically to check understanding of syntax or concept application, such as dragging and dropping code blocks to complete a program or predicting the output of a given snippet.

Guided Practice

Students will engage in a series of guided coding challenges designed to reinforce the concepts taught. These activities will be performed collaboratively using Breakout Rooms for small group discussions and problem-solving. Each group will be assigned specific tasks to complete in their online IDE.

Challenge 1: Personal Greeting Program. Students will write a program that asks the user for their name using the 'input()' function and then prints a personalized greeting using a variable.

Challenge 2: Area Calculator. Students will write a program that prompts the user for the length and width of a rectangle, stores these values in variables, calculates the area, and prints the result.

Challenge 3: Dice Roll Simulator. Students will use the 'random' library to simulate the roll of a six-sided die, generating a random integer between 1 and 6, and printing the result.

Challenge 4: Square Root Finder. Students will use the 'math' library to ask the user for a number, calculate its square root, and display the result.

During these challenges, the teacher will circulate between breakout rooms, providing support, answering questions, and offering hints. Groups will be encouraged to discuss their approaches and troubleshoot errors together. After a set time, groups will briefly share their solutions or challenges with the whole class, fostering a collaborative learning environment.

Assessment

Formative assessment will be ongoing throughout the lesson as the teacher observes students' participation in the interactive games, their engagement during live coding, and their collaborative efforts in the breakout rooms. The teacher will note students' ability to follow instructions, write basic code, and apply new concepts.

For a more structured assessment, students will complete an exit ticket during the last 10 minutes of the

lesson. The exit ticket will consist of three prompts:

1. Write one key concept you learned about Python programming today.
2. Explain in your own words what a Python library is and give an example of one we used.
3. Write a simple Python program that uses both the 'print()' function and a variable to display a message like "My favorite number is X", where X is a number of your choice stored in a variable.

This exit ticket will allow the teacher to gauge individual understanding of Python basics and library usage.

Differentiation

To cater to diverse learning needs, several differentiation strategies will be employed.

For students who require additional support: The teacher will provide pre-written code snippets with blanks for them to fill in, reducing the amount of initial typing required. These students will also be paired with stronger peers for peer mentoring and collaborative problem-solving. Simplified challenges with fewer steps or more explicit instructions will be available. The teacher will conduct more frequent check-ins and provide direct, step-by-step guidance. Access to simplified syntax cheat sheets will also be provided.

For students who require more challenge or have prior experience: These students will be encouraged to extend the guided practice challenges by adding more complex features, such as error handling for user input, incorporating simple conditional statements (if/else), or adding loops to repeat actions. They may also be challenged to research another Python library of their choice and briefly explain its purpose and a potential use case to the class or in a short written response. Another option is to combine functionalities from multiple libraries or to design a small mini-project that integrates several concepts learned during the lesson. For example, they could create a program that rolls two dice and reports if it was a 'double'.