Python Programming Foundations: Young Developers Workshop

SECTION 1: PROGRAMMING FOUNDATIONS EXPLORATION

Learning Objectives:

- Understand basic programming concepts
- Explore Python's fundamental data types
- Practice creating and manipulating variables

Variables and Data Types Deep Dive

Key Concept: What is a Variable?

A variable is like a container that stores information. Think of it as a labeled box where you can store different types of data that can change during your program's execution.

Complete the following activities to demonstrate your understanding:

1. Create and explain three different variables:

```
# Example Variable Declarations student_name = "Alex Johnson" # String variable age = 14 # Integer variable average_score = 85.5 # Float variable
```

Explain each variable's purpose and data type: a) student_name: b) age: c) average_score:

2. Data Type Conversion Challenge:

```
# Convert between data types age_as_string = str(14) score_as_integer = int(85.5)
```

What happens in each conversion? Explain the results:

Computational Thinking Exercise

Design a simple algorithm to solve a real-world problem:

1. Problem: Create a program to calculate a student's final grade

Steps to solve this problem: 1. 2. 3. 4.

2. Flowchart your solution:

[Draw your flowchart here]

SECTION 2: CODING CHALLENGES AND PROBLEM SOLVING

Conditional Logic Mastery

> **Key Concept: Conditional Statements**
>
> Conditional statements allow your program to make decisions based on different conditions. They use keywords like 'if', 'elif', and 'else' to create logical branching.

Complete the following coding challenges:

1. Create a grade classification function:

```
def classify_grade(score): # Your code here to determine letter grade pass
```

Implement the function to return: - A for 90-100 - B for 80-89 - C for 70-79 - D for 60-69 - F for below 60

2. Age Verification Challenge:

```
def check_eligibility(age): # Determine eligibility for activities pass
```

Create a function that: - Allows entry to teen club (13-17) - Provides different messages for different age groups

SECTION 3: PYTHON DATA STRUCTURES EXPLORATION

Learning Objectives:

- Understand lists, tuples, and dictionaries
- Practice data structure manipulation
- Develop complex data handling skills

Lists and List Comprehensions

Key Concept: Lists as Dynamic Collections

Lists are ordered, mutable collections that can store multiple types of data. They provide powerful methods for data manipulation and transformation.

```
# List Creation and Manipulation students = ["Alice", "Bob", "Charlie", "David"] scores = [85, 92, 78, 95] # List Comprehension Example high_performers = [name for name, score in zip(students, scores) if score > 90]
```

Complete the following list manipulation challenges:

1. Create a list of student scores and perform operations:

   Tasks: a) Calculate average score b) Find highest and lowest scores c) Sort the scores in ascending order

Dictionary Mastery

Key Concept: Dictionaries as Key-Value Stores

Dictionaries provide a powerful way to store and retrieve data using unique keys, enabling complex data relationships.

```
# Student Grade Management Dictionary student_grades = { "Alice": {"Math": 95, "Science": 88, "English": 92}, "Bob": {"Math": 85, "Science": 90, "English": 78} } # Dictionary Comprehension advanced_students = {name: grades for name, grades in student_grades.items() if all(score > 85 for score in grades.values())}
```

Dictionary Manipulation Challenge:

1. Create a student record system:

Tasks: a) Add new student records b) Calculate individual student averages c) Find students with all subjects above 85

SECTION 4: FUNCTIONS AND MODULAR PROGRAMMING

Learning Objectives:

- Design and implement custom functions
- Understand function parameters and return values
- Practice modular programming techniques

Function Design Principles

Key Concept: Functions as Reusable Code Blocks

Functions allow you to encapsulate logic, promote code reusability, and create more organized and maintainable programs.

```
# Advanced Function Design def calculate_student_stats(scores, weight_dict=None): """ Calculate weighted average and performance metrics :param scores: List of student scores :param weight_dict: Optional dictionary for weighted calculations :return: Dictionary of statistical information """ if weight_dict is None: weight_dict = {"default": 1} # Complex calculation logic here return { "average": sum(scores) / len(scores), "max_score": max(scores), "min_score": min(scores) }
```

Function Design Challenges:

1. Create a multi-purpose grade analysis function:

   Requirements: a) Calculate basic statistics b) Support optional weighted scoring c) Provide performance insights

Lambda and Higher-Order Functions

Key Concept: Functional Programming Techniques

Lambda functions and higher-order functions provide powerful ways to write concise and flexible code.

```
# Lambda and Map/Filter Examples students = [ {"name": "Alice", "age": 15, "score": 92}, {"name": "Bob", "age": 14, "score": 85}, {"name": "Charlie", "age": 16, "score": 88} ] # Filter high-performing students high_performers = list(filter(lambda student: student['score'] > 90, students)) # Transform student data student_names = list(map(lambda student: student['name'], students))
```

Functional Programming Challenge:

1. Implement data transformation using lambda and higher-order functions:

   Tasks: a) Filter students by age and score b) Transform student data c) Create summary reports

Python Programming Foundations: Young Developers Workshop

SECTION 1: PROGRAMMING FOUNDATIONS EXPLORATION

Learning Objectives:

- Understand basic programming concepts
- Explore Python's fundamental data types
- Practice creating and manipulating variables

Variables and Data Types Deep Dive

Key Concept: What is a Variable?

A variable is like a container that stores information. Think of it as a labeled box where you can store different types of data that can change during your program's execution.

Complete the following activities to demonstrate your understanding:

1. Create and explain three different variables:

```
# Example Variable Declarations student_name = "Alex Johnson" # String variable age = 14 # Integer variable average_score = 85.5 # Float variable
```

Explain each variable's purpose and data type: a) student_name: b) age: c) average_score:

2. Data Type Conversion Challenge:

```
# Convert between data types age_as_string = str(14) score_as_integer = int(85.5)
```

What happens in each conversion? Explain the results:

Computational Thinking Exercise

Design a simple algorithm to solve a real-world problem:

1. Problem: Create a program to calculate a student's final grade

Steps to solve this problem: 1. 2. 3. 4.

2. Flowchart your solution:

[Draw your flowchart here]

SECTION 2: CODING CHALLENGES AND PROBLEM SOLVING

Conditional Logic Mastery

> **Key Concept: Conditional Statements**
>
> Conditional statements allow your program to make decisions based on different conditions. They use keywords like 'if', 'elif', and 'else' to create logical branching.

Complete the following coding challenges:

1. Create a grade classification function:

```
def classify_grade(score): # Your code here to determine letter grade pass
```

Implement the function to return: - A for 90-100 - B for 80-89 - C for 70-79 - D for 60-69 - F for below 60

2. Age Verification Challenge:

```
def check_eligibility(age): # Determine eligibility for activities pass
```

Create a function that: - Allows entry to teen club (13-17) - Provides different messages for different age groups

Workshop Completion

Congratulations on completing the Python Programming Foundations Workshop! You've taken your first steps into the exciting world of coding and computational thinking.

🏆 You are now ready to continue your programming journey! 🏆