Python Lab Work: A Bridge from C++

Welcome to this lab work designed to help you transition your programming skills from C++ to Python. While Python and C++ have many differences, they share fundamental concepts. This lab will focus on practicing Python's unique syntax for things you already know, and then applying that knowledge to convert simple C++ programs.

Part 1: Python Basics - Hands-on Practice

In this section, you will write and run simple Python scripts to solidify your understanding of the concepts we've covered.

1. Variables and Data Types

In Python, you don't need to declare a variable's type. The interpreter infers it.

Task: Open a new Python file and write the following script.

```
# C++ equivalent: int age = 20;
age = 20
# C++ equivalent: std::string name = "Ahmad";
name = "Ahmad"
# C++ equivalent: double pi = 3.14159;
pi = 3.14159
# C++ equivalent: bool is_student = true;
is_student = True

print(age)
print(name)
print(pi)
print(is_student)
```

What to notice:

- Variable names are like C++.
- No semicolons at the end of a statement.
- Boolean values are capitalized: `True` or `False`.

2. Basic Input/Output

Getting input from the user is different in Python. The `input()` function always returns a string. You'll often need to cast it to another type, like an `int` or `float`.

Task: Write a script that asks the user for their name and age, then prints a greeting.

```
# Get a string input from the user
name = input("Enter your name: ")

# Get a string input and convert it to an integer
age = int(input("Enter your age: "))

print("Salam,", name)
print("You are", age, "years old.")
```

3. Conditional Statements (If-Else)

Python uses a colon (`:`) and indentation to define code blocks, replacing the curly braces `{}` from C++.

Task: Write a script that checks if a number is positive, negative, or zero.

```
# Get a number from the user
number = int(input("Enter a number: "))

if number > 0:
    print("The number is positive.")
elif number < 0:
    print("The number is negative.")
else:
    print("The number is zero.")
```

What to notice:

- The `if`, `elif` (else if), and `else` statements end with a colon.
- The code inside each block is indented by four spaces. Python enforces this indentation.

4. Scope and Indentation

In C++, curly braces `{}` define the scope of a variable. In Python, **indentation** defines the scope. A variable created inside an indented block is local to that block.

Task: Write the Python equivalent of the following C++ code. Pay attention to what happens when you try to access the variable `y` after the `if` statement.

C++ Code:

```
#include <iostream>

int main() {
    int x = 10;
    if (x > 5) {
        int y = 20; // y is local to this if block
        std::cout << "Inside if: y = " << y << std::endl;
    }
    // std::cout << "Outside if: y = " << y << std::endl; // This would cause a
compile error
    return 0;
}
```

Your Task (Python):

```
x = 10
if x > 5:
    y = 20 # y is local to this indented block
    print("Inside if: y =", y)

# This line will cause an error because y is not in scope here
# print("Outside if: y =", y)
```

What to notice:

- The Python code behaves similarly to the C++ code. The variable `y` is only accessible inside the `if` block because of its indentation.
- If you uncomment the last line in the Python code, you will get a `NameError` because `y` does not exist in that scope.

Part 2: C++ to Python Conversion Exercises

Now, let's apply what you've learned. For each of the following C++ programs, write the equivalent code in Python.

Exercise 1: Area of a Rectangle

C++ Program:

```
#include <iostream>

int main() {
    double width, height, area;
    std::cout << "Enter the width: ";
    std::cin >> width;
    std::cout << "Enter the height: ";
    std::cin >> height;

    area = width * height;
    std::cout << "The area is: " << area << std::endl;

    return 0;
}
```

Your Task (Python): Convert this simple C++ program into Python. Pay attention to how you declare variables and how you get input from the user.

Exercise 2: Checking Even or Odd

C++ Program:

```
#include <iostream>

int main() {
    int number;
    std::cout << "Enter an integer: ";
    std::cin >> number;

    if (number % 2 == 0) {
        std::cout << number << " is even.";
    } else {
        std::cout << number << " is odd.";
    }

    return 0;
}
```

Your Task (Python): Write the equivalent Python script for the C++ program above. Pay attention to the input function and conditional statements.

Challenge Task: Nested Conditional Statements

This challenge will test your understanding of indentation for creating nested code blocks.

C++ Program:

```
#include <iostream>

int main() {
    int score;
    std::cout << "Enter your score: ";
    std::cin >> score;

    if (score >= 90) {
        std::cout << "You got an A.";
        if (score > 95) {
            std::cout << " Excellent!";
        }
    } else {
        std::cout << "You did not get an A.";
    }
    std::cout << std::endl;
    return 0;
}
```

Your Task (Python): Convert the C++ program above into Python. Pay close attention to the indentation required for the inner `if` statement to ensure it is nested correctly within the outer `if` block.

Solutions

Solution for Exercise 1: Area of a Rectangle (Python)

```
width = float(input("Enter the width: "))
height = float(input("Enter the height: "))

area = width * height
print("The area is:", area)
```

Solution for Exercise 2: Checking Even or Odd (Python)

```
number = int(input("Enter an integer: "))

if number % 2 == 0:
    print(number, "is even.")
else:
    print(number, "is odd.")
```

Solution for Challenge Task (Python)

```
score = int(input("Enter your score: "))

if score >= 90:
    print("You got an A.", end="")
    if score > 95:
        print(" Excellent!", end="")
```

```
else:
    print("You did not get an A.", end="")

print()
```