COMP11124 Object Oriented Programming - Week 1 Practical Lab Summary

Introduction

This document summarizes the solutions for the Week 1 Python lab exercises. Each section includes:

- A brief explanation of the task
- The relevant Python code
- The output produced by the code
- Key concepts highlighted in bullet points

Exercise 1: Variables and Types

Explanation: This activity looks into the introductory aspects of data types and type casting in Python. It shows how to create variables of various types, how to convert one to another, and shows much more complex types such as lists, dictionaries and tuples. In programming, variables are very basic because they enable you to save and alter data. Knowledge about the data types available assists you in selecting the best type of data along with the operation. Type casting is significant in cases when you want to change data type of data i.e., translation between string type and integer type. Control of variables and types is a crucial concept behind all programming activities since it will help you design versatile and dependable code. Being aware of and utilizing the type will eliminate some of the mistakes and it will structurally and visually build stronger and more readable code.

Code:

```python
is_active = True
student_age = 42
pi_value = 3.14
welcome_message = "Hello World"

print("Exercise 1: Variables and Types")
print(f"Type of is_active: {type(is_active)}")
print(f"Type of student_age: {type(student_age)}")
print(f"Type of pi_value: {type(pi_value)}")
print(f"Type of welcome_message: {type(welcome_message)}")

# Additional variable types
favorite_numbers = [7, 13, 21]  # List
student_info = {'name': 'Alice', 'id': 12345}  # Dictionary
coordinates = (10.5, 20.3)  # Tuple
no_value = None  # NoneType
print(f"Type of favorite_numbers: {type(favorite_numbers)}")
print(f"Type of student_info: {type(student_info)}")
print(f"Type of coordinates: {type(coordinates)}")
print(f"Type of no_value: {type(no_value)}")
```

```python
# Casting variables
integer_number = 5
floating_number = 5.5
is_logged_in = True

print(f"integer_number: {integer_number}")
print(f"floating_number: {floating_number}")
print(f"is_logged_in: {is_logged_in}")

# Casting to different types
int_to_float = float(integer_number)
float_to_int = int(floating_number)
bool_to_int = int(is_logged_in)

print("\nCasting Results:")
print(f"int to float: {int_to_float}")
print(f"float to int: {float_to_int}")
print(f"bool to int: {bool_to_int}")

# More casting examples
string_number = "123"
converted_int = int(string_number)
converted_str = str(456)
print(f"string to int: {converted_int}")
print(f"int to string: {converted_str}")

# Demonstrate boolean casting
zero_value = 0
nonzero_value = 15
print(f"bool(0): {bool(zero_value)}")
print(f"bool(15): {bool(nonzero_value)}")
```

Sample Output:

```

Exercise 1: Types and variables
Type of is_active: <class 'bool'>
Studentage: <class 'int'>
Pi_value type: float
Welcome_message type: <class 'str'>
Type of favorite_numbers: <class list>
Type of student_info: <class dict>:
Type of coordinates: <class 'tuple'>
Type of no_value: <type 'NoneType'>
integer_number: 5
floating_number: 5.5
is_logged_in: True

Casting Results:
int to float: 5.0
float to int: 5
bool to int: 1
string to int: 123
```

```
int to string: 456
bool(0): False
bool(15): True
```

Key Concepts:

- Python supports multiple data types: `bool`, `int`, `float`, `str`, `list`, `dict`, `tuple`, `NoneType`
- Type casting allows conversion between types
- Use `type()` to check a variable's type
- Boolean casting: `bool(0)` is `False`, any nonzero number is `True`

Exercise 2: Arithmetic Operators

Explanation: It is the practice to illustrate the use of arithmetic operators in Python, e.g. addition, subtraction, multiplication, division, floor division, the modulus and exponentiation in Python. It also involves finding averages, areas and perimeters of objects and the calculation of area of a triangle is also introduced under it. The simplest operation in programming is the use of arithmetic operators and this is the basis of every computation in programming. One of the most important skills of a programmer is knowing how to work with arithmetic operators and how to transfer them to the real life problems. These are necessary because they equip one with skills that are needed to address more serious challenges in the future in terms of programming. Giving yourself routine with these operations, you learn to think consistently and to solve problems, and you also learn how to solve large problems by reducing them into smaller and simpler calculations.

Code:

```
add_result = 10 + 5
print("Addition:", add_result)
subtract_result = 20 - 8
print("Subtraction:", subtract_result)
multiply_result = 6 * 4
print("Multiplication:", multiply_result)
divide_result = 15 / 3
print("Division:", divide_result)
floor_div_result = 17 // 4
print("Floor Division:", floor_div_result)
modulus_result = 17 % 4
print("Modulus:", modulus_result)
power_result = 2 ** 3
print("Exponentiation:", power_result)

# Additional arithmetic examples
apples = 7
oranges = 3
print(f"Total fruits: {apples + oranges}")
print(f"Difference in fruits: {apples - oranges}")
print(f"Product of fruits: {apples * oranges}")
print(f"Ratio of apples to oranges: {apples / oranges}")
print(f"Remainder when apples divided by oranges: {apples % oranges}")
```

```
# Calculating the Average
math_score = 25
science_score = 75
average_score = (math_score + science_score) / 2
print(f"\nAverage of {math_score} and {science_score} is: {average_score}")

# Calculate the Area and Perimeter of a Rectangle
rect_length = 10
rect_width = 5
rect_area = rect_length * rect_width
rect_perimeter = 2 * (rect_length + rect_width)
print(f"Rectangle with length {rect_length} and width {rect_width} has area: {rect_area}")
print(f"Perimeter of rectangle: {rect_perimeter}")

# Calculate the area of a triangle
triangle_base = 8
triangle_height = 6
triangle_area = 0.5 * triangle_base * triangle_height
print(f"Area of triangle with base {triangle_base} and height {triangle_height}: {triangle_area}")
```

Sample Output:

```
Addition: 15
Subtraction: 12
Multiplication: 24
Division: 5.0
Floor Division: 4
Modulus: 1
Exponentiation: 8
Total fruits: 10
Difference in fruits: 4
Product of fruits: 21
Ratio of apples to oranges: 2.3333333333333335
Remainder when apples divided by oranges: 1

Average of 25 and 75 is: 50.0
Rectangle with length 10 and width 5 has area: 50
Perimeter of rectangle: 30
Area of triangle with base 8 and height 6: 24.0
```

Key Concepts:

- Arithmetic operators: `+`, `-`, `*`, `/`, `//`, `%`, `**`
- Calculating averages, areas, perimeters
- Order of operations (PEMDAS/BODMAS)
- Use of variables in real-world calculations

Exercise 3: Strings and f-Strings

Explanation: String manipulation includes changing case, string replacement, length measures, split and join and f-string to create a formatted output. It also shows formatting of floats and aggregation of values in f-strings. Strings are sets of characters that are utilized to portray the text and you will be taught how to transform the case of a string, substitute parts of a string as well as the length of the given string. They come in handy when processing and controlling text data that pertains to most programming work. The use of string operations is needed to perform such tasks as verification of user input values, data cleaning, and reports. String efficiency will allow you to cope with a variety of data processing options.

Code:

```
course_description = "This class covers OOP."
print(f"Original string: {course_description}")

upper_description = course_description.upper()
lower_description = course_description.lower()
expanded_description = course_description.replace("OOP", "Object Oriented
Programming")
description_length = len(course_description)

print(f"Uppercase: {upper_description}")
print(f"Lowercase: {lower_description}")
print(f"Replaced: {expanded_description}")
print(f"Length: {description_length}")

# More string operations
words = course_description.split()
print(f"Split string: {words}")
joined_words = "-".join(words)
print(f"Joined string: {joined_words}")

# f-String Task
student_name = "Peter"
class_count = 10
campus_name = "Paisley"
intro_text = f"My name is {student_name} and I am studying {class_count} classes
in {campus_name}"
print(intro_text)

# f-string with calculations
num_books = 5
num_notebooks = 3
print(f"I have {num_books} books and {num_notebooks} notebooks. Total: {num_books
+ num_notebooks}")

# String formatting with float
gpa = 3.6789
print(f"My GPA is {gpa:.2f}")
```

Sample Output:

```
Original string: This class covers OOP.
Uppercase: THIS CLASS COVERS OOP.
Lowercase: this class covers oop.
Replaced: This class covers Object Oriented Programming.
Length: 22
Split string: ['This', 'class', 'covers', 'OOP.']
Joined string: This-class-covers-OOP.
My name is Peter and I am studying 10 classes in Paisley
I have 5 books and 3 notebooks. Total: 8
My GPA is 3.68
```

Key Concepts:

- String methods: `.upper()`, `.lower()`, `.replace()`, `.split()`, `.join()`, `len()`
- f-Strings for formatted output and calculations
- Formatting floats in f-strings

Exercise 4: Temperature Converter

Explanation: Through this exercise, it applies temperature converter which converts Celsius to Fahrenheit and Kelvin, and also showing conversion of Fahrenheit and Kelvin to the other units. It has user input (safe-ty commented). You get to know how to apply the formulas in converting temperature and present the outcome in a friendly manner to the user. This program makes your knowledge of variables, arithmetic, and string formats stronger, as well as demonstrating that real-world problems can be solved through the usage of programming. Translating mathematical formulas into coded form makes it a good skill to possess particularly in the scientific and engineering field. The exercise also emphasizes on the need to have clear output, a factor that makes users trust and understand the results of your program.

Code:

```
celsius_temp = 25  # Example input
fahrenheit_temp = (celsius_temp * 9/5) + 32  # Celsius to Fahrenheit
kelvin_temp = celsius_temp + 273.15     # Celsius to Kelvin

print("Welcome to the Temperature Converter!")
print(f"The temperature you have entered is {celsius_temp} degree Celsius.")
print("\nConverted Temperatures:")
print(f"{celsius_temp} degree Celsius is equal to {fahrenheit_temp} Fahrenheit.")
print(f"{celsius_temp} degree Celsius is equal to {kelvin_temp} Kelvin.")
print("\nThank you for using the Temperature Converter!")

# Convert Fahrenheit to Celsius and Kelvin
input_fahrenheit = 77
celsius_from_f = (input_fahrenheit - 32) * 5/9
kelvin_from_f = celsius_from_f + 273.15
print(f"\n{input_fahrenheit} Fahrenheit is {celsius_from_f} Celsius and
{kelvin_from_f} Kelvin.")

# Convert Kelvin to Celsius and Fahrenheit
input_kelvin = 300
celsius_from_k = input_kelvin - 273.15
fahrenheit_from_k = (celsius_from_k * 9/5) + 32
print(f"{input_kelvin} Kelvin is {celsius_from_k} Celsius and {fahrenheit_from_k}
Fahrenheit.")
```

Sample Output:

```
You are welcome to the Temperature Converter!
The temperature that you have set is 25 degree Celsius.

Converted Temperatures:
77.0 Fahrenheit converts into 25 degree Celsius.
The temperature of 25 degree Celsius means 298.15 Kelvin.

Thank you visiting Temperature Converter!

At 77 Fahrenheit that would be 25.0 Celsius or 298.15 Kelvin.
26.850000000000023 Celsius is 300 Kelvin and 80.33000000000004 Fahrenheit.
```

Key Concepts:

- Arithmetic operations for unit conversion
- Printing formatted output
- Multi-step calculations
- User input (with `input()`, commented for safety)

Conclusion

These exercises introduce fundamental Python concepts, including variables, types, arithmetic, strings, and basic input/output. Mastery of these basics is essential for progressing in object-oriented programming and more advanced topics.