**Introduction to Python, Day 1**

**Python in the job market**

There are various job fields that python has to offer. Integrating applications with MySQL is in high demand because MySQL and Python are both open-source applications. Network programming in Python is another option, which requires an extensive knowledge of how networking is controlled.

**History of Python**

Python was created in the late 1980s. It is a relatively simple language that includes a standard library that provides modules for many processes that programs deal with.

**Invoking the Interpreter**

Python was created in the late 1980s. It is a relatively simple language that includes a standard library that provides modules for many processes that programs deal with.

**Interactive Mode**

The interactive mode of the Python interpreter allows programmers to interactively run Python code. It provides a prompt where we can type and execute Python statements one at a time. This is a great way to quickly test small snippets of code, experiment with language features, and get immediate feedback.

**Comments in Python**

Comments are a programming language construct used to insert human-readable code of a program. They are ignored by the compiler and interpreter but can be potentially significant to programmers. Comments in python start with the hash character, #.

**Using Python**

Python is a versatile and popular programming language that can be used for a wide range of tasks, from web development to scientific computing and more.

**Day 1 Reflections**

Python is a high-level, versatile, and easy-to-learn programming language. It's known for its clean and readable syntax, which makes it a great choice for beginners and professionals alike. It can be used for a wide range of applications such as web development, scientific computing, data analysis, artificial intelligence, automation, and more.

**Day 2**

**Introduction to Variables**

Variables are a temporary storage space in a computer’s memory. When a variable’s value changes the program’s current state also changes. A variable act as a container to hold a different number of data items or values.

**Daily Notes - Using variables**

 # Define variables  
num1 = 10  
num2 = 20  
sum\_result = num1 + num2  
difference = num1 - num2  
product = num1 \* num2  
print("Sum:", sum\_result)  
print("Difference:", difference)  
print("Product:", product)

**Casting**

 Casting in Python refers to the process of converting a variable from one data type to another. Python provides several built-in functions for type casting. int(), float(), str(), bool(), etc.

**Consolidating Learnings**  
first\_number = 2  
second\_number = 4  
total = first\_number + second\_number  
# Subtract 3 from the total  
result = total - 3  
print(f"The first number is {first\_number}")  
print(f"The second number is {second\_number}")  
print(f"{first\_number} + {second\_number} = {total}")  
print(f"{total} - 3 = {result}")

**Reflections**

 In Python, variables are used to store data values, they act as symbolic names that reference specific memory locations containing data. Variables allow us to easily manipulate and work with data in your programs.

**Day 3**

**Data Types**

Data types are classifications that categorize various types of data based on their nature, characteristics, and the operations that can be performed on them. They define the type of values that a variable can hold, and they determine how those values behave when used in operations. We have integers, Booleans, floating point and many other data types in python.

**Integers**

Integers (int) are a data type used to represent whole numbers, both positive and negative, without any decimal or fractional component. They can be of any length and are limited only by the memory available on a machine.

**Floating point numbers**

Floating-point numbers in are a data type used to represent real numbers, which can have both an integer part and a fractional part. They are called "floating-point" because the decimal point can "float"; that is, it can support a variable number of digits before and after it.

**Unpacking Argument Lists**

A string is a data type used to represent text. It is a sequence of characters enclosed in either single quotes (') or double quotes ("). Python doesn't distinguish between characters and strings of length one; they are both treated as strings.

**Lambda Expressions**

A lambda expression in Python is a small, anonymous function that can have any number of parameters, but it can only have one expression. Lambda functions are also known as "anonymous functions" because they don't have a name like a regular function defined with the def keyword.  
The syntax is**: lambda arguments: expression**

**Conventions about the content and formatting of documentation strings**

Documentation strings, often referred to as docstrings, are used to provide information about the purpose and usage of functions, classes, modules, and packages in Python. They help improve code readability and serve as a form of inline documentation.

**Consolidating Learnings**

x = bool()  
y = bool()  
print ('Enter x as 1 or 0:')  
x = int(input())  
print ('Enter y as 1 or 0:')  
y = int(input())  
z = str(not bool(x or y))  
print ('The Boolean value of x is', str(bool(x)))  
print ('The Boolean value of y is', str(bool(y)))  
print ('The Boolean value of (x or y) is', str(bool(x or y)))  
print ('The Boolean value of (x not y) is', z).

**Activity 2**

def print\_animal\_info(animalType, animalSpecies, waterConsumption):  
print(f"The {animalSpecies} is a type of {animalType}.")  
print(f"It drinks {waterConsumption:.2f} litres of water per day.")  
animalType = input("Enter the type of animal: ")  
animalSpecies = input("Enter the species of the animal: ")  
waterConsumption = float(input("Enter how many litres of water the animal drinks a day: "))  
print\_animal\_info(animalType, animalSpecies, waterConsumption)

**Day 4**

**Introduction to operators**

Operators in Python are special symbols or keywords that are used to perform operations on one or more operands (values or variables). They can perform a wide range of tasks, from basic arithmetic calculations to logical comparisons and more complex operations.

**Using Operators**

Operators in Python are symbols or keywords that perform operations on one or more operands (values). They can be used to perform various computations and manipulate data. There are eight comparison operations in Python. They all have the same priority, but their priority is higher than the Boolean operations (and-or-not operators).

**Activity 1**

def fill\_bottles(litres):  
bottle\_size = 0.5 # Size of a bottle in liters  
bottles = int(litres / bottle\_size)  
remaining\_litres = round(litres % bottle\_size, 2)  
print(f"{litres:.2f}L water will fill {bottles} bottles ({remaining\_litres:.2f}L remains)")  
# Taking user input for the number of litres  
litres = float(input("Enter number of litres: "))  
# Calling the function to calculate and print the result  
fill\_bottles(litres)

**Activity 2**

 def generate\_report(released\_millilitres):  
megalitre\_capacity = 1000000 # 1 megalitre in millilitres  
# Convert released millilitres to litres  
released\_litres = released\_millilitres / 1000  
# Calculate the remaining litres  
remaining\_litres = megalitre\_capacity - released\_litres  
# Calculate the percentage of litres left  
percentage\_left = (remaining\_litres / megalitre\_capacity) \* 100  
# Print the report  
print(f"Litres Released: {released\_litres}L")  
print(f"Litres Remaining: {remaining\_litres:.2f}L")  
print(f"Percentage Left: {percentage\_left:.2f}%")  
# Taking user input for the millilitres released  
released\_millilitres = float(input("Enter millilitres of water released: "))  
# Calling the function to generate the report  
generate\_report(released\_millilitres)