**Week 1: Python Introduction**

**Day 1**

Functions in Python:

Functions in Python are blocks of reusable code that perform a specific task or a set of tasks. They are defined using the def keyword followed by the function name and a pair of parentheses. Functions can take parameters (input data) and return values (output data). Here are some key points about functions in Python:

A screenshot of a computer

Description automatically generated

Calling Functions: You can call a function by using its name followed by parentheses and passing any required arguments.

A computer code on a black background

Description automatically generated

Comments in Python:

Comments in Python are used to add explanatory notes or documentation to your code. They are not executed by the Python interpreter and are only meant for human readers. Python supports two types of comments:

Single-line Comments: These comments start with the # symbol and continue to the end of the line. They are typically used for short comments.

A screenshot of a computer

Description automatically generated

**Day 2**

**Certainly, here's a summary of the topics covered:**

**1. \*\*Two Different Data Types in Python:\*\***

**Python uses two main categories of data types:**

**- Primitive Data Types: These include `int` (integers), `float` (floating-point numbers), `str` (strings), `bool` (Booleans), and `NoneType`. They represent basic data elements.**

**- Composite Data Types: These include `list`, `tuple`, `set`, and `dict`. They allow you to group and manipulate collections of data.**

**2. \*\*Converting Between Data Types:\*\***

**Python provides methods for converting between data types, both implicitly and explicitly. Implicit conversion occurs automatically in some situations, while explicit type conversion, also known as casting, is done using functions like `int()`, `float()`, and others. This allows you to change the data type of a value when needed.**

**3. \*\*Naming Conventions in Python:\*\***

**Python follows naming conventions to maintain code readability and consistency:**

**- Variables and function names are in lowercase with words separated by underscores.**

**- Constants are in uppercase with underscores.**

**- Module names are in lowercase.**

**- Class names follow CamelCase (CapWords).**

**- Method names are in lowercase with underscores.**

**4. \*\*Casting:\*\***

**Casting is the process of converting a value from one data type to another. It's essential when you need data in a specific format or for performing operations that require compatible data types. Python provides casting functions like `int()`, `float()`, `str()`, and others for explicit type conversion.**

**Day 3:**

**1. \*\*Defining Functions:\*\***

**- Functions are reusable blocks of code defined using the `def` keyword.**

**- They can take input parameters, perform operations, and return values.**

**2. \*\*Keyword Arguments:\*\***

**- Keyword arguments allow you to specify values for function parameters using parameter names.**

**- This enhances code readability, especially in functions with many parameters.**

**3. \*\*Arbitrary Argument Lists:\*\***

**- `\*args` and `\*\*kwargs` enable functions to accept variable numbers of non-keyword and keyword arguments, respectively.**

**- They allow for flexibility when dealing with different inputs.**

**4. \*\*Unpacking Argument Lists:\*\***

**- Unpacking uses `\*` and `\*\*` to take elements from sequences and pass them as separate arguments to a function.**

**- It simplifies the passing of multiple arguments, making code more concise.**

**5. \*\*Lambda Expressions:\*\***

**- Lambda expressions are small, anonymous functions defined with the `lambda` keyword.**

**- They are used for simple, one-time operations, often within functions like `map()`, `filter()`, and `sorted()`.**

**DAY FOUR:**

**1. \*\*Modules:\*\***

**- Modules are files containing Python code for organizing and reusing code.**

**- They can define functions, classes, and variables.**

**- Python has built-in modules and allows the creation of custom modules.**

**2. \*\*Executing Modules as Scripts:\*\***

**- Modules can be run as scripts by checking the `\_\_name\_\_` variable.**

**- Code under `if \_\_name\_\_ == '\_\_main\_\_':` executes when the module is run directly.**

**3. \*\*Module Search Path:\*\***

**- Python searches for modules in specific directories, including the current directory and paths specified in `sys.path`.**

**4. \*\*Standard Modules:\*\***

**- Python includes a wide range of built-in modules for various tasks, such as `math`, `os`, and `datetime`.**

**5. \*\*Packages:\*\***

**- Packages are a way to organize related modules into directories with an `\_\_init\_\_.py` file.**

**- They help manage and structure larger Python projects.**

**6. \*\*Importing from a Package:\*\***

**- Modules from a package are imported using dot notation, specifying the package and module names.**

**7. \*\*Intra-Package References:\*\***

**- Within a package, you can use relative imports to refer to other modules from the same package.**

**8. \*\*Packages in Multiple Directories:\*\***

**- Python allows packages to span multiple directories, making it easier to manage and structure code across different folders.**

**Understanding modules and packages is crucial for organizing and maintaining Python code, whether you're working on small scripts or complex applications. Packages and their use of directories provide a way to create scalable and well-structured projects.**