**ASSIGNMENT**

**(By: Joel Jacob John**

**ID: STB03-T0003)**

**Python**

**Python Datatypes:**

Python supports a variety of data types that facilitate the storage and manipulation of information. These include:

- int: Integer data type for whole numbers.

- float: Floating-point data type for decimal numbers.

- str: String data type for text.

- list: Ordered collection of elements.

- tuple: Immutable ordered collection of elements.

- dict: Associative array or dictionary, consisting of key-value pairs.

- set: Unordered collection of unique elements.

- bool: Boolean data type representing True or False.

- None: Represents the absence of a value or a null value.

**Functions:**

Functions in Python encapsulate a set of instructions and can be defined using the `def` keyword. Key concepts include:

1. Function Definition:

- def function\_name(parameters):

- Defines a function with a specified name and parameters.

2. Return Statement:

- return expression

- Specifies the value to be returned by the function.

3. Arguments:

- Functions can take positional and keyword arguments.

4. Scope:

- Variables defined within a function have local scope.

5. Lambda Functions:

- Anonymous functions created using the `lambda` keyword.

**If-Else Statements:**

Conditional statements in Python enable the execution of different code blocks based on specified conditions.

1. If Statement:

- Executes a block of code if a condition is True.

2. Else Statement:

- Executes a block of code if the preceding if condition is False.

3. Elif Statement:

- Stands for "else if" and is used for multiple condition checks.

4. Nested If-Else:

- Conditional statements can be nested for complex logic.

**Loops:**

Loops are used to iterate over sequences or execute a block of code repeatedly.

1. For Loop:

- Iterates over a sequence (e.g., list, tuple) or other iterable objects.

2. While Loop:

- Repeatedly executes a block of code as long as a specified condition is True.

3. Break and Continue:

- `break`: Exits the loop prematurely.

- `continue`: Skips the rest of the loop's code and continues to the next iteration.

**Object-Oriented Programming (OOP):**

OOP is a programming paradigm that uses objects and classes. Key concepts include:

1. Class:

- Blueprint for creating objects. Defines attributes and methods.

2. Object:

- Instance of a class, representing a real-world entity.

3. Inheritance:

- Allows a class to inherit properties and methods from another class.

4. Encapsulation:

- Bundling of data (attributes) and methods that operate on the data within a single unit (class).

5. Polymorphism:

- Ability of a class to take on multiple forms by implementing methods with the same name but different functionality.

Understanding these concepts provides a solid foundation for Python programming, enabling developers to create efficient and maintainable code.

**SPARK**

**Spark Architecture:**

1. Driver:

- The driver is the main program responsible for executing the main() function of a Spark application.

- It converts the user's code into tasks and schedules them on worker nodes.

- Maintains information about the application, including the SparkContext, DAG, and metadata for fault recovery.

- For each job, a new Spark session is created and terminated upon completion.

2. Cluster Manager:

- Acquires and allocates resources on the cluster for Spark applications.

- Supports multiple cluster managers like standalone, Mesos, and YARN.

- Efficiently manages resources to ensure tasks are executed on available worker nodes.

- Handles resource negotiations between the driver and worker nodes.

3. Worker Nodes:

- Individual machines that execute tasks for a Spark application.

- Each worker node has its own executor processes, responsible for running tasks and storing data.

- Executors communicate with the driver and execute assigned tasks.

- Follows a master-slave architecture, with the cluster manager allocating resources for each worker.

**Spark APIs:**

1. Resilient Distributed Dataset (RDD):

- Not serializable, leading to slower performance.

- Limited memory power.

2. Dataset:

- Faces similar issues as RDD.

3. DataFrame:

- Preferred API as it combines the best features of RDD and Dataset.

- Offers improved performance and optimization capabilities.

- Represents distributed data in tabular form, enhancing ease of use.

**Spark Operations:**

1. Transformations:

- Create a new DataFrame after each operation.

- Follows the Directed Acyclic Graph (DAG) model.

- Responsible for generating output at each operation.

2. Actions:

- Perform operations that return final results to the driver program.

- Initiates the execution of transformations and triggers the DAG.

In summary, Apache Spark leverages a distributed architecture with a driver, cluster manager, and worker nodes. The DataFrame API, with its combination of RDD and Dataset features, provides a powerful and efficient way to process and analyze distributed data. Spark operations, categorized into transformations and actions, enable the creation of complex data processing pipelines using the DAG model.