Link to the server: jup.dev.iitm.ac.in
Log in to this server with your LDAP credentials

Documentation Guide

This document contains instructions on the documentation expected in assignment submissions. Please follow the pointers mentioned in this document to ensure your code is esily understandable and debuggable.

- **Naming Variables :** Variable names should be easily understandable and appropriate to the situation. Please avoid contractions that are ambiguous.

- **Function Parameters and Return Type :** Use the typing module to add data type information to all parameters in a function and their return type as well. This module is not needed for primitive data types, but is required to fully specify compound data types such as lists and dictionaries. The main data types you might use for documentation and an example for their usage are listed below:

  - **Primitive datatypes :** `int`, `float`, `str`

```
def person_info(name: str, age: int, height: float) ->
   str:
    return f"{name} is {age} years old and {height:.2f}
       meters tall."
```

  - **List :** The `List` type represents a list of items where all items have the same type. You specify the type of the items in the list using square brackets.

```
from typing import List

def process_numbers(numbers: List[int]) -> int:
    return sum(numbers)
```

  - **Dict :** The `Dict` type represents a dictionary where keys and values each have a specific type. You specify the types for keys and values within square brackets.

```
from typing import Dict

def get_user_info() -> Dict[str, int]:
    return {"age": 25, "years_of_experience": 5}
```

  - **Tuple :** The `Tuple` type represents a tuple with a fixed number of elements, each of a specified type.

```
from typing import Tuple

def get_coordinates() -> Tuple[float, float]:
    return (19.07, 72.87)
```

- **Callable :** The `Callable` type represents a function or any callable object. You specify the types of the arguments the callable takes and the type it returns. Within the square brackets, the first element is the data types of parameters in square brackets and the second element is the return type.

```
from typing import Callable

def apply_operation(x: int, y: int, operation: Callable
    [[int, int], int]) -> int:
     return operation(x, y)
```

- **Set :** The `Set` type represents a set of unique items, all of the same type.

```
from typing import Set

def unique_numbers(numbers: Set[int]) -> int:
    return len(numbers)
```

- **numpy Array :** `np.ndarrays` or numpy arrays are very useful for various array manipulation tasks as well as for numerous mathematical problems. When using a numpy array make sure to specify it as in the example below:

```
import numpy as np
from typing import Tuple

def calculate_mean(arr: np.ndarray) -> float:
    return np.mean(arr)
```

- **pandas Dataframe :** `pd.DataFrame` or pandas dataframe is very useful for data manipulation tasks. The example below illustrates marking a variable as type dataframe:

```
import pandas as pd

def column_mean(df: pd.DataFrame, column: str) -> float:
    return df[column].mean()
```

For syntax, note that:

- `variable : data type` is used to specify the type for a parameter
- `-> : data type` is used to specify the return type of a function

- **Comments :** Comments should be used to clearly explain important steps in the algorithm and describe functions. The example below illustrates the use of multiline and single line comments and we expect such documentation in assignments as well:

```python
from typing import List

def find_max(numbers: List[int]) -> int:
    """
    Find the maximum number in a list.

    Parameters:
    numbers (List[int]): A list of integers.

    Returns:
    int: The maximum integer in the list.

    This function iterates through the list to find the
       maximum value.
    """

    # Initialize max_num to the smallest possible integer
    max_num = -1e9

    # Iterate over all numbers in the list
    for num in numbers:
        # If the current number is greater update max_num
        if num > max_num:
            max_num = num

    # Return the maximum number found
    return max_num
```

- **Black Autoformatter :** A tool that automatically formats your code for you. We have already installed it for you on the server, but in case you want to install it locally, type `pip install black` into your terminal.

  To format any file (.py or .ipynb), run `black <path_to_file>` in the terminal. Make sure you format your code before submitting.