Chapter 54 - Type Defination

Theory:

This chapter covers the topic of Type Defination.

Code Example:

```
# ■ For advanced types like Union and Optional
from typing import List, Tuple, Union, Optional
# 1■■ Returns an integer
def add(x: int, y: int) -> int:
return x + y
# 2■■ Returns a string
def greet(name: str) -> str:
return f"Hello, {name}!"
# 3■■ Returns nothing (None)
def print_sum(x: int, y: int) -> None:
print(f"Sum is: {x + y}")
# 4■■ Returns a list of strings
def get_fruits() -> List[str]:
return ["apple", "banana", "mango"]
# 5■■ Returns a tuple of int and str
def get_user_info(user_id: int) -> Tuple[int, str]:
return (user_id, "John Doe")
# 6■■ Returns either int or str
def parse_value(value: str) -> Union[int, str]:
if value.isdigit():
return int(value)
return value
# 7■■ Returns an optional string (can be None)
def find_name(name_list: List[str], query: str) -> Optional[str]:
if query in name_list:
return query
return None
# ■ Using all functions
if __name__ == "__main__":
print("1. Add:", add(5, 3)) # int
print("2. Greet:", greet("Milan")) # str
print("3. Print sum:", end=" "); print_sum(10, 20) # None
print("4. Fruits:", get_fruits()) # List[str]
print("5. User Info:", get_user_info(101)) # Tuple[int, str]
print("6. Parse '42':", parse_value("42")) # Union[int, str]
print("7. Find Name:", find_name(["Ali", "Milan"], "Milan")) # Optional[str]
print("7. Find Name (Not Found):", find_name(["Ali", "Milan"], "Sara")) # None
```