

SCALE FOR PROJECT PYTHON MODULE (/PROJECTS/PYTHON-MODULE-05)

You should evaluate 1 student in this team



Git repository

<git@vogsphere.1337.ma:vogsphere/intra-uuid-b5b22e21-301e-4774-9>

Introduction

- Remain polite, courteous, respectful and constructive throughout the evaluation process. The well-being of the community depends on it.

- Identify with the person (or the group) evaluated the eventual dysfunctions of the work. Take the time to discuss and debate the problems you have identified.

- You must consider that there might be some difference in how your peers might have understood the project's instructions and the scope of its functionalities. Always keep an open mind and grade him/her as honestly as possible. The pedagogy is valid only and only if peer-evaluation is conducted seriously.

Guidelines

Please respect the following rules:

- Test the main.py file first to verify basic functionality and type annotations
- Focus on method overriding and polymorphic behavior using abstract base classes (ABC)
- Verify that classes demonstrate proper inheritance relationships
- Check that overridden methods provide meaningful specialization
- Ensure comprehensive type annotations are used throughout all code
- Verify proper use of ABC and @abstractmethod decorators where required

Attachments

subject.pdf (<https://cdn.intra.42.fr/pdf/pdf/191345/en.subject.pdf>)

main.tar.gz (<https://cdn.intra.42.fr/document/document/43694/main.tar.gz>)

Preliminaries

Basics

- Only grade the work that is in the learner's GiT repository.
- Check that only the requested files are available in the git repository. If not, the evaluation stops here.
- Run python3 main.py to test basic functionality and type checking. The main.py should successfully import and test the learner's classes. If main.py fails completely, the evaluation stops here.
- Verify that all files include proper type annotations from the typing module. If type annotations are missing or incomplete, the evaluation stops here.

Yes

No

Exercises

Exercise 0 - Data Processor Foundation

If needed, ask the assessed person to help you with the tests.

Does the program correctly implement ALL the following features?

- File named stream_processor.py exists and can be imported
- All code includes comprehensive type annotations (typing imports, parameter types, return types)
- Proper imports: from abc import ABC, abstractmethod and from typing import Any, List, Dict,
- DataProcessor is an abstract base class (inherits from ABC)
- DataProcessor has @abstractmethod decorators on process() and validate() methods
- Method signatures match requirements:
 - process(self, data: Any) -> str
 - validate(self, data: Any) -> bool
 - format_output(self, result: str) -> str
- NumericProcessor(), TextProcessor(), LogProcessor() classes inherit from DataProcessor (no parameters required)
- All subclasses properly override the abstract methods with specialized behavior
- NumericProcessor handles numeric data (lists, individual numbers)
- TextProcessor handles string data (character/word counting)
- LogProcessor handles log entries (extracts log levels, generates alerts)
- Demonstrates polymorphic usage (same method calls on different objects)
- Run python3 stream_processor.py and verify the output matches expected format
- Ask the assessed person to explain what method overriding is and why abstract base classes are used

Yes

No

Exercise 1 - Polymorphic Streams

If needed, ask the assessed person to help you with the tests.

Does the program correctly implement ALL the following features?

- File named data_stream.py exists and can be imported
- All code includes comprehensive type annotations (typing imports, parameter types, return types)
- Proper imports: from abc import ABC, abstractmethod and from typing import Any, List, Dict,
- DataStream is an abstract base class (inherits from ABC)
- DataStream has @abstractmethod decorator on process_batch() method
- Method signatures match requirements:
 - process_batch(self, data_batch: List[Any]) -> str (abstract)
 - filter_data(self, data_batch: List[Any], criteria: Optional[str] = None) -> List[Any]
 - get_stats(self) -> Dict[str, Union[str, int, float]]
- SensorStream(stream_id), TransactionStream(stream_id), EventStream(stream_id) classes inherit from DataStream
- All subclasses properly override the abstract method and optionally override other methods with specific behavior
- SensorStream processes temperature/sensor data with alerts for extreme values
- TransactionStream processes buy/sell operations with net flow calculations
- EventStream processes system events with error detection and categorization
- StreamProcessor class can handle multiple stream types polymorphically
- Demonstrates filtering capabilities specific to each stream type
- Run python3 data_stream.py and verify the output shows polymorphic processing
- Does the implementation include comprehensive error handling and recovery mechanisms?
- Ask the assessed person to explain how polymorphism enables the StreamProcessor to handle different stream types

Yes

No

Exercise 2 - Nexus Integration

If needed, ask the assessed person to help you with the tests.

Does the program correctly implement ALL the following features?

- File named nexus_pipeline.py exists and can be imported
- All code includes comprehensive type annotations (typing imports, parameter types, return types)
- Proper imports: from abc import ABC, abstractmethod, from typing import Any, List, Dict, Union, Protocol, and collections
- ProcessingPipeline is an abstract base class with configurable stages
- Stage classes (InputStage, TransformStage, OutputStage) implement process(self, data: Any) (no constructor parameters required)
- Stages can use Protocol or duck typing (must have a process() method)
- JSONAdapter(pipeline_id), CSVAdapter(pipeline_id), StreamAdapter(pipeline_id) inherit from ProcessingPipeline

Intra Projects Python Module 05 Edit

- Each adapter overrides the process(self, data: Any) -> Union[str, Any] method for format-specificity
- NexusManager can orchestrate multiple pipelines polymorphically
- Demonstrates pipeline chaining functionality
- Includes error recovery simulation
- All classes show proper method overriding with meaningful specialization
- Run python3 nexus_pipeline.py and verify complex pipeline processing works
- Ask the assessed person to explain how the pipeline system demonstrates both inheritance and polymorphism

 Yes No

Code Quality and Understanding

Code Quality and Understanding

Evaluate the overall code quality and learner understanding:

- Code follows Python conventions and is well-structured
- Comprehensive type annotations are used throughout (All functions have parameter and return type annotations)
- Proper use of ABC and @abstractmethod decorators for abstract base classes
- Proper use of super() where appropriate in overridden methods
- Method signatures are consistent between parent and child classes
- Each subclass provides meaningful, distinct behavior (not just cosmetic changes)
- Error handling is implemented appropriately
- The learner can explain what abstract base classes (ABC) are and why they're useful
- The learner can explain the difference between method overriding and method overloading
- The learner understands when and why to use polymorphism
- The learner can demonstrate how the same interface works with different implementations
- The learner can explain the importance of type annotations in Python development
- The learner understands that @abstractmethod forces subclasses to implement specific methods

 Yes No

Ratings

Don't forget to check the flag corresponding to the defense

 Ok Outstanding project

Empty work

 Incomplete work Invalid compilation Norme Cheat Concerning situation Leaks Forbidden function Can't support / ...

Conclusion

Leave a comment on this evaluation (2048 chars max)

Finish evaluation

API General Terms of Use
(<https://profile.intra.42.fr/legal/terms/33>)

Declaration on the use of cookies
(<https://profile.intra.42.fr/legal/terms/2>)

Privacy policy
(<https://profile.intra.42.fr/legal/terms/5>)

General term of use of the site
(<https://profile.intra.42.fr/legal/terms/6>)

Rules of protection
(<https://profile.intra.42.fr/legal/terms/1>)