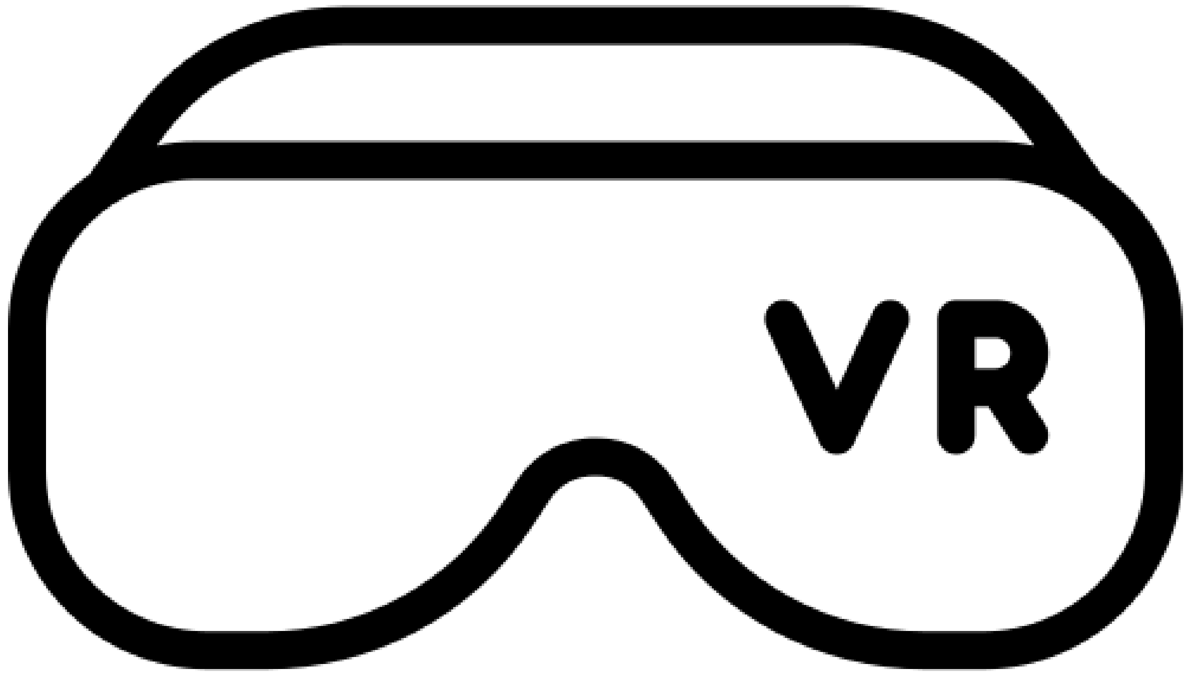
Google Developer Student Clubs
Dongguk University

AI 모델 어떻게 제공해야 할까?

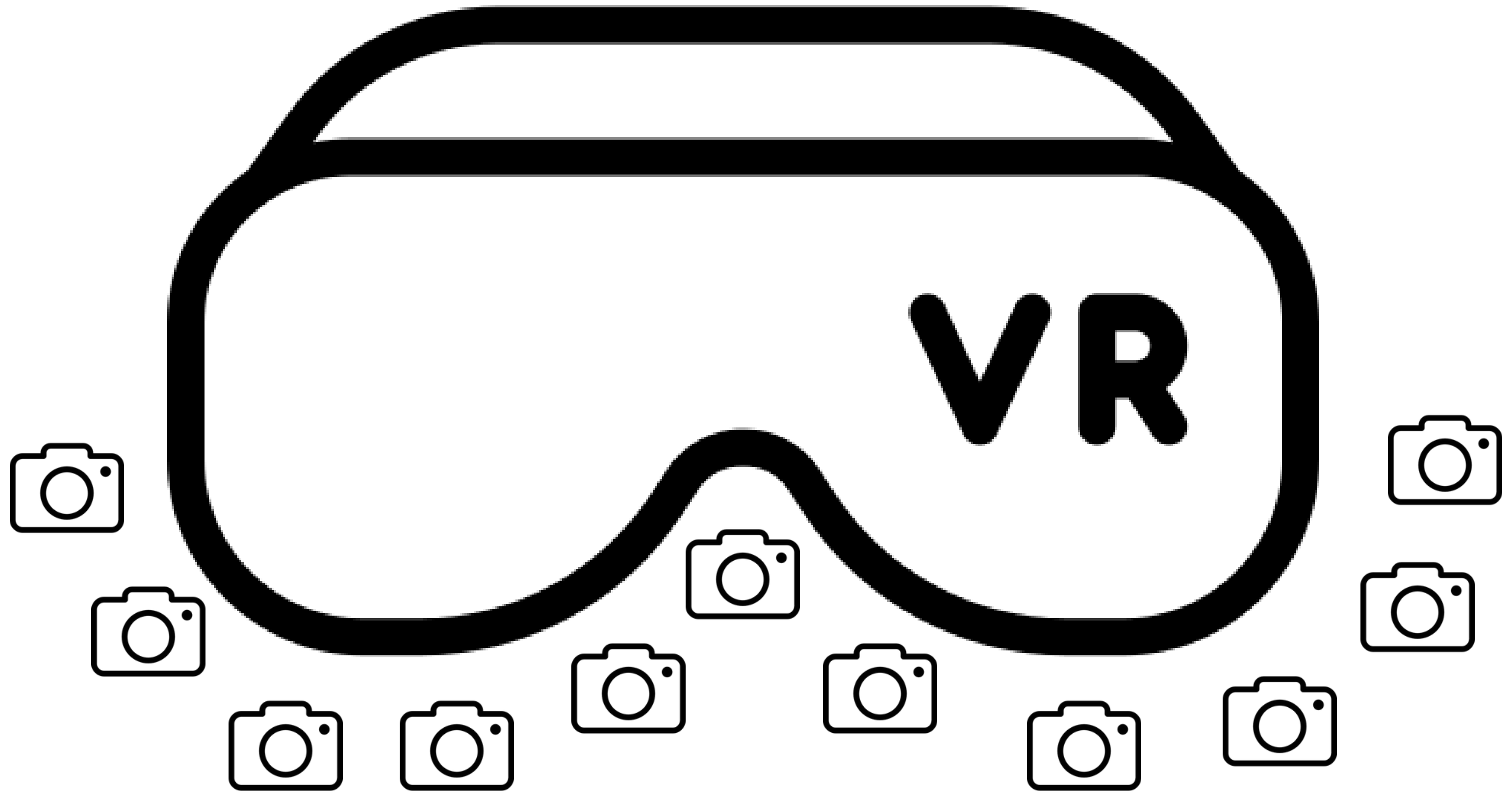
Machine Learning Design Pattern

2021111896
서버/클라우드
고나연

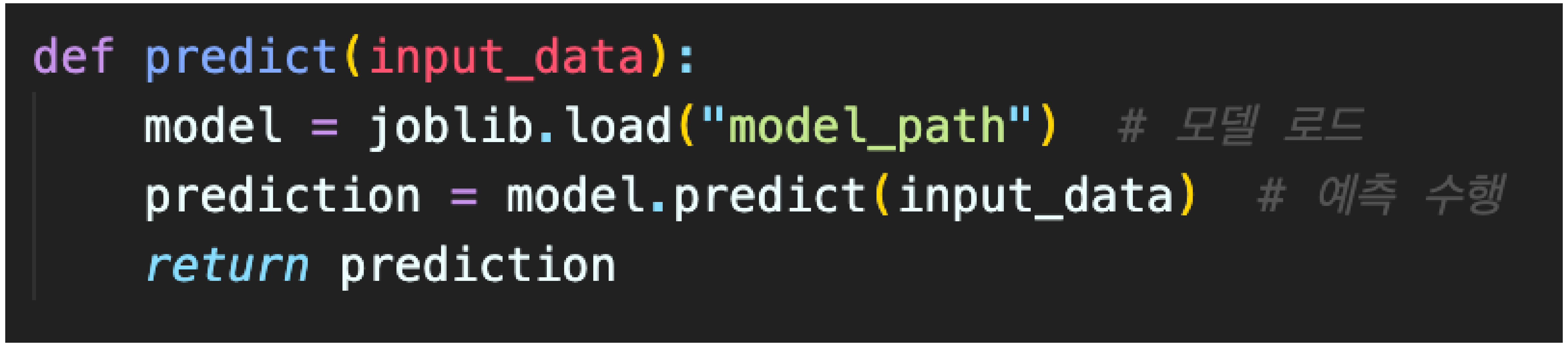
Google Developer Student Clubs
Dongguk University



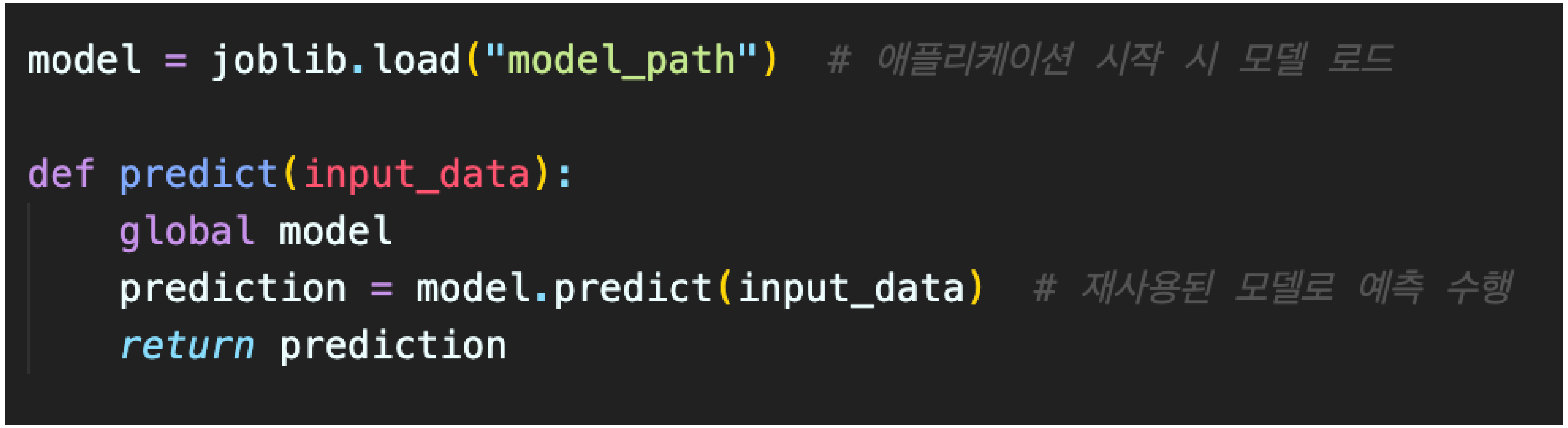
Fitness+
Beginner to advanced


VR

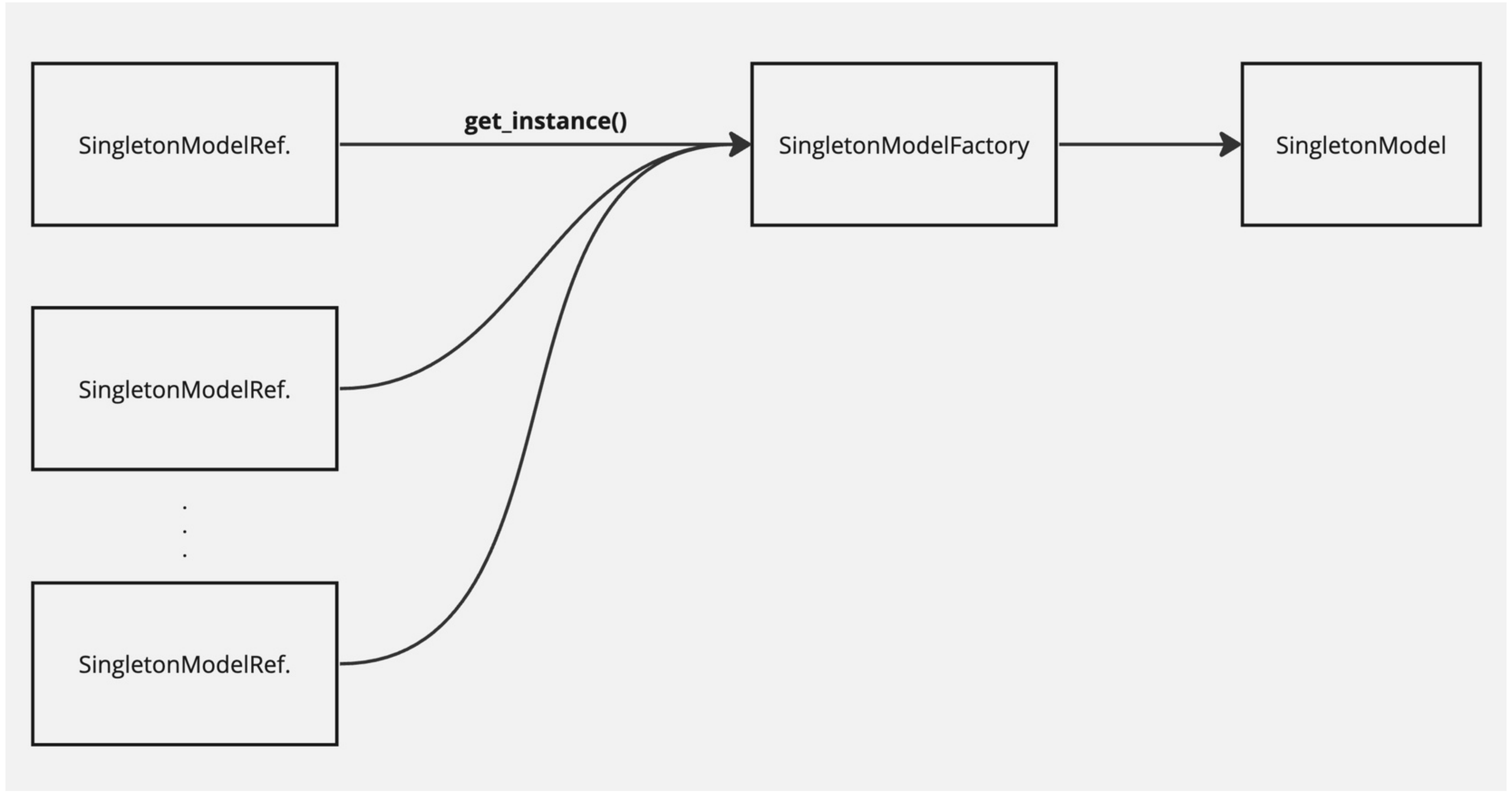
AI SERVING

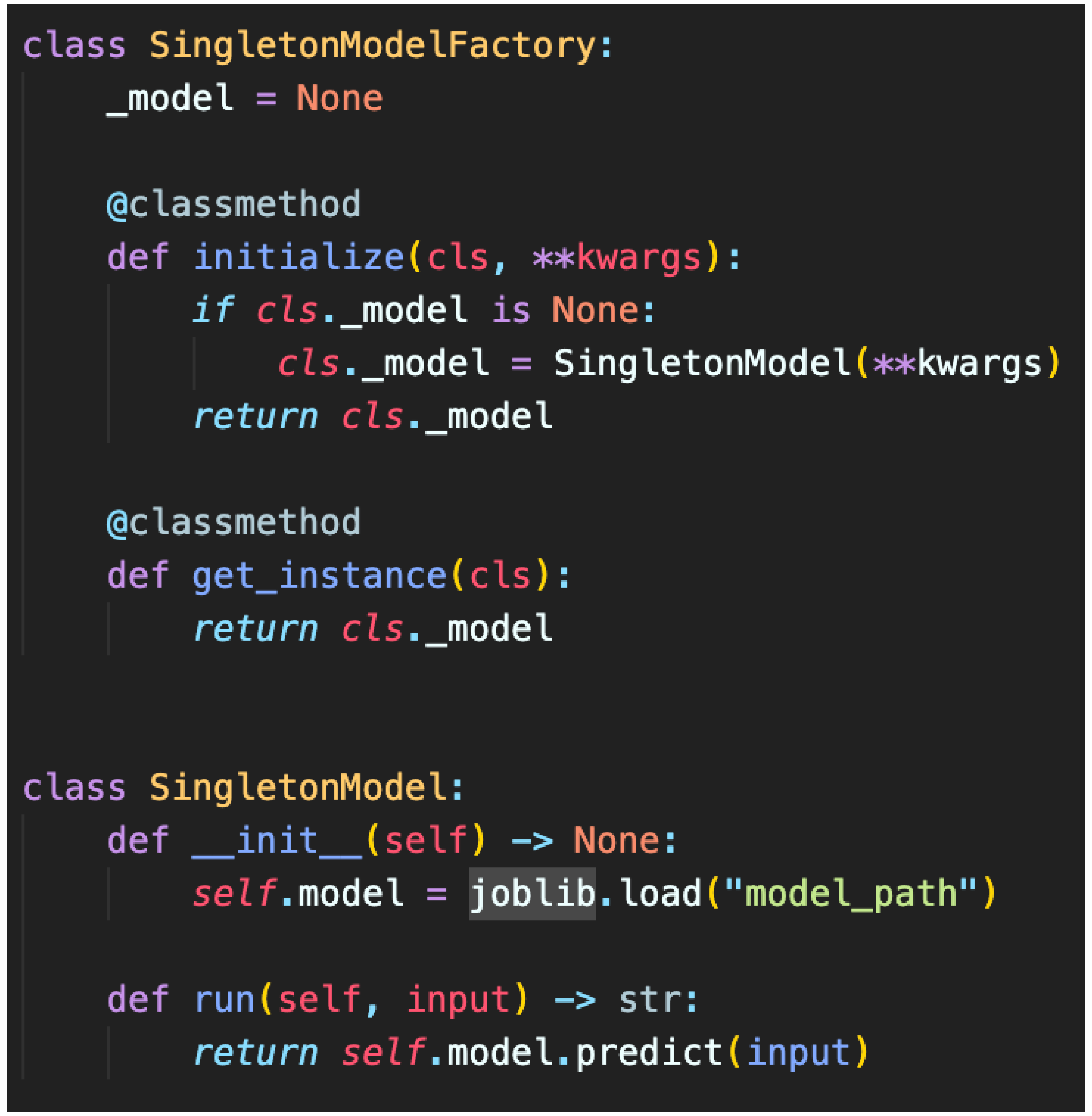


```python
def predict(input_data):
    model = joblib.load("model_path")  # 모델 로드
    prediction = model.predict(input_data)  # 예측 수행
    return prediction
```



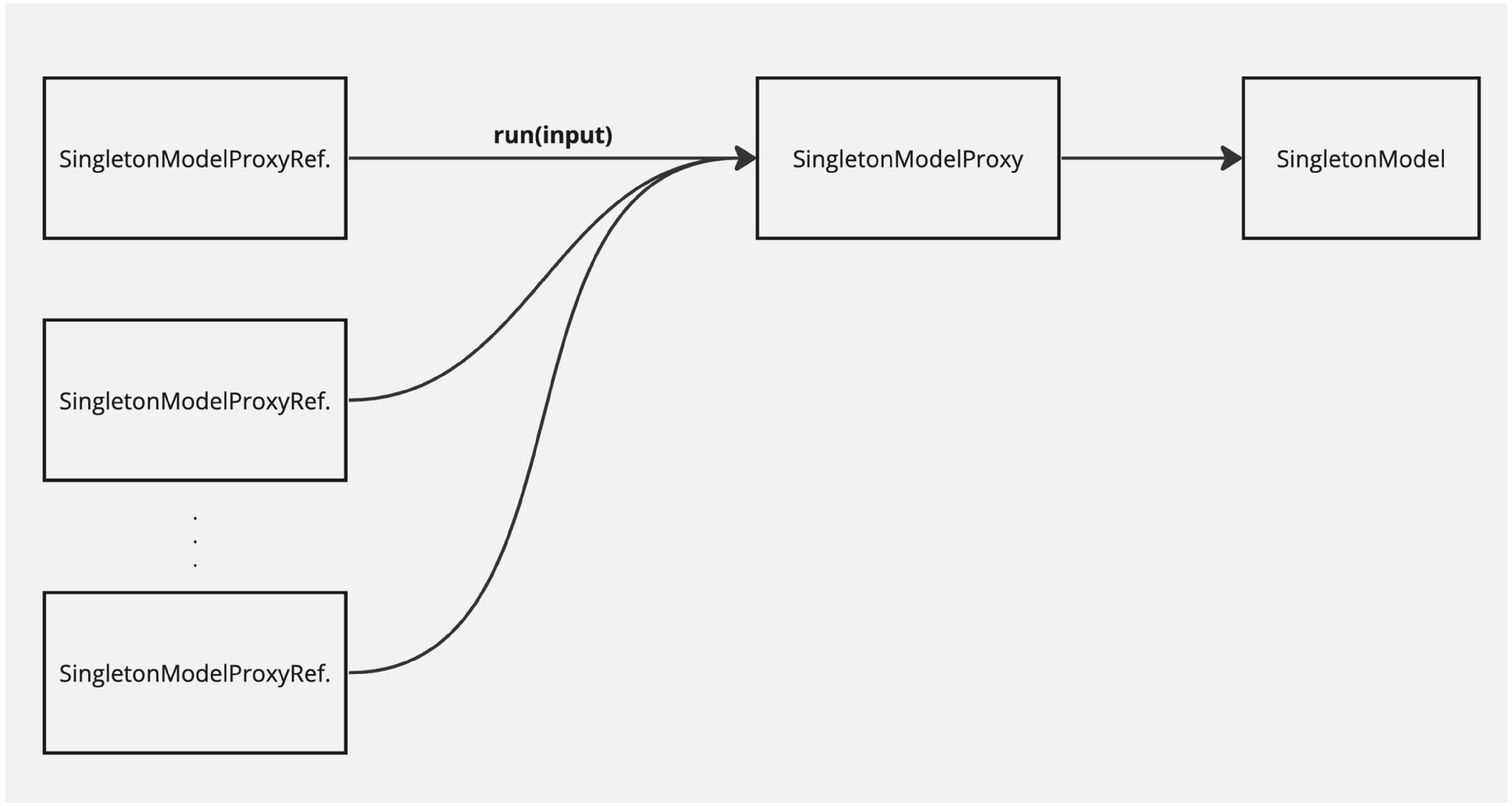
```python
model = joblib.load("model_path")  # 애플리케이션 시작 시 모델 로드

def predict(input_data):
    global model
    prediction = model.predict(input_data)  # 재사용된 모델로 예측 수행
    return prediction
```

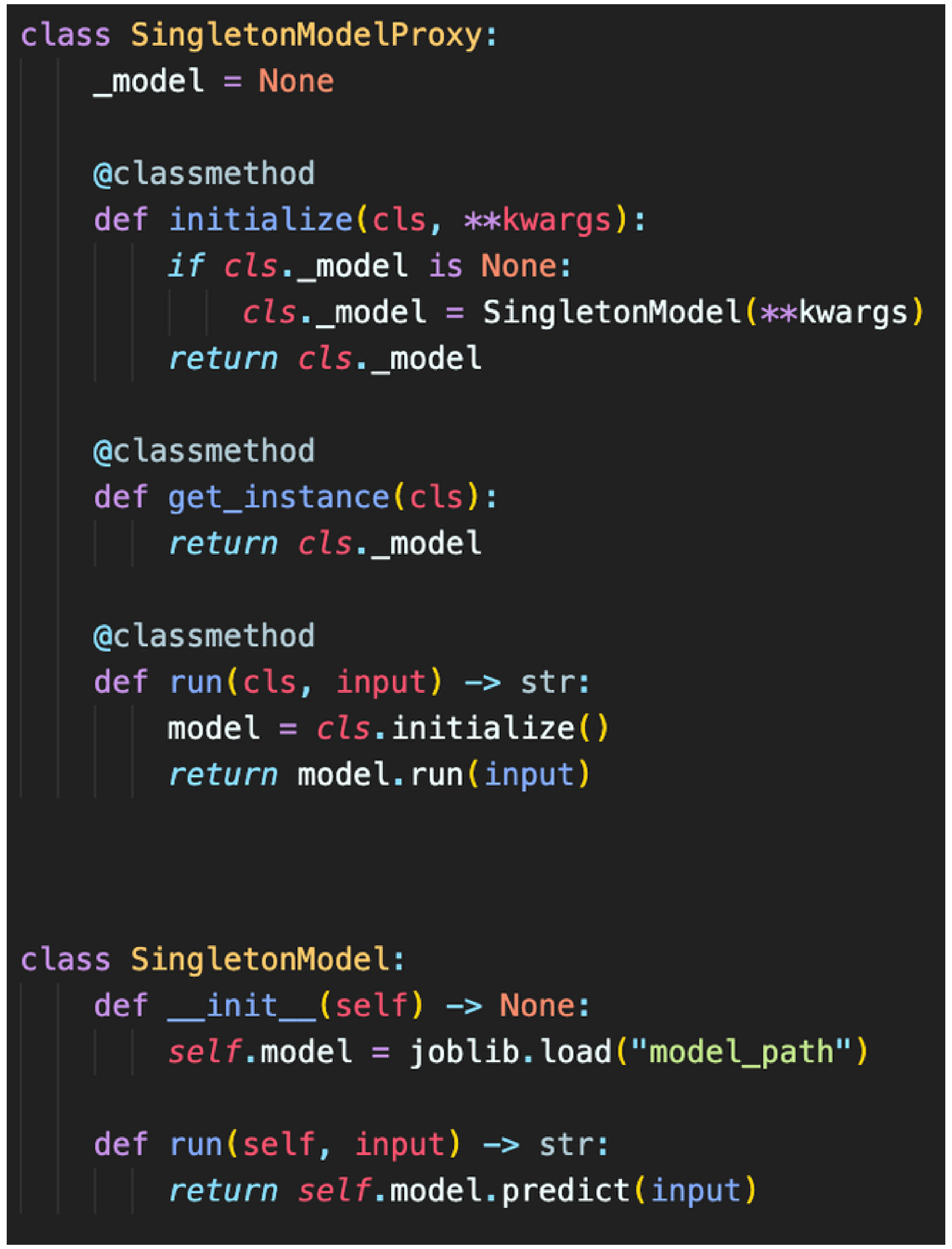




```python
class SingletonModelFactory:
    _model = None

    @classmethod
    def initialize(cls, **kwargs):
        if cls._model is None:
            cls._model = SingletonModel(**kwargs)
        return cls._model

    @classmethod
    def get_instance(cls):
        return cls._model


class SingletonModel:
    def __init__(self) -> None:
        self.model = joblib.load("model_path")

    def run(self, input) -> str:
        return self.model.predict(input)
```

Google Developer Student Clubs
Dongguk University



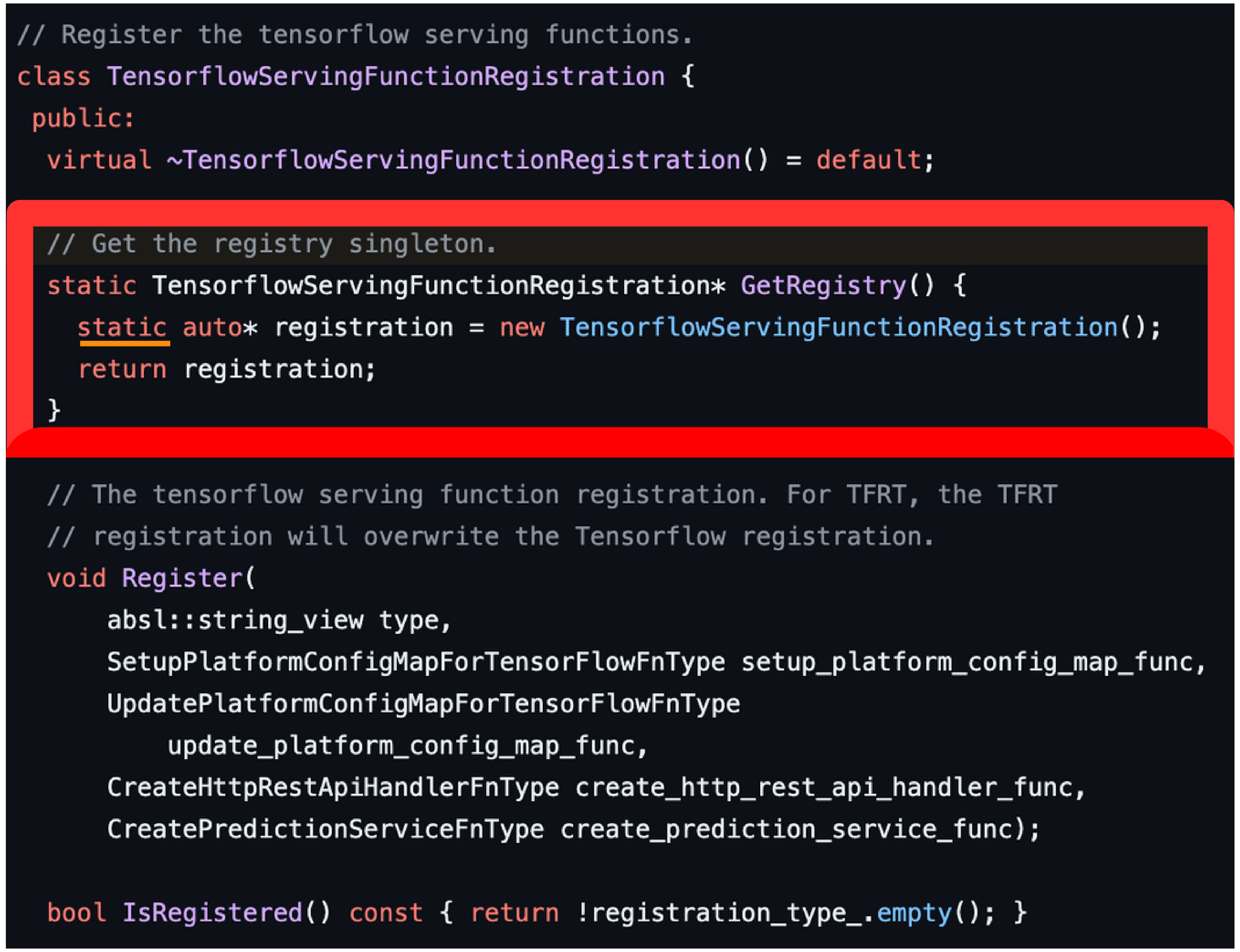


```python
class SingletonModelProxy:
    _model = None

    @classmethod
    def initialize(cls, **kwargs):
        if cls._model is None:
            cls._model = SingletonModel(**kwargs)
        return cls._model

    @classmethod
    def get_instance(cls):
        return cls._model

    @classmethod
    def run(cls, input) -> str:
        model = cls.initialize()
        return model.run(input)


class SingletonModel:
    def __init__(self) -> None:
        self.model = joblib.load("model_path")

    def run(self, input) -> str:
        return self.model.predict(input)
```

TensorFlow

/serving

tensorflow_serving/model_servers/server_init.h



```
// Register the tensorflow serving functions.
class TensorflowServingFunctionRegistration {
 public:
  virtual ~TensorflowServingFunctionRegistration() = default;

  // Get the registry singleton.
  static TensorflowServingFunctionRegistration* GetRegistry() {
    static auto* registration = new TensorflowServingFunctionRegistration();
    return registration;
  }

  // The tensorflow serving function registration. For TFRT, the TFRT
  // registration will overwrite the Tensorflow registration.
  void Register(
      absl::string_view type,
      SetupPlatformConfigMapForTensorFlowFnType setup_platform_config_map_func,
      UpdatePlatformConfigMapForTensorFlowFnType
          update_platform_config_map_func,
      CreateHttpRestApiHandlerFnType create_http_rest_api_handler_func,
      CreatePredictionServiceFnType create_prediction_service_func);

  bool IsRegistered() const { return !registration_type_.empty(); }
```

- **Registration**

  텐서플로우 서빙에 필요한 기능의 구현체를 싱글톤으로 등록

- **GetRegistry**

  싱글톤 객체에 대한 참조형을 반환

감사합니다