소프트웨어연구부문
Software
Research Laboratory

Libsalt 설계 및 구현

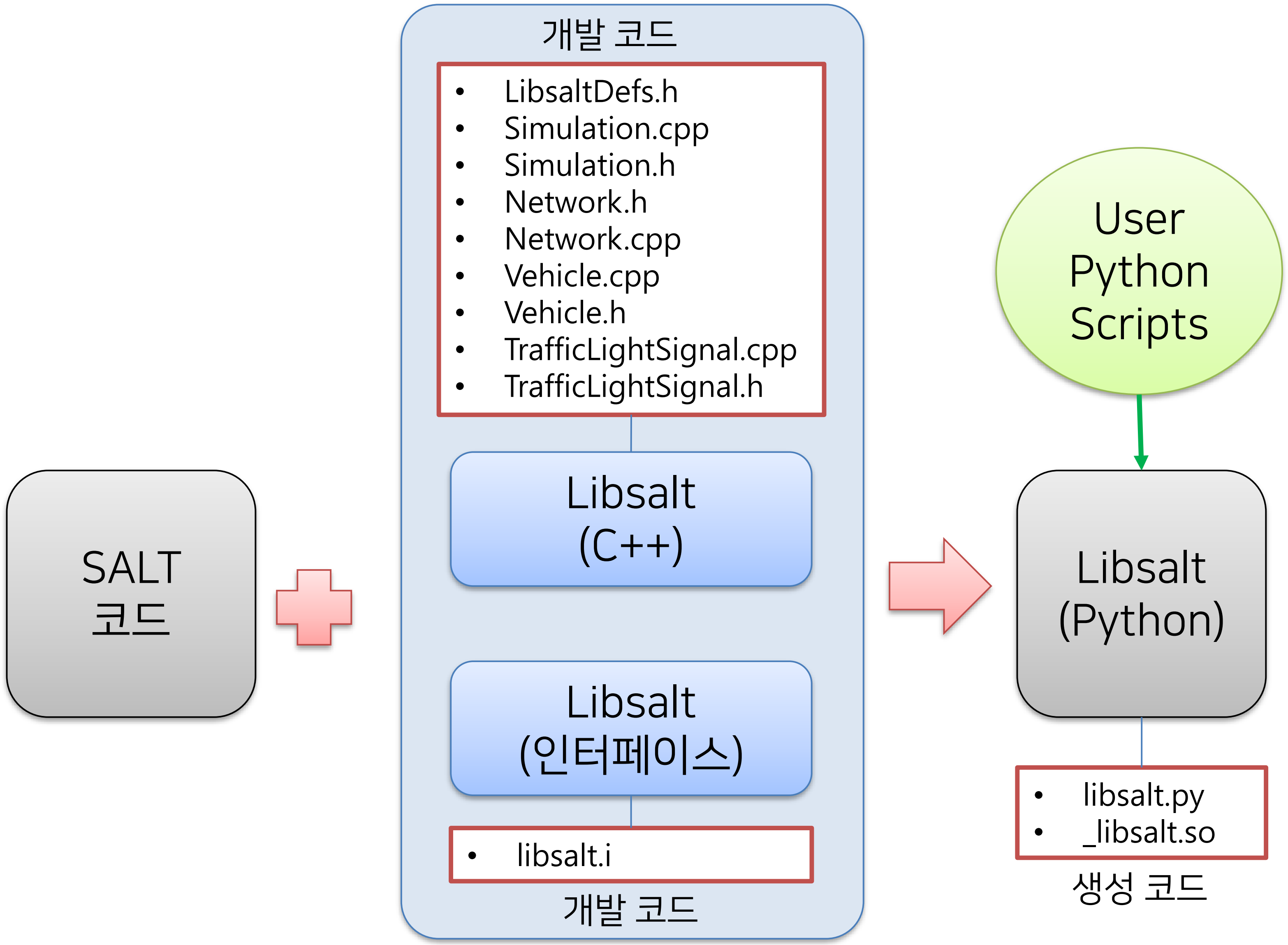
- Simulation, Network, Vehicle -

이명철
(스마트데이터연구실)

ETRI 한국전자통신연구원
www.etri.re.kr

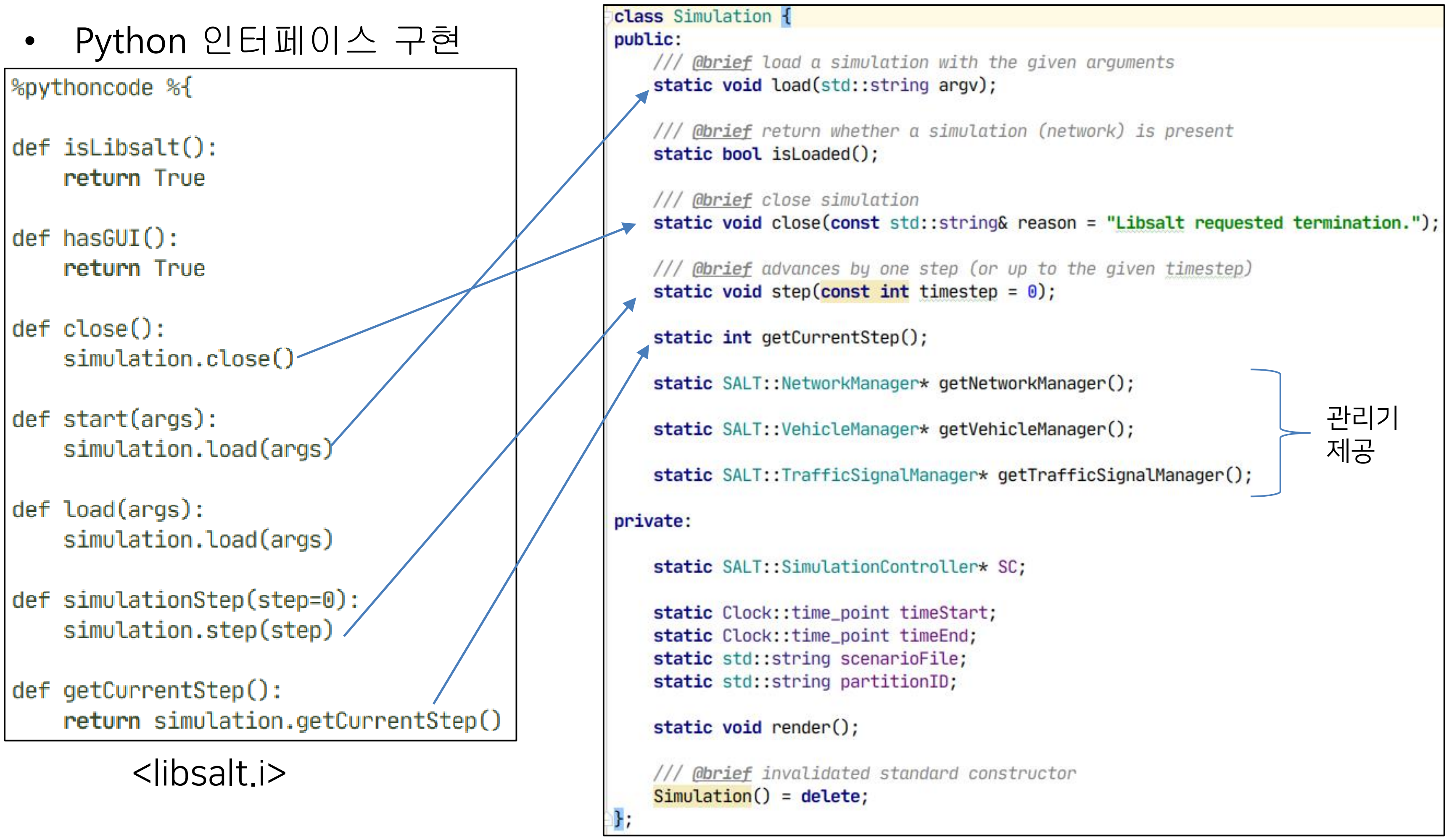
2020.08.25(화)

Libsalt 설계 및 구현 결과



Simulation 인터페이스 (C++/Python)

- 시뮬레이션 제어 인터페이스 제공
  - Python 인터페이스 구현
  - C++ 인터페이스 구현



```
%pythoncode %{

def isLibsalt():
    return True

def hasGUI():
    return True

def close():
    simulation.close()

def start(args):
    simulation.load(args)

def load(args):
    simulation.load(args)

def simulationStep(step=0):
    simulation.step(step)

def getCurrentStep():
    return simulation.getCurrentStep()
```

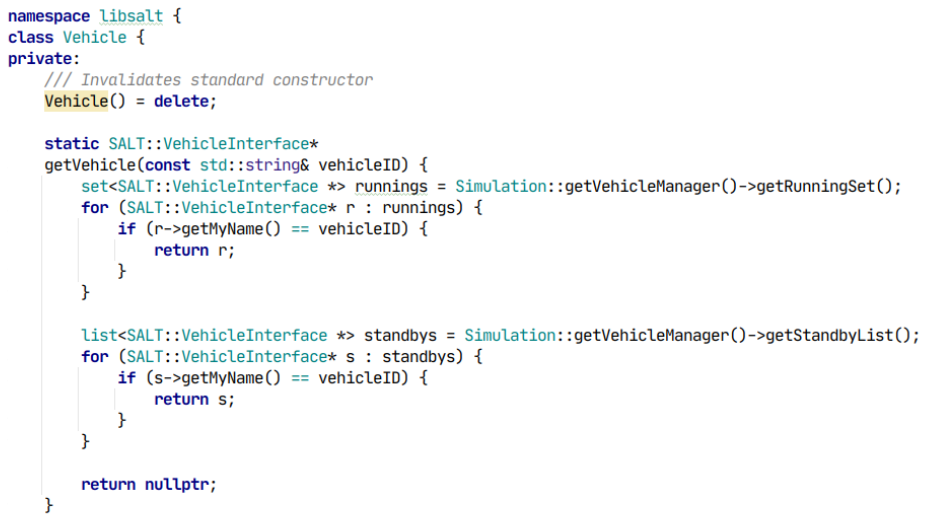
<libsalt.i>

```
class Simulation {
public:
    /// @brief load a simulation with the given arguments
    static void load(std::string argv);

    /// @brief return whether a simulation (network) is present
    static bool isLoaded();

    /// @brief close simulation
    static void close(const std::string& reason = "Libsalt requested termination.");

    /// @brief advances by one step (or up to the given timestep)
    static void step(const int timestep = 0);

    static int getCurrentStep();

    static SALT::NetworkManager* getNetworkManager();

    static SALT::VehicleManager* getVehicleManager();

    static SALT::TrafficSignalManager* getTrafficSignalManager();

private:

    static SALT::SimulationController* SC;

    static Clock::time_point timeStart;
    static Clock::time_point timeEnd;
    static std::string scenarioFile;
    static std::string partitionID;

    static void render();

    /// @brief invalidated standard constructor
    Simulation() = delete;
};
```

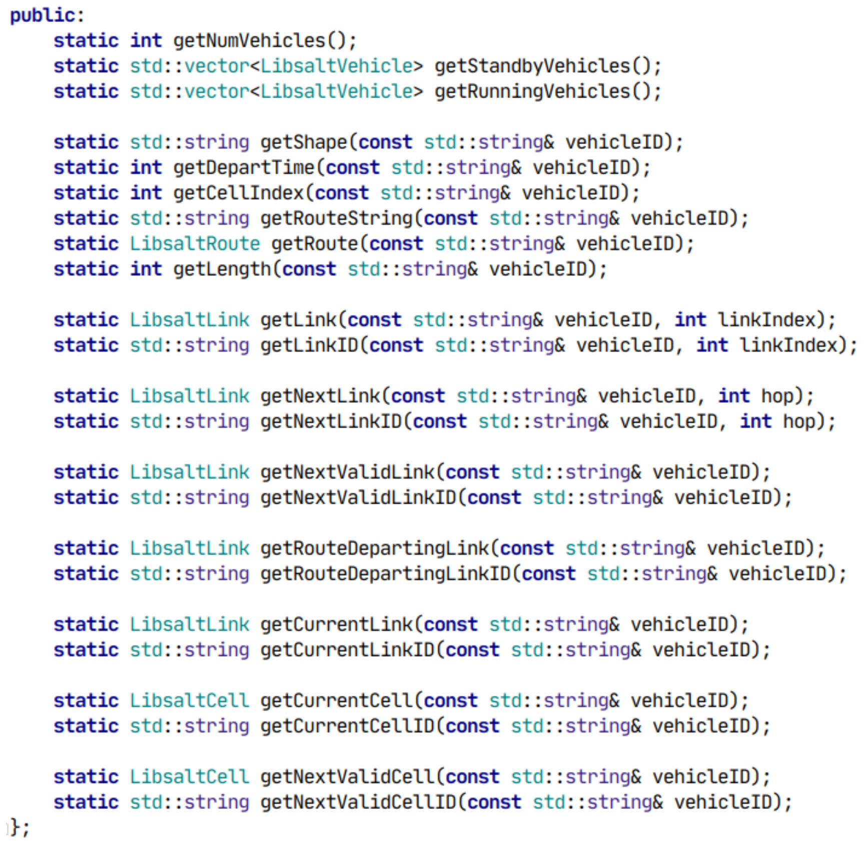
<Simulation.h>

Vehicle 인터페이스 (C++)

- 차량 정보 접근 인터페이스 제공



```cpp
namespace libsalt {
class Vehicle {
private:
    /// Invalidates standard constructor
    Vehicle() = delete;

    static SALT::VehicleInterface*
    getVehicle(const std::string& vehicleID) {
        set<SALT::VehicleInterface *> runnings = Simulation::getVehicleManager()->getRunningSet();
        for (SALT::VehicleInterface* r : runnings) {
            if (r->getMyName() == vehicleID) {
                return r;
            }
        }

        list<SALT::VehicleInterface *> standbys = Simulation::getVehicleManager()->getStandbyList();
        for (SALT::VehicleInterface* s : standbys) {
            if (s->getMyName() == vehicleID) {
                return s;
            }
        }

        return nullptr;
    }
```

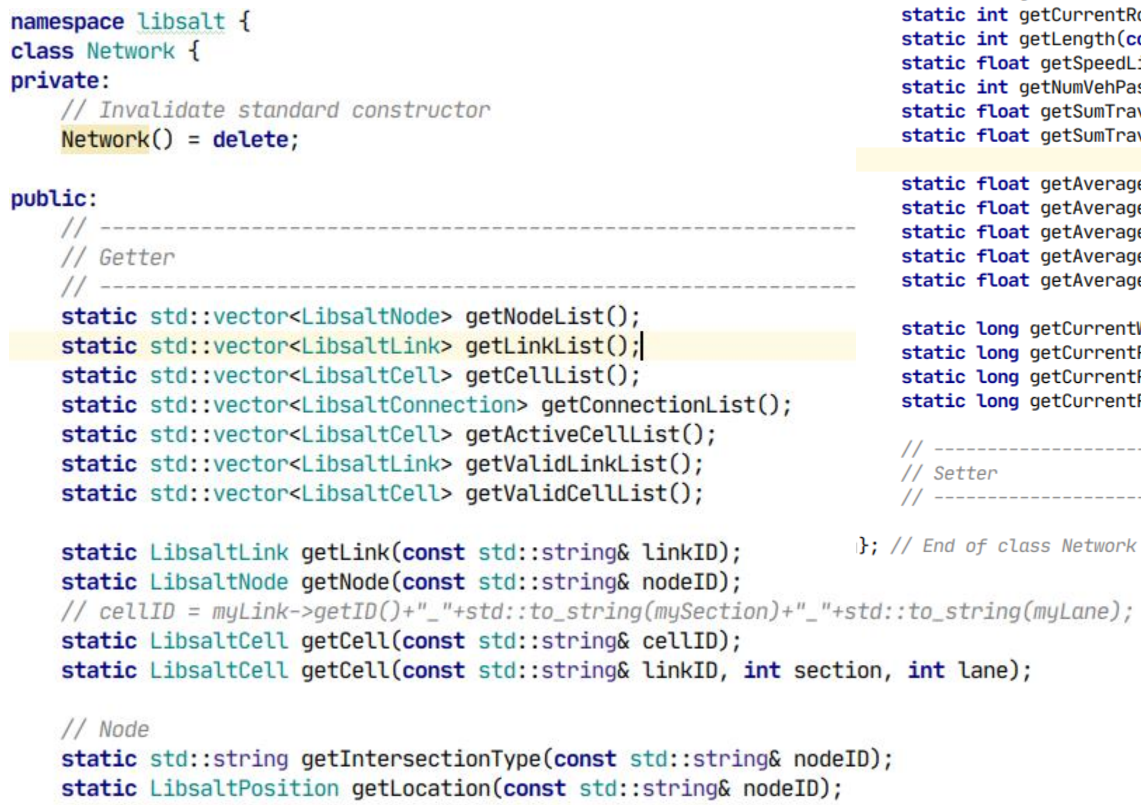


```cpp
public:
    static int getNumVehicles();
    static std::vector<LibsaltVehicle> getStandbyVehicles();
    static std::vector<LibsaltVehicle> getRunningVehicles();

    static std::string getShape(const std::string& vehicleID);
    static int getDepartTime(const std::string& vehicleID);
    static int getCellIndex(const std::string& vehicleID);
    static std::string getRouteString(const std::string& vehicleID);
    static LibsaltRoute getRoute(const std::string& vehicleID);
    static int getLength(const std::string& vehicleID);

    static LibsaltLink getLink(const std::string& vehicleID, int linkIndex);
    static std::string getLinkID(const std::string& vehicleID, int linkIndex);

    static LibsaltLink getNextLink(const std::string& vehicleID, int hop);
    static std::string getNextLinkID(const std::string& vehicleID, int hop);

    static LibsaltLink getNextValidLink(const std::string& vehicleID);
    static std::string getNextValidLinkID(const std::string& vehicleID);

    static LibsaltLink getRouteDepartingLink(const std::string& vehicleID);
    static std::string getRouteDepartingLinkID(const std::string& vehicleID);

    static LibsaltLink getCurrentLink(const std::string& vehicleID);
    static std::string getCurrentLinkID(const std::string& vehicleID);

    static LibsaltCell getCurrentCell(const std::string& vehicleID);
    static std::string getCurrentCellID(const std::string& vehicleID);

    static LibsaltCell getNextValidCell(const std::string& vehicleID);
    static std::string getNextValidCellID(const std::string& vehicleID);
};
```

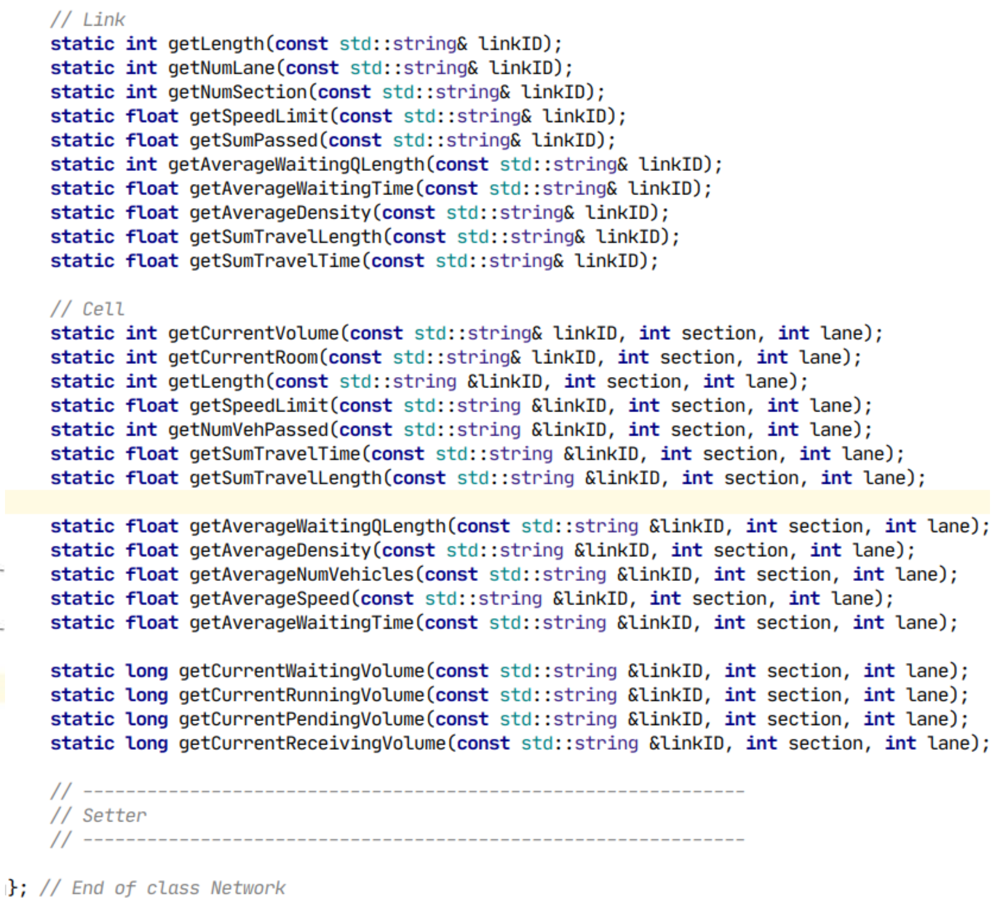
<Vehicle.h>

Network 인터페이스 (C++)

- 교차로, 도로, 셀, 연결 정보 접근 인터페이스 제공



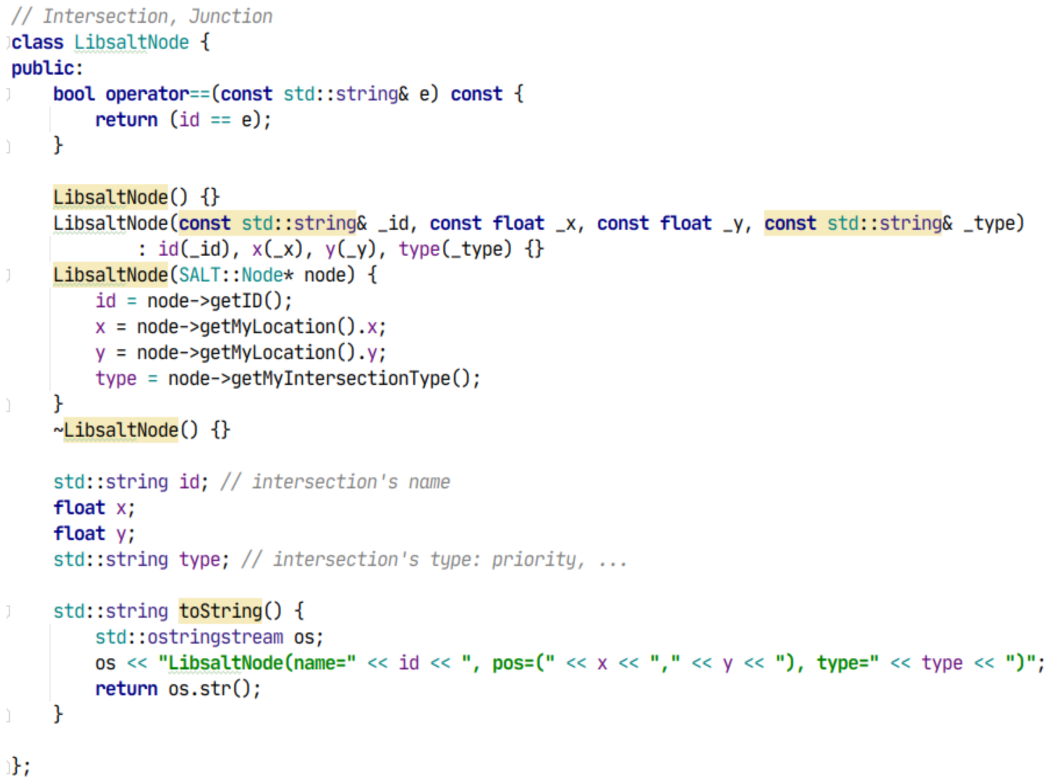
```cpp
namespace libsalt {
class Network {
private:
    // Invalidate standard constructor
    Network() = delete;

public:
    // ------------------------------------------------------------
    // Getter
    // ------------------------------------------------------------
    static std::vector<LibsaltNode> getNodeList();
    static std::vector<LibsaltLink> getLinkList();
    static std::vector<LibsaltCell> getCellList();
    static std::vector<LibsaltConnection> getConnectionList();
    static std::vector<LibsaltCell> getActiveCellList();
    static std::vector<LibsaltLink> getValidLinkList();
    static std::vector<LibsaltCell> getValidCellList();

    static LibsaltLink getLink(const std::string& linkID);
    static LibsaltNode getNode(const std::string& nodeID);
    // cellID = myLink->getID()+"_"+std::to_string(mySection)+"_"+std::to_string(myLane);
    static LibsaltCell getCell(const std::string& cellID);
    static LibsaltCell getCell(const std::string& linkID, int section, int lane);

    // Node
    static std::string getIntersectionType(const std::string& nodeID);
    static LibsaltPosition getLocation(const std::string& nodeID);
```



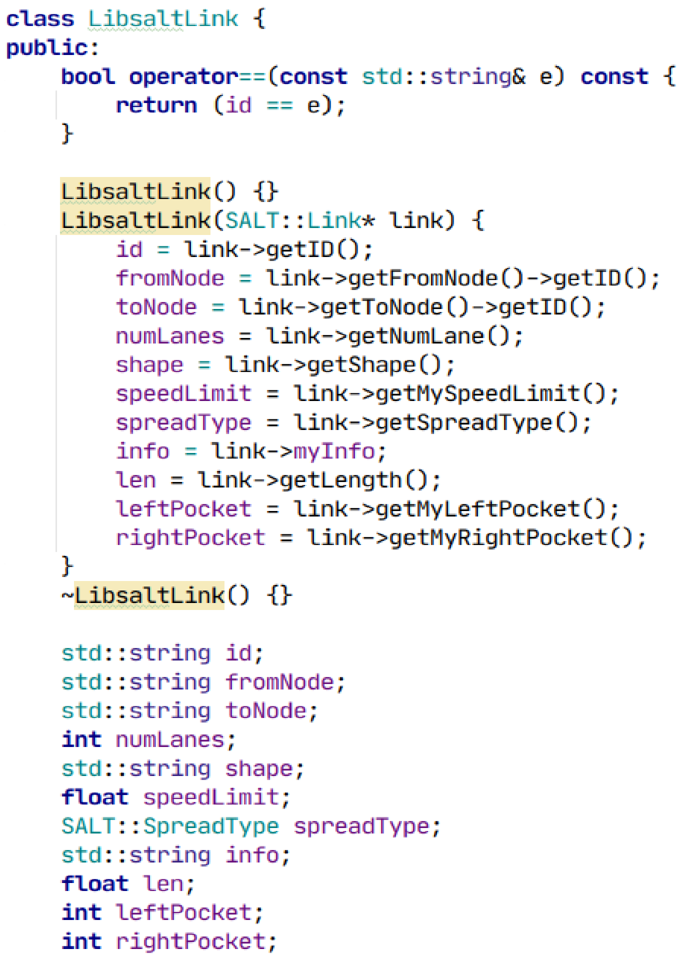
```cpp
    // Link
    static int getLength(const std::string& linkID);
    static int getNumLane(const std::string& linkID);
    static int getNumSection(const std::string& linkID);
    static float getSpeedLimit(const std::string& linkID);
    static float getSumPassed(const std::string& linkID);
    static int getAverageWaitingQLength(const std::string& linkID);
    static float getAverageWaitingTime(const std::string& linkID);
    static float getAverageDensity(const std::string& linkID);
    static float getSumTravelLength(const std::string& linkID);
    static float getSumTravelTime(const std::string& linkID);

    // Cell
    static int getCurrentVolume(const std::string& linkID, int section, int lane);
    static int getCurrentRoom(const std::string& linkID, int section, int lane);
    static int getLength(const std::string &linkID, int section, int lane);
    static float getSpeedLimit(const std::string &linkID, int section, int lane);
    static int getNumVehPassed(const std::string &linkID, int section, int lane);
    static float getSumTravelTime(const std::string &linkID, int section, int lane);
    static float getSumTravelLength(const std::string &linkID, int section, int lane);

    static float getAverageWaitingQLength(const std::string &linkID, int section, int lane);
    static float getAverageDensity(const std::string &linkID, int section, int lane);
    static float getAverageNumVehicles(const std::string &linkID, int section, int lane);
    static float getAverageSpeed(const std::string &linkID, int section, int lane);
    static float getAverageWaitingTime(const std::string &linkID, int section, int lane);

    static long getCurrentWaitingVolume(const std::string &linkID, int section, int lane);
    static long getCurrentRunningVolume(const std::string &linkID, int section, int lane);
    static long getCurrentPendingVolume(const std::string &linkID, int section, int lane);
    static long getCurrentReceivingVolume(const std::string &linkID, int section, int lane);

    // ------------------------------------------------------------
    // Setter
    // ------------------------------------------------------------

}; // End of class Network
```

<Network.h>

Python 전달 자료 구조



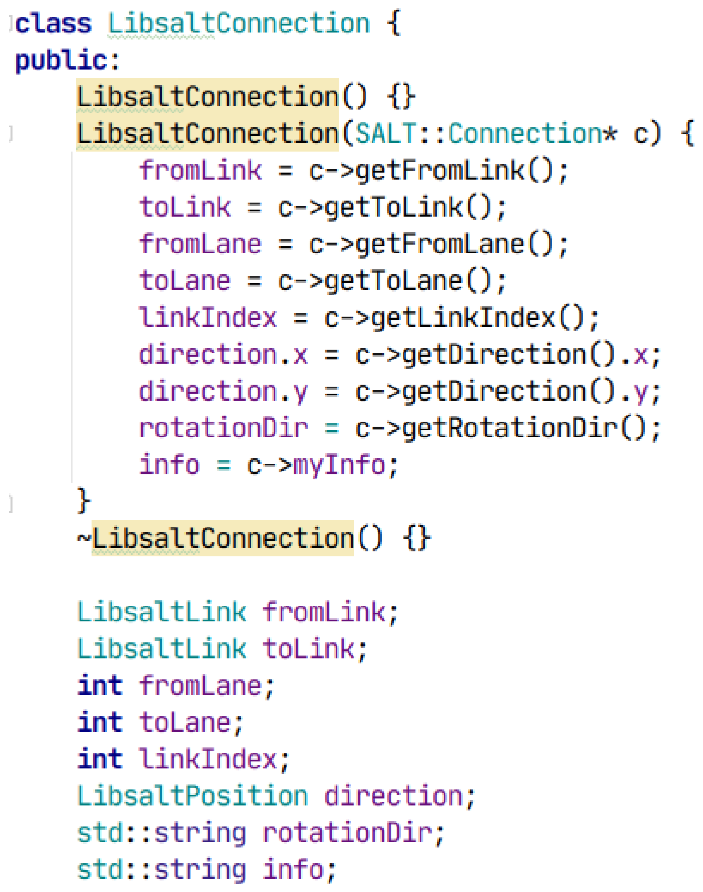
```cpp
// Intersection, Junction
class LibsaltNode {
public:
    bool operator==(const std::string& e) const {
        return (id == e);
    }

    LibsaltNode() {}
    LibsaltNode(const std::string& _id, const float _x, const float _y, const std::string& _type)
            : id(_id), x(_x), y(_y), type(_type) {}
    LibsaltNode(SALT::Node* node) {
        id = node->getID();
        x = node->getMyLocation().x;
        y = node->getMyLocation().y;
        type = node->getMyIntersectionType();
    }
    ~LibsaltNode() {}

    std::string id; // intersection's name
    float x;
    float y;
    std::string type; // intersection's type: priority, ...

    std::string toString() {
        std::ostringstream os;
        os << "LibsaltNode(name=" << id << ", pos=(" << x << "," << y << "), type=" << type << ")";
        return os.str();
    }

};
```



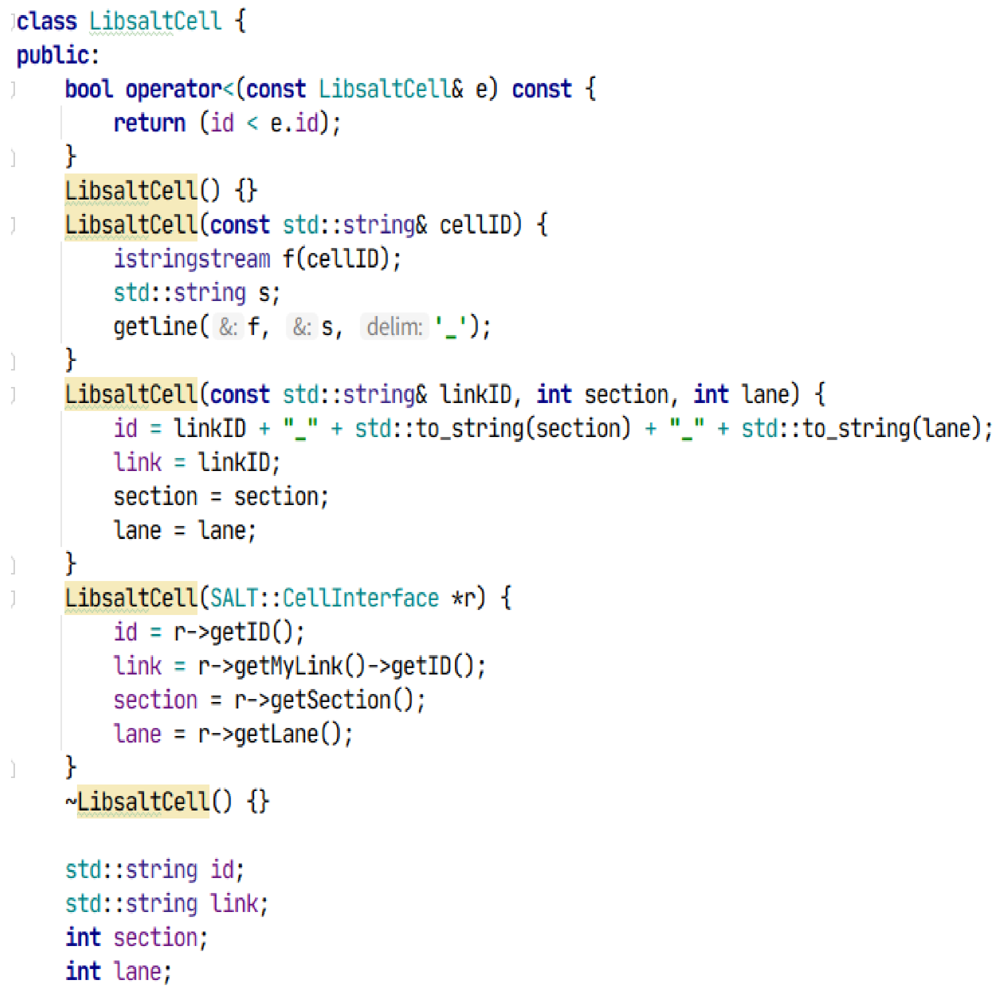
```cpp
class LibsaltLink {
public:
    bool operator==(const std::string& e) const {
        return (id == e);
    }

    LibsaltLink() {}
    LibsaltLink(SALT::Link* link) {
        id = link->getID();
        fromNode = link->getFromNode()->getID();
        toNode = link->getToNode()->getID();
        numLanes = link->getNumLane();
        shape = link->getShape();
        speedLimit = link->getMySpeedLimit();
        spreadType = link->getSpreadType();
        info = link->myInfo;
        len = link->getLength();
        leftPocket = link->getMyLeftPocket();
        rightPocket = link->getMyRightPocket();
    }
    ~LibsaltLink() {}

    std::string id;
    std::string fromNode;
    std::string toNode;
    int numLanes;
    std::string shape;
    float speedLimit;
    SALT::SpreadType spreadType;
    std::string info;
    float len;
    int leftPocket;
    int rightPocket;
```

<LibsaltDefs.h>

Python 전달 자료 구조 정의



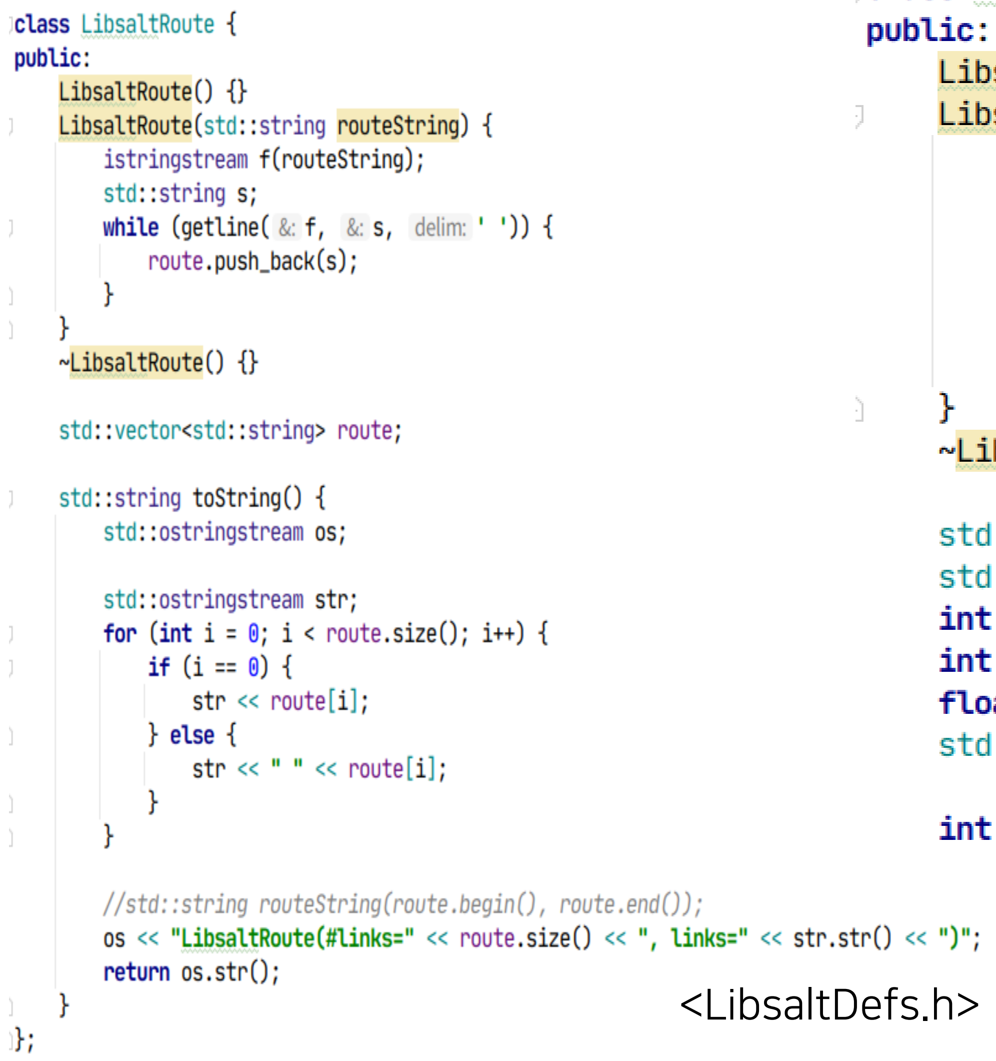
```cpp
class LibsaltConnection {
public:
    LibsaltConnection() {}
    LibsaltConnection(SALT::Connection* c) {
        fromLink = c->getFromLink();
        toLink = c->getToLink();
        fromLane = c->getFromLane();
        toLane = c->getToLane();
        linkIndex = c->getLinkIndex();
        direction.x = c->getDirection().x;
        direction.y = c->getDirection().y;
        rotationDir = c->getRotationDir();
        info = c->myInfo;
    }
    ~LibsaltConnection() {}

    LibsaltLink fromLink;
    LibsaltLink toLink;
    int fromLane;
    int toLane;
    int linkIndex;
    LibsaltPosition direction;
    std::string rotationDir;
    std::string info;
```



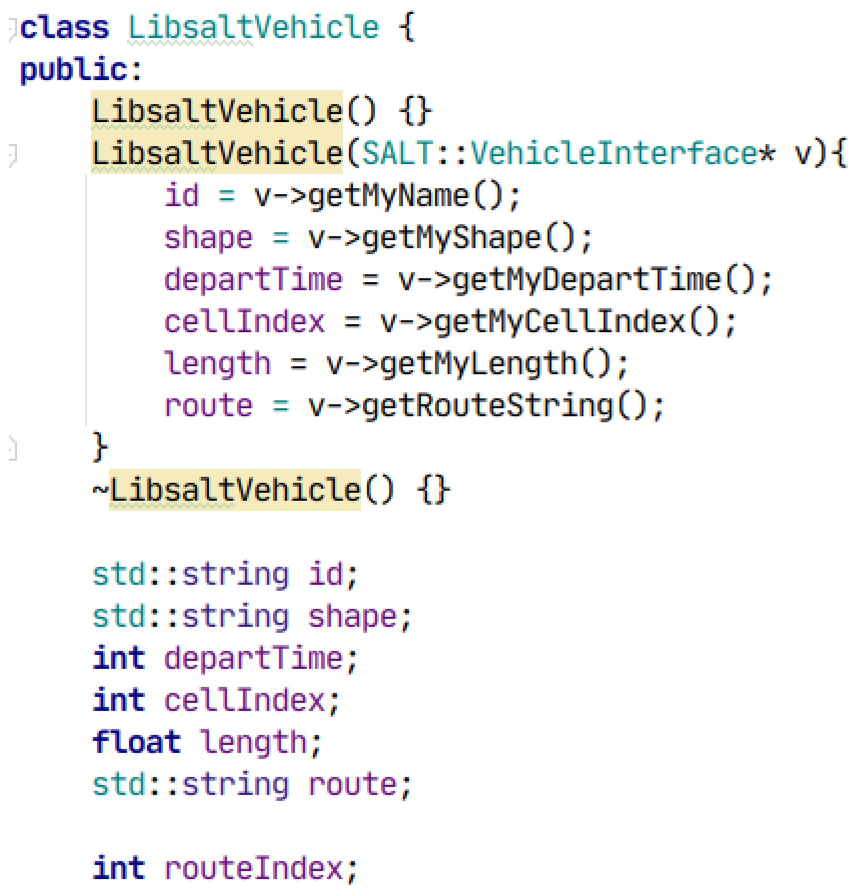
```cpp
class LibsaltCell {
public:
    bool operator<(const LibsaltCell& e) const {
        return (id < e.id);
    }
    LibsaltCell() {}
    LibsaltCell(const std::string& cellID) {
        istringstream f(cellID);
        std::string s;
        getline( &: f,  &: s,  delim: '_');
    }
    LibsaltCell(const std::string& linkID, int section, int lane) {
        id = linkID + "_" + std::to_string(section) + "_" + std::to_string(lane);
        link = linkID;
        section = section;
        lane = lane;
    }
    LibsaltCell(SALT::CellInterface *r) {
        id = r->getID();
        link = r->getMyLink()->getID();
        section = r->getSection();
        lane = r->getLane();
    }
    ~LibsaltCell() {}

    std::string id;
    std::string link;
    int section;
    int lane;
```

<LibsaltDefs.h>

Python 전달 자료 구조 정의



```cpp
class LibsaltRoute {
public:
    LibsaltRoute() {}
    LibsaltRoute(std::string routeString) {
        istringstream f(routeString);
        std::string s;
        while (getline( &: f,  &: s,  delim: ' ')) {
            route.push_back(s);
        }
    }
    ~LibsaltRoute() {}

    std::vector<std::string> route;

    std::string toString() {
        std::ostringstream os;

        std::ostringstream str;
        for (int i = 0; i < route.size(); i++) {
            if (i == 0) {
                str << route[i];
            } else {
                str << " " << route[i];
            }
        }

        //std::string routeString(route.begin(), route.end());
        os << "LibsaltRoute(#links=" << route.size() << ", links=" << str.str() << ")";
        return os.str();
    }
};
```

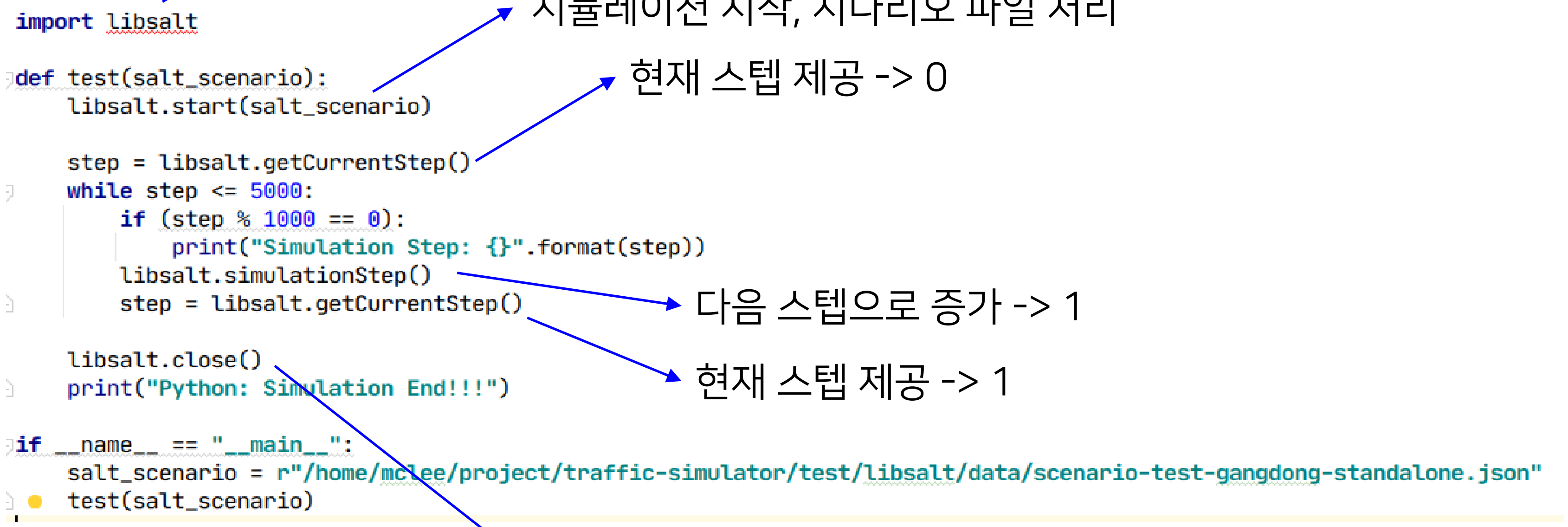


```cpp
class LibsaltVehicle {
public:
    LibsaltVehicle() {}
    LibsaltVehicle(SALT::VehicleInterface* v){
        id = v->getMyName();
        shape = v->getMyShape();
        departTime = v->getMyDepartTime();
        cellIndex = v->getMyCellIndex();
        length = v->getMyLength();
        route = v->getRouteString();
    }
    ~LibsaltVehicle() {}

    std::string id;
    std::string shape;
    int departTime;
    int cellIndex;
    float length;
    std::string route;

    int routeIndex;
```

<LibsaltDefs.h>

테스트 코드 - Simulation

- test_simulation.py



```python
import libsalt

def test(salt_scenario):
    libsalt.start(salt_scenario)

    step = libsalt.getCurrentStep()
    while step <= 5000:
        if (step % 1000 == 0):
            print("Simulation Step: {}".format(step))
        libsalt.simulationStep()
        step = libsalt.getCurrentStep()

    libsalt.close()
    print("Python: Simulation End!!!")

if __name__ == "__main__":
    salt_scenario = r"/home/mclee/project/traffic-simulator/test/libsalt/data/scenario-test-gangdong-standalone.json"
    test(salt_scenario)
```

libsalt 라이브러리 임포트

시뮬레이션 시작, 시나리오 파일 처리

현재 스텝 제공 -> 0

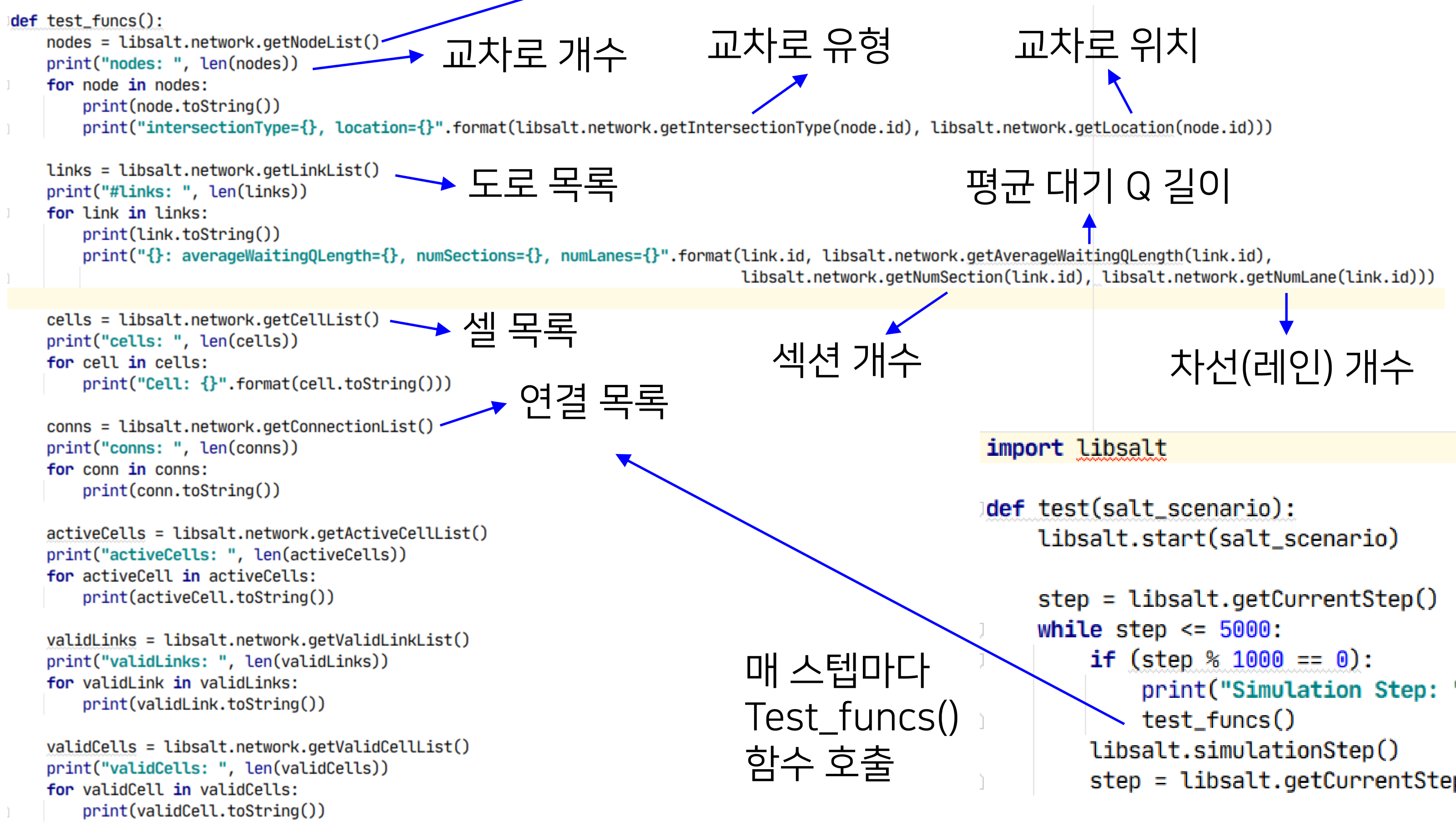
다음 스텝으로 증가 -> 1

현재 스텝 제공 -> 1

시뮬레이션 종료

테스트 코드 - Network

- test_network.py



```
def test_funcs():
    nodes = libsalt.network.getNodeList()
    print("nodes: ", len(nodes))
    for node in nodes:
        print(node.toString())
        print("intersectionType={}, location={}".format(libsalt.network.getIntersectionType(node.id), libsalt.network.getLocation(node.id)))

    links = libsalt.network.getLinkList()
    print("#links: ", len(links))
    for link in links:
        print(link.toString())
        print("{}: averageWaitingQLength={}, numSections={}, numLanes={}".format(link.id, libsalt.network.getAverageWaitingQLength(link.id),
                                                                                 libsalt.network.getNumSection(link.id), libsalt.network.getNumLane(link.id)))

    cells = libsalt.network.getCellList()
    print("cells: ", len(cells))
    for cell in cells:
        print("Cell: {}".format(cell.toString()))

    conns = libsalt.network.getConnectionList()
    print("conns: ", len(conns))
    for conn in conns:
        print(conn.toString())

    activeCells = libsalt.network.getActiveCellList()
    print("activeCells: ", len(activeCells))
    for activeCell in activeCells:
        print(activeCell.toString())

    validLinks = libsalt.network.getValidLinkList()
    print("validLinks: ", len(validLinks))
    for validLink in validLinks:
        print(validLink.toString())

    validCells = libsalt.network.getValidCellList()
    print("validCells: ", len(validCells))
    for validCell in validCells:
        print(validCell.toString())
```

교차로 목록

교차로 개수

교차로 유형

교차로 위치

도로 목록

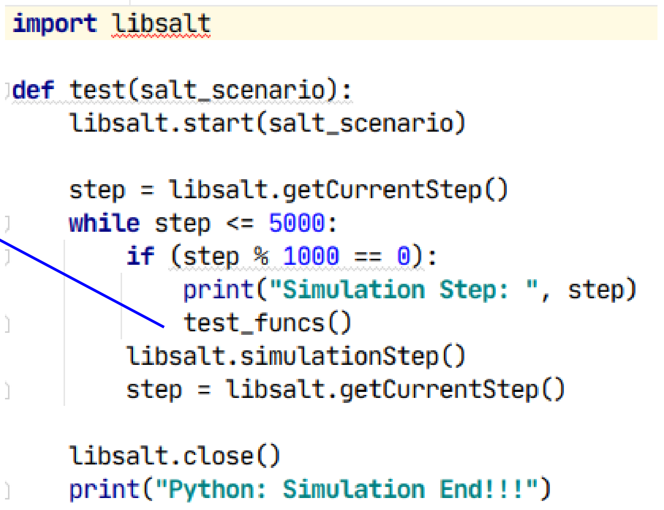
평균 대기 Q 길이

셀 목록

섹션 개수

차선(레인) 개수

연결 목록

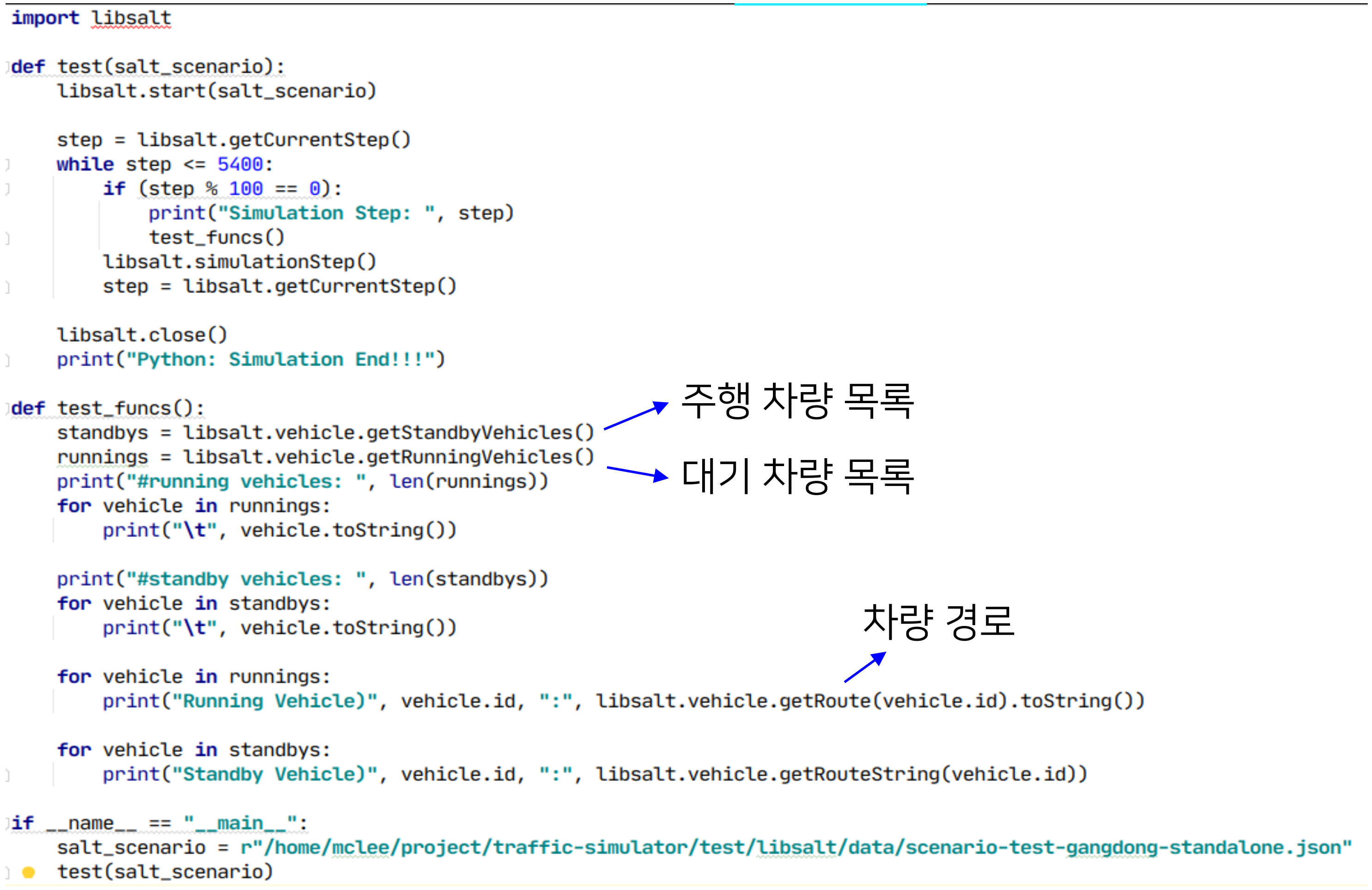


```
import libsalt

def test(salt_scenario):
    libsalt.start(salt_scenario)

    step = libsalt.getCurrentStep()
    while step <= 5000:
        if (step % 1000 == 0):
            print("Simulation Step: ", step)
            test_funcs()
        libsalt.simulationStep()
        step = libsalt.getCurrentStep()

    libsalt.close()
    print("Python: Simulation End!!!")
```

매 스텝마다
Test_funcs()
함수 호출

테스트 코드 - Vehicle

- test_vehicle.py



```python
import libsalt

def test(salt_scenario):
    libsalt.start(salt_scenario)

    step = libsalt.getCurrentStep()
    while step <= 5400:
        if (step % 100 == 0):
            print("Simulation Step: ", step)
            test_funcs()
        libsalt.simulationStep()
        step = libsalt.getCurrentStep()

    libsalt.close()
    print("Python: Simulation End!!!")

def test_funcs():
    standbys = libsalt.vehicle.getStandbyVehicles()
    runnings = libsalt.vehicle.getRunningVehicles()
    print("#running vehicles: ", len(runnings))
    for vehicle in runnings:
        print("\t", vehicle.toString())

    print("#standby vehicles: ", len(standbys))
    for vehicle in standbys:
        print("\t", vehicle.toString())

    for vehicle in runnings:
        print("Running Vehicle)", vehicle.id, ":", libsalt.vehicle.getRoute(vehicle.id).toString())

    for vehicle in standbys:
        print("Standby Vehicle)", vehicle.id, ":", libsalt.vehicle.getRouteString(vehicle.id))

if __name__ == "__main__":
    salt_scenario = r"/home/mclee/project/traffic-simulator/test/libsalt/data/scenario-test-gangdong-standalone.json"
    test(salt_scenario)
```

주행 차량 목록

대기 차량 목록

차량 경로

감사합니다