MATLAB EXPO

# Model-Based Design에서 Simulink와 Python 연동하기

신행재 부장, 매스웍스코리아

MathWorks

# Text vs Image

Which one is better to understand?

건물의 외부 형태는 가로 세로의 크기는 각각 6.36미터와 최대 11.73미터로 이루어진 직사각형 형태이다.
건물 안에는 총 4개의 룸으로 이루어져 있는데, 안방은 직사각 형태의 방 좌측 하단에 배치되어 있다. 우측 하단에는 거실이 배치되어 있고, 북쪽에 가운데 공간은 부엌이 있다. 부엌의 바로 좌측에 방1이 배치되어있고, 방1 아래에는 화장실이 2개 존재한다. 부엌의 오른편에는 배란다가 존재하고, 그 아래에 방2가 존재한다.

안방의 사이즈는 4.86mx4.25m형태의 직사각형 형태이다. 거실은 3.98mx5.02m이다. 부엌은 2.5mx4.8m이다. 방1의 사이즈는 4.59x3.1m 이다. 방2의 사이즈는 3.4x2.9m 이다. 벽의 두께는 12cm이다.

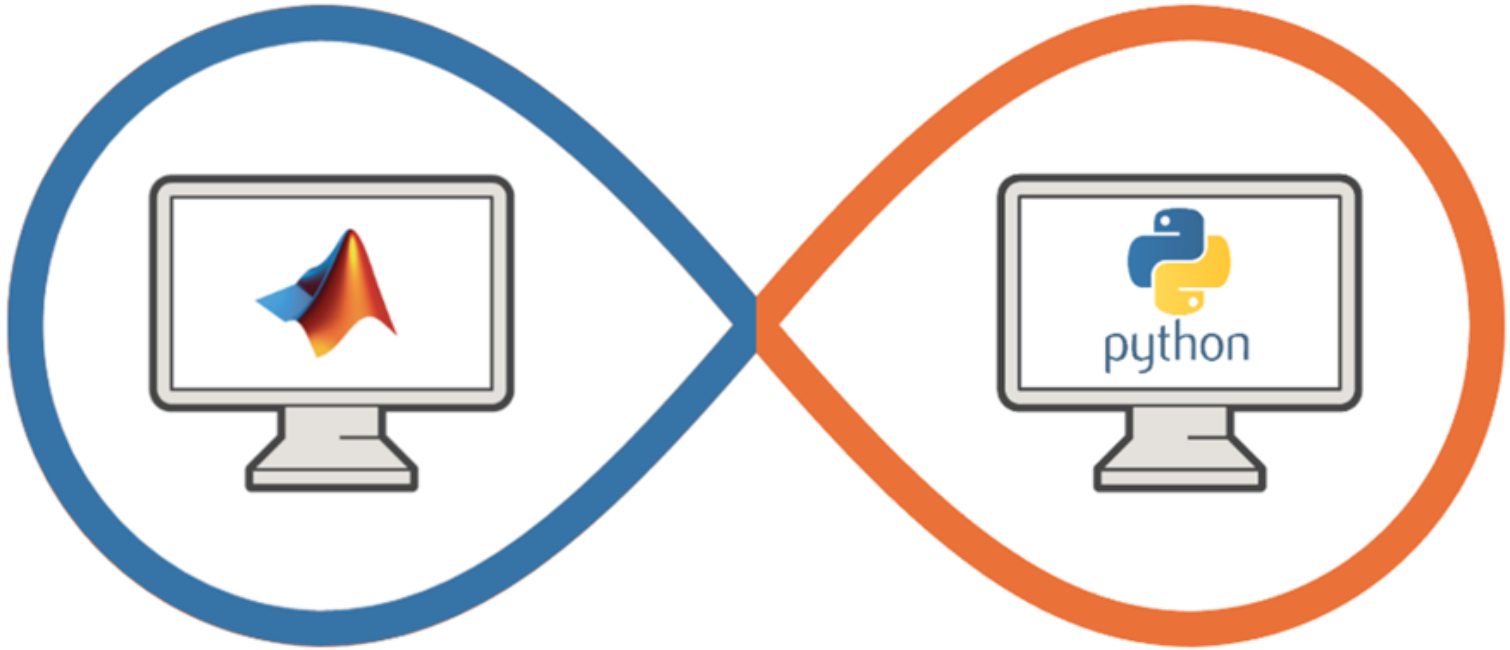
안방 가운데에는 싱글사이즈 침대가 2개 배치되는데, 남쪽으로 머리를 향하고 있으며 동쪽 벽면에 가깝게 배치가 되어있다. 안방의 화장실1앞에는 장롱이 배치되어있는데, 마치 복도처럼 배치가 되어 2개의 공간으로 파티션을 해주는 역할도 수행한다.
안방 방문은 안방 중심 기준으로 우측 위쪽 벽면에 존재해야 한다.

거실에는 공부를 위한 2개의 책상이 좌측 벽면에 배치가 된다. 2개의 책상 우측에는 공기청정기를 배치한다. 에어컨은 거실의 남동쪽 모서리에 배치가 되어야 한다.

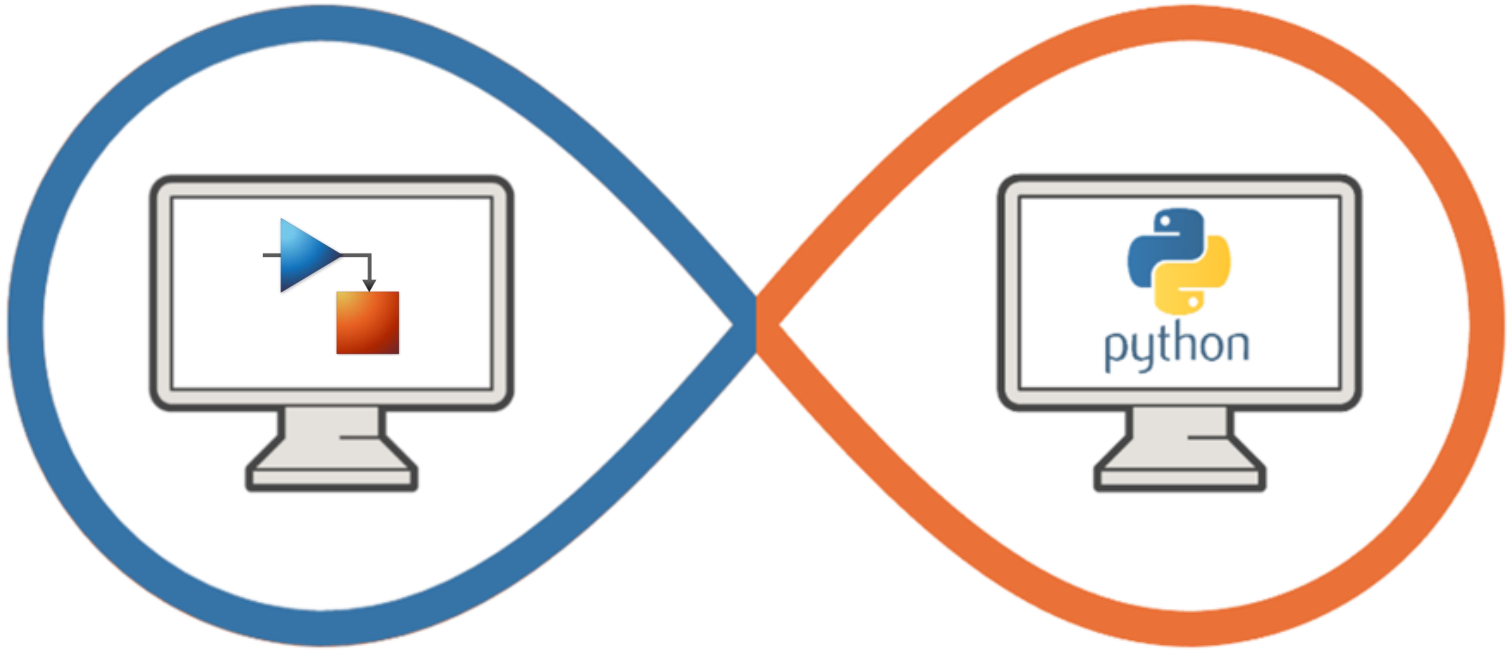
부엌은 북쪽 상단에 존재해야 하는데, 냉장고와 김치냉장고가 좌측하부에 배치되어야 한다.
부엌의 싱크대는 상단 가운데 벽면에 배치가 되어있다. 부엌의 우측의 문은 작은 베란다로 향하는데, 베란다에도 간단한 조리시설이 존재한다. 작은 베란다는 3.3x1.3m 사이즈의 공간인데, 세탁기와 건조기가 배치되어 있어야 한다.

방1에는 북쪽에서 1.3m 아래쪽 좌측 벽면에 싱글침대가 북쪽방향으로 배치가 되어있다. 공부를 위한 책상은 북동쪽 모서리에 배치되어 동쪽을 향하여 공부하도록 배치가 되어있고, 그 옆에 책장이 배치되어 있다. 책장 옆에는 작은 옷장을 책상과 나란히 배치하여 옷을 수납할 수 있도록 하였다.
북동쪽 모서리에 피아노를 배치하였고, 북쪽을 바라보면서 피아노를 칠 수 있다. 방1의 남쪽 벽면에는 별도의 수납공간을 배치하였다.
……
……

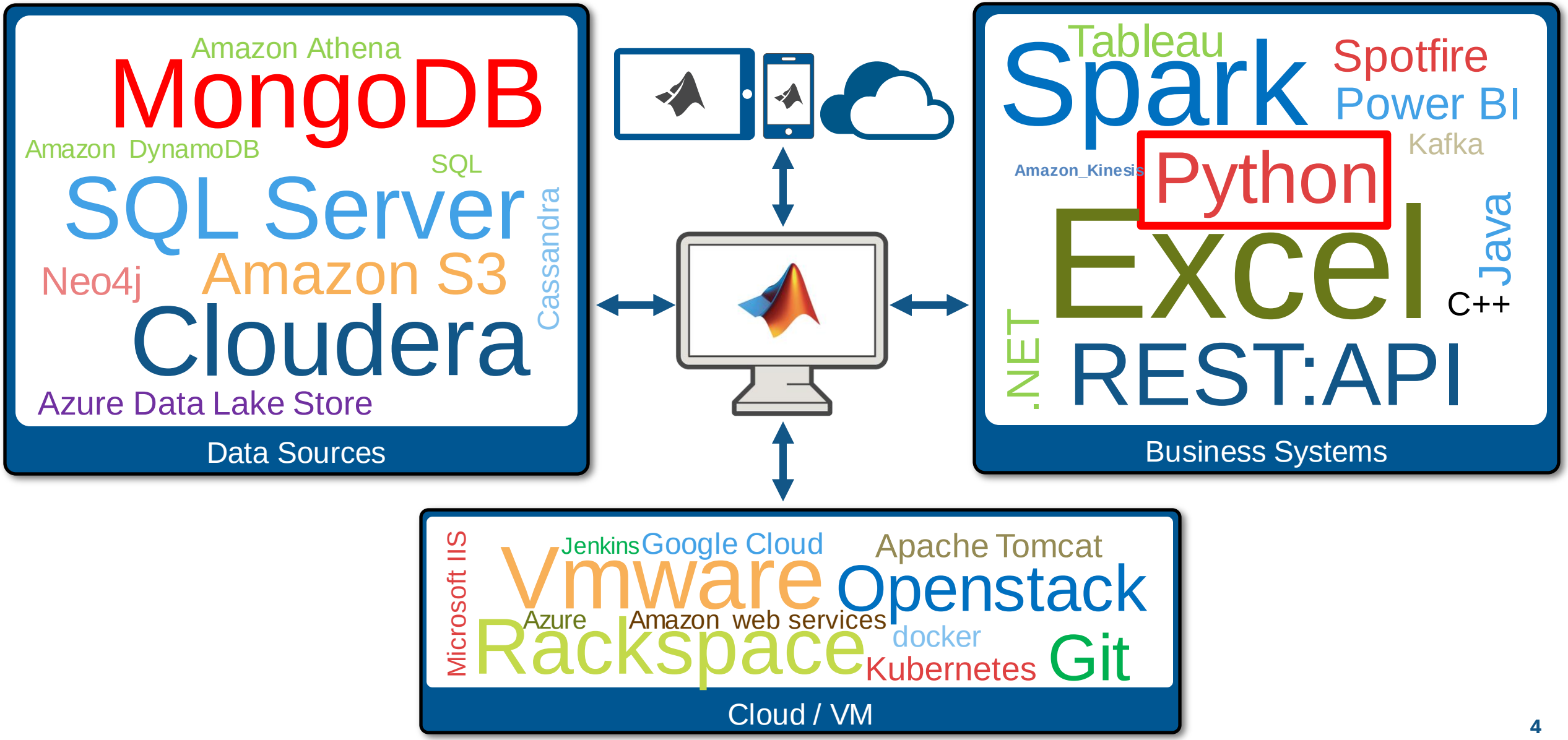
# You probably have heard a lot about using MATLAB and Python



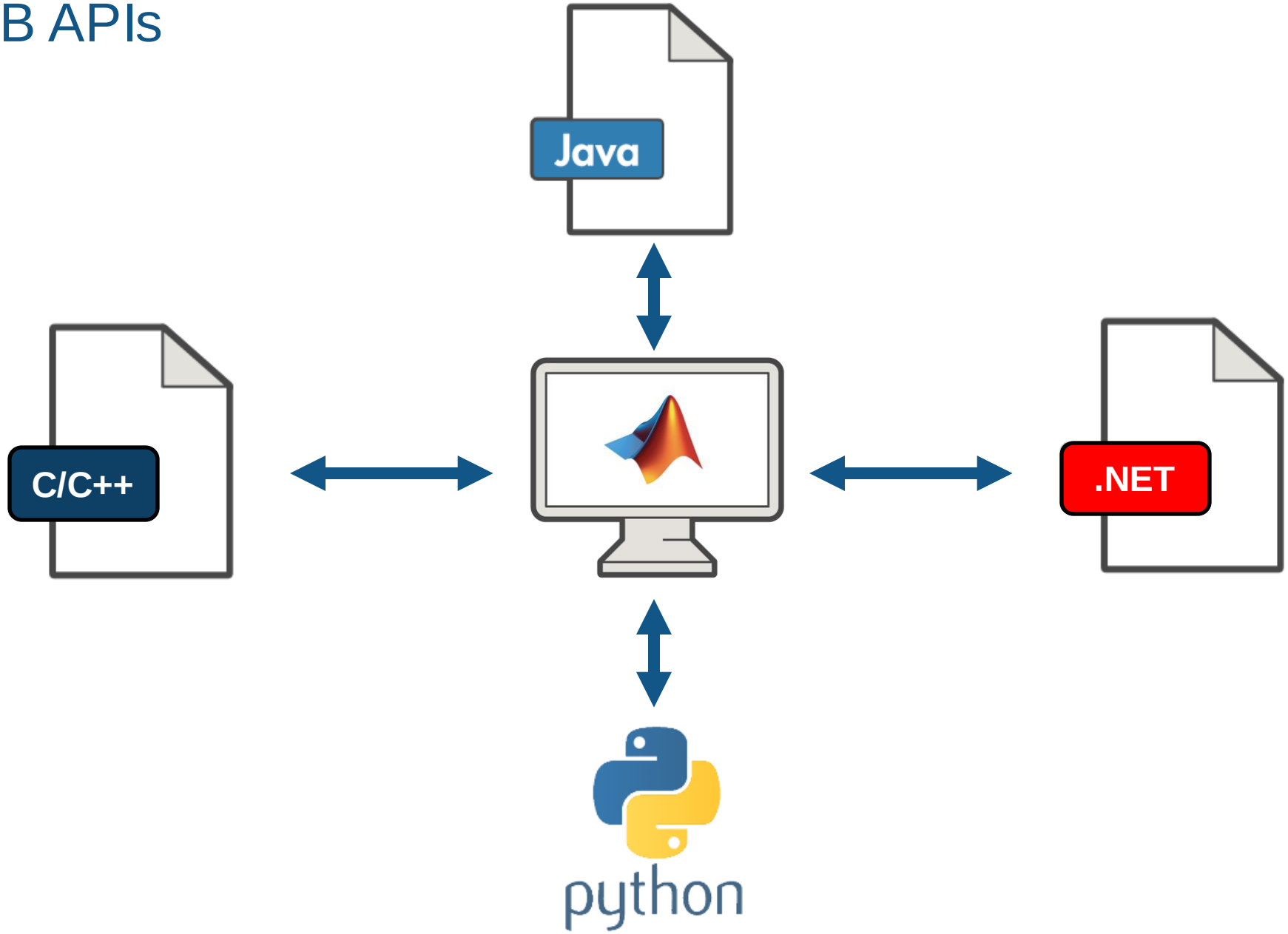
# But wondered what about using Python with Simulink?



# MATLAB and the Analytics Ecosystem

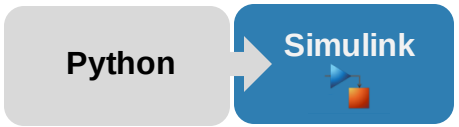


# MATLAB APIs



# Key takeaways

- Simulink as an open simulation platform supports versatile ways to interoperate with Python:
  - Bring Python code into Simulink as a library for co-execution




  - Integrate TensorFlow and PyTorch models for both simulation and code generation


  - Simulate a Simulink model directly from Python


  - Export a Simulink model as a Python package for deployment

# Why use Python and Simulink together?


- Need to **integrate** code from a colleague
- **Facilitate** development by enhancing **an AI workflow**
- Need **functionality** available in MATLAB and Simulink or in Python
- Leverage the work from the **community**

# The best way to use Simulink with Python is case specific

- Let's illustrate that through 4 typical scenarios

- In a team setting:
  - Jayden is a Simulink user and James is a Python user
  - James and Jayden need to work with each other to deliver a project

Jayden

Simulink user

James

Python user

# Scenario #1


James

나는 Python Algorithm 개발자입니다. Image Processing과 Computer Vision등의 Algorithm을 개발합니다.

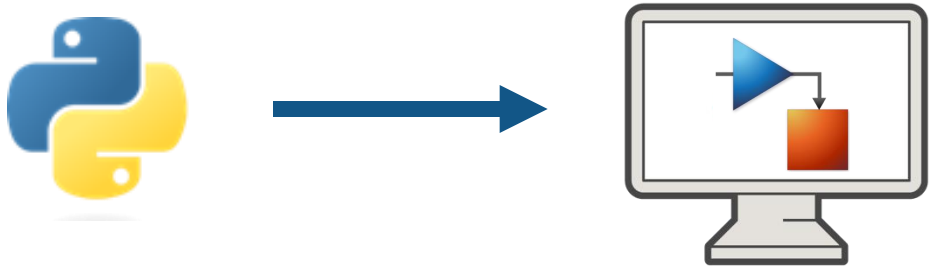
Jayden

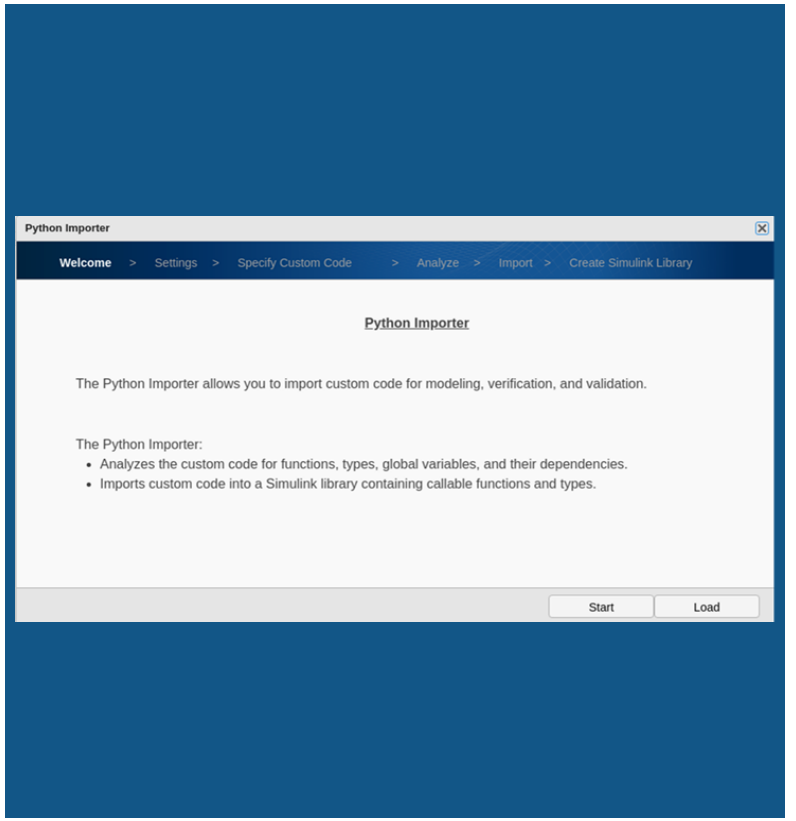
나는 다양한 Component들을 통합하는 System Engineer입니다. James가 만든 Algorithm을 포함하여 전체 System Level Simulation을 Simulink에서 진행해보고 싶습니다.


# Calling Python from Simulink

## - Use Python Importer

Use **Python Importer** to bring Python functions into Simulink





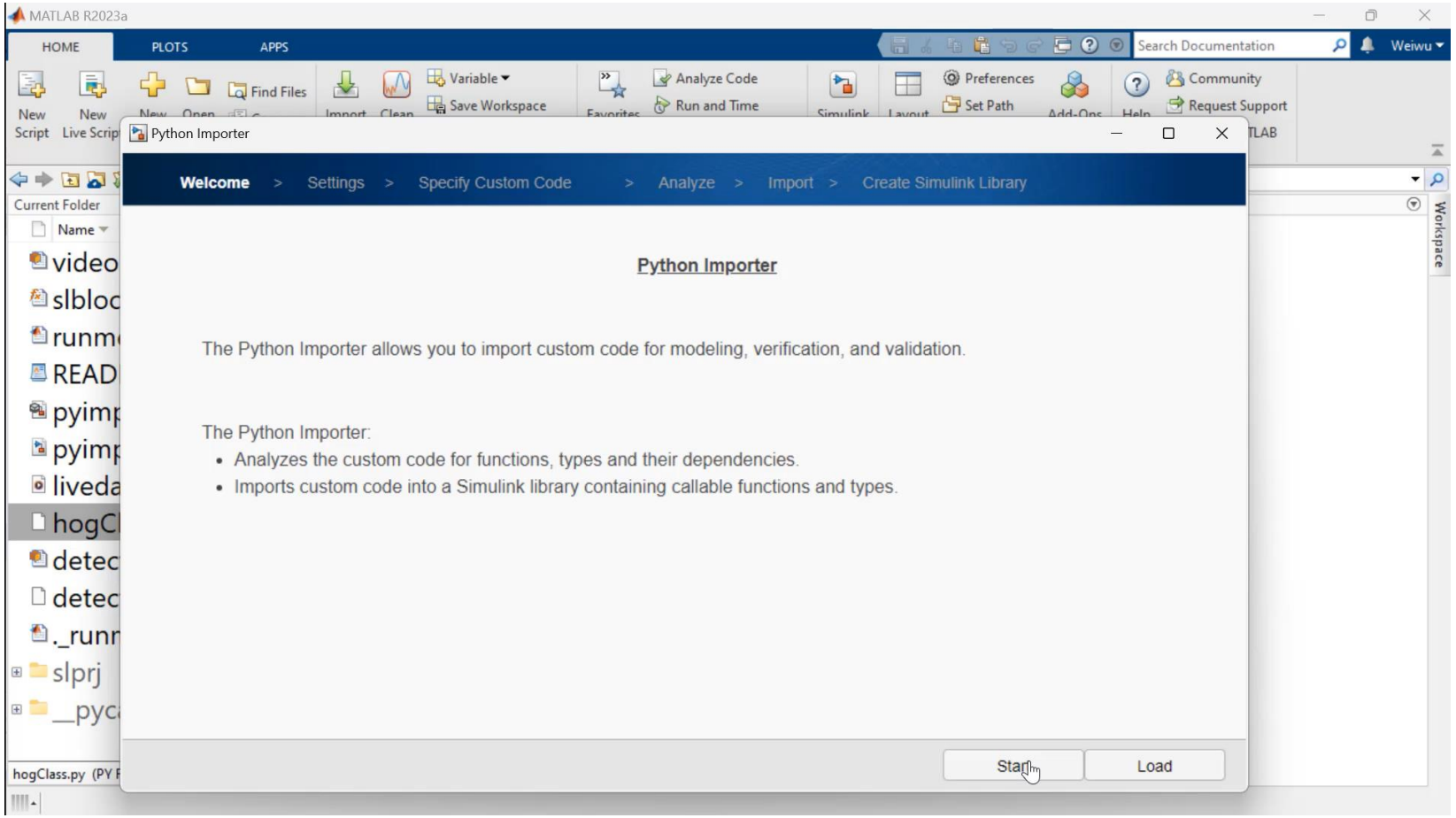

- Graphical wizard for step-by-step guidance, no/minimal manual code
- Integrating a package of Python functions with each Python function corresponding to a library block
- Convenient for re-use or building a custom blockset

R2023a


# Calling Python from Simulink

## - Use Python Importer

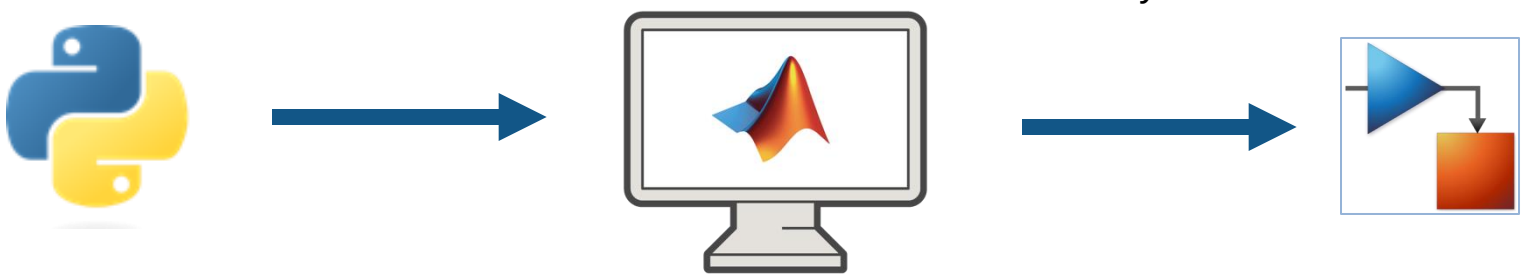
- Demo: Integrate human detection algorithm in Python

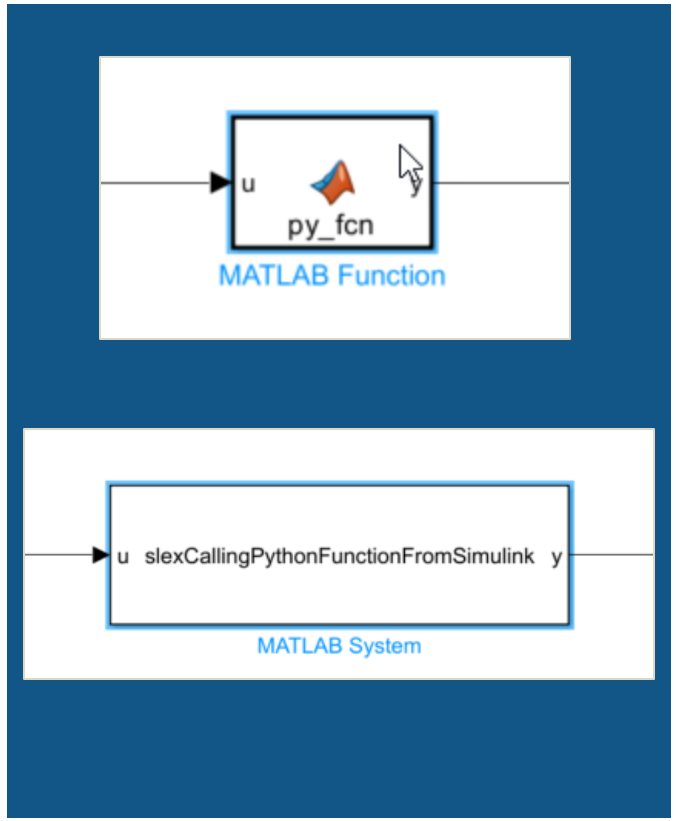



# Calling Python from Simulink
## - Using MATLAB Interface for Python

Use Python Interface API

Embed MATLAB code inside a MATLAB Function block or a MATLAB System block






If you are using a version earlier than R2023a, or you would like to write code manually:

- Write MATLAB functions or MATLAB System objects
- Access Python libraries directly by adding the py. prefix to the Python name


# Scenario #2

James

나는 TensorFlow와 PyTorch를 사용하여 Deep Learning Model을 만드는 Data Scientist입니다.(Battery SOC[1] 추정)

Jayden

나는 James가 만든 Deep Learning Model을 Simulink로 가져와서 System-Level 검증을 수행하고자 합니다.
하지만 Co-simulation뿐만 아니라 실제 H/W Test를 위한 코드생성도 하고 싶군요.

1)SOC: State of Charge


# There are two options to bring your deep learning model into Simulink if code gen is a must have

- Import your deep learning model in MATLAB directly

  \+ Multi-platform code generation: library-free C/C++ code, optimized code for Intel and ARM processors, and CUDA code for NVIDIA® GPU

  Δ import process can be painful, need for custom code, and validation testing

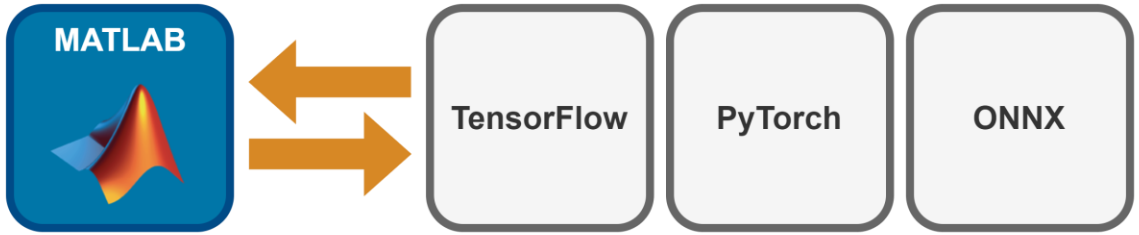
- Simulate and generate code for TensorFlow™ Lite model

  \+ requiring only a simple Python code to compile the model

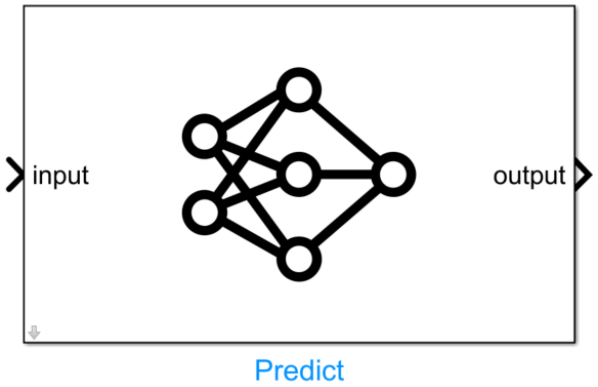
  Δ requires the TensorFlow Lite interpreter and libraries built on the target hardware, which is currently limited to Windows and Linux targets


# Integrate deep learning models from Python

## -using MATLAB model converters for TensorFlow, PyTorch, and ONNX



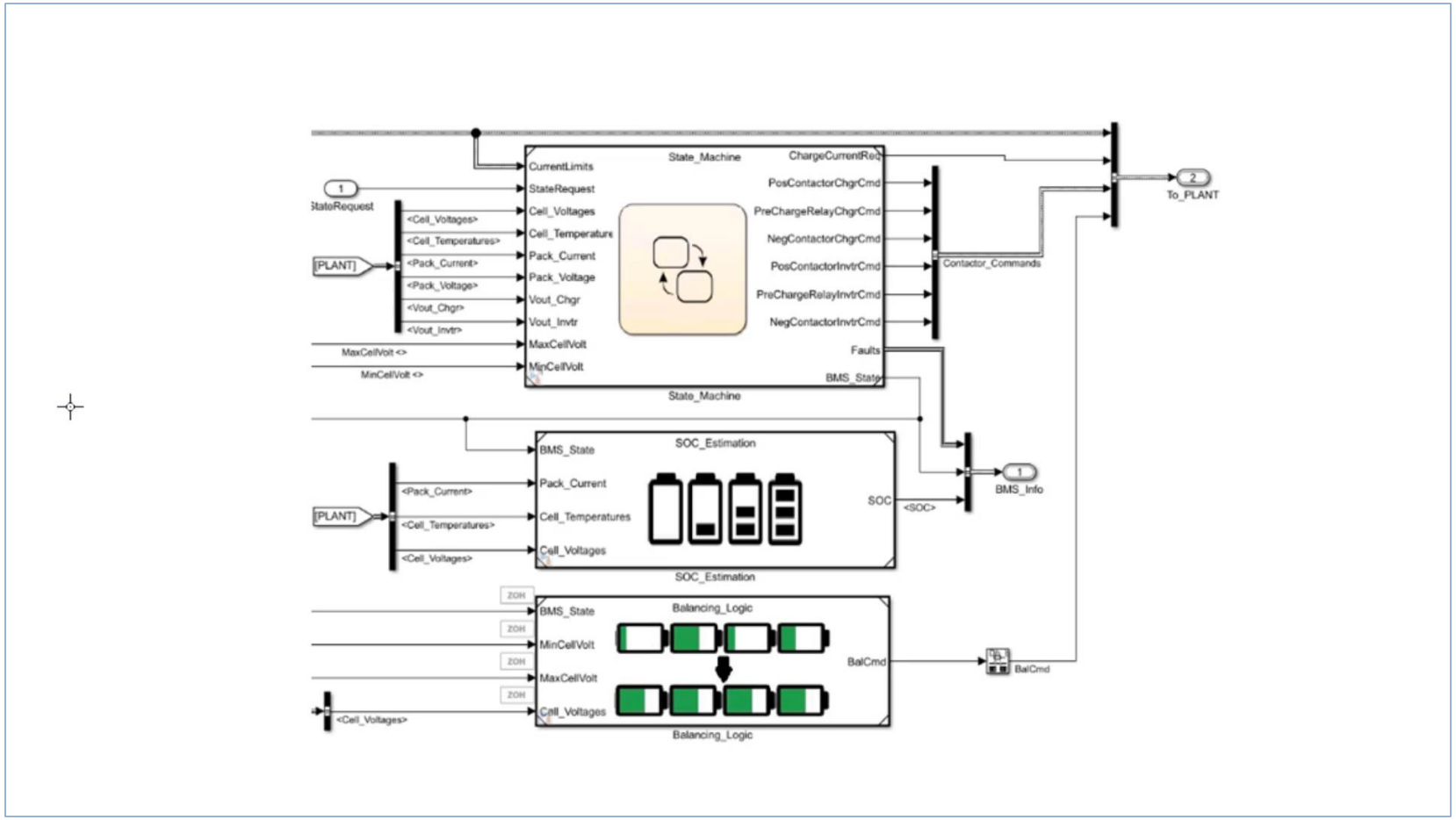

- Once the model is converted in MATLAB, use the deep neural networks blocks to bring it into Simulink, for both simulation and code generation




# Integrate deep learning models from Python

## -using MATLAB model converters for TensorFlow, PyTorch and ONNX

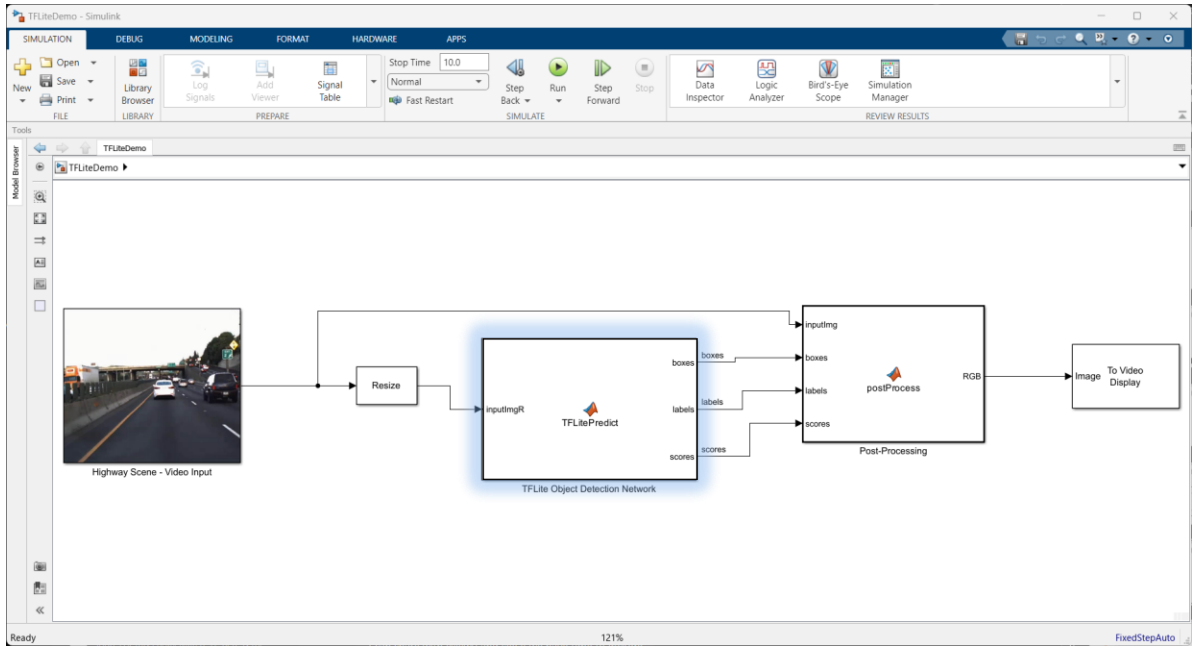
- Demo: Integrate a TensorFlow model for battery SoC estimation




# Integrate deep learning models from TensorFlow Lite (TFLite)

## -using the MATLAB function to load a pre-trained TFLite model in Simulink

*Example: TFLite Object Detector integrated with Simulink*




```
169     memcpy(&b[0], &x[0], 100U * sizeof(real32_T));
```

```
171
172   // Function for MATLAB Function: '<Root>/TFLite Object Detection Network'
173   static void TFLiteDemo_TFLiteModel_predict(coder_TFLiteModel_TFLiteDemo_T
174     *b_this, const uint8_T varargin_1[307200], real32_T varargout_1[400], real32_T
175     varargout_2[100], real32_T varargout_3[100], real32_T *varargout_4)
176   {
177     real_T count;
178     std::mem_fn(&invokeinterpreter::setVerbose)(b_this->Network, b_this->Verbose);
179     std::mem_fn(&invokeinterpreter::setProfiling)(b_this->Network,
180       b_this->EnableProfiling);
181     std::mem_fn(&invokeinterpreter::setNumThreads)(b_this->Network,
182       b_this->NumThreads);
183     std::mem_fn(&invokeinterpreter::setInputMean)(b_this->Network, b_this->Mean);
184     std::mem_fn(&invokeinterpreter::setInputStdDeviation)(b_this->Network,
185       b_this->StandardDeviation);
```

Import TFLite models using a MATLAB Function block

# Scenario #3

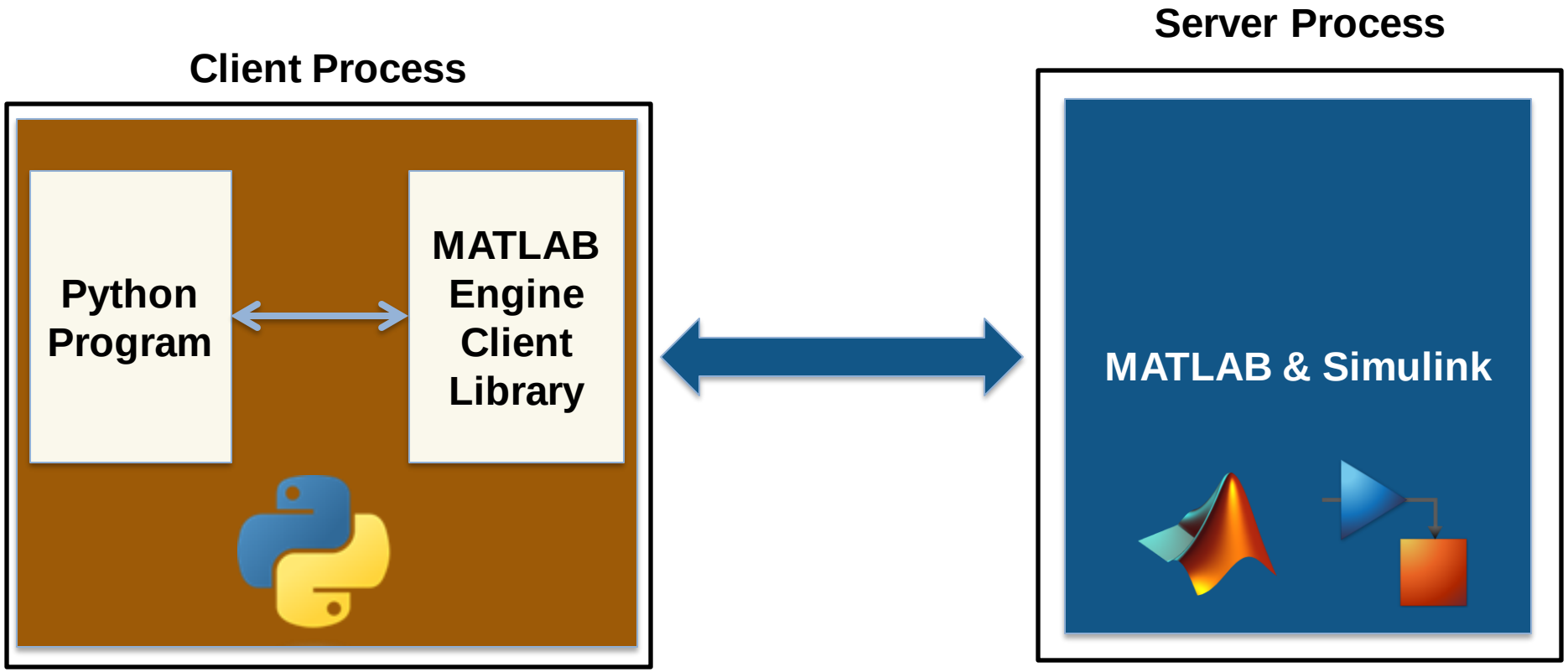
Jayden

나는 Dynamic System Modeling을 위해서 Simulink를 사용합니다. 가령 차량 Suspension System을 모델링하죠.

James

나는 Python 기반의 자동화 Framework을 이용해서 Simulink Simulation을 하고 싶습니다. Jayden이 만든 Simulink 모델을 Python에서 테스트해보고 싶군요.


# Simulate a Simulink Model from Python
## - Using MATLAB Engine API



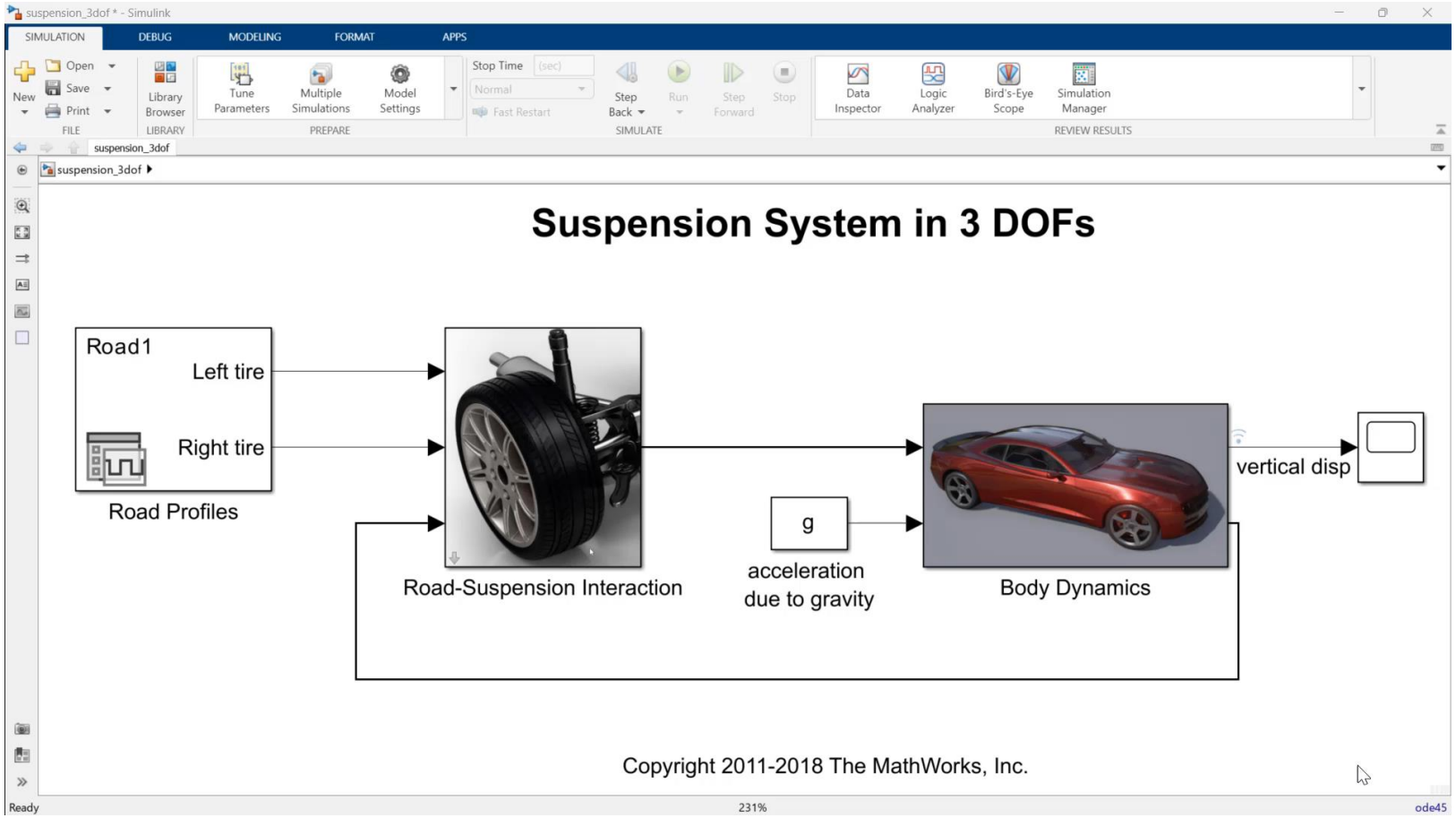

```
mle = matlab.engine.start_matlab(); # start the MATLAB engine
res[0] = mle.sim_the_model();  # # run Simulink simulation within a MATLAB function
```


# Simulate a Simulink Model from Python
## - Using MATLAB Engine API

- Create/terminate MATLAB
- Put variable into MATLAB workspace
- Get variable from MATLAB workspace
- Provide flexible Simulink simulation capabilities including changing non-tunable parameters and running simulations in normal mode


# Simulate a Simulink Model from Python
## - Using MATLAB Engine API

- Demo: Simulate a road suspension model in Python




# Scenario #4

Jayden

나는 Dynamic System Modeling을 위해서 Simulink를 사용합니다. 가령 차량 Suspension System을 Modeling하죠.

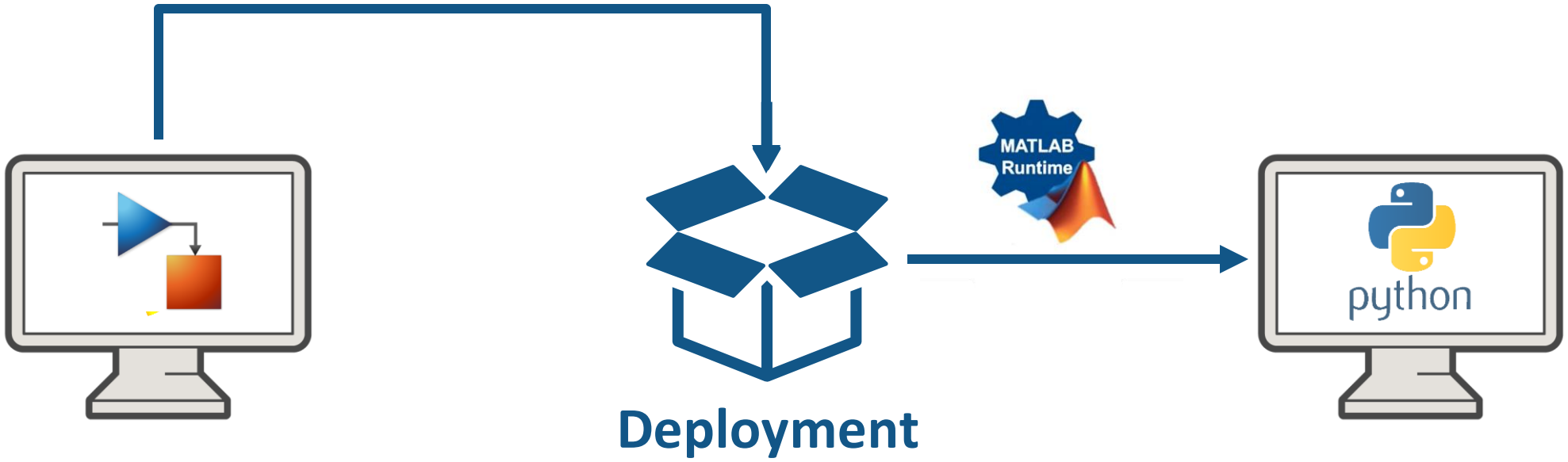
James

나는 Jayden이 만든 Simulink Model을 Python 기반의 제품에 적용하고 싶습니다. Simulink Simulation을 Python Package화 하여 적용하고 싶군요.


# Call a compiled Simulink model from Python

## - Using MATLAB Runtime

Generate a Python package from a MATLAB function that encapsulates a Simulink simulation

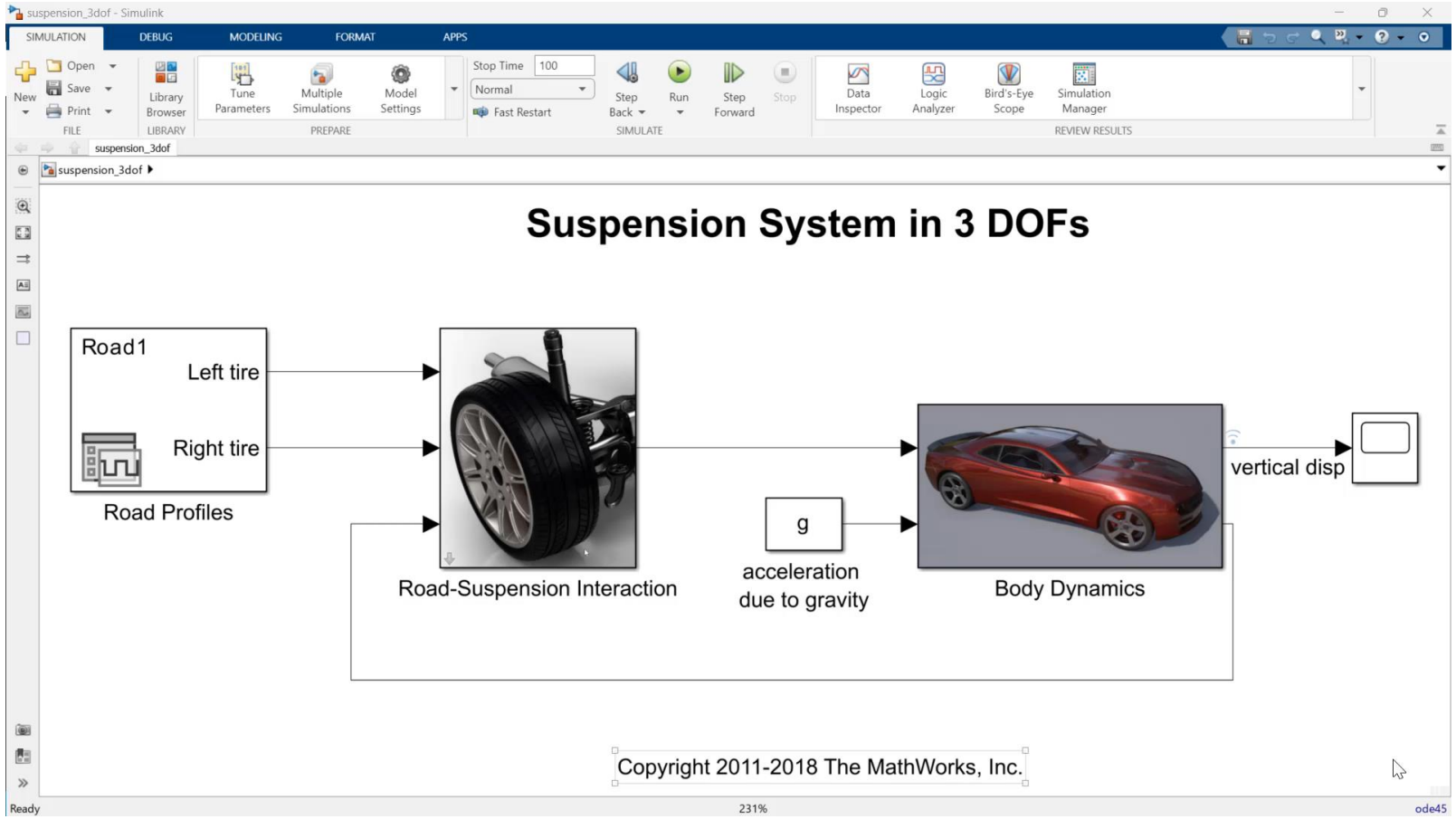



```
import sim_the_model
mlr = sim_the_model.initialize()
res[0] = mlr.sim_the_model()
```


# Call a compiled Simulink model from Python

## - Using MATLAB Runtime

- Demo: Simulate the compiled suspension system model as a Python package



# Other ways to call a compiled Simulink model in Python


- Package the Simulink model as a Functional Mockup Unit (FMU)
  - Call the FMU from Python using third-party libraries such as FMPy
- Package the Simulink model as a simulation service API (using MATLAB Production Server)
  - RESTful API for scalable applications
- Generate C/C++ code or shared library from the Simulink model
  - Call the generated code using CTYPES or related wrappers

# Key takeaways

- Simulink as an open simulation platform supports versatile ways to interoperate with Python:
  - Bring Python code into Simulink as a library for co-execution
  - Integrate TensorFlow and PyTorch models for both simulation and code generation
  - Simulate a Simulink model directly from Python
  - Export a Simulink model as a Python package for deployment




# Customer Reference: Mercedes-Benz Simulates Hardware Sensors with Deep Neural Networks

## Challenge

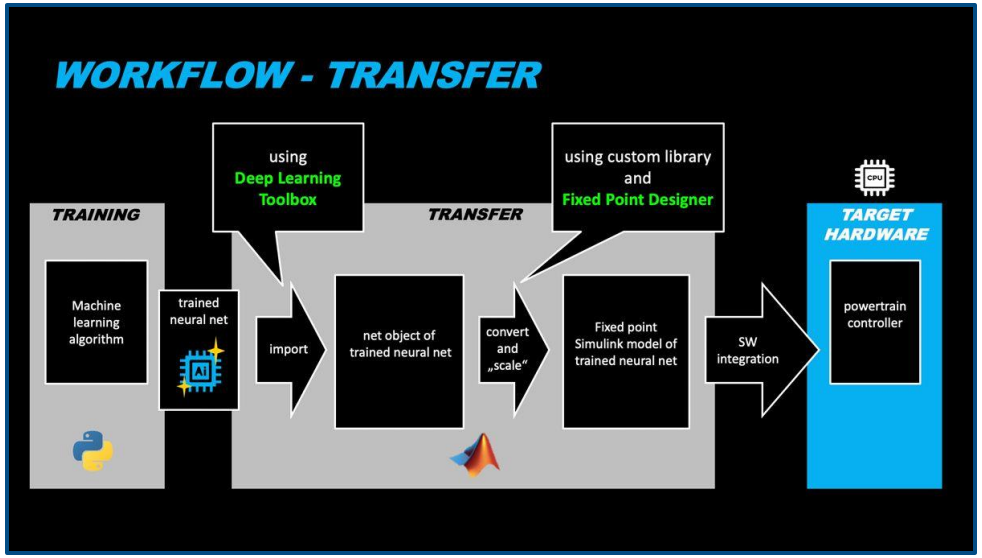
Simulate automotive hardware sensors with deep neural networks

## Solution

Use MATLAB, Simulink, Deep Learning Toolbox, and Fixed-Point Designer to convert Qkeras(Python) deep learning models into code that can be deployed to an automotive ECU

## Results

- CPU, memory, and performance requirements met
- Flexible process established
- Development speed increased 600%




**Automated workflow for deploying virtual sensors to powertrain ECU.**

*"This was the first time we were simulating sensors with neural networks on one of our powertrain ECUs. Without MATLAB and Simulink, we would have to use a tedious manual coding process that was very slow and error-prone."*

*- Katja Deuschl, AI developer at Mercedes-Benz*

Link to user story

# To learn more

- Integrate Python Code with Simulink
- Deep learning with Simulink
- Importing Models from TensorFlow, PyTorch, and ONNX
- MATLAB Engine API
- Call Simulink from Python
- Python Package Integration

## EXPO Special Track

| Time | Session | | | | |
|---|---|---|---|---|---|
| 9:00 | 등록 | | | | |
| 9:40 | 인사말 이종민 대표이사, 매스웍스코리아 | | | | |
| 9:50 | [고객 기조연설] SW 중심으로 진화되는 차량제어 시스템 김치경 상무, 현대자동차 | | | | |
| 10:30 | [기조연설] 불가능을 가능하게 만든 과학자와 엔지니어의 도전 Sameer Prabhu, 매스웍스 | | | | |
| 10:50 | 휴식 및 부스 관람 | | | | |
| | Track 1 - 인공지능 | Track 2 - 모델링과 시뮬레이션 | Track 4 - 자율시스템 개발 | 스페셜 트랙 - 컨설팅 | 스페셜 트랙 - 테크니컬 컴퓨팅 |
| 11:20 | (1-1) 인공지능 모델의 이해와 검증<br>김종남 부장, 매스웍스코리아 | 국방, 우주 개발의 MBD 프로세스 적용<br>한상설 팀장, 국방과학연구소 | MATLAB을 이용한 AI 기반 자율로봇 구현<br>김종현 부장, 매스웍스코리아 | | |
| 11:50 | 점심 식사 | | | | |
| 13:00 | MATLAB 기반 딥러닝 모델을 활용한 감시 카메라 영상 내 차량 속력 계산 시스템 개발<br>최영수 연구원, 국립과학수사연구원<br>대전과학수사연구소 | Model-Based Design 설계에서 Simulink와 Python 연동하기<br>신행재 부장, 매스웍스코리아 | 양산형 Safety/자율주행 솔루션 개발을 위한 프로세스 전 단계에서의 MATLAB/Simulink 활용 노하우<br>하대근 CTO, ADUS Inc | MATLAB 프로그래밍 아키텍처 디자인<br>임형득 부장, 매스웍스코리아 | |
| 13:30 | 휴식 | | | | 휴식 |
| 13:40 | AI 기반 차원 축소 모델을 사용한 Simulink에서의 시스템 분석 및 설계 가속화<br>엄준상 부장, 매스웍스코리아 | DO Qualification Kit와 모델기반 설계를 이용한 항공 Software 개발<br>이승현 책임연구원, 현대자동차 | Track 5 - 전동화<br>전동식 브레이크 시스템의 전자-유압 모델을 이용한 모터 최적 변수 설계<br>오선열 책임연구원, HL 만도 | | Python과 함께 사용하는 MATLAB 워크샵<br>박인용 과장, 매스웍스코리아 |
| 14:10 | 휴식 | | | | |
| 14:20 | Low Code 데이터 분석 및 인공지능<br>장규환 부장, 매스웍스코리아 | Track 3 - 예측 정비<br>펌프 시스템 건전성 관리를 위한 진단 및 예지 기술<br>선경호 책임연구원, 한국기계연구원 | 시스템 모델링을 이용한 EV용 배터리 팩 성능 최적화<br>강효석 부장, 매스웍스코리아 | 다양한 엔지니어링 최적화 문제를 해결하는 효과적인 방법<br>최규웅 부장, 매스웍스코리아 | |
| 14:50 | 휴식 및 부스 관람 | | | | 금융과 인공지능 |
| 15:30 | 클라우드 기반의 MATLAB 영상 검사 시스템 개발<br>성호현 부장, 매스웍스코리아 | 건전성 예측관리 시스템의 개발/운용 효율을 위한 DevOps 구축<br>엄준상 부장, 매스웍스코리아 | Track 6 - 무선 및 신호처리<br>MATLAB을 활용한 6G 연구 및 개발 가속하기<br>서기환 부장, 매스웍스코리아 | | 금융 분야의 MATLAB의 활용 사례와 성과<br>장규환 부장, 매스웍스코리아 |
| 16:00 | 휴식 | | | | |
| 16:10 | 클라우드 기반 교량의 장기 성능 평가 시스템 개발<br>박종웅 교수, 중앙대학교 | MATLAB을 활용한 수소충전소 이상감지 시스템 개발<br>한국가스공사 | MATLAB 환경에서 레이다 및 안테나 시스템의 설계 및 운용에 대한 최적화 기법<br>김석 부장, 매스웍스코리아 | | MATLAB을 이용한 eXplainlable AI 모형의 금융자산 예측 적용<br>최선용 교수, 가천대학교 |
| 16:40 | 경품 추첨 및 맺음말 | | | | |
| 17:00 | 행사종료 | | | | |

# MATLAB EXPO

## Thank you