

How to harness the power of Python programming within the Simba simulation environment

By Peyman Razmi

March 28th, 2024

How to harness
the power of
Python
programming
within the Simba
simulation
environment

AGENDA

- Introduction to Python Programming
- Python environment and Libraries
- Introduction to Simba environment,
its capabilities
- Power electronic simulation in Simba
- Management of Simba Simulation
with Python
- Introduction to Julia programming in
Simba simulation



What is Python?

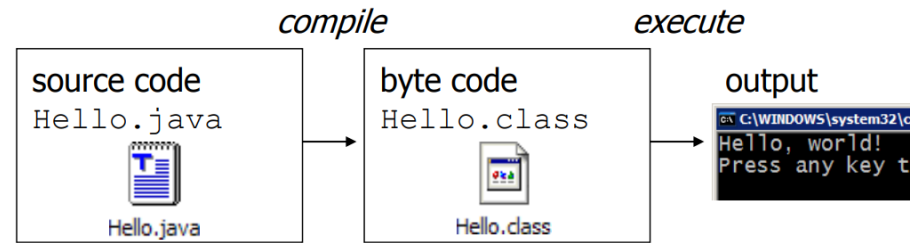
Python is a popular high-level programming language used in various applications

- Python is an easy language to learn because of its simple syntax
- Python can be used for simple tasks such as plotting or for more complex tasks like Algebra programming, optimization and machine learning algorithms.



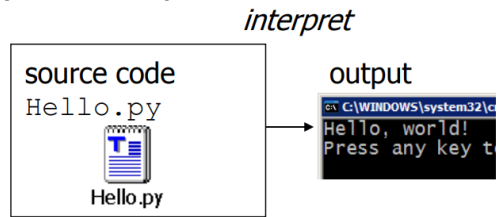
Introduction to Python

Many languages require you to compile (translate) your program into a form that the machine understands.



Python

Python is instead directly interpreted into machine instructions.

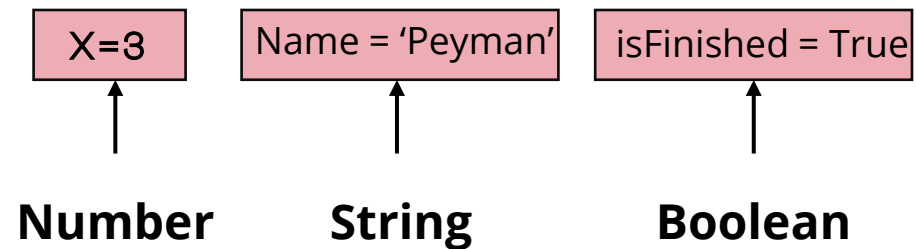


The interpreter provides an interactive environment to play with the language



— Variables, Objects, and Classes

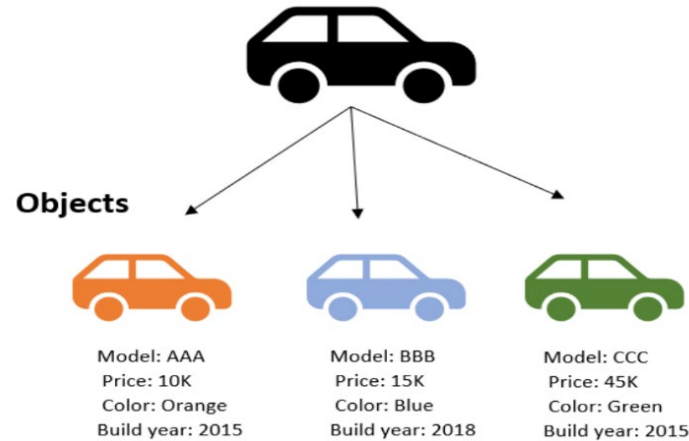
- A variable is a reference to a value stored in a computer's memory.
- Variables can be sorted into a variety of categories (or data types) such as numbers (int/float etc), Boolean values (true/false), and sequences (strings, lists etc).



- An object is a collection of data from a computer's memory that can be manipulated.



Object Example



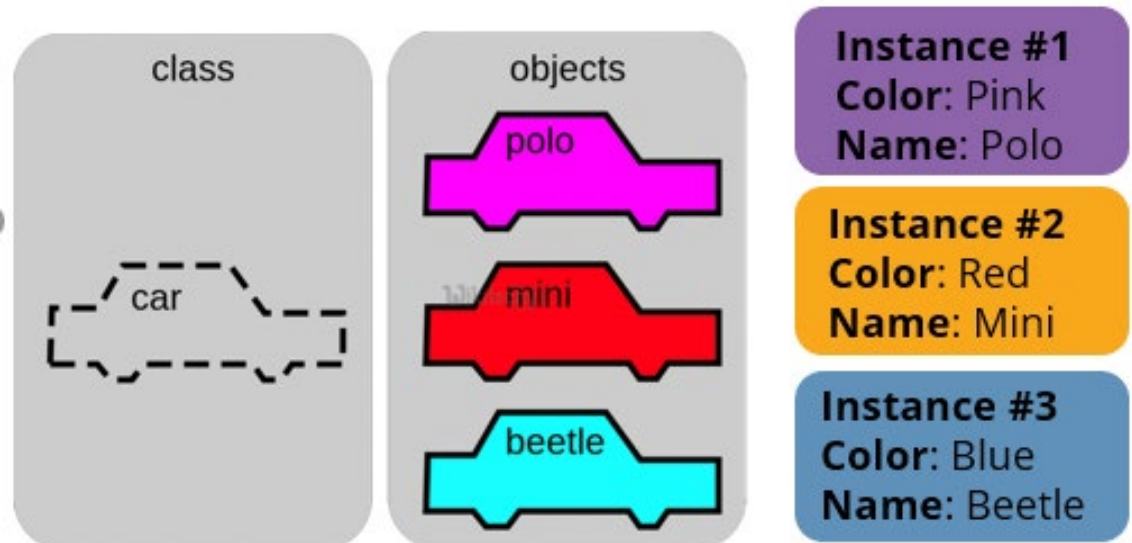
- **Color:** The color of the car (e.g., red, blue, black).
- **Make:** The manufacturer of the car (e.g., Ford).
- **Model:** The model name or number (e.g., Camry, F-150, X5).
- **Year:** The manufacturing year of the car.
- **Engine Size:** The size of the engine (e.g., 2.0L, 3.5L).
- **Fuel Type:** The type of fuel the car uses (e.g., gasoline, diesel, electric).
- **Mileage:** The number of miles or kilometers the car has traveled.
- **IsNew:** A boolean variable indicating whether the car is new or used.



Classes

A **class** is a collection of objects who share the same set of variables.

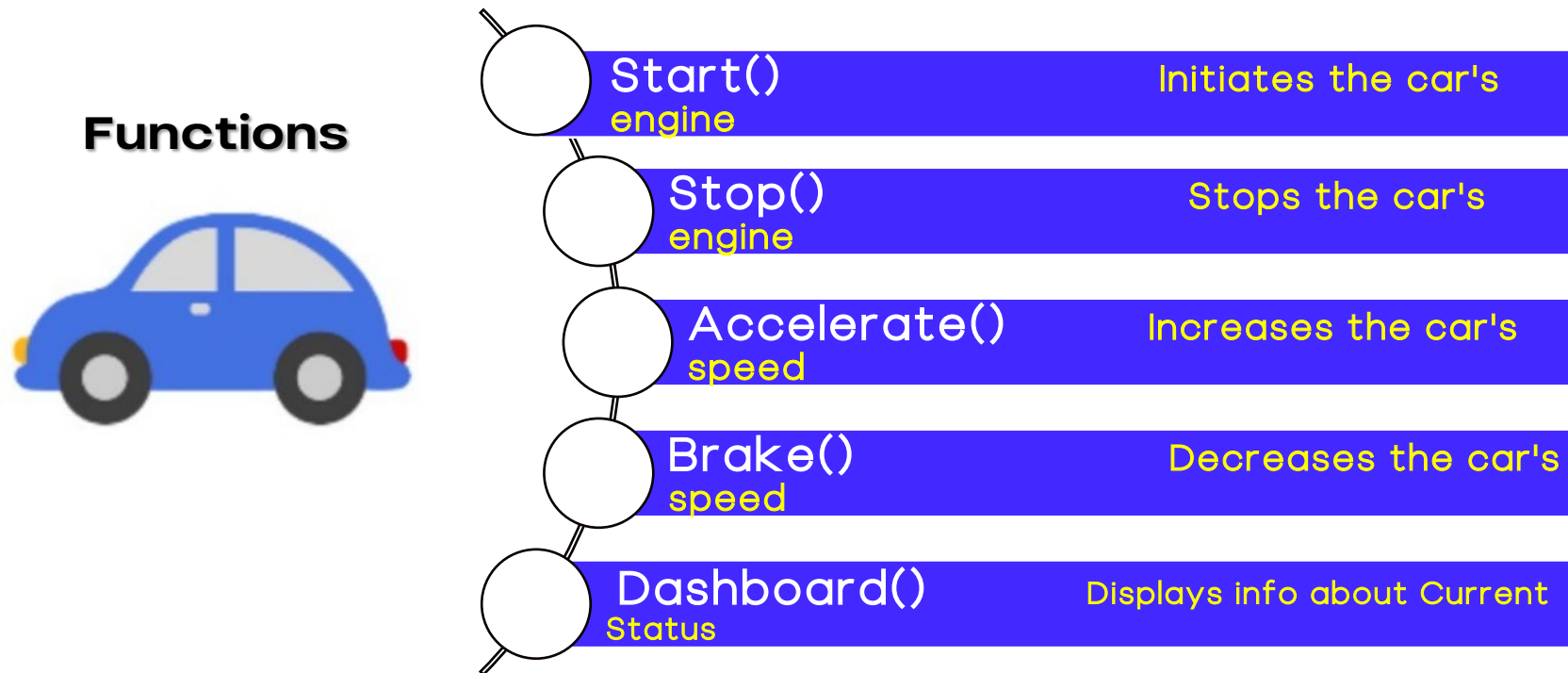
- The definition of the class provides a blueprint for all the objects within it (**instances**).
- Instances may share the same variables (color, size, shape, etc.), but they do **NOT** share the same values for each variable (blue/red/pink, small/large, square/circular etc.)



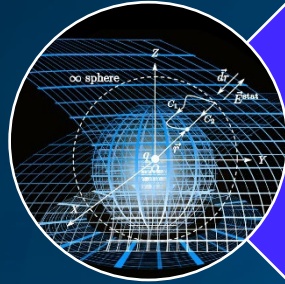
Methods

Methods are the functions used to act on/alter an object's data. They describe what your object can "do."

Methods define actions or behaviors that the car can perform

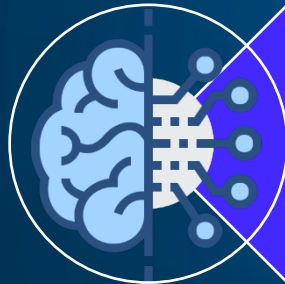


Python Libraries



Scientific Computing

- Numpy
- Pandas
- Scipy
- Matplotlib



Machine Learning AI

- SckitLear
- TensorFlow
- Keras
- Pythorch

SIMBA Platform

Simba is a software platform used for simulating power electronics and motor drives. It provides tools and features for modeling and analyzing electrical systems and components, allowing engineers and researchers to simulate the behavior of power electronic circuits and motor drive systems.

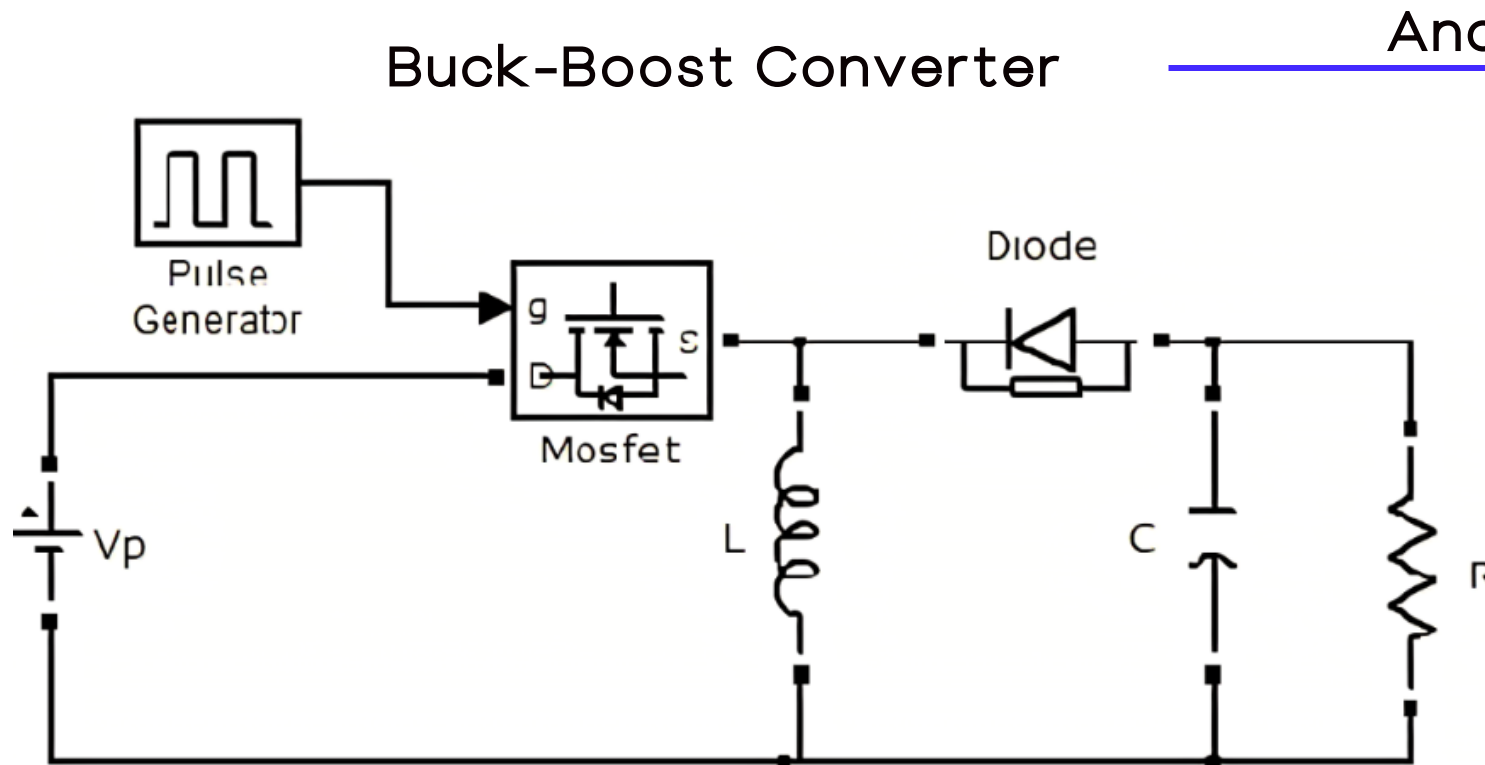


Simba Simulation Environment

The screenshot displays the Simba Simulation Environment interface. At the top left, there is a menu icon and a 'Time setting' label pointing to a gear icon. Next to it is a 'Run Button' with a play icon. The top center shows a 'Test1' window. On the right, there are three tabs: 'Test Bench', 'Design' (which is selected and highlighted with a red circle), and 'Result'. Above these tabs is the label 'Output waveform'. Below the tabs are zoom controls (-, +, Fit) and a 'DESIGN VARIABLES' section. This section contains a text box with instructions: 'Click "New Variable" to define variables that can be reused in device parameter expressions. Variables can also be defined in an external variable file using the syntax "variable_name = expression".' Below the text is a 'New Variable' button and a 'Variable File Path' input field with a dropdown arrow. On the left side, there is a sidebar with 'Designs' and 'Libraries' tabs. The 'Libraries' tab is active, showing a search bar and a list of components: Electrical, Rotational Mechanical, Thermal, Control, Utility, and Annotation. A red box highlights this list, with the label 'Components' next to it. A red line points from the 'Design' tab to the central 'Design environment' area.



Simba Simulation Environment



Analysis

- ❖ Simba Environment
- ❖ Python programming



Required components/Elements

The screenshot displays the SIMBA software interface. At the top left, the SIMBA logo is visible. The main window is titled 'Proj1' and includes a 'Test Bench' button on the right. A sidebar on the left shows 'Designs' and 'Libraries' tabs, with 'Proj1' listed under 'Designs'. The central area displays a grid of components:

- PWM**: A green square icon with a pulse-width modulation waveform, labeled 'C1'.
- Inductor**: A blue icon of an inductor, labeled 'L1 100uH'.
- Capacitor**: A blue icon of a capacitor, labeled 'C2 10uF'.
- Resistance**: A blue icon of a resistor, labeled 'R1 80Ω'.
- Diode**: A blue icon of a diode, labeled 'D1'.
- Switch/Mosfet**: A blue icon of a MOSFET circuit, labeled 'M1'.
- Voltage Source**: A blue icon of a DC voltage source, labeled 'DC1 20V'.

Control buttons for zooming (-, +) and fitting (Fit) are located in the top right corner of the component grid.



Design Buck-Boost Converter

The screenshot displays a circuit design software interface. On the left, a sidebar titled "Designs and Libraries" shows a search bar and a list of categories: Electrical, Rotational Mechanical, Thermal, Control, Utility, and Annotation. The main workspace shows a circuit diagram of a buck-boost converter. The circuit includes a DC voltage source labeled "DC1 8V", a MOSFET labeled "T1", an inductor labeled "L1 100uH", a diode labeled "D3", a capacitor labeled "C1 10uF", and a load resistor labeled "R2 5Ω". A pulse generator labeled "C2" is connected to the gate of the MOSFET. The resistor "R2" is highlighted with a red circle. On the right, a "RESISTOR PROPERTIES" panel is open, showing the name "R2" and value "5". Under the "Scopes" section, "Current" and "Voltage" are both checked. A "Result" tab is circled in red at the top of the panel, with red arrows pointing from it to the resistor in the circuit and the checked checkboxes. A "Help" button is located at the bottom of the properties panel. The top of the interface has a "Test Bench" tab and a "Design" tab, with "Design" being the active tab.





Sam
1 job completed.



Test Bench

Design

Result

Results

Filter Results... ('X' to filter Checked signals) 🔍

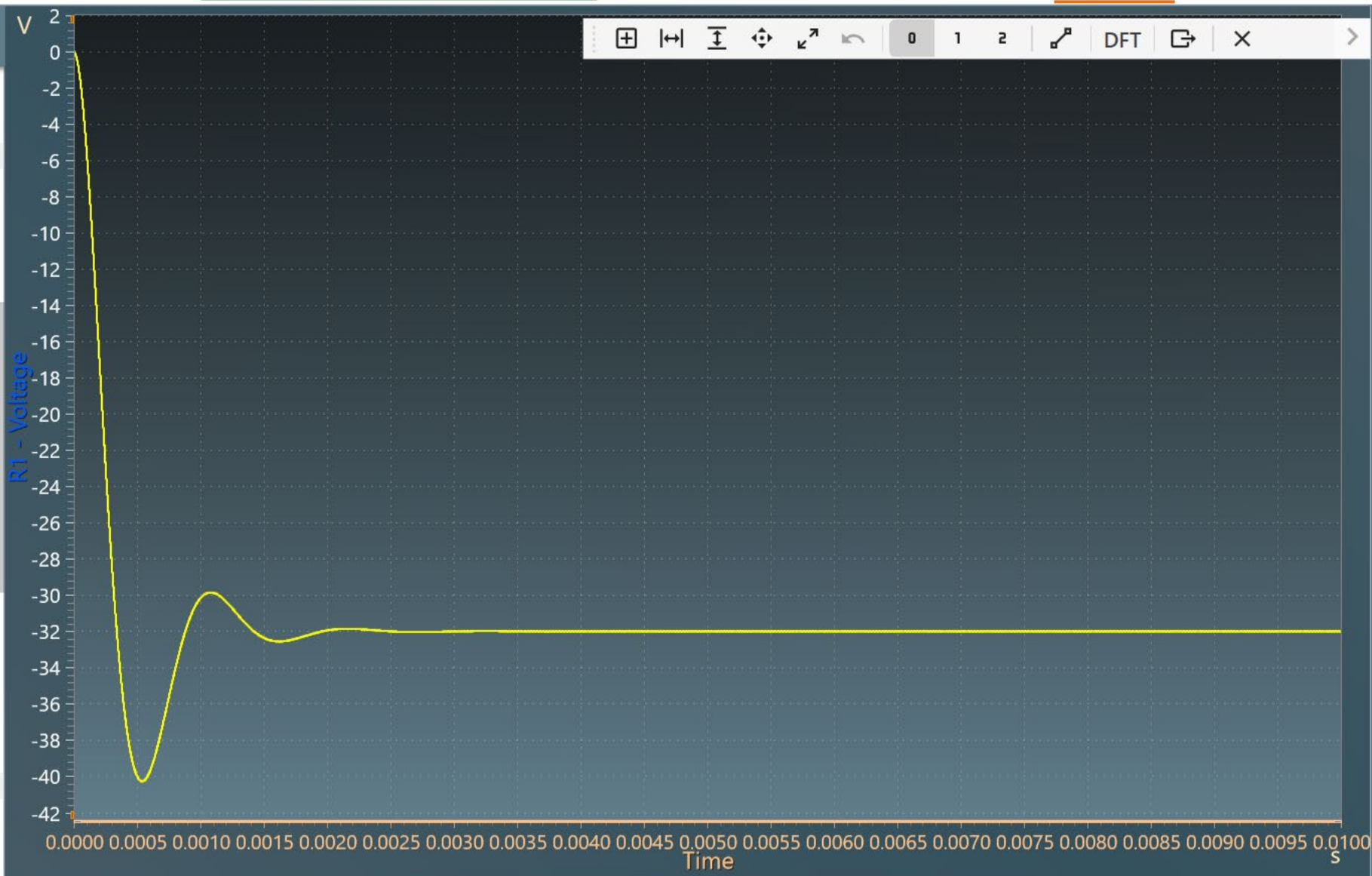
- D3 - Voltage
- DC1 - Current
- DC1 - Voltage
- L1 - Current
- L1 - Voltage
- R1 - Current
- R1 - Voltage Edit
- T1 - Voltage

▶ Sam - Transient
3/4/2024 12:27:20 AM

▶ Sam - Transient
3/4/2024 12:17:22 AM

▶ Sam - Transient
3/4/2024 12:17:18 AM

Delete Results... New Chart



Python Programming and Import Designed simulation

Required Libraries

```
1  #%% Load modules
2  import matplotlib.pyplot as plt
3  from aesim.simba import JsonProjectRepository
4  import os
5  #%% Load project
6
7  #%% Get project
8  script_folder = os.path.realpath(os.path.dirname(__file__))
9  file_path = os.path.join(script_folder, "Proj3.jsimba")
10 project = JsonProjectRepository(file_path)
11 BuckBoost = project.GetDesignByName('pro')
12
```

Import created design

- ✓ **script_folder = os.path.realpath(os.path.dirname(__file__)):**
This line finds the real path of the directory where our current Python script is located.
- ✓ **file_path = os.path.join(script_folder, "ConPro.jsimba"):**
Here, we're creating the full path to our Simba project file, 'Proj3.jsimba', by joining the script's folder path with the project file's name.
- ✓ **project = JsonProjectRepository(file_path):**
Lastly, this line loads our Simba project file into the Python script using a class called `JsonProjectRepository`. By doing this, our script can directly interact with the project, allowing for manipulation, analysis, or automation within the Simba environment."



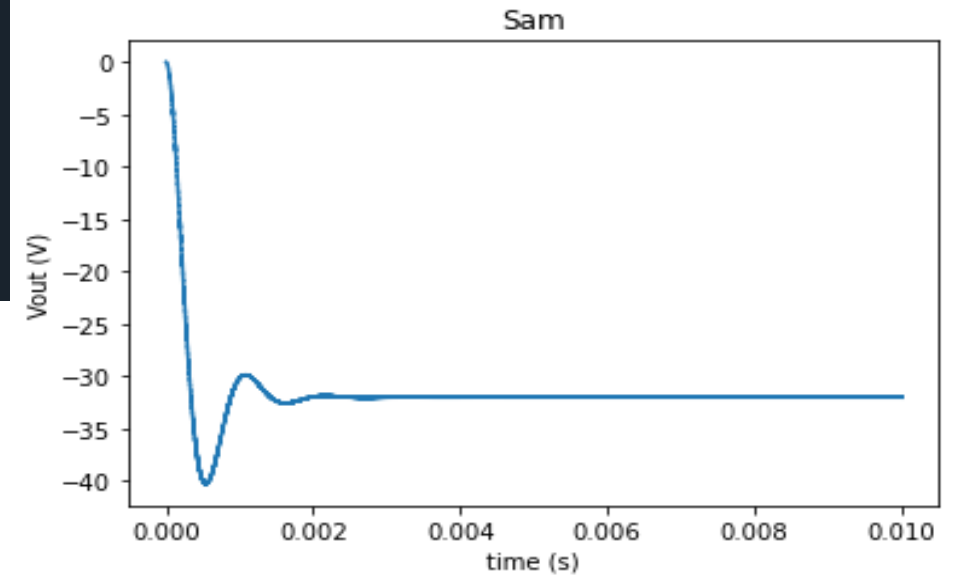
Python Programming Explanations

Command Transient Analyze

```
16 job = BuckBoost.TransientAnalysis.NewJob()
17 status = job.Run()
18
19 #%% Get results
20 t = job.TimePoints
21 Vout = job.GetSignalByName('R1 - Instantaneous Voltage').DataPoints
22
23 #%% Plot Curve
24 fig, ax = plt.subplots()
25 ax.set_title(BuckBoost.Name)
26 ax.set_ylabel('Vout (V)')
27 ax.set_xlabel('time (s)')
28 ax.plot(t, Vout)
29 plt.show()
30
```

Plotting code
with Matplot

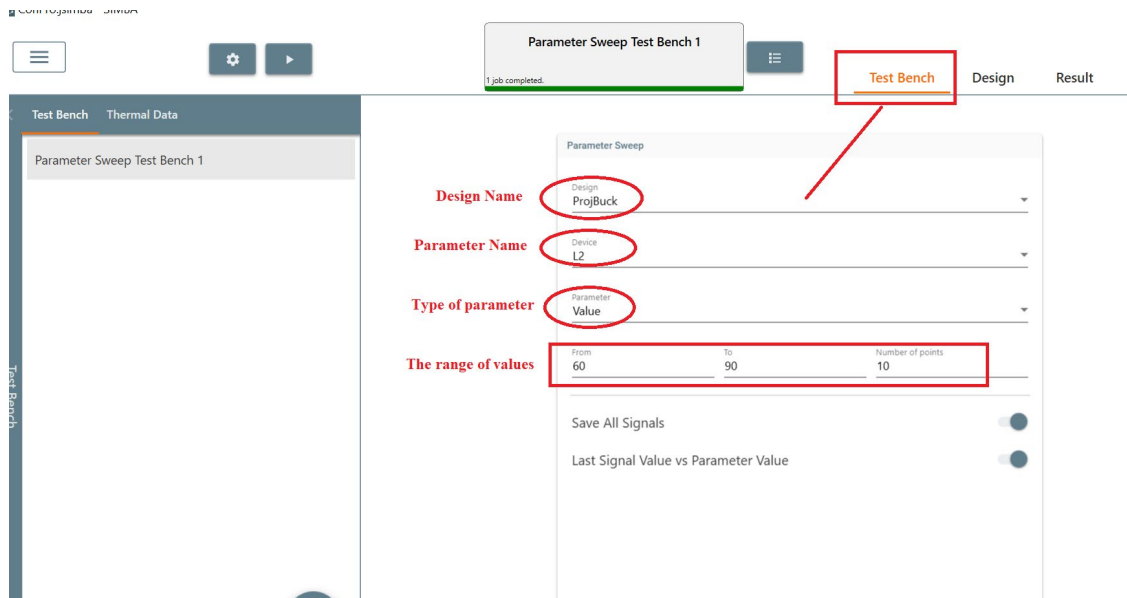
Define Output



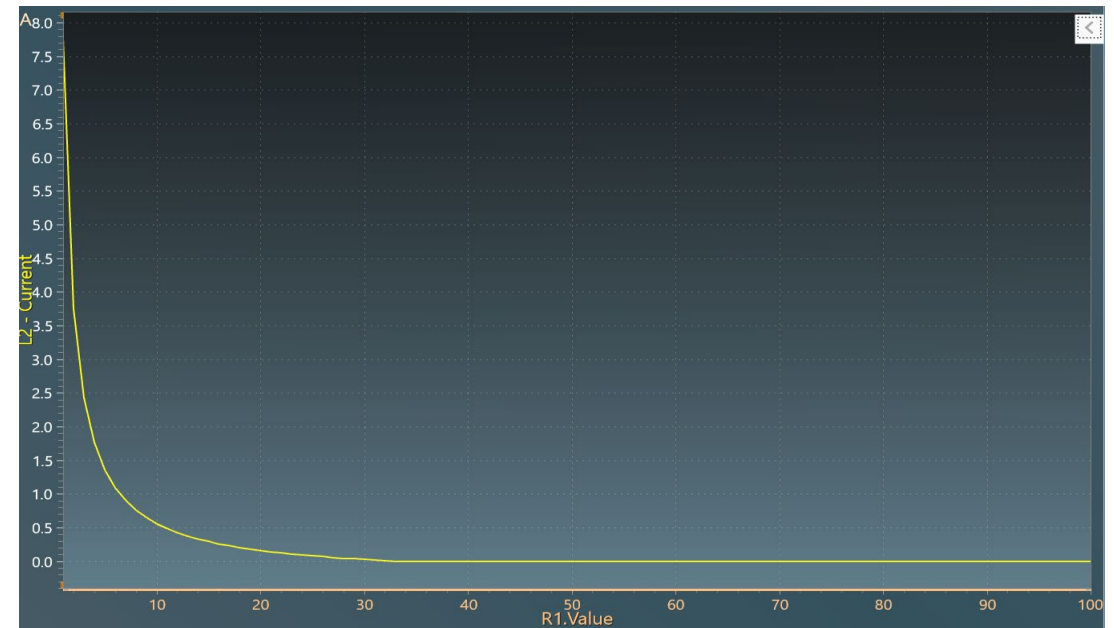
Parameter Sweep in Simba

A "Parameter Sweep" is a process in which a series of simulations are run while systematically varying the parameters of a model to analyze the effects on its behavior or performance.

Parameter Sweep in Simba environment

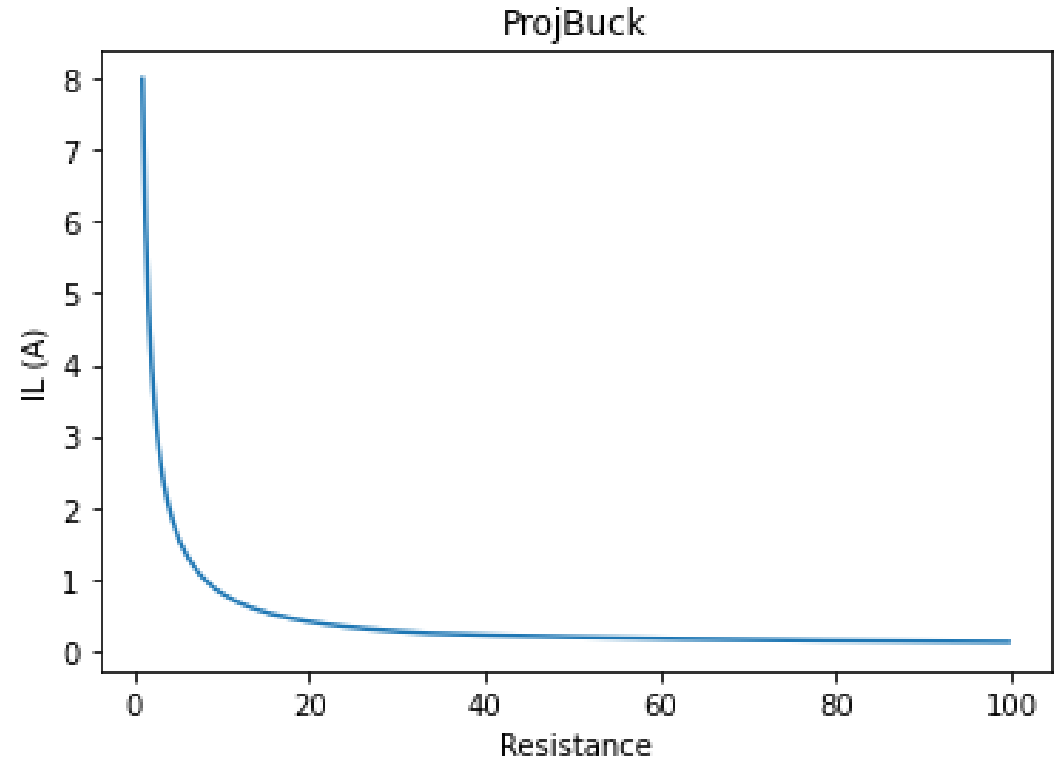


$R = 1\Omega \longrightarrow 100\Omega$



Parameter Sweep with python code

```
R = np.arange(1, 100, 10/30)
ILs = []
for RR in R:
    R2 = BuckBoostConverter.Circuit.GetDeviceByName('R1')
    R2.Value=RR
    # Run calculation
    job = BuckBoostConverter.TransientAnalysis.NewJob()
    status = job.Run()
    # Retrieve results
    t = np.array(job.TimePoints)
    IL = np.array(job.GetSignalByName('L2 - Current').DataPoints)
    # Average output voltage for t > 5ms
    indices = np.where(t >= 0.005)
    IL = np.take(IL, indices)
    IL = np.average(IL)
    # Save results
    ILs.append(IL)
```



Julia libraries in Simba

```
using PyCall  
using PyPlot
```

```
# Import Python libraries in Julia  
aesim = pyimport("aesim.simba")  
os = pyimport("os")  
np = pyimport("numpy")
```

