**NAME:ROHAN NYATI**

**SAP ID:500075940**

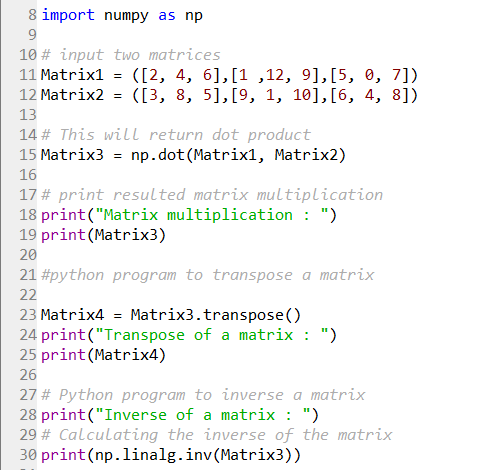
**ROLL NO. : R177219148**

**BATCH-5(AI&ML)**

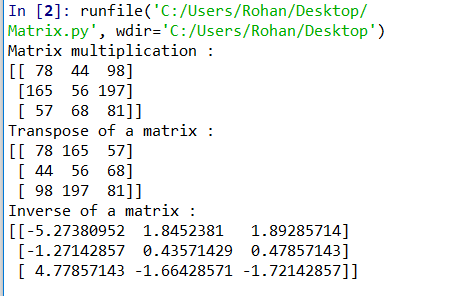
**Assignment - 2**

**Q.1) Understand how to inverse, transpose, multiply two matrices of any dimension in python.**

**Code ->**



**Output ->**



**Q.2)Understand the https://pypi.org/project/control/ library. How can time domain and frequency domain of a signal be represented by the control library.**

**Python Control Systems Library**

The Python Control Systems Library is a Python module that implements basic operations for analysis and design of feedback control systems.

**Features**

* Linear input/output systems in state-space and frequency domain
* Block diagram algebra: serial, parallel, and feedback interconnections
* Time response: initial, step, impulse
* Frequency response: Bode and Nyquist plots
* Control analysis: stability, reachability, observability, stability margins
* Control design: eigenvalue placement, linear quadratic regulator
* Estimator design: linear quadratic estimator (Kalman filter)

## Frequency domain plotting

|  |  |
| --- | --- |
| [**bode\_plot**](https://python-control.readthedocs.io/en/0.8.4/generated/control.bode_plot.html#control.bode_plot)(syslist[, omega, plot, …]) | Bode plot for a system |
| [**nyquist\_plot**](https://python-control.readthedocs.io/en/0.8.4/generated/control.nyquist_plot.html#control.nyquist_plot)(syslist[, omega, plot, …]) | Nyquist plot for a system |
| [**gangof4\_plot**](https://python-control.readthedocs.io/en/0.8.4/generated/control.gangof4_plot.html#control.gangof4_plot)(P, C[, omega]) | Plot the “Gang of 4” transfer functions for a system |
| [**nichols\_plot**](https://python-control.readthedocs.io/en/0.8.4/generated/control.nichols_plot.html#control.nichols_plot)(sys\_list[, omega, grid]) | Nichols plot for a system |
| [**nichols\_grid**](https://python-control.readthedocs.io/en/0.8.4/generated/control.nichols_grid.html#control.nichols_grid)([cl\_mags, cl\_phases, line\_style]) | Nichols chart grid |

## Time domain simulation

|  |  |
| --- | --- |
| [**forced\_response**](https://python-control.readthedocs.io/en/0.8.4/generated/control.forced_response.html#control.forced_response)(sys[, T, U, X0, transpose, …]) | Simulate the output of a linear system. |
| [**impulse\_response**](https://python-control.readthedocs.io/en/0.8.4/generated/control.impulse_response.html#control.impulse_response)(sys[, T, X0, input, …]) | Impulse response of a linear system |
| [**initial\_response**](https://python-control.readthedocs.io/en/0.8.4/generated/control.initial_response.html#control.initial_response)(sys[, T, X0, input, …]) | Initial condition response of a linear system |
| [**input\_output\_response**](https://python-control.readthedocs.io/en/0.8.4/generated/control.input_output_response.html#control.input_output_response)(sys, T[, U, X0, …]) | Compute the output response of a system to a given input. |
| [**step\_response**](https://python-control.readthedocs.io/en/0.8.4/generated/control.step_response.html#control.step_response)(sys[, T, X0, input, output, …]) | Step response of a linear system |
| [**phase\_plot**](https://python-control.readthedocs.io/en/0.8.4/generated/control.phase_plot.html#control.phase_plot)(odefun[, X, Y, scale, X0, T, …]) | Phase plot for 2D dynamical systems |

### State space systems

The State space class is used to represent state-space realizations of linear time-invariant (LTI) systems:

\frac{dx}{dt} &= A x + B u \\
y &= C x + D u

where u is the input, y is the output, and x is the state.

### Transfer functions

The Transfer function class is used to represent input/output transfer functions

G(s) = \frac{\text{num}(s)}{\text{den}(s)}
     = \frac{a_0 s^m + a_1 s^{m-1} + \cdots + a_m}
            {b_0 s^n + b_1 s^{n-1} + \cdots + b_n},

where n is generally greater than or equal to m (for a proper transfer function).

### FRD (frequency response data) systems

The Frequency response data (FRD) class is used to represent systems in frequency response data form.

### Discrete time systems

A discrete time system is created by specifying a nonzero ‘timebase’, dt. The timebase argument can be given when a system is constructed:

* dt = None: no timebase specified (default)
* dt = 0: continuous time system
* dt > 0: discrete time system with sampling period ‘dt’
* dt = True: discrete time with unspecified sampling period