# PHT.313UF

## Modeling and simulation of semiconductors and semiconductor devices

## necessary programs to run your scripts from home

# Using your on programs and scripts

## (1) Which programming languages are possible?

**MATLAB**

available at
servers and
computer at the
two universities

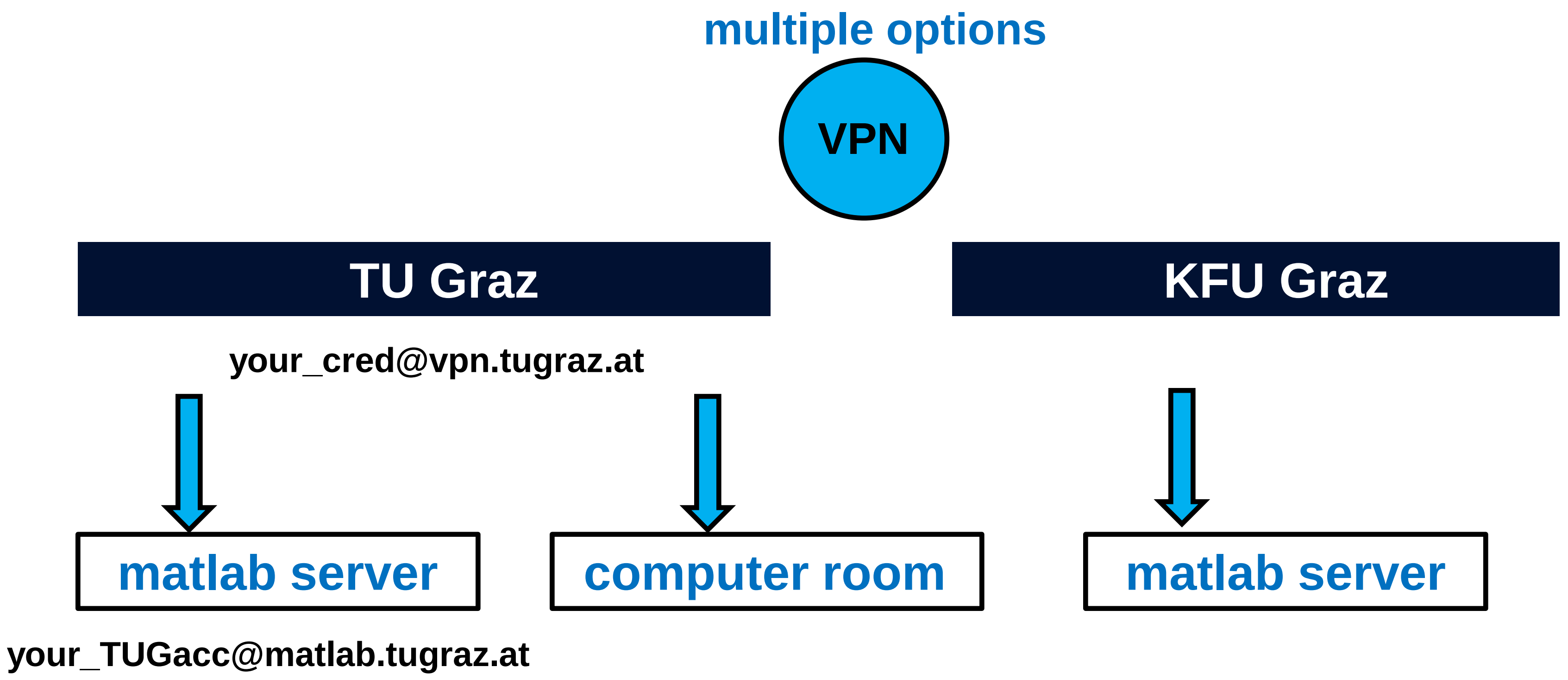
**GNU Octave**

**Python**

free,
can be installed on
Windows and Linux

## (2) What are you options to run your program in your language?

## Remote use of Matlab from home



multiple options

VPN

TU Graz

KFU Graz

your_cred@vpn.tugraz.at

matlab server

computer room

matlab server

your_TUGacc@matlab.tugraz.at

experienced students: transfer, open, and run matlab scripts

# GNU Octave

**Download @**

**https://www.gnu.org/software/octave/**

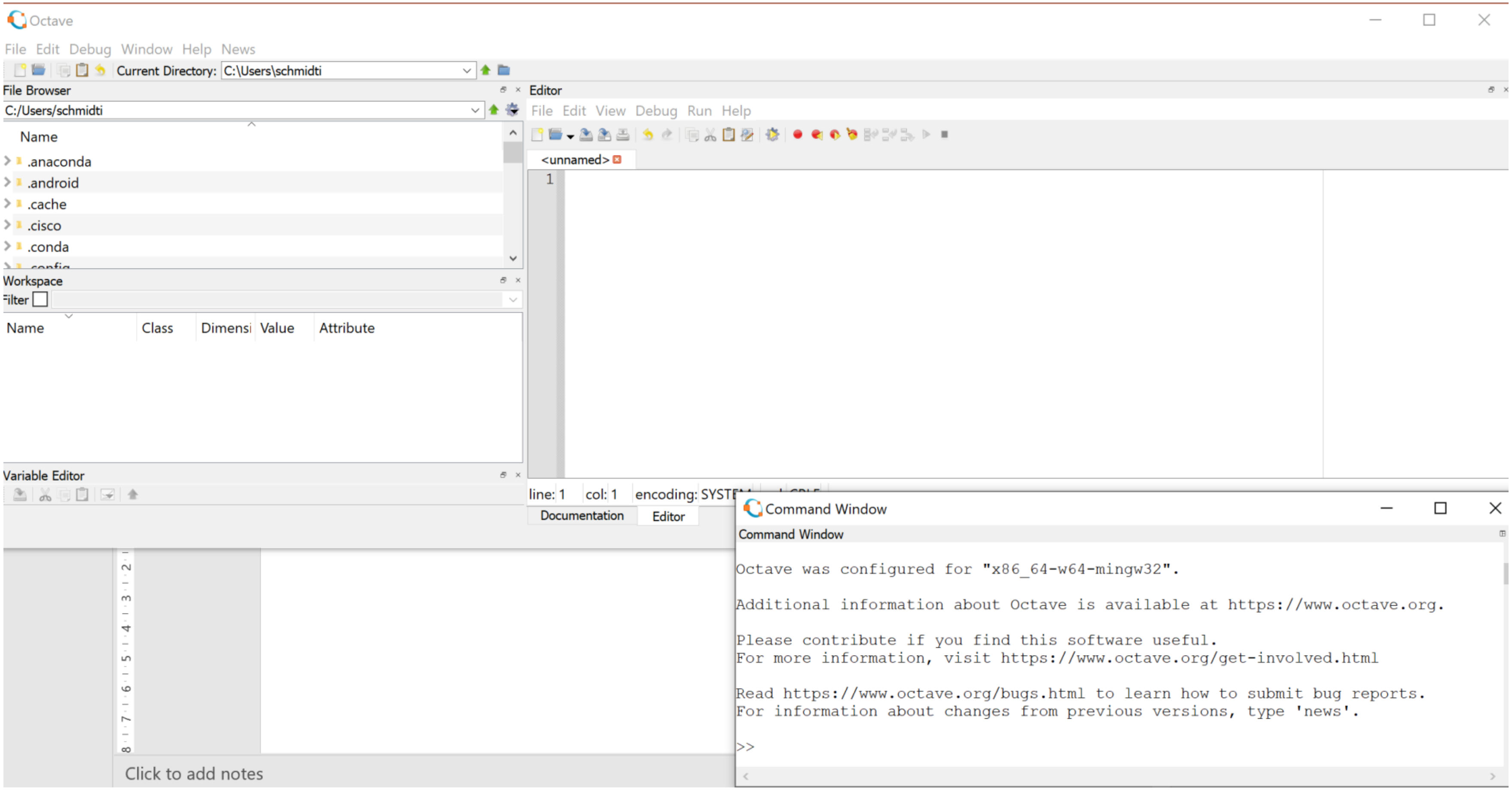
**Linux**

**Windows**

**https://www.youtube.com/watch?v=6ARUo9KglqM**

**Suitable for beginners: Launch Octave GUI (graphical user interface)**

## Launch Octave GUI: work bench



```
Octave was configured for "x86_64-w64-mingw32".

Additional information about Octave is available at https://www.octave.org.

Please contribute if you find this software useful.
For more information, visit https://www.octave.org/get-involved.html

Read https://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.

>>
```

# Python

**scripts: routinely run in Python 3.7.3**

**with required packages: numpy, matplotlib, scipy, seaborn**

**purists: Python 3 distribution with required packages + editor of choice**

**for a swift and low maintenance start:**

**Python within Anaconda and Spyder editor**

**https://www.youtube.com/watch?v=saFio54AEUw**

**https://www.youtube.com/watch?v=ou65T_mC8Z8**

**Anaconda contains also Jupyter notebooks**

# Setting up a list of logarithmically spaced numbers

## MATLAB

## GNU Octave

```
N_dop = logspace(13,25,100); %used dopant densities
```

y = logspace(a,b,n) generates n points between decades 10^a and 10^b.

## Python

```
    dopmin = 1e18
    dopmax = 1e25
    N_steps = 500    # Number of steps between dopmin and dopmax
```

either

```
    doping_densities_exp = np.linspace(np.log10(dopmin),np.log10(dopmax),N_steps)
    doping_densities = np.power(10, doping_densities_exp)
```

or

```
    doping_densities = np.logspace(dopmin, dopmax,num = N_steps)
```

numpy.logspace(start, stop, num=50, endpoint=True, base=10.0, dtype=None, axis=0)
Return numbers spaced evenly on a log scale for a base. Default is base = 10, i.e., one gets N_steps numbers between 10^start and 10^end