

[Table of contents \(../toc.ipynb\)](#)

# Tools for Python developers



- Upon now, you used either python through terminal or some cloud service like [Python anywhere \(https://www.pythonanywhere.com/try-ipython/\)](https://www.pythonanywhere.com/try-ipython/), or the respective jupyter notebooks for this class on [Binder \(https://mybinder.org/v2/gh/StephanRhode/py-algorithms-4-automotive-engineering/master\)](https://mybinder.org/v2/gh/StephanRhode/py-algorithms-4-automotive-engineering/master).
- This chapter introduces an integrated development environment (Pycharm) and local interactive notebooks (Jupyter).

# Pycharm



- Pycharm is one possible integrated development environment (IDE) for Python.
- An IDE is basically a program which supports all steps of software development like
  - Coding with code completion, static code analyses, templates,...
  - Testing
  - Debugging
  - Version control
  - Environment and package management
  - Code refactoring
  - Build chains,...
  - Please find [here all features \(https://www.jetbrains.com/pycharm/features/\)](https://www.jetbrains.com/pycharm/features/).

- Other popular choices next to Pycharm are [Visual Studio Code](https://code.visualstudio.com/) (<https://code.visualstudio.com/>), and [Spyder](https://www.spyder-ide.org/) (<https://www.spyder-ide.org/>). Just try which one is best for you.
- There is a free community edition of Pycharm [link to installer](https://www.jetbrains.com/pycharm/download) (<https://www.jetbrains.com/pycharm/download>).
- And as Pycharm is a professional software, there is much [video recorded training material](https://www.jetbrains.com/pycharm/learning-center/) (<https://www.jetbrains.com/pycharm/learning-center/>).

## Pycharm introduction video

Please find here a [video which presents Pycharm features \(https://www.youtube.com/watch?v=BPC-bGdBSM8\)](https://www.youtube.com/watch?v=BPC-bGdBSM8).

# A small Pycharm live demo

The screenshot displays the PyCharm IDE interface for a project named "py-algorithms-4-automotive-engineering". The main editor shows the "test\_notebooks.py" file, which contains a list of objectives for a course. The sidebar on the right lists the following objectives:

- Anaconda, Pycharm, Jupyter
- NumPy, Matplotlib, SymPy, Scikit-Learn
- Methods and tools for creating software
- Version management GitHub, git
- Testing software pytest, Pylint
- Documentation Sphinx
- Continuous Integration (CI) Travis CI
- Workflows in Open Source and Inner Source, Kanban, Scrum
- Practical programming projects to:
- Road sign recognition
- Vehicle state estimation
- Calibration of vehicle models by mathematical optimization
- Data-based modelling of the powertrain of an electric vehicle

The bottom panel shows the "Version Control" tab, which displays a list of files and their status. The "test\_notebooks.py" file is highlighted, and the "Unified viewer" shows the code for the "test\_00\_intro" function, which is a pytest parametrized test.

```
except CellExecutionError:
    raise

@pytest.mark.parametrize("notebook", ["00_intro", "01_py-installation"])
@pytest.mark.parametrize("notebook", ["00_intro", "01_py-installation"])

def test_00_intro(notebook):
    execute_notebook(notebook_file="00_intro/" + notebook + ".ipynb")

@pytest.mark.parametrize("notebook", ["00_syntax", "01_semantics", "02_data-types", "03_conditions-and-loops", "04_functions"])
```

# Exercise: First steps in Pycharm (15 minutes)



Please complete the following tasks:

- Install Pycharm from <https://www.jetbrains.com/pycharm/download> (<https://www.jetbrains.com/pycharm/download>)
- Create a new project
- Create an environment (pip, conda)

- Add this Python code as file

```
def my_hello():  
    print("Hello world")
```

- Execute this file in Pycharm

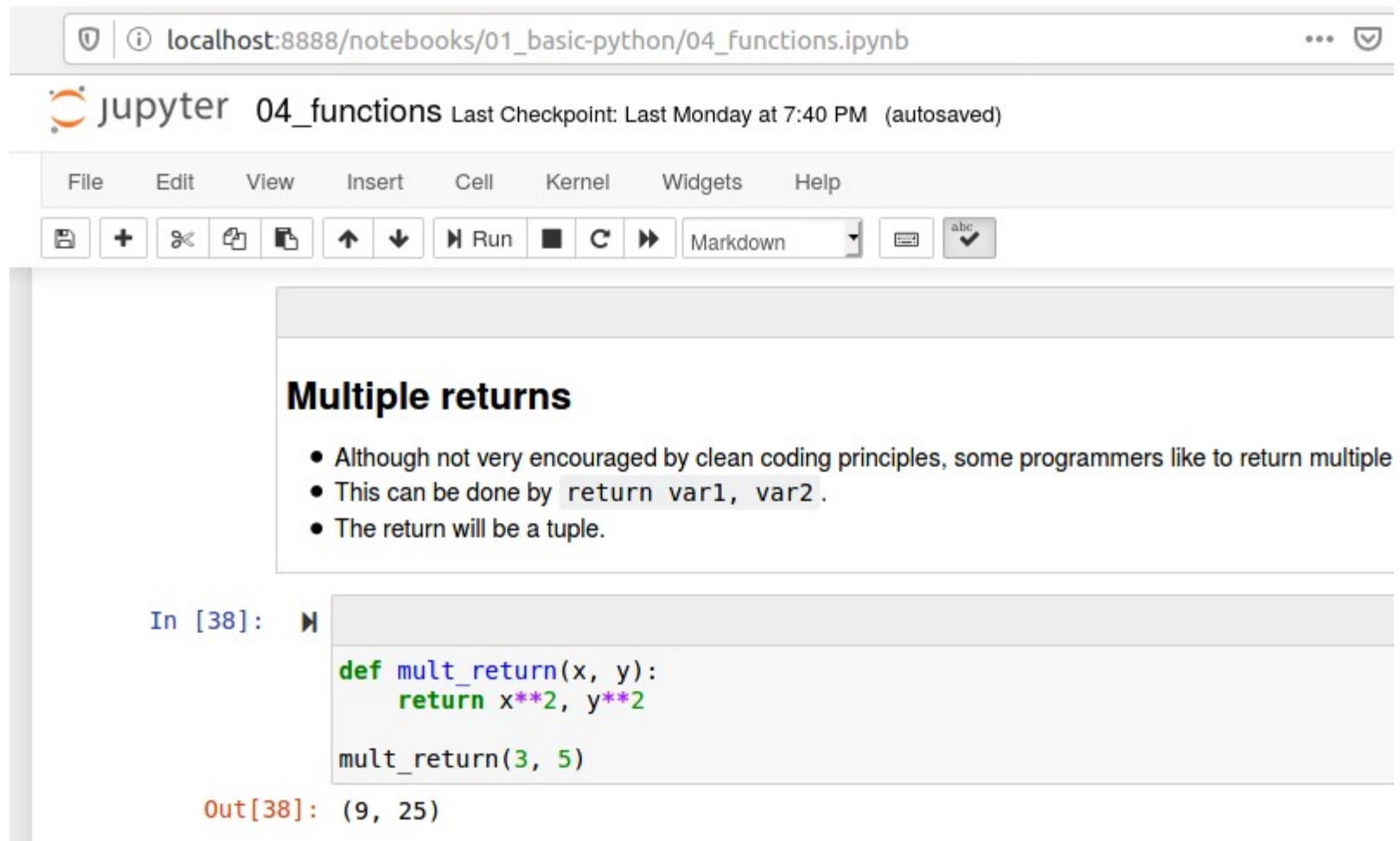
# Jupyter



- Jupyter is an interactive programming environment, where you can combine programming, presentation of results, and explanation with text and equations in one web page.
- Hence Jupyter is a way to communicate scientific computing like it is done since ages. Leonardo da Vinci used Notebooks as well!
- Jupyter works with Python kernel, as well as with Julia, R, Rubi, Matlab.
- Relatively recent, [Jupyter Lab \(https://jupyterlab.readthedocs.io/en/stable/\)](https://jupyterlab.readthedocs.io/en/stable/) was released, which is the successor of Jupyter.
- Note the entire course material is written in Jupyter. It is very convenient :)

# Jupyter installation

- Jupyter is a package. Hence, just type `conda install jupyter` to extend your environment.
- You can also try Jupyter in the cloud here <https://jupyter.org/try> (<https://jupyter.org/try>).
- The command to start the Jupyter notebook server is `jupyter notebook`.



The screenshot shows a Jupyter Notebook interface in a web browser. The address bar displays `localhost:8888/notebooks/01_basic-python/04_functions.ipynb`. The notebook title is `04_functions`, and it indicates the last checkpoint was saved on Monday at 7:40 PM. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for saving, adding cells, undo, redo, and running code. The current cell is a markdown cell with the title **Multiple returns** and three bullet points:

- Although not very encouraged by clean coding principles, some programmers like to return multiple
- This can be done by `return var1, var2`.
- The return will be a tuple.

Below the markdown cell is a code cell labeled `In [38]:` containing the following Python code:

```
def mult_return(x, y):  
    return x**2, y**2  
  
mult_return(3, 5)
```

The output of the code cell is displayed as `Out[38]: (9, 25)`.





## Some Jupyter commands

- There is edit mode and navigation mode. You can switch between them with `enter` and `esc`.
- `shift + enter` runs a cell and selects the next cell below.
- If you use functions, you can use auto completion with `tab` or read the doc string with `shift + tab`.
- Many more keyboard shortcuts are on top of Jupyter panel in the keyboard icon.
- Add to this, there are some so called magic commands, which start with `%`. Quite common is for instance `%matplotlib notebook`, which embeds plots.

# Jupyter tutorial

- There is a great Jupyter tutorial on Binder. Hence, please consult this tutorial to learn Jupyter.
- <https://hub.mybinder.turing.ac.uk/user/ipython-ipython-in-depth-zgalw2wh/notebooks/binder/Index.ipynb> (<https://hub.mybinder.turing.ac.uk/user/ipython-ipython-in-depth-zgalw2wh/notebooks/binder/Index.ipynb>)

# Exercise: Jupyter (10 minutes)



Here the task:

- Activate your local Python environment and install jupyter with `conda install jupyter`.
- Open the notebook server with `jupyter notebook` command.
- Create a new notebook and try some Python code there.
- Add text in markdown cells.

# Jupyter extensions

There are many extensions for Jupyter notebooks like

- table of contents bar,
- variable inspector,
- spell checkers,
- auto code style checks,... available in [jupyter-contrib-nbextensions \(https://jupyter-contrib-nbextensions.readthedocs.io/en/latest/index.html\)](https://jupyter-contrib-nbextensions.readthedocs.io/en/latest/index.html).

With these extensions, Jupyter becomes a very powerful interactive development environment.

Please find here a screen shot of Jupyter with table of contents and variable inspector extension.

jupyter 02\_data-types Last Checkpoint: a minute ago (unsaved changes)

File Edit View Insert Cell Kernel Navigate Widgets Help

Run Code

### Contents

- 1 Built in data types and structures
  - 1.1 Operations with strings
  - 1.2 Numbers
    - 1.2.1 Integers
    - 1.2.2 Floats
    - 1.2.3 Fixed precision of floating point numbers
    - 1.2.4 Floats take aways
    - 1.2.5 Complex numbers
  - 1.3 Lists
    - 1.3.1 List comprehensions
  - 1.4 Tuple
  - 1.5 Set
  - 1.6 Dictionaries
  - 1.7 Exercise: Try dictionaries (15 min)
  - 1.8 Solution

### Variable Inspector

X	Name	Type	Size	Value
x	a	float	24	0.1
x	b	float	24	0.2
x	my_string	str	63	Say some

### 1.2.3 Fixed precision of floating point numbers

- Remember that numbers are actually stored in binary form in computers.
- The precision of floating point numbers is always limited, due to fixed number
- This drawback of floating point numbers is not Python specific, it will also occ

```
In [5]: # let's look a bit deeper in floats
a = 0.1
b = 0.2
# print a with 21 digits
print("{0:.21f}".format(a))
print("{0:.21f}".format(b))

0.100000000000000005551
0.200000000000000011102
```

```
In [6]: # Hence, tests like this will fail
0.1 + 0.2 == 0.3

Out[6]: False
```