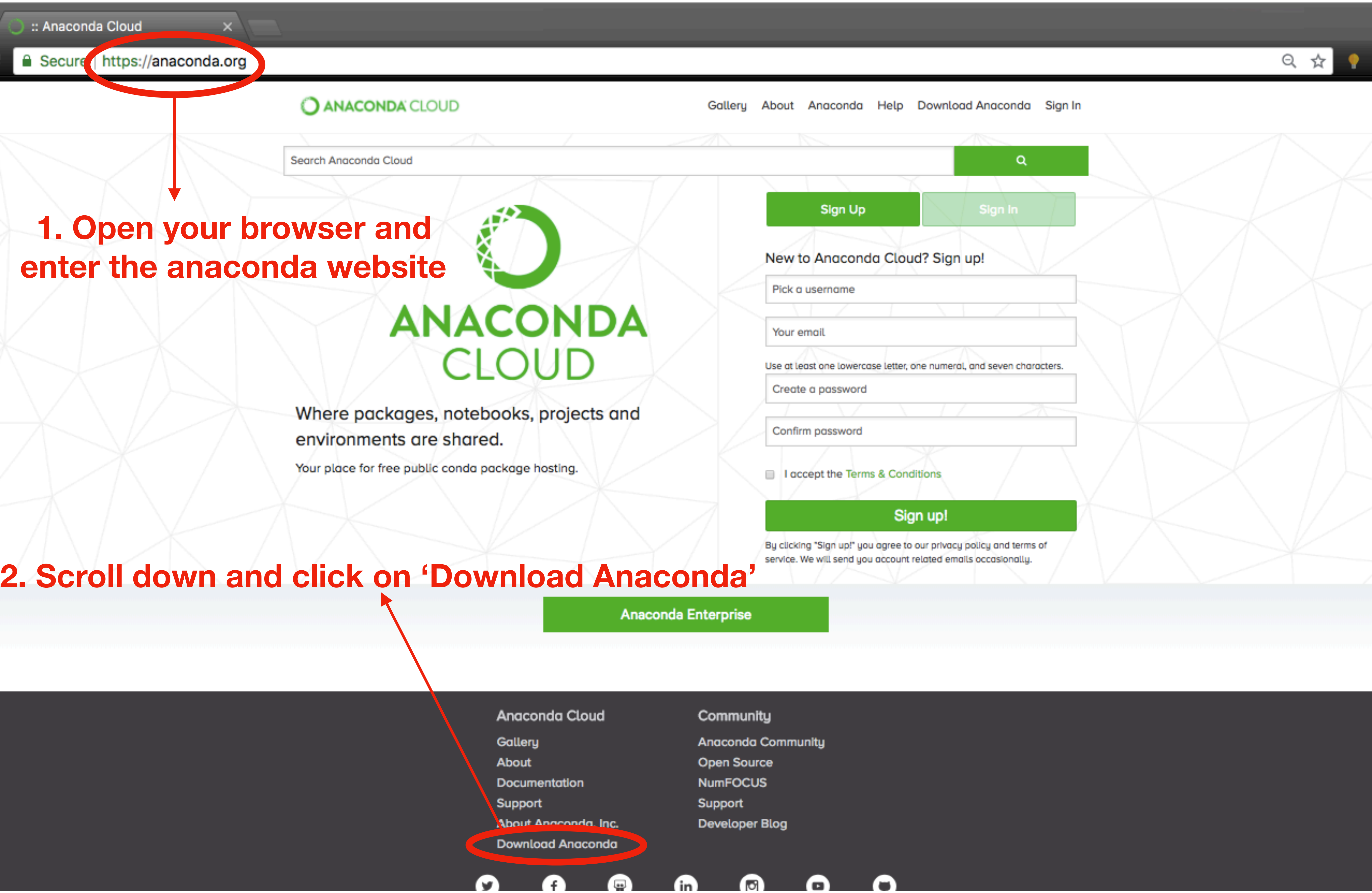


1. Open your browser and enter the anaconda website

ANACONDA CLOUD

Where packages, notebooks, projects and environments are shared.

Your place for free public conda package hosting.

New to Anaconda Cloud? Sign up!

Pick a username

Your email

Use at least one lowercase letter, one numeral, and seven characters.

Create a password

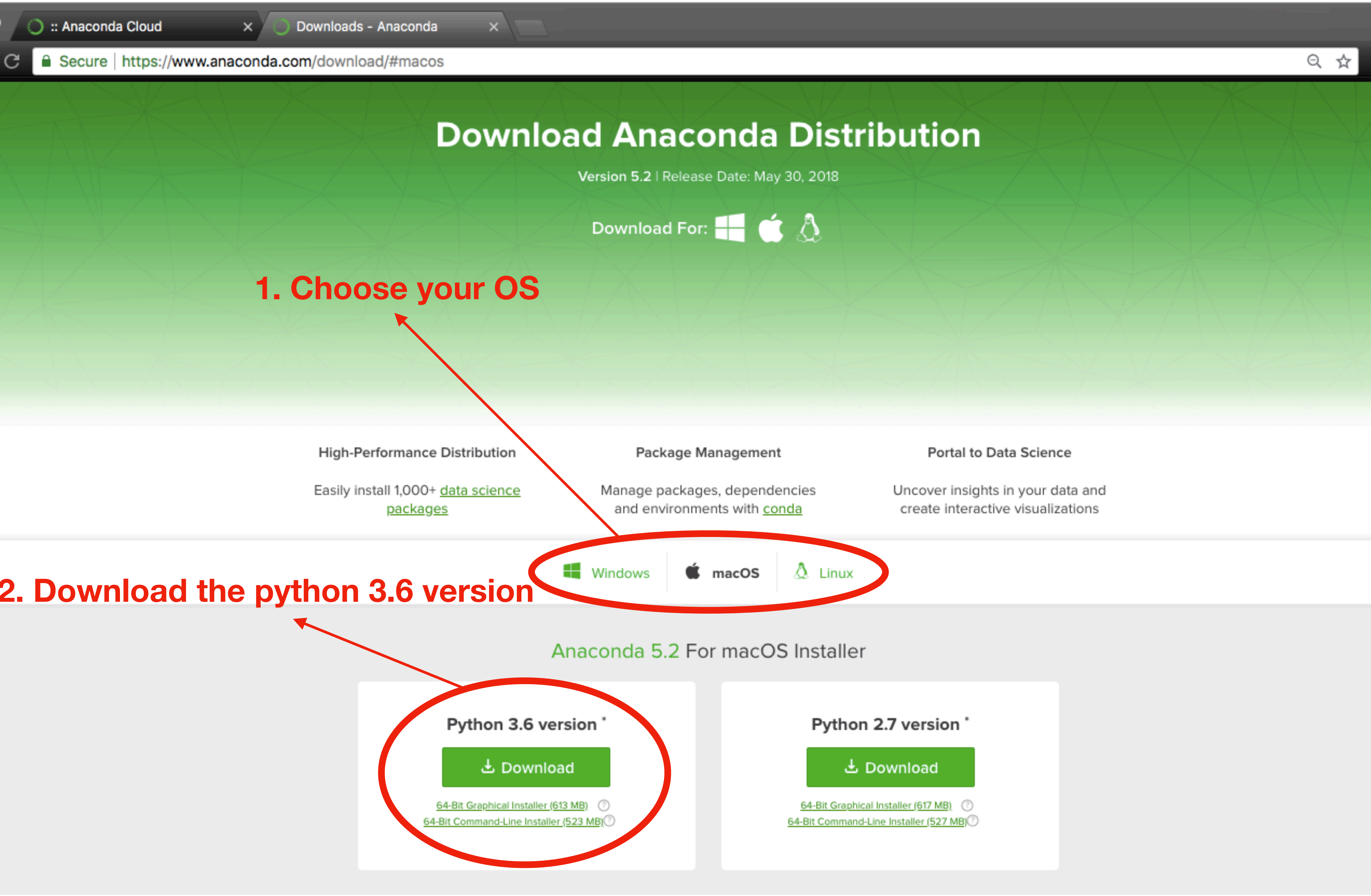
Confirm password

□ I accept the Terms & Conditions

Sign up!

By clicking "Sign up!" you agree to our privacy policy and terms of service. We will send you account related emails occasionally.

2. Scroll down and click on ‘Download Anaconda’



Download Anaconda Distribution

Version 5.2 | Release Date: May 30, 2018

Download For:

1. Choose your OS

High-Performance Distribution

Easily install 1,000+ data science packages

Package Management

Manage packages, dependencies and environments with conda

Portal to Data Science

Uncover insights in your data and create interactive visualizations

2. Download the python 3.6 version

Anaconda 5.2 For macOS Installer

Python 3.6 version *

64-Bit Graphical Installer (613 MB)
64-Bit Command-Line Installer (523 MB)

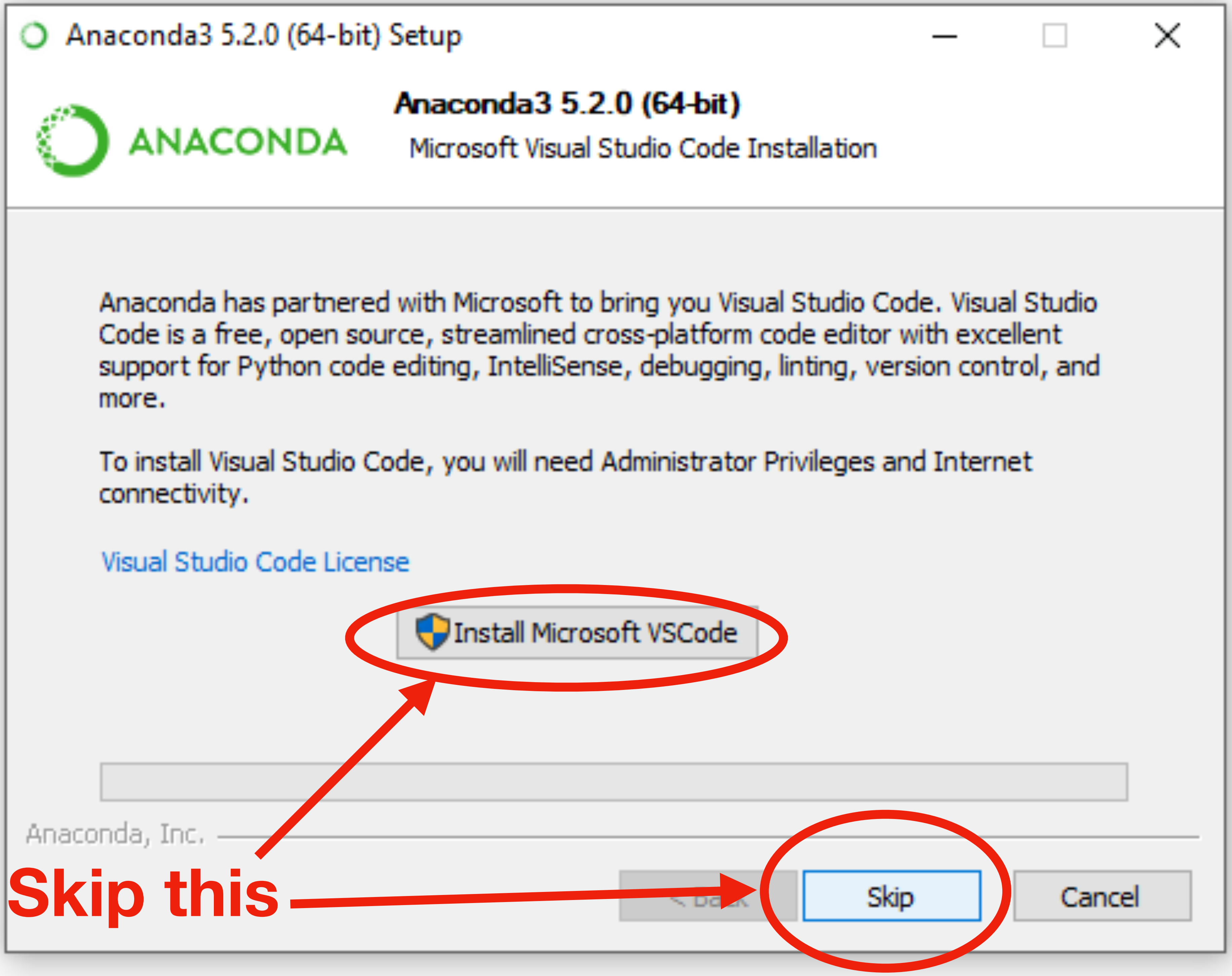
Python 2.7 version *

64-Bit Graphical Installer (617 MB)
64-Bit Command-Line Installer (527 MB)

- Depending on your operating system, a .sh (linux), .pkg (mac OS) or .exe (Windows) will be downloaded.

- Execute the respective files and follow the installation instructions

- When installing anaconda, skip the last part where it suggests you to install Microsoft VSCode



Anaconda3 5.2.0 (64-bit) Setup

Anaconda3 5.2.0 (64-bit)

Microsoft Visual Studio Code Installation

Anaconda has partnered with Microsoft to bring you Visual Studio Code. Visual Studio Code is a free, open source, streamlined cross-platform code editor with excellent support for Python code editing, IntelliSense, debugging, linting, version control, and more.

To install Visual Studio Code, you will need Administrator Privileges and Internet connectivity.

Visual Studio Code License

Anaconda, Inc.

- Mac OS and Linux:

1. Go into the folder where you saved the jupyter notebook (.ipynb data format)

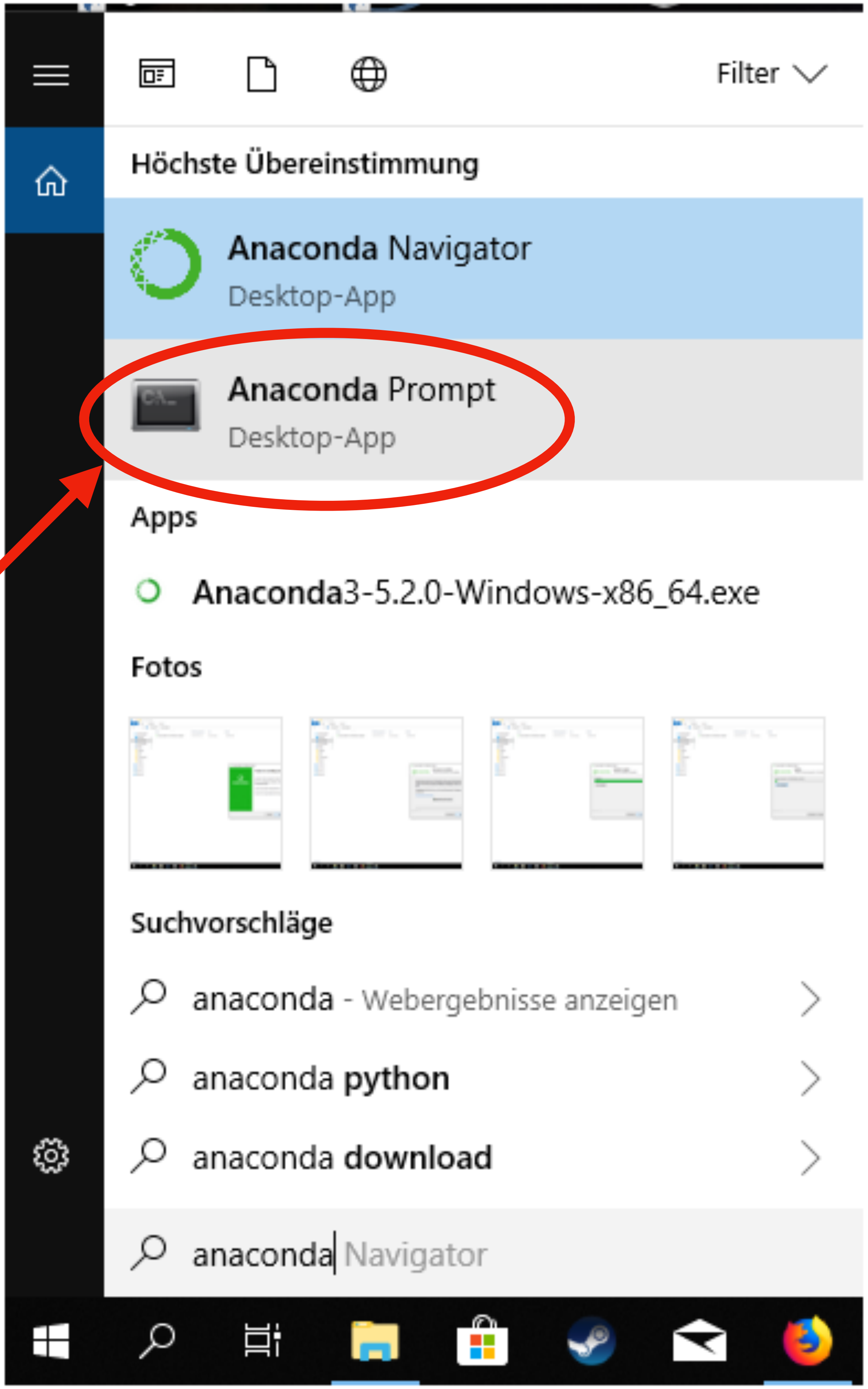
2. Open the terminal

3. Type: ‘jupyter notebook’

4. A window in your web browser will open where you can click on the notebooks or create new ones (see slide 9)

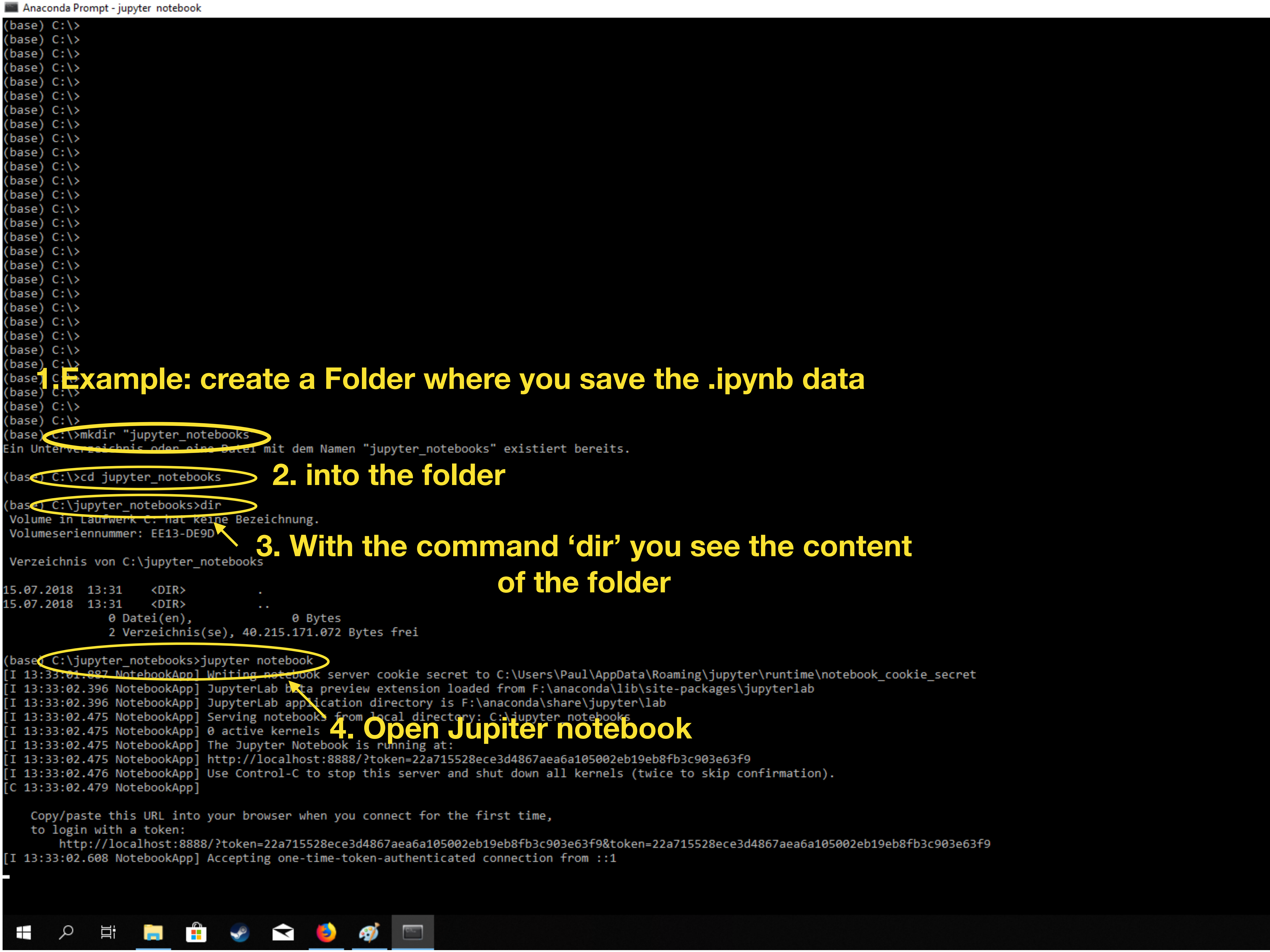
- Windows OS:

1. Open the Windows menu bar
2. Search for: ‘anaconda promt’
3. A terminal will open
4. Switch into the folder you saved the .ipynb files with ‘cd FOLDER’
5. Type: ‘jupyter notebook’



Filter
Höchste Übereinstimmung
Anaconda Navigator
Desktop-App
Anaconda Prompt
Desktop-App
Apps
Anaconda3-5.2.0-Windows-x86_64.exe
Fotos
Suchvorschläge
anaconda - Webergebnisse anzeigen
anaconda python
anaconda download
anaconda Navigator

Open



Anaconda Prompt - jupyter notebook

```
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
(base) C:\>
```

1.Example: create a Folder where you save the .ipynb data

```
(base) C:\>mkdir "jupyter_notebooks
Ein Unterverzeichnis oder eine Datei mit dem Namen "jupyter_notebooks" existiert bereits.
```

```
(base) C:\>cd jupyter_notebooks
```

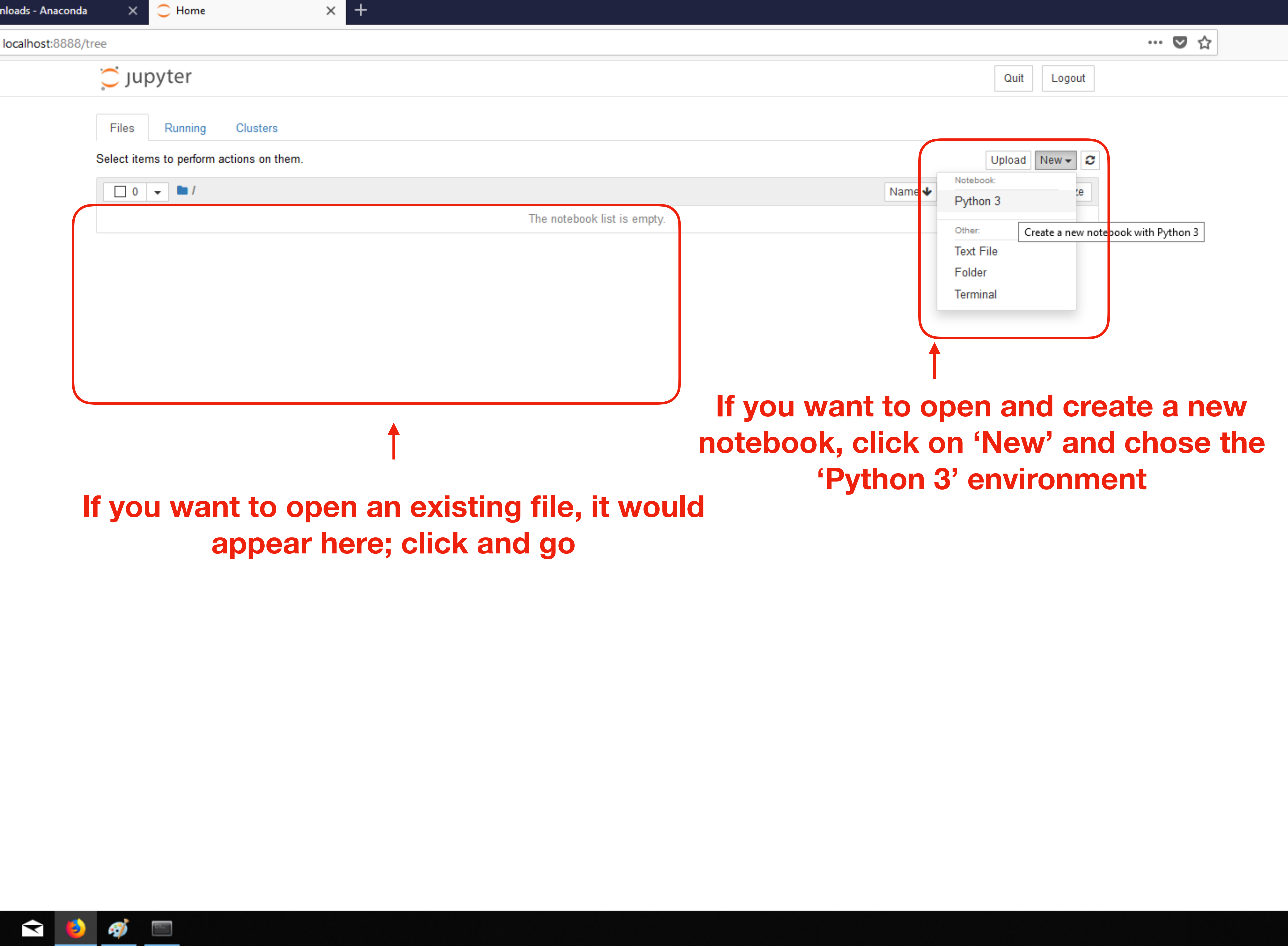
2. into the folder

```
(base) C:\jupyter_notebooks>dir
 Volume in Laufwerk C: hat keine Bezeichnung.
 Volumeseriennummer: EE13-DE9D
```

3. With the command ‘dir’ you see the content of the folder

```
 Verzeichnis von C:\jupyter_notebooks

15.07.2018  13:31    <DIR>          .
15.07.2018  13:31    <DIR>          ..
               0 Datei(en),              0 Bytes
               2 Verzeichnis(se), 40.215.171.072 Bytes frei

(base) C:\jupyter_notebooks>jupyter notebook
[I 13:33:01.887 NotebookApp] Writing notebook server cookie secret to C:\Users\Paul\AppData\Roaming\jupyter\runtime\notebook_cookie_secret
[I 13:33:02.396 NotebookApp] JupyterLab beta preview extension loaded from F:\anaconda\lib\site-packages\jupyterlab
[I 13:33:02.396 NotebookApp] JupyterLab application directory is F:\anaconda\share\jupyter\lab
[I 13:33:02.475 NotebookApp] Serving notebooks from local directory: C:\jupyter_notebooks
[I 13:33:02.475 NotebookApp] 0 active kernels
[I 13:33:02.475 NotebookApp] The Jupyter Notebook is running at:
[I 13:33:02.475 NotebookApp] http://localhost:8888/?token=22a715528ece3d4867aea6a105002eb19eb8fb3c903e63f9
[I 13:33:02.476 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 13:33:02.479 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://localhost:8888/?token=22a715528ece3d4867aea6a105002eb19eb8fb3c903e63f9&token=22a715528ece3d4867aea6a105002eb19eb8fb3c903e63f9
[I 13:33:02.608 NotebookApp] Accepting one-time-token-authenticated connection from ::1
```

4. Open Jupiter notebook



Select items to perform actions on them.

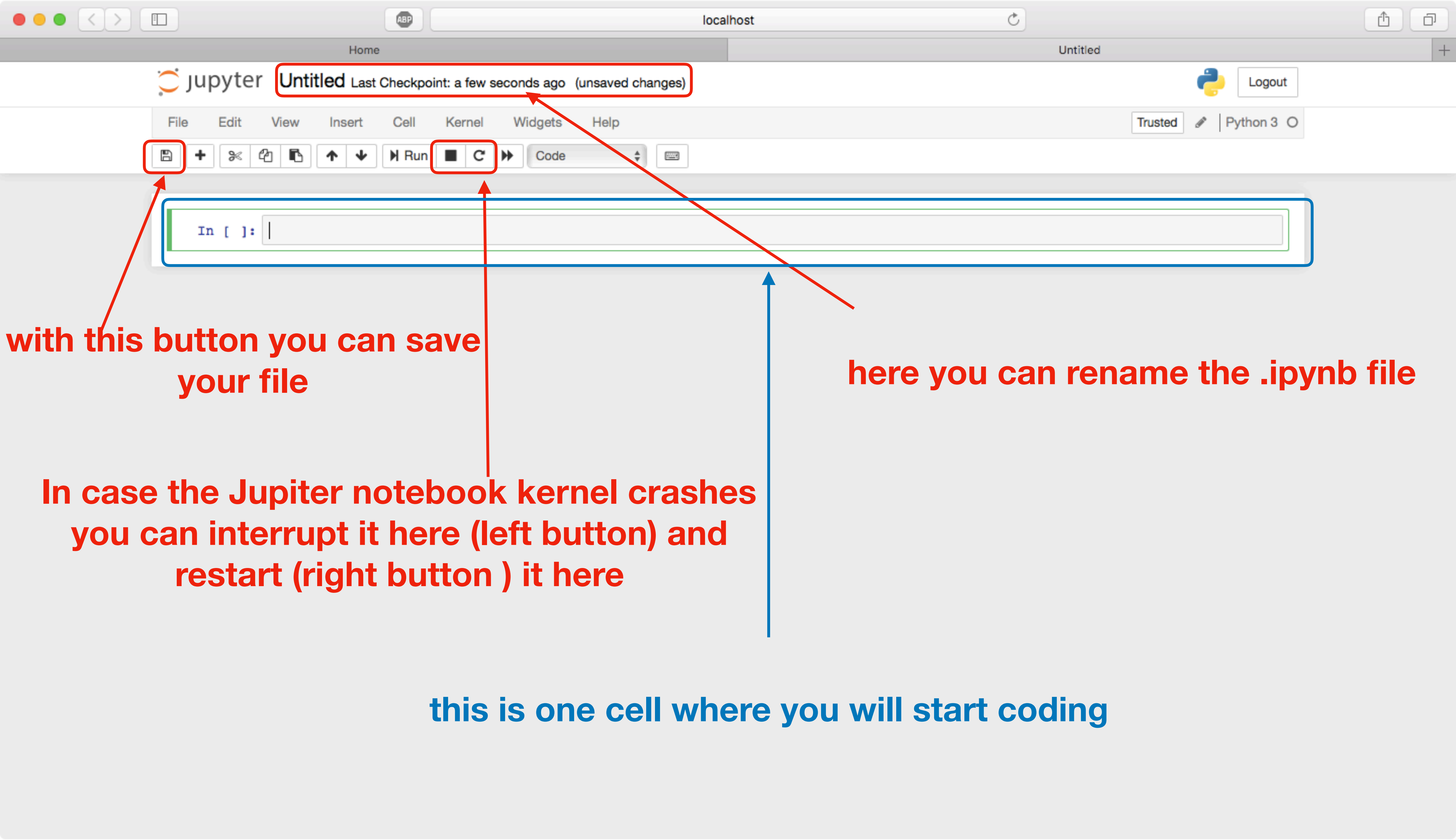
The notebook list is empty.

Create a new notebook with Python 3

If you want to open and create a new notebook, click on ‘New’ and chose the ‘Python 3’ environment

If you want to open an existing file, it would appear here; click and go

a new notebook - navigation

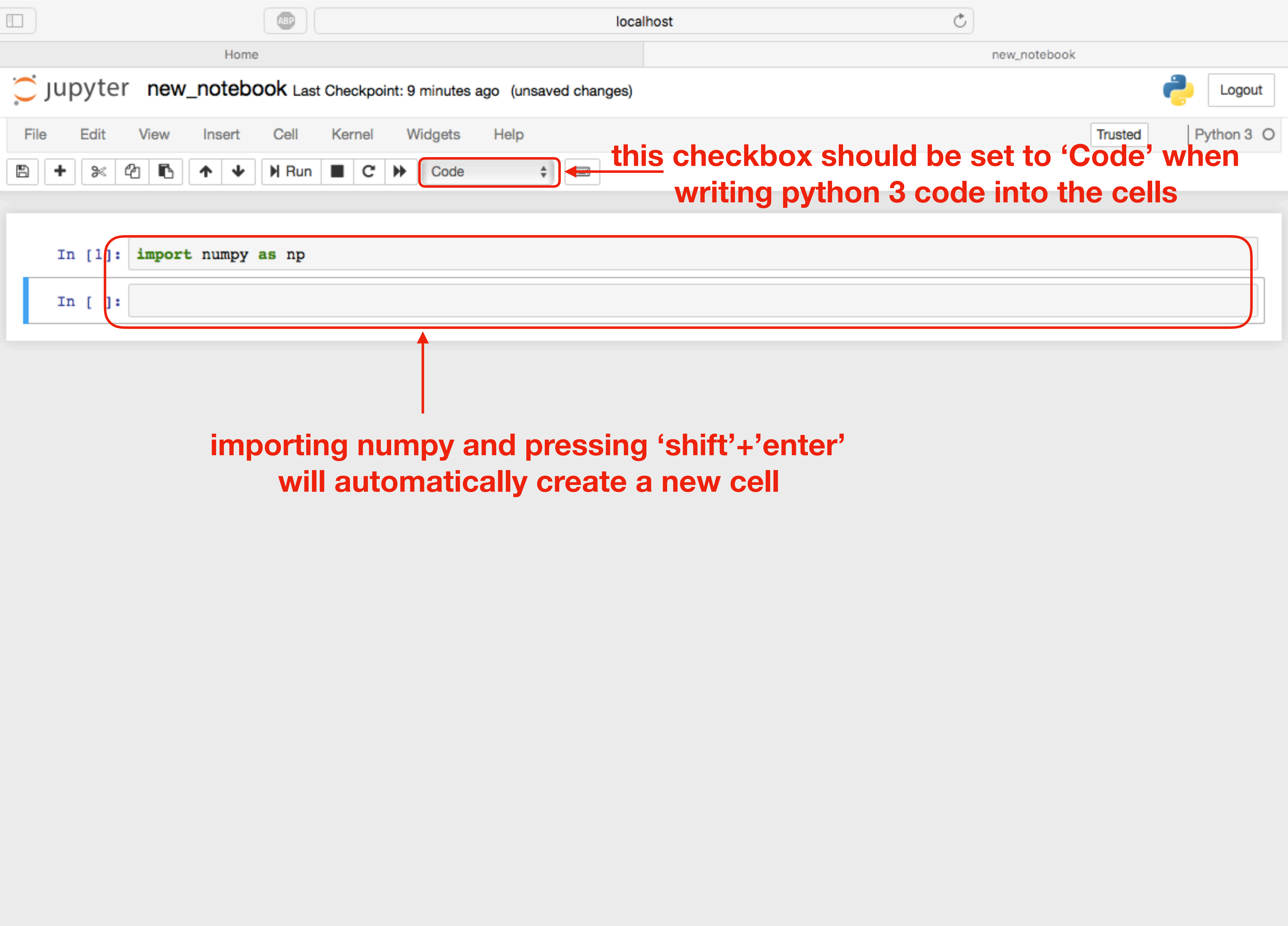


with this button you can save your file

here you can rename the .ipynb file

In case the Jupiter notebook kernel crashes you can interrupt it here (left button) and restart (right button) it here

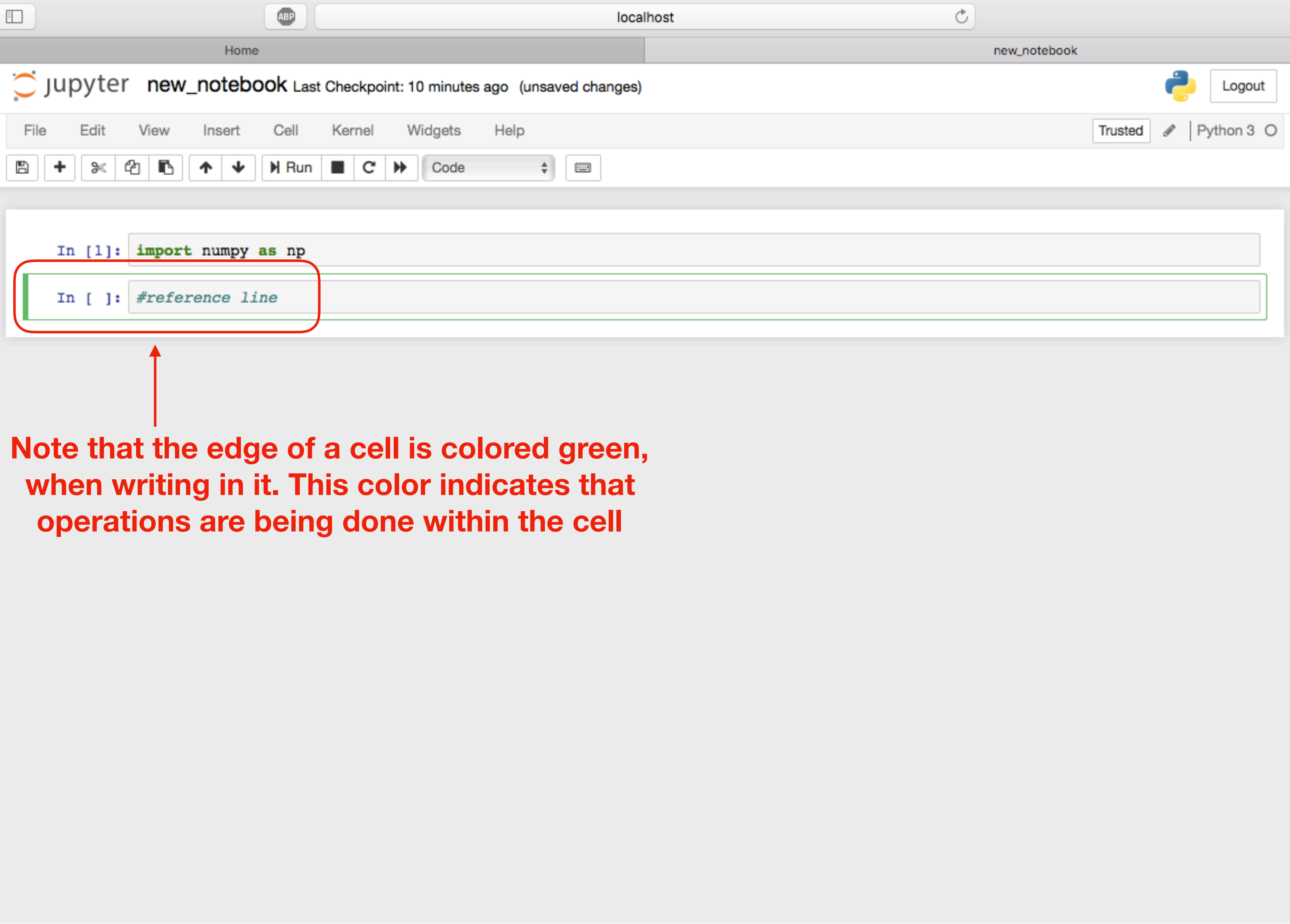
this is one cell where you will start coding



this checkbox should be set to 'Code' when writing python 3 code into the cells

```
In [1]: import numpy as np
In [ ]:
```

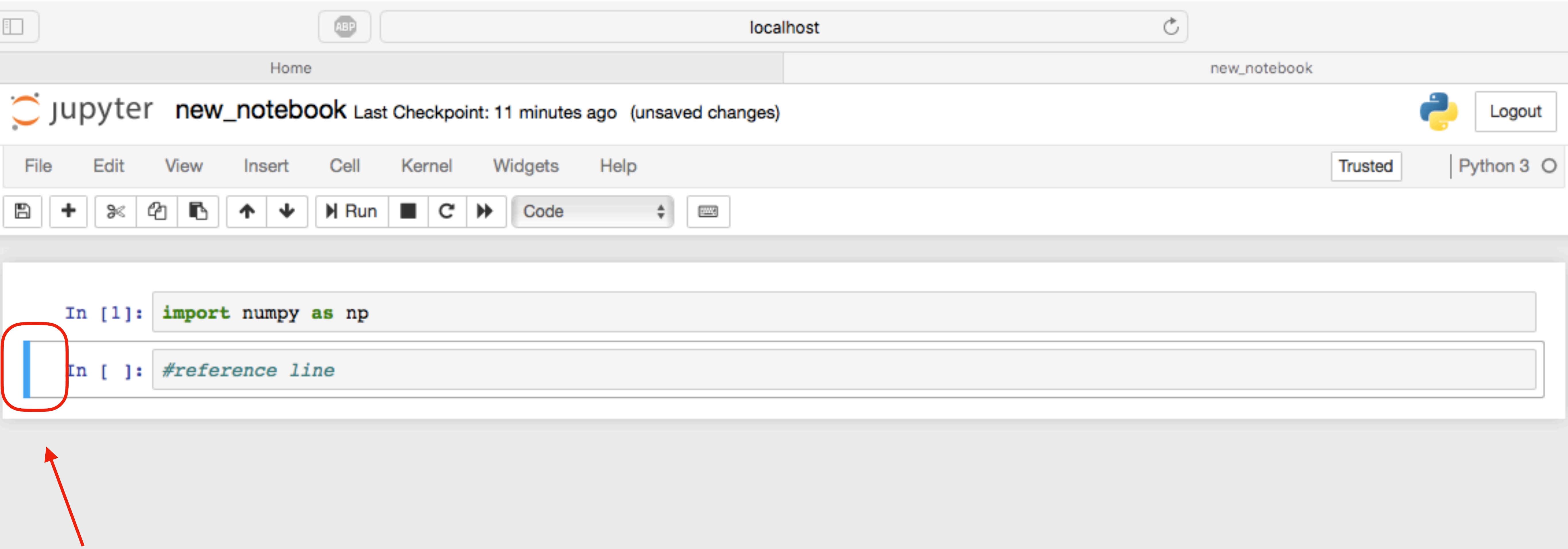
importing numpy and pressing 'shift'+'enter' will automatically create a new cell



```
In [1]: import numpy as np
```

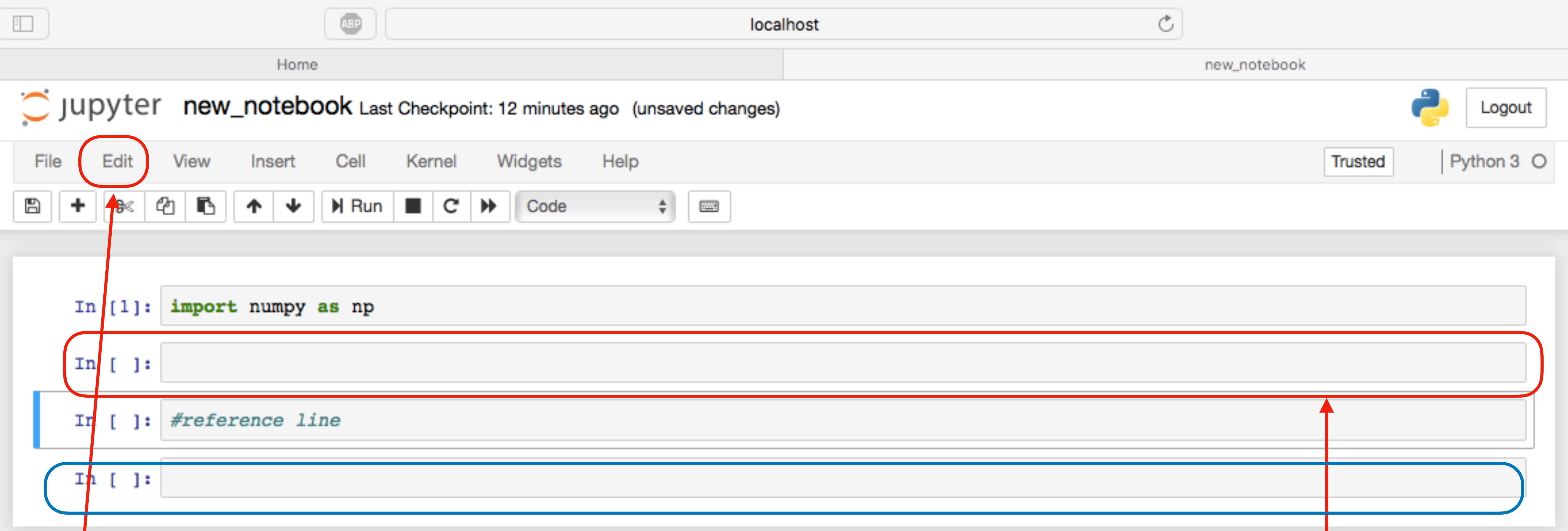
```
In [ ]: #reference line
```

Note that the edge of a cell is colored green, when writing in it. This color indicates that operations are being done within the cell



```
In [1]: import numpy as np

In [ ]: #reference line
```

By pressing into the the line on the left of the cell, the manipulation within the cell is deactivated. Note how the color of the edge turns blue. This color indicates that not operations, regarding the cells itself can be done. Also note the mark of the line as ‘reference line’ in the following slide.



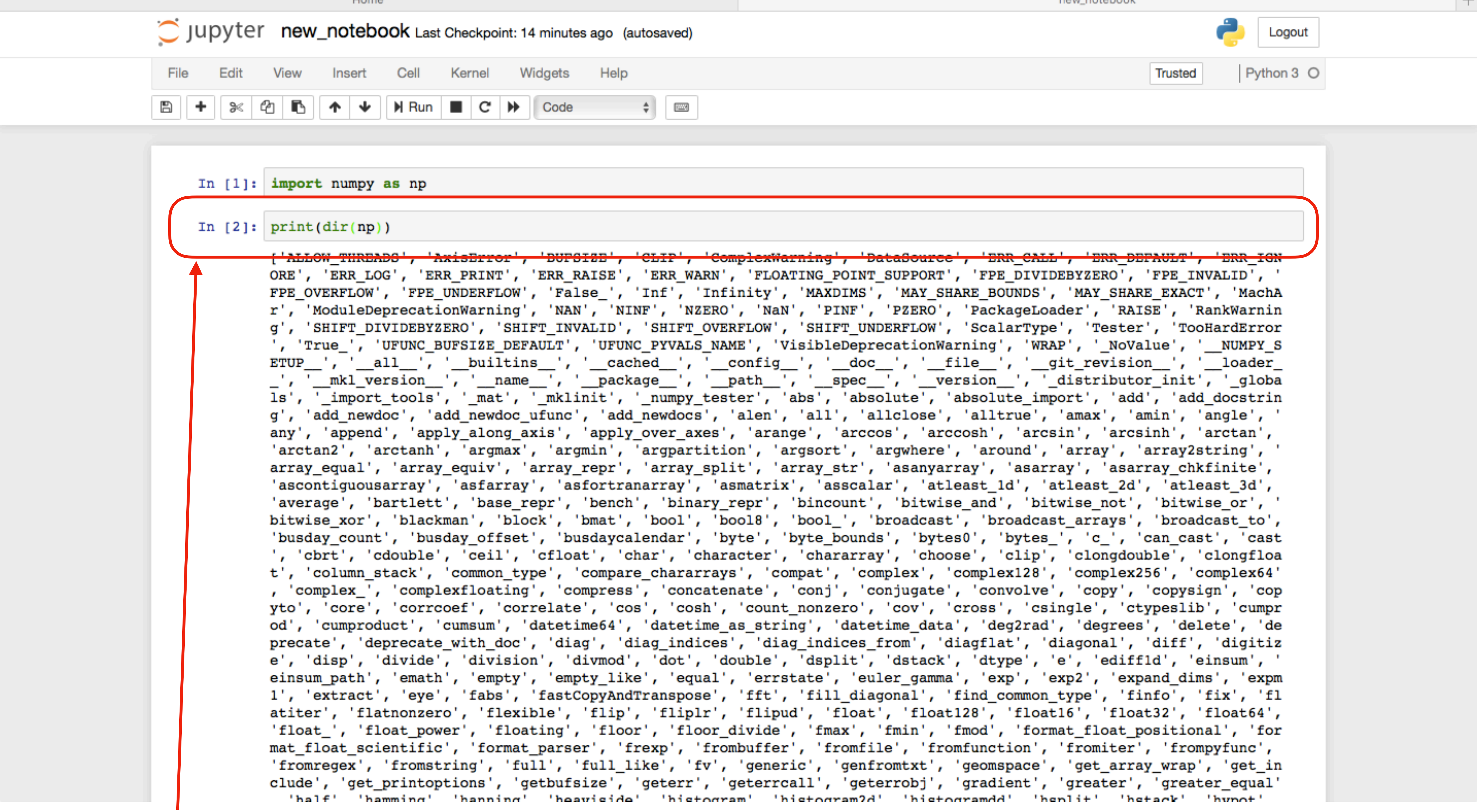
```
In [1]: import numpy as np

In [ ]:

In [ ]: #reference line

In [ ]:
```

By pressing ‘b’ on the keyboard, a new line below the reference line is created

By pressing ‘a’ on the keyboard, a new line on top the reference line is created

Note that the box around the reference line is still marked as blue, therefore, if you want to work in one cell you created, click into it, such that the box around it becomes green. If you want to perform cell operations with respect to the new cells, activate the respective cell, sah that the box around it appears blue, and do as you wish.

By pressing ‘x’ and a cell, where cell operations are activated (blue edge), the cell can be deleted. If you accidentally delete a cell and want to undo it, click on edit and ‘Undo Delete Cells’. This will restore the deleted cell.

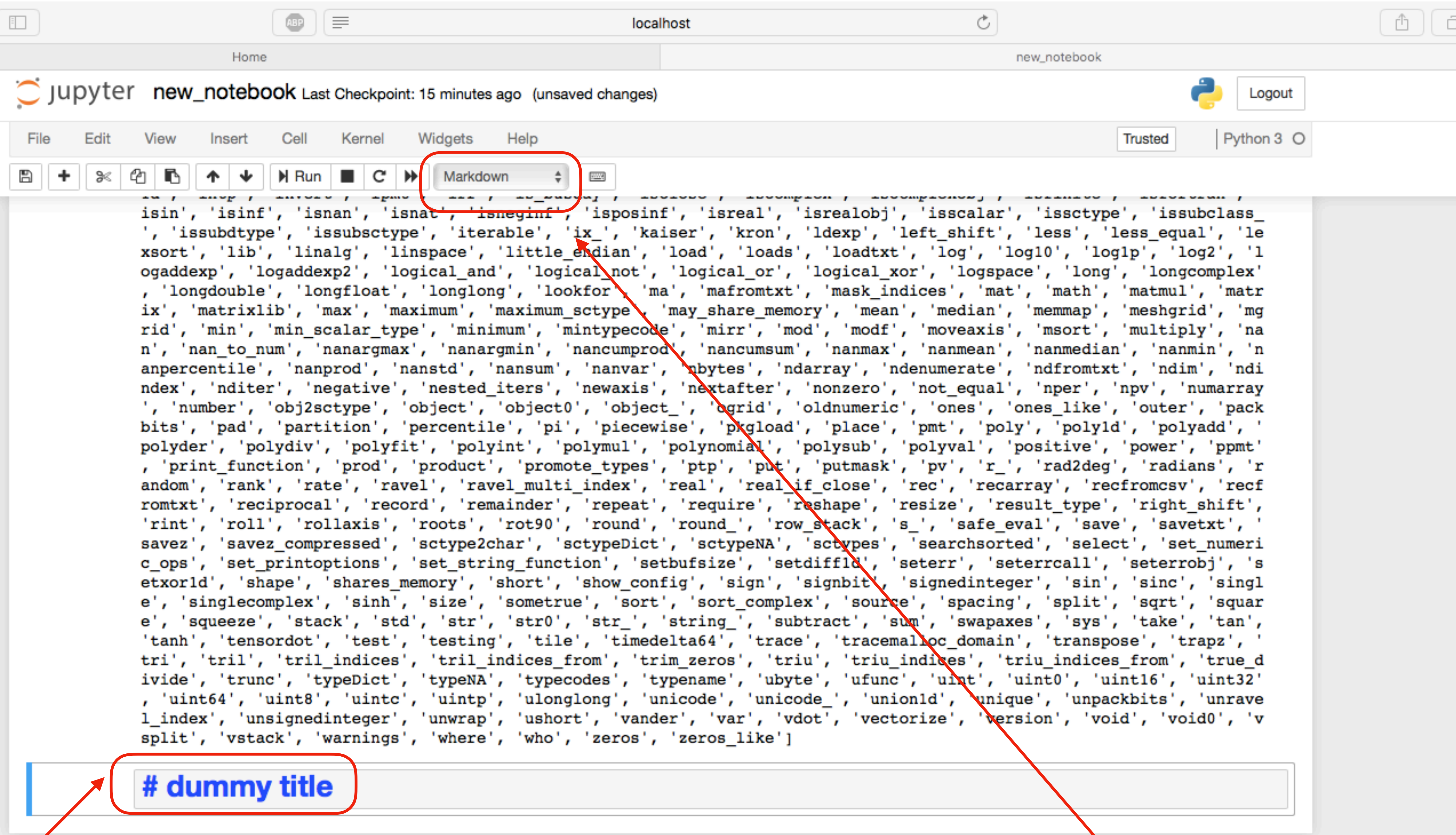


```
In [1]: import numpy as np
```

```
In [2]: print(dir(np))
```

```
['ALLOW_THREADS', 'AxisError', 'BUFSIZE', 'CLIP', 'ComplexWarning', 'DataSource', 'ERR_CALL', 'ERR_DEFAULT', 'ERR_IGN
ORE', 'ERR_LOG', 'ERR_PRINT', 'ERR_RAISE', 'ERR_WARN', 'FLOATING_POINT_SUPPORT', 'FPE_DIVIDEBYZERO', 'FPE_INVALID', '
FPE_OVERFLOW', 'FPE_UNDERFLOW', 'False_', 'Inf', 'Infinity', 'MAXDIMS', 'MAY_SHARE_BOUNDS', 'MAY_SHARE_EXACT', 'MachA
r', 'ModuleDeprecationWarning', 'NAN', 'NINF', 'NZERO', 'NaN', 'PINF', 'PZERO', 'PackageLoader', 'RAISE', 'RankWarnin
g', 'SHIFT_DIVIDEBYZERO', 'SHIFT_INVALID', 'SHIFT_OVERFLOW', 'SHIFT_UNDERFLOW', 'ScalarType', 'Tester', 'TooHardError
', 'True_', 'UFUNC_BUFSIZE_DEFAULT', 'UFUNC_PYVALS_NAME', 'VisibleDeprecationWarning', 'WRAP', '_NoValue', '__NUMPY_S
ETUP__', '__all__', '__builtins__', '__cached__', '__config__', '__doc__', '__file__', '__git_revision__', '__loader_
_', '__mkl_version__', '__name__', '__package__', '__path__', '__spec__', '__version__', '_distributor_init', '_globa
ls', '_import_tools', '_mat', '_mklinit', '_numpy_tester', 'abs', 'absolute', 'absolute_import', 'add', 'add_docstrin
g', 'add_newdoc', 'add_newdoc_ufunc', 'add_newdocs', 'alen', 'all', 'allclose', 'alltrue', 'amax', 'amin', 'angle', '
any', 'append', 'apply_along_axis', 'apply_over_axes', 'arange', 'arccos', 'arccosh', 'arcsin', 'arcsinh', 'arctan',
'arctan2', 'arctanh', 'argmax', 'argmin', 'argpartition', 'argsort', 'argwhere', 'around', 'array', 'array2string', '
array_equal', 'array_equiv', 'array_repr', 'array_split', 'array_str', 'asanyarray', 'asarray', 'asarray_chkfinite',
'ascontiguousarray', 'asfarray', 'asfortranarray', 'asmatrix', 'asscalar', 'atleast_1d', 'atleast_2d', 'atleast_3d',
'average', 'bartlett', 'base_repr', 'bench', 'binary_repr', 'bincount', 'bitwise_and', 'bitwise_not', 'bitwise_or', '
bitwise_xor', 'blackman', 'block', 'bmat', 'bool', 'bool8', 'bool_', 'broadcast', 'broadcast_arrays', 'broadcast_to',
'busday_count', 'busday_offset', 'busdaycalendar', 'byte', 'byte_bounds', 'bytes0', 'bytes_', 'c_', 'can_cast', 'cast
', 'cbrt', 'cdouble', 'ceil', 'cfloat', 'char', 'character', 'chararray', 'choose', 'clip', 'clongdouble', 'clongfloa
t', 'column_stack', 'common_type', 'compare_chararrays', 'compat', 'complex', 'complex128', 'complex256', 'complex64'
, 'complex_', 'complexfloating', 'compress', 'concatenate', 'conj', 'conjugate', 'convolve', 'copy', 'copysign', 'cop
yto', 'core', 'corrcoef', 'correlate', 'cos', 'cosh', 'count_nonzero', 'cov', 'cross', 'csingle', 'ctypeslib', 'cumpr
od', 'cumproduct', 'cumsum', 'datetime64', 'datetime_as_string', 'datetime_data', 'deg2rad', 'degrees', 'delete', 'de
precate', 'deprecate_with_doc', 'diag', 'diag_indices', 'diag_indices_from', 'diagflat', 'diagonal', 'diff', 'digitiz
e', 'disp', 'divide', 'division', 'divmod', 'dot', 'double', 'dsplit', 'dstack', 'dtype', 'e', 'ediff1d', 'einsum', '
einsum_path', 'emath', 'empty', 'empty_like', 'equal', 'errstate', 'euler_gamma', 'exp', 'exp2', 'expand_dims', 'expm
1', 'extract', 'eye', 'fabs', 'fastCopyAndTranspose', 'fft', 'fill_diagonal', 'find_common_type', 'finfo', 'fix', 'fl
atiter', 'flatnonzero', 'flexible', 'flip', 'fliplr', 'flipud', 'float', 'float128', 'float16', 'float32', 'float64',
'float_', 'float_power', 'floating', 'floor', 'floor_divide', 'fmax', 'fmin', 'fmod', 'format_float_positional', 'for
mat_float_scientific', 'format_parser', 'frexp', 'frombuffer', 'fromfile', 'fromfunction', 'fromiter', 'frompyfunc',
'fromregex', 'fromstring', 'full', 'full_like', 'fv', 'generic', 'genfromtxt', 'geomspace', 'get_array_wrap', 'get_in
clude', 'get_printoptions', 'getbufsize', 'geterr', 'geterrcall', 'geterrobj', 'gradient', 'greater', 'greater_equal'
, 'half', 'hamming', 'hanning', 'heaviside', 'histogram', 'histogram2d', 'histogramdd', 'hsplit', 'hstack', 'hypot',
```

Giving a print order in a cell and pressing ‘shit’+’enter’ will yield the result directly under the active cell. This reflects a nice feature of the jupyter notebooks, namely that the result of a piece of code can be seen directly, which makes it very comprehensive how parts of a code perform different things. Of course in the end one can always write a script without fragmenting everything into cells, for the coding process itself however, especially in scientific analysis, this comes in handy. Here all the elements within numpy are printed.

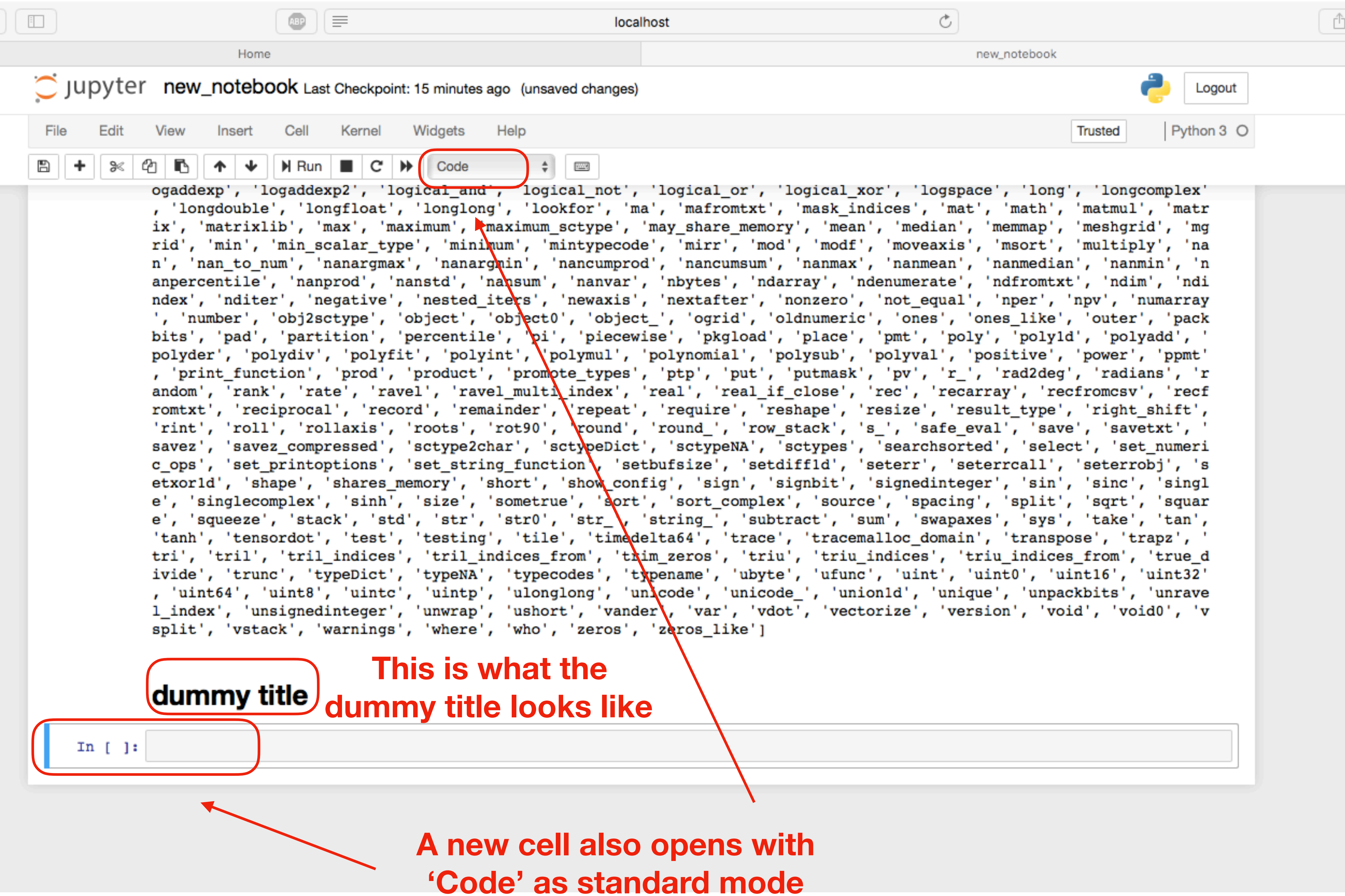


```
isin', 'isinf', 'isnan', 'isnat', 'isneginf', 'isposinf', 'isreal', 'isrealobj', 'isscalar', 'issctype', 'issubclass_
', 'issubdtype', 'issubsctype', 'iterable', 'ix_', 'kaiser', 'kron', 'ldexp', 'left_shift', 'less', 'less_equal', 'le
xsort', 'lib', 'linalg', 'linspace', 'little_endian', 'load', 'loads', 'loadtxt', 'log', 'log10', 'log1p', 'log2', 'l
ogaddexp', 'logaddexp2', 'logical_and', 'logical_not', 'logical_or', 'logical_xor', 'logspace', 'long', 'longcomplex'
, 'longdouble', 'longfloat', 'longlong', 'lookfor', 'ma', 'mafromtxt', 'mask_indices', 'mat', 'math', 'matmul', 'matr
ix', 'matrixlib', 'max', 'maximum', 'maximum_sctype', 'may_share_memory', 'mean', 'median', 'memmap', 'meshgrid', 'mg
rid', 'min', 'min_scalar_type', 'minimum', 'mintypecode', 'mirr', 'mod', 'modf', 'moveaxis', 'msort', 'multiply', 'na
n', 'nan_to_num', 'nanargmax', 'nanargmin', 'nancumprod', 'nancumsum', 'nanmax', 'nanmean', 'nanmedian', 'nanmin', 'n
anpercentile', 'nanprod', 'nanstd', 'nansum', 'nanvar', 'nbytes', 'ndarray', 'ndenumerate', 'ndfromtxt', 'ndim', 'ndi
ndex', 'nditer', 'negative', 'nested_iters', 'newaxis', 'nextafter', 'nonzero', 'not_equal', 'nper', 'npv', 'numarray
', 'number', 'obj2sctype', 'object', 'object0', 'object_', 'ogrid', 'oldnumeric', 'ones', 'ones_like', 'outer', 'pack
bits', 'pad', 'partition', 'percentile', 'pi', 'piecewise', 'pkgload', 'place', 'pmt', 'poly', 'poly1d', 'polyadd', '
polyder', 'polydiv', 'polyfit', 'polyint', 'polymul', 'polynomial', 'polysub', 'polyval', 'positive', 'power', 'ppmt'
, 'print_function', 'prod', 'product', 'promote_types', 'ptp', 'put', 'putmask', 'pv', 'r_', 'rad2deg', 'radians', 'r
andom', 'rank', 'rate', 'ravel', 'ravel_multi_index', 'real', 'real_if_close', 'rec', 'recarray', 'recfromcsv', 'recf
romtxt', 'reciprocal', 'record', 'remainder', 'repeat', 'require', 'reshape', 'resize', 'result_type', 'right_shift',
'rint', 'roll', 'rollaxis', 'roots', 'rot90', 'round', 'round_', 'row_stack', 's_', 'safe_eval', 'save', 'savetxt', '
savez', 'savez_compressed', 'sctype2char', 'sctypeDict', 'sctypeNA', 'sctypes', 'searchsorted', 'select', 'set_numeri
c_ops', 'set_printoptions', 'set_string_function', 'setbufsize', 'setdiff1d', 'seterr', 'seterrcall', 'seterrobj', 's
etxor1d', 'shape', 'shares_memory', 'short', 'show_config', 'sign', 'signbit', 'signedinteger', 'sin', 'sinc', 'singl
e', 'singlecomplex', 'sinh', 'size', 'sometrue', 'sort', 'sort_complex', 'source', 'spacing', 'split', 'sqrt', 'squar
e', 'squeeze', 'stack', 'std', 'str', 'str0', 'str_', 'string_', 'subtract', 'sum', 'swapaxes', 'sys', 'take', 'tan',
'tanh', 'tensordot', 'test', 'testing', 'tile', 'timedelta64', 'trace', 'tracemalloc_domain', 'transpose', 'trapz', '
tri', 'tril', 'tril_indices', 'tril_indices_from', 'trim_zeros', 'triu', 'triu_indices', 'triu_indices_from', 'true_d
ivide', 'trunc', 'typeDict', 'typeNA', 'typecodes', 'typename', 'ubyte', 'ufunc', 'uint', 'uint0', 'uint16', 'uint32'
, 'uint64', 'uint8', 'uintc', 'uintp', 'ulonglong', 'unicode', 'unicode_', 'union1d', 'unique', 'unpackbits', 'unrave
l_index', 'unsignedinteger', 'unwrap', 'ushort', 'vander', 'var', 'vdot', 'vectorize', 'version', 'void', 'void0', 'v
split', 'vstack', 'warnings', 'where', 'who', 'zeros', 'zeros_like']
```

```
# dummy title
```

Setting a #+'space' before the text, will create a title like object. This is nice for structuring your exercises

'Code' is not the only option in this checkbox. When writing longer comments for example which should be detached from the code (not only commented out with #text for one line or '"text"' more lines), on can use a cell to activate 'Markdown' within it. Just start writing your text and 'shift'+'enter' when done to get a plane text within this cell.



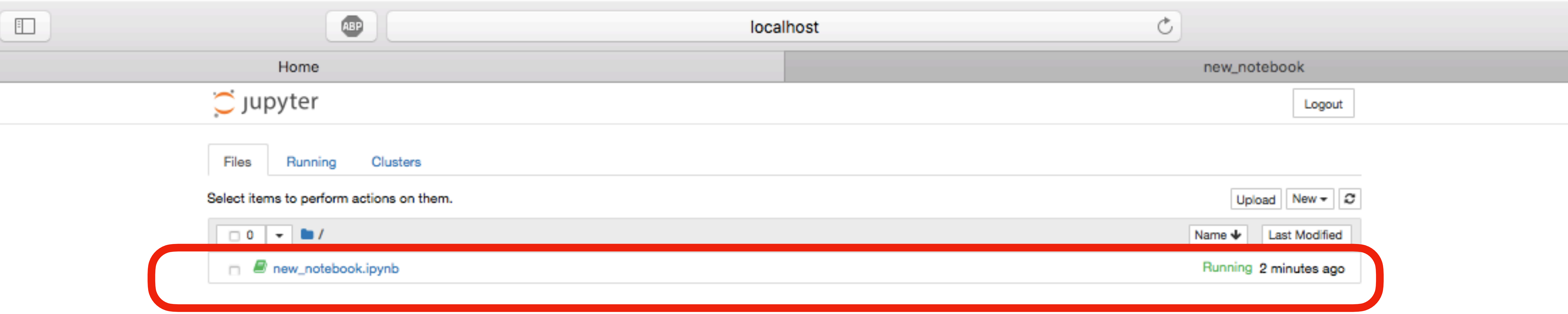
```
ogaddexp', 'logaddexp2', 'logical_and', 'logical_not', 'logical_or', 'logical_xor', 'logspace', 'long', 'longcomplex'
, 'longdouble', 'longfloat', 'longlong', 'lookfor', 'ma', 'mafromtxt', 'mask_indices', 'mat', 'math', 'matmul', 'matr
ix', 'matrixlib', 'max', 'maximum', 'maximum_sctype', 'may_share_memory', 'mean', 'median', 'memmap', 'meshgrid', 'mg
rid', 'min', 'min_scalar_type', 'minimum', 'mintypecode', 'mirr', 'mod', 'modf', 'moveaxis', 'msort', 'multiply', 'na
n', 'nan_to_num', 'nanargmax', 'nanargmin', 'nancumprod', 'nancumsum', 'nanmax', 'nanmean', 'nanmedian', 'nanmin', 'n
anpercentile', 'nanprod', 'nanstd', 'nansum', 'nanvar', 'nbytes', 'ndarray', 'ndenumerate', 'ndfromtxt', 'ndim', 'ndi
ndex', 'nditer', 'negative', 'nested_iters', 'newaxis', 'nextafter', 'nonzero', 'not_equal', 'nper', 'npv', 'numarray
', 'number', 'obj2sctype', 'object', 'object0', 'object_', 'ogrid', 'oldnumeric', 'ones', 'ones_like', 'outer', 'pack
bits', 'pad', 'partition', 'percentile', 'pi', 'piecewise', 'pkgload', 'place', 'pmt', 'poly', 'poly1d', 'polyadd', '
polyder', 'polydiv', 'polyfit', 'polyint', 'polymul', 'polynomial', 'polysub', 'polyval', 'positive', 'power', 'ppmt'
, 'print_function', 'prod', 'product', 'promote_types', 'ptp', 'put', 'putmask', 'pv', 'r_', 'rad2deg', 'radians', 'r
andom', 'rank', 'rate', 'ravel', 'ravel_multi_index', 'real', 'real_if_close', 'rec', 'recarray', 'recfromcsv', 'recf
romtxt', 'reciprocal', 'record', 'remainder', 'repeat', 'require', 'reshape', 'resize', 'result_type', 'right_shift',
'rint', 'roll', 'rollaxis', 'roots', 'rot90', 'round', 'round_', 'row_stack', 's_', 'safe_eval', 'save', 'savetxt', '
savez', 'savez_compressed', 'sctype2char', 'sctypeDict', 'sctypeNA', 'sctypes', 'searchsorted', 'select', 'set_numeri
c_ops', 'set_printoptions', 'set_string_function', 'setbufsize', 'setdiff1d', 'seterr', 'seterrcall', 'seterrobj', 's
etxor1d', 'shape', 'shares_memory', 'short', 'show_config', 'sign', 'signbit', 'signedinteger', 'sin', 'sinc', 'singl
e', 'singlecomplex', 'sinh', 'size', 'sometrue', 'sort', 'sort_complex', 'source', 'spacing', 'split', 'sqrt', 'squar
e', 'squeeze', 'stack', 'std', 'str', 'str0', 'str_', 'string_', 'subtract', 'sum', 'swapaxes', 'sys', 'take', 'tan',
'tanh', 'tensordot', 'test', 'testing', 'tile', 'timedelta64', 'trace', 'tracemalloc_domain', 'transpose', 'trapz', '
tri', 'tril', 'tril_indices', 'tril_indices_from', 'trim_zeros', 'triu', 'triu_indices', 'triu_indices_from', 'true_d
ivide', 'trunc', 'typeDict', 'typeNA', 'typecodes', 'typename', 'ubyte', 'ufunc', 'uint', 'uint0', 'uint16', 'uint32'
, 'uint64', 'uint8', 'uintc', 'uintp', 'ulonglong', 'unicode', 'unicode_', 'union1d', 'unique', 'unpackbits', 'unrave
l_index', 'unsignedinteger', 'unwrap', 'ushort', 'vander', 'var', 'vdot', 'vectorize', 'version', 'void', 'void0', 'v
split', 'vstack', 'warnings', 'where', 'who', 'zeros', 'zeros_like']
```

dummy title

This is what the dummy title looks like

A new cell also opens with ‘Code’ as standard mode

a new notebook - navigation



Select items to perform actions on them.

□ 0 /

□ new_notebook.ipynb Running 2 minutes ago

if you go back into the directory where you started Jupiter notebook, the notebook will appear here, ready to be opened and worked with again