TUTORIAL ON VSCODE-PVS

PAOLO MASCI
PAOLO.MASCI@NIANET.ORG

NATIONAL INSTITUTE OF AEROSPACE
LANGLEY RESEARCH CENTER

MAY 2022

TOPICS COVERED IN THIS TUTORIAL

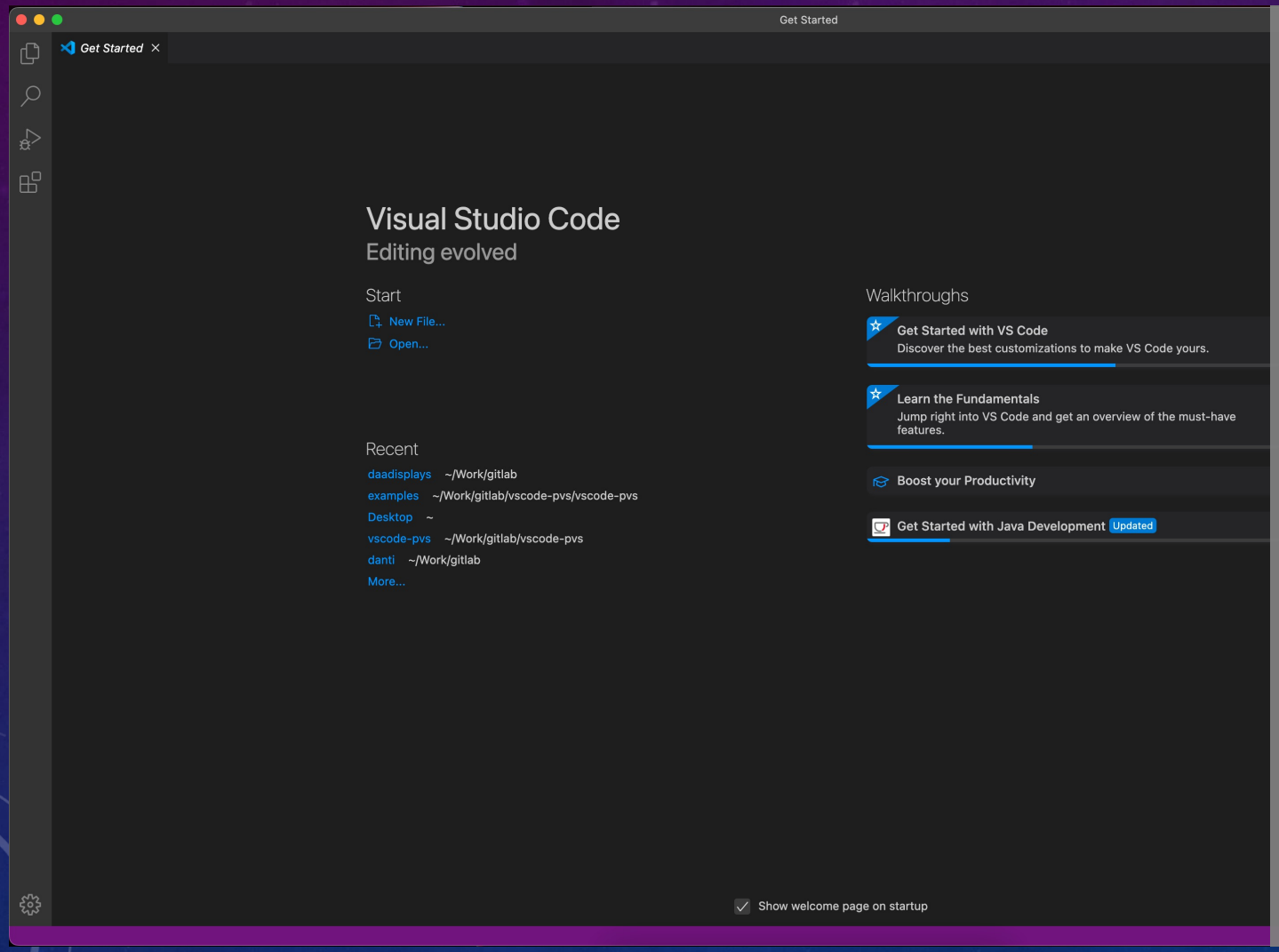
1. Installation of VSCode-PVS
2. Creation and editing of PVS theories
3. Typechecking and debugging PVS theories
4. Development of PVS proofs in VSCode-PVS
5. Documenting your PVS files
6. Prototyping functions provided by VSCode-PVS

INSTALLATION OF VSCODE-PVS

- VSCode-PVS can be installed from the Visual Studio Code marketplace

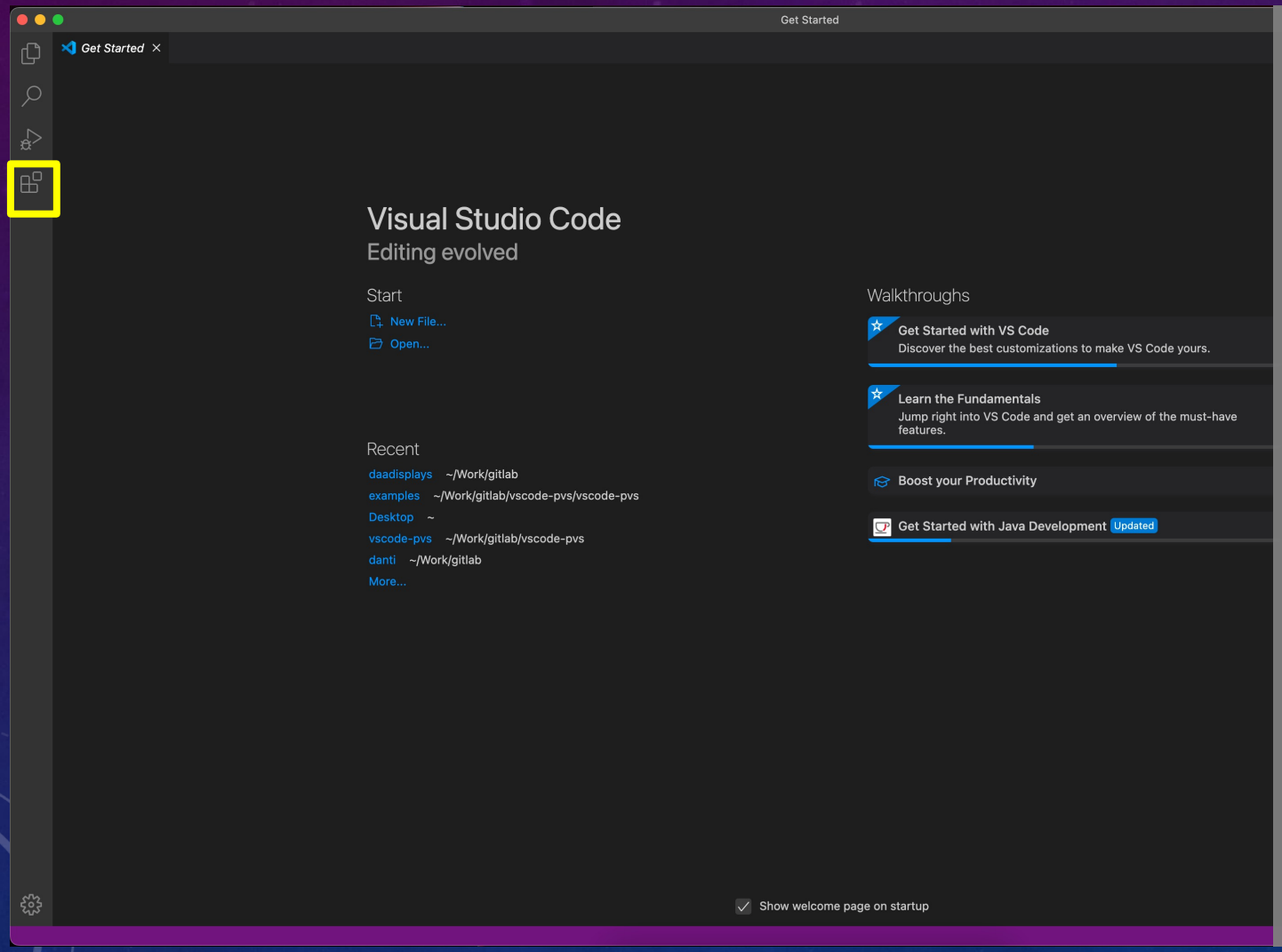
- Requirements
  - Visual Studio Code (https://code.visualstudio.com/download)
  - NodeJS (https://nodejs.org/en/download/)
  - Linux or Intel Mac

INSTALLATION OF VSCODE-PVS



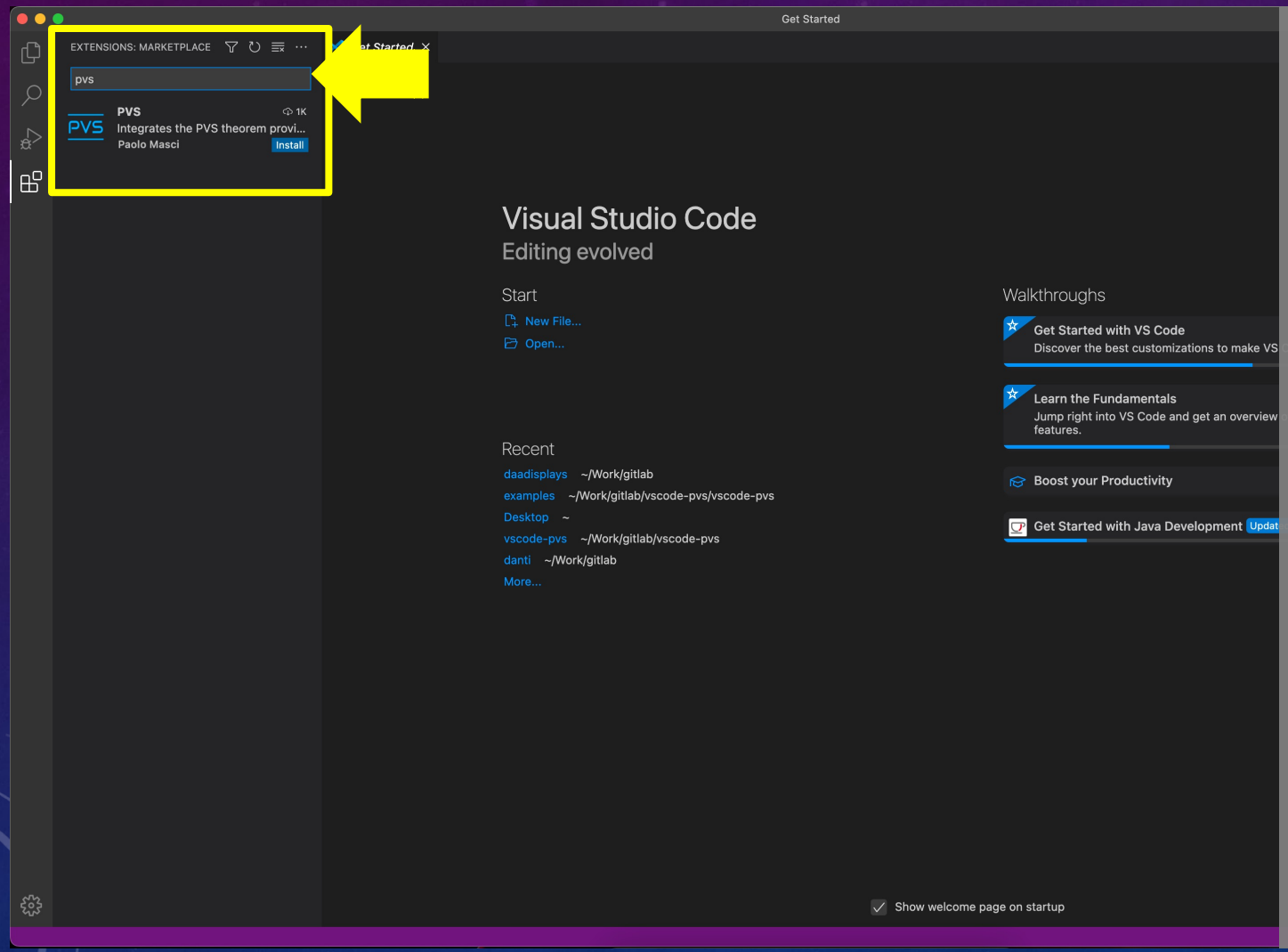
1. Open Visual Studio Code

INSTALLATION OF VSCODE-PVS



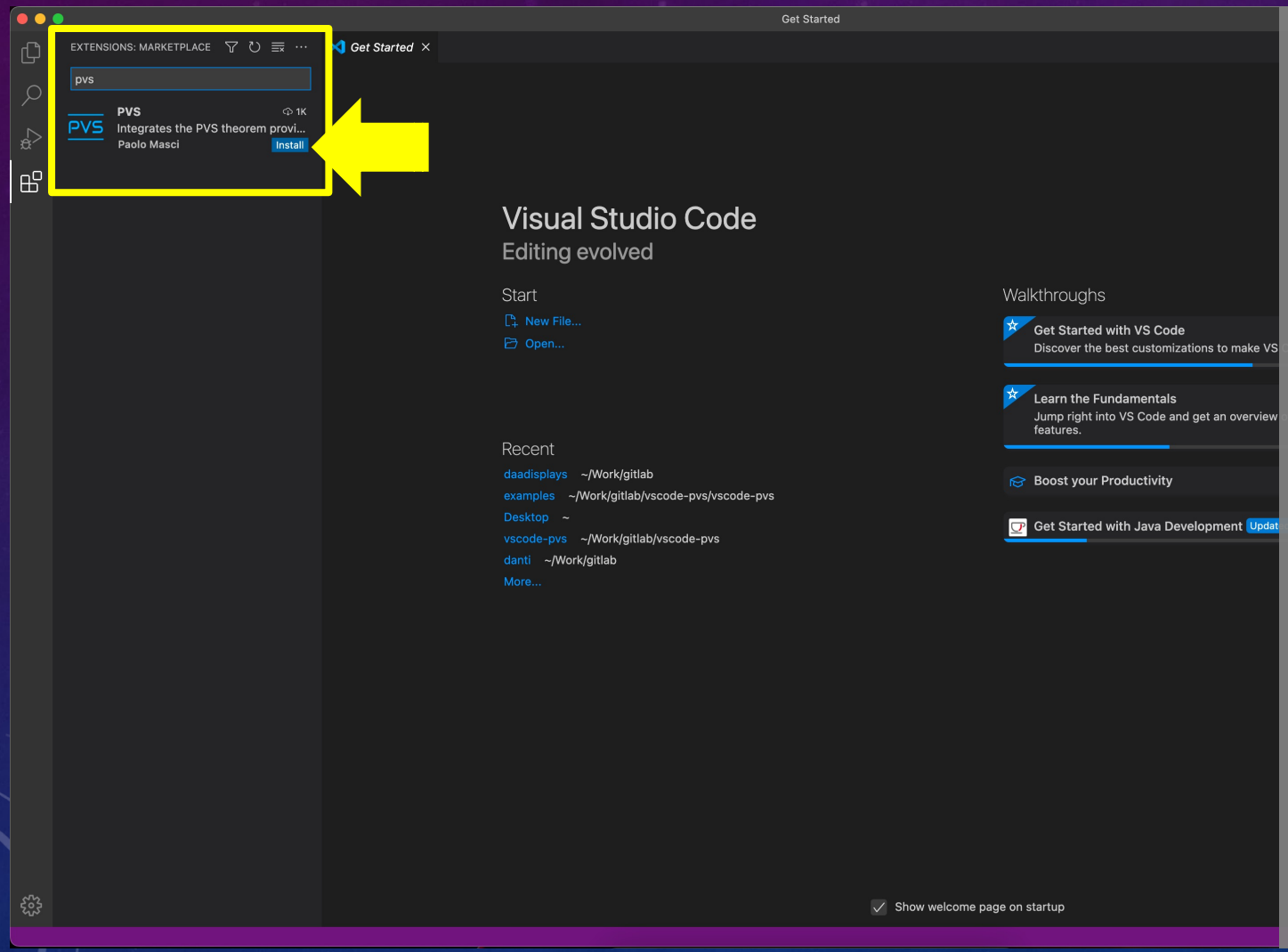
1. Open Visual Studio Code
2. Click on the Extensions icon

INSTALLATION OF VSCODE-PVS



1. Open Visual Studio Code
2. Click on the Extensions icon
3. Search PVS in the Marketplace

INSTALLATION OF VSCODE-PVS

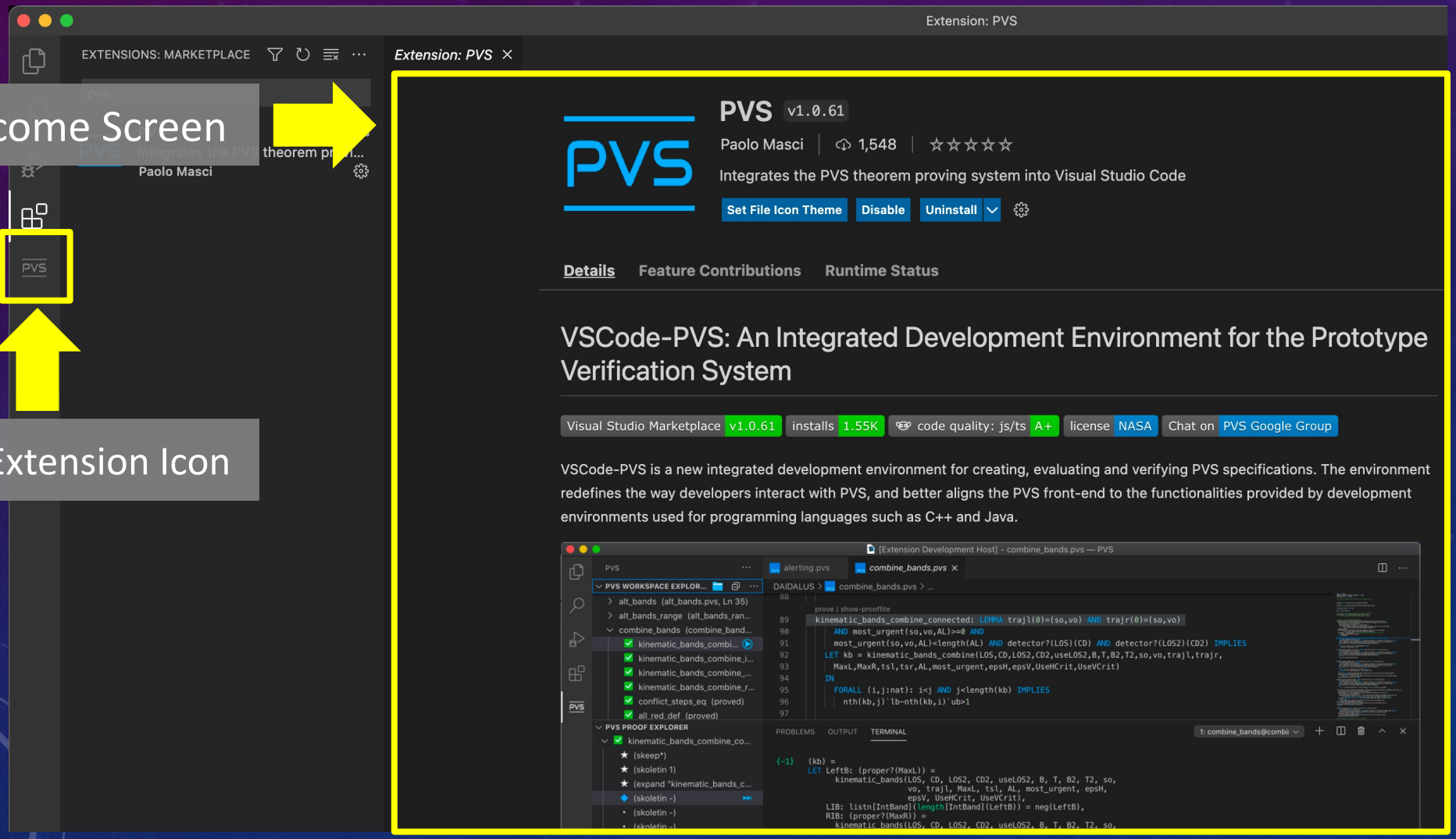


1. Open Visual Studio Code
2. Click on the Extensions icon
3. Search PVS in the Marketplace
4. Click Install

INSTALLATION OF VSCODE-PVS

Welcome Screen

PVS Extension Icon



PVS ALLEGRO + NASALIB

VSCode-PVS will check if PVS Allegro and NASALib are already present on your system (if they are not present, they will be download by VSCode-PVS)

Default directory structure when you choose your home folder as base folder for the installation of PVS Allegro

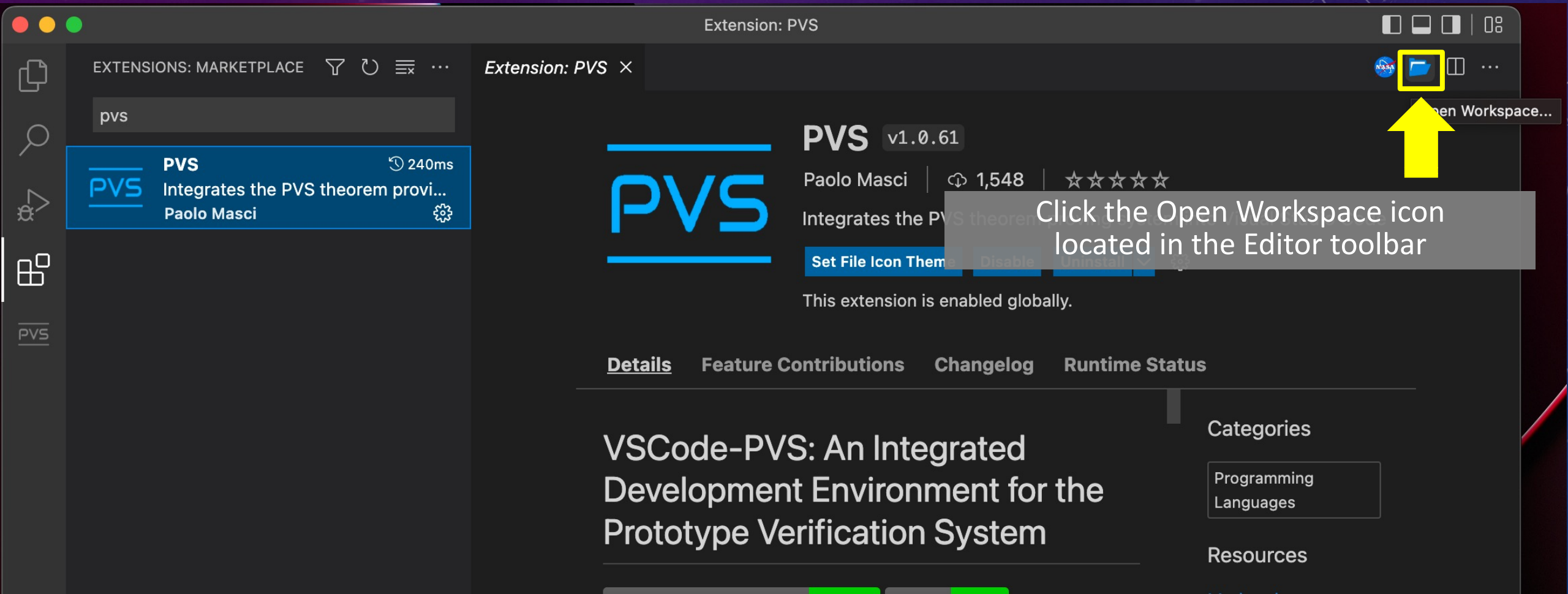
- PVS Allegro: ~/pvs-7.1.0
- NASALib: ~/pvs-7.1.0/nasalib
- Your PVS developments: ~/Workspaces

CREATION AND EDITING OF PVS THEORIES

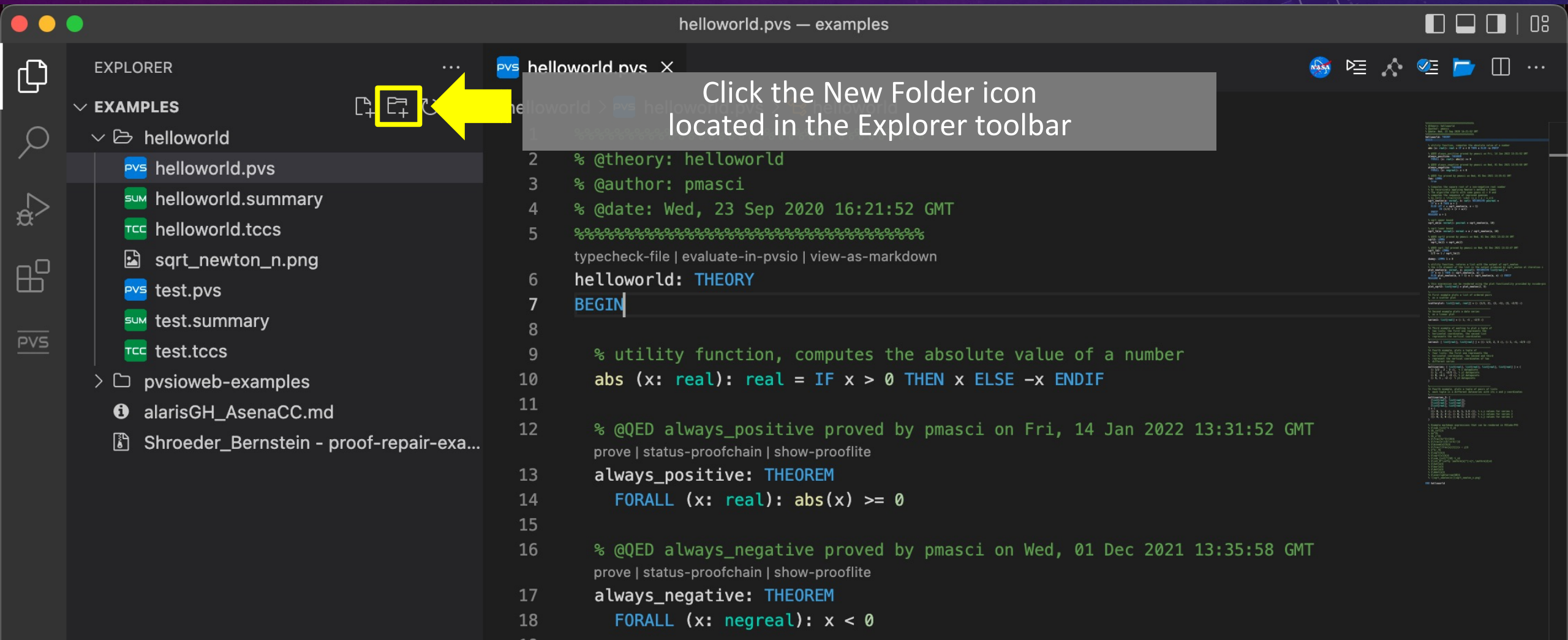
To develop a PVS theory, you need to perform 3 steps in VSCode-PVS

1. Open a workspace
2. Create a folder that will contain your .pvs files
3. Edit the .pvs files to develop the theory specification

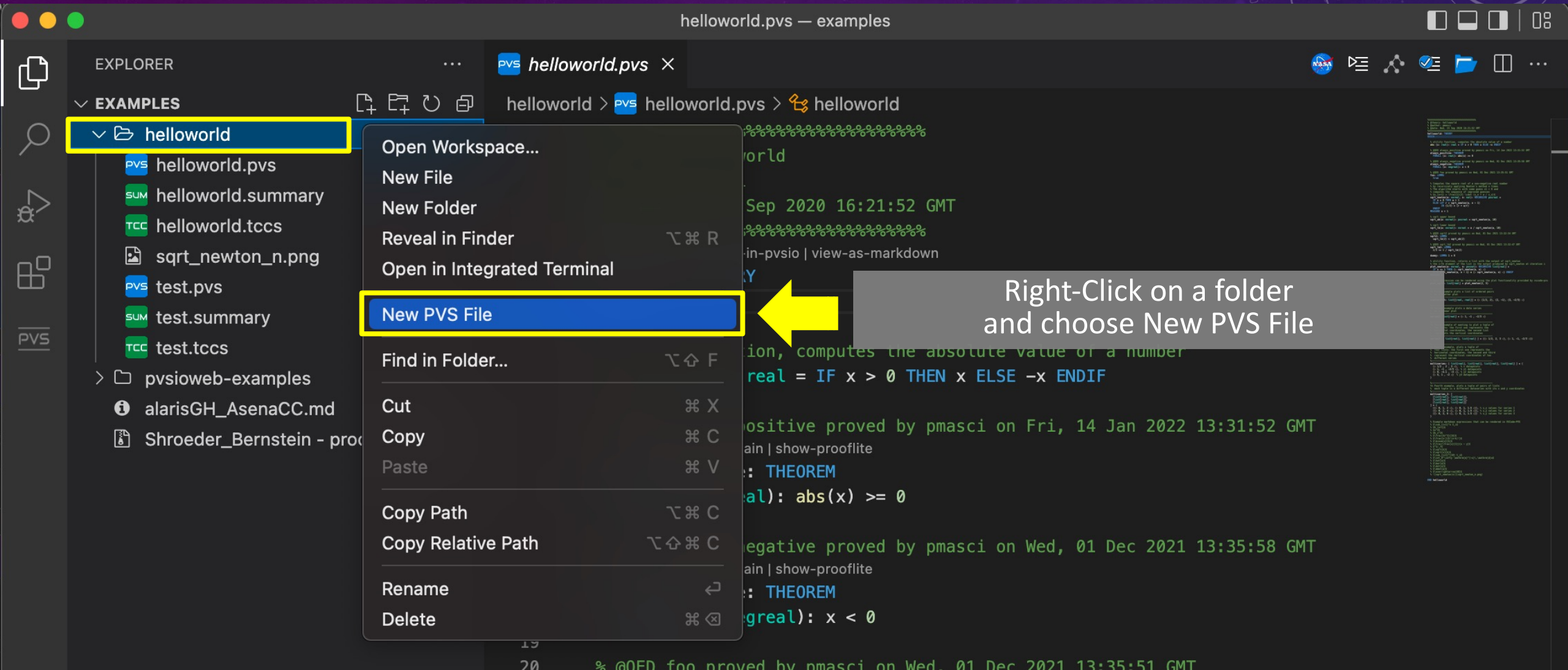
OPENING A WORKSPACE



CREATING A NEW FOLDER IN THE WORKSPACE



CREATING A .PVS FILE



THE CREATED .PVS FILE

helloworld.pvs

helloworld > helloworld.pvs > helloworld

```
1   %%
2   % @theory: helloworld
3   % @author: pmasci
4   % @date: Fri, 20 May 2022 17:05:04 GMT
5   %%
    typecheck-file | evaluate-in-pvsio | view-as-markdown
6   helloworld: THEORY
7     BEGIN
8
9     END helloworld
```

The created .pvs file will contain a documentation header and a theory declaration

PVS language reference: https://pvs.csl.sri.com/documentation.html

EDITING A PVS THEORY

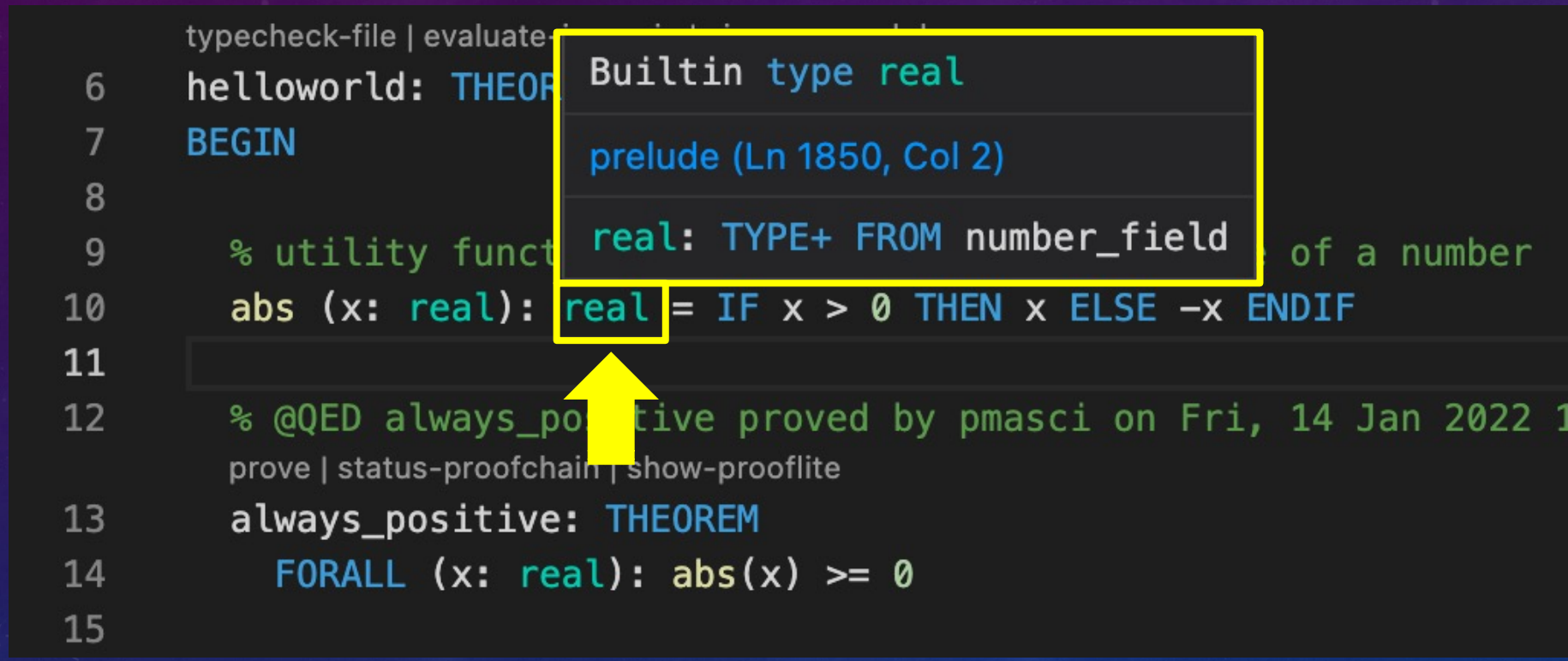
typecheck-file | evaluate-in-pvsio | view-as-markdown

```
6   helloworld: THEORY
7   BEGIN
8
9     % utility function, computes the absolute value of a number
10    abs (x: real): real = IF x > 0 THEN x ELSE -x ENDIF
11
12    % @QED always_positive proved by pmasci on Fri, 14 Jan 2022
      prove | status-proofchain | show-prooflite
13    always_positive: THEOREM
14      FORALL (x: real): abs(x) >= 0
15
```

Syntax highlighting for keywords, types and library functions

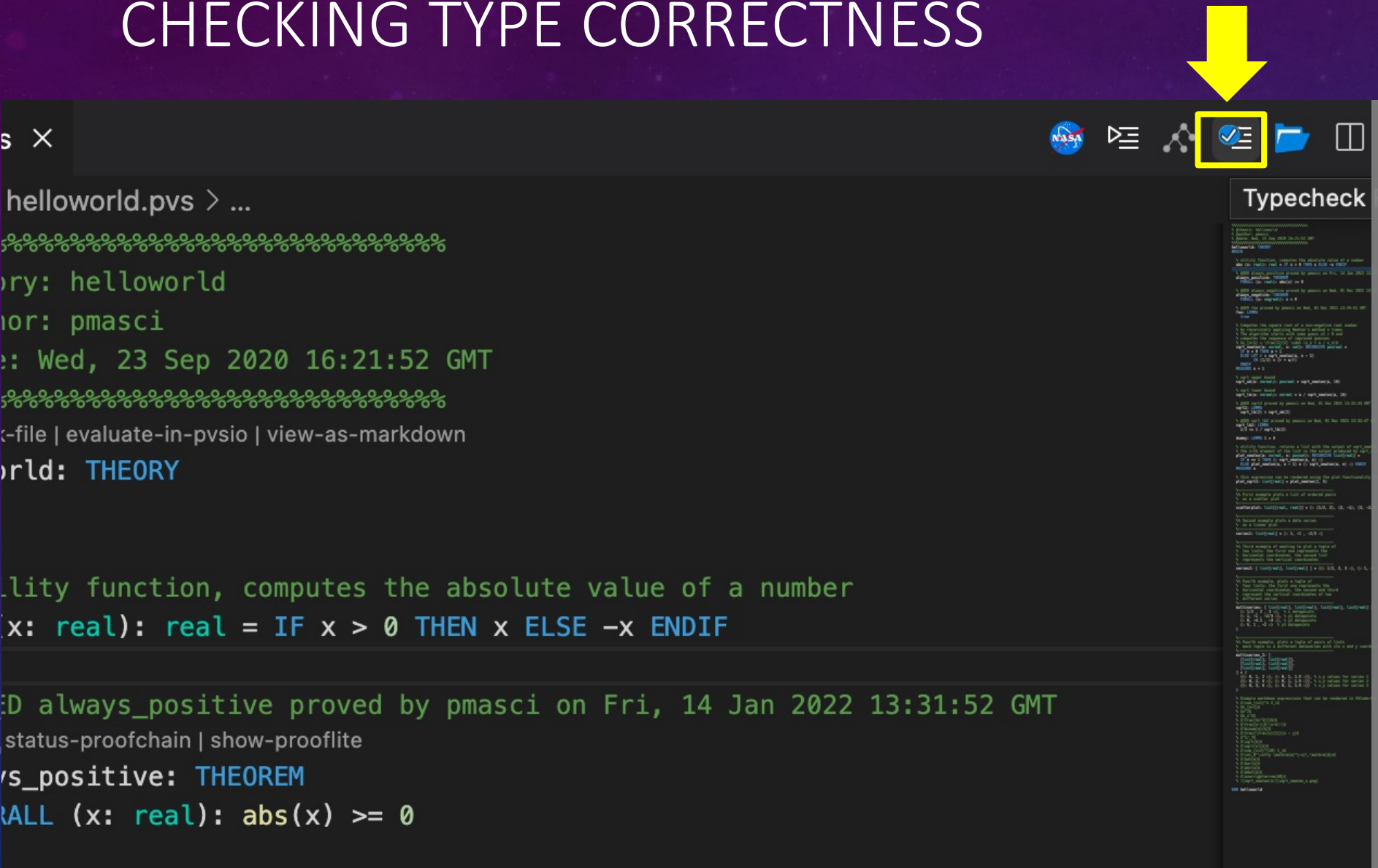
PVS language reference: https://pvs.csl.sri.com/documentation.html

NAVIGATING DEFINITIONS



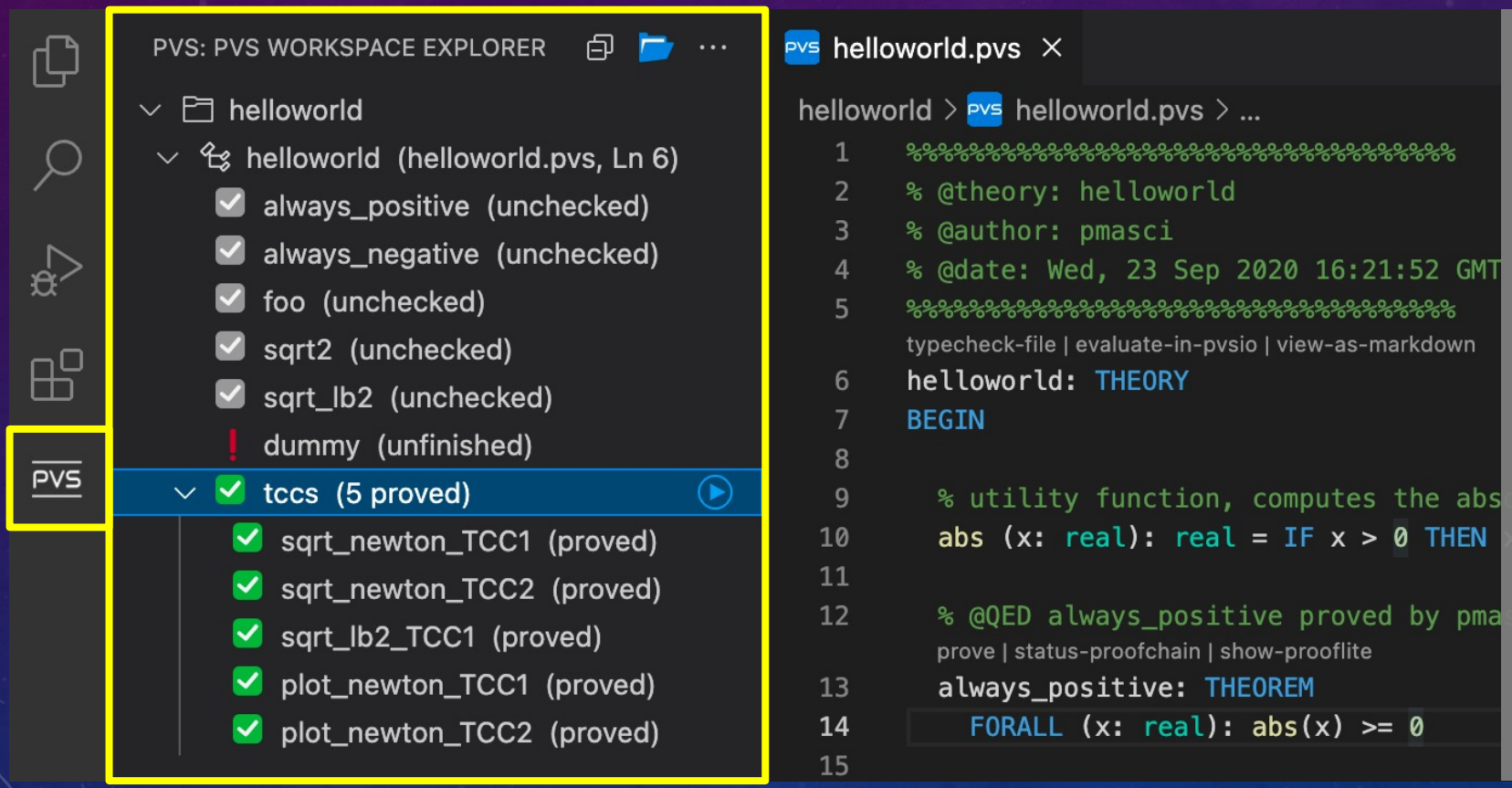
Place the mouse pointer of a term to see the definition of the term

CHECKING TYPE CORRECTNESS



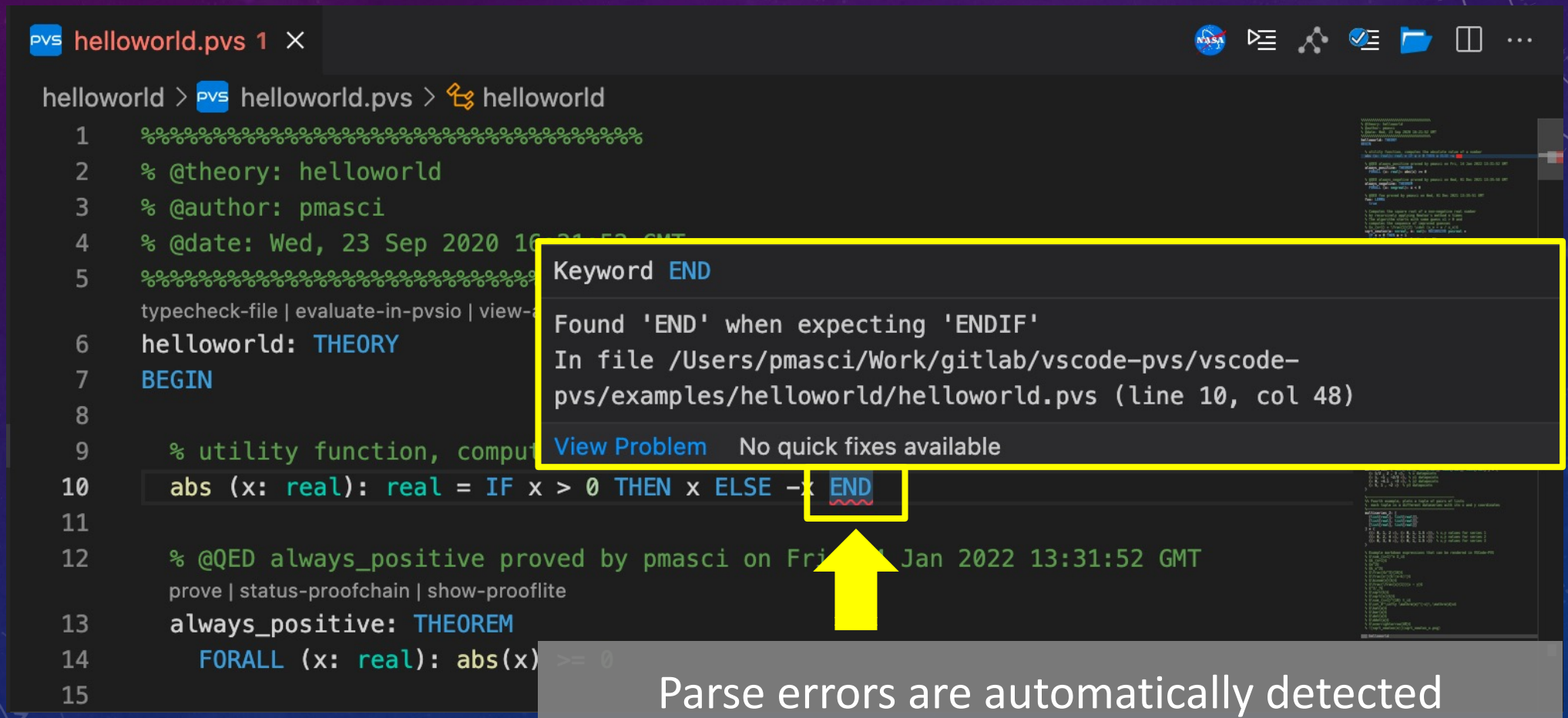
Click the Typecheck icon located in the Editor toolbar to check type correctness

PROOF OBLIGATIONS (TCCS)



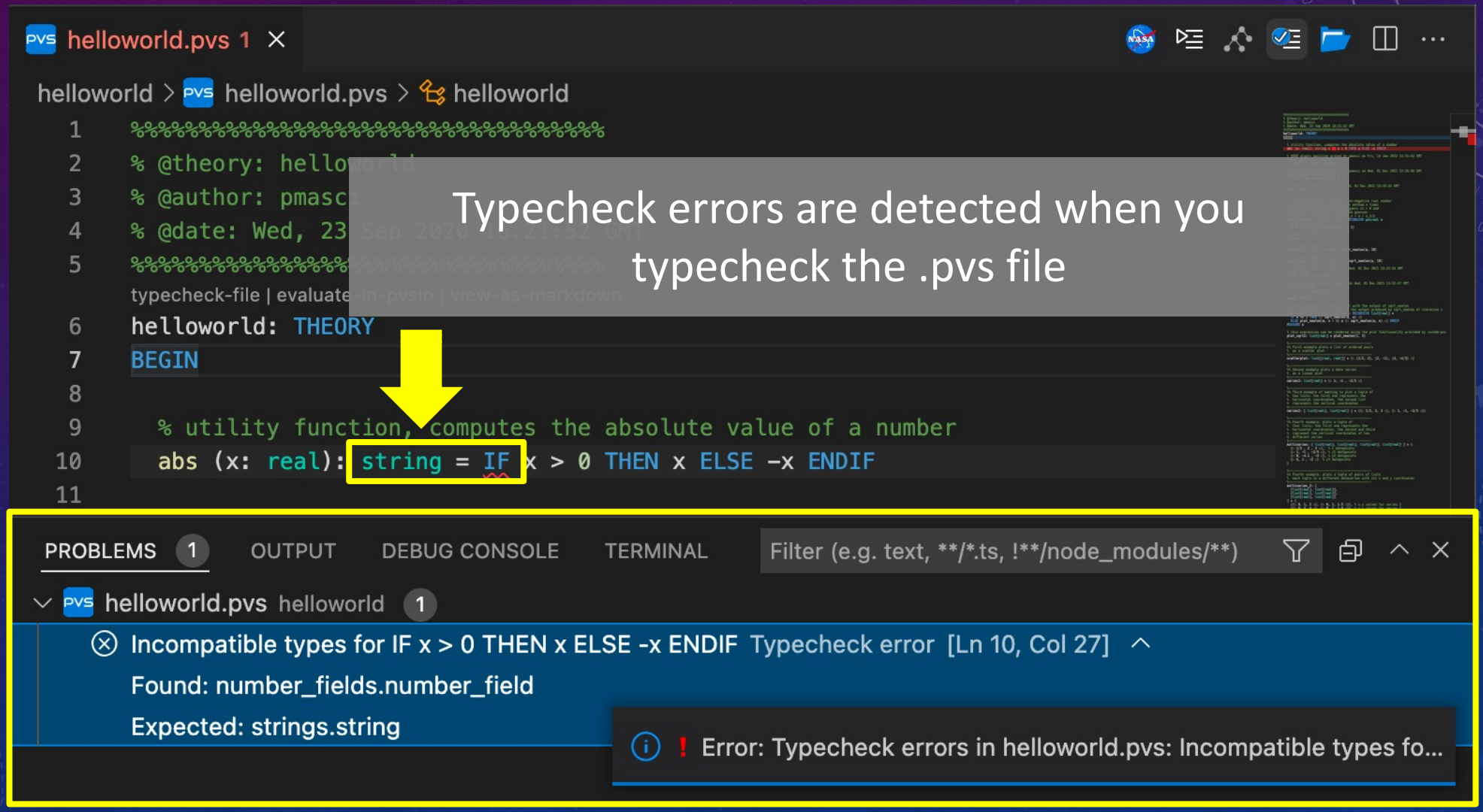
Click the PVS icon to view the list of proof obligations and theorems

LIVE DIAGNOSTICS FOR PARSE ERRORS

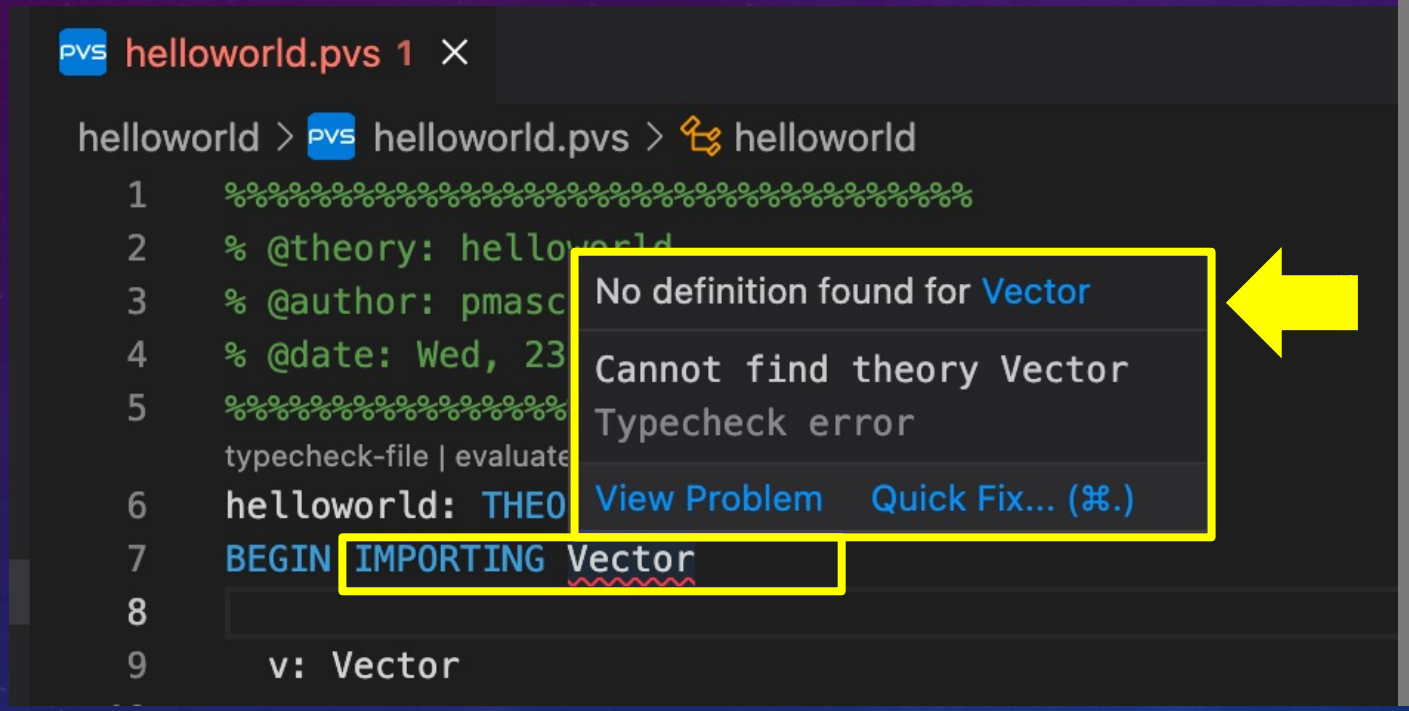


Parse errors are automatically detected when the .pvs file is saved

LIVE DIAGNOSTICS FOR TYPECHECK ERRORS

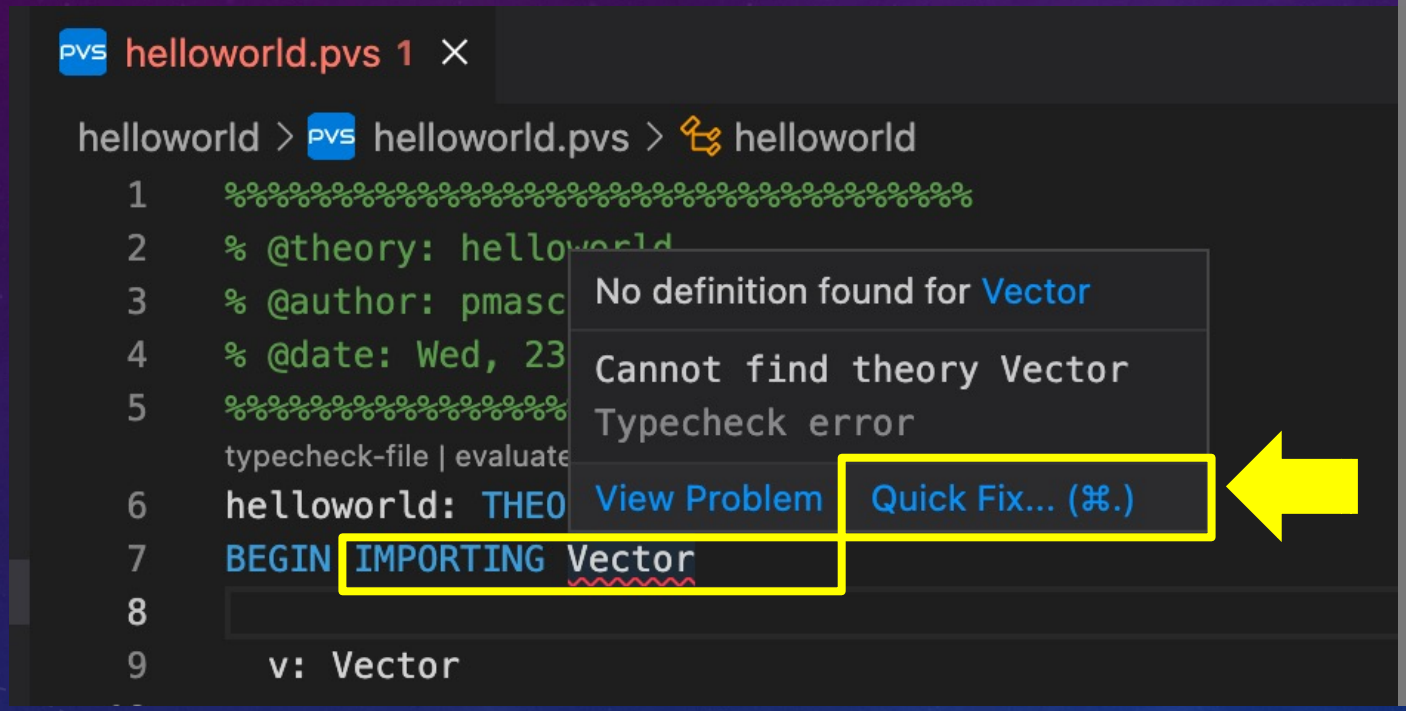


LIVE DIAGNOSTICS FOR IMPORTING ERRORS



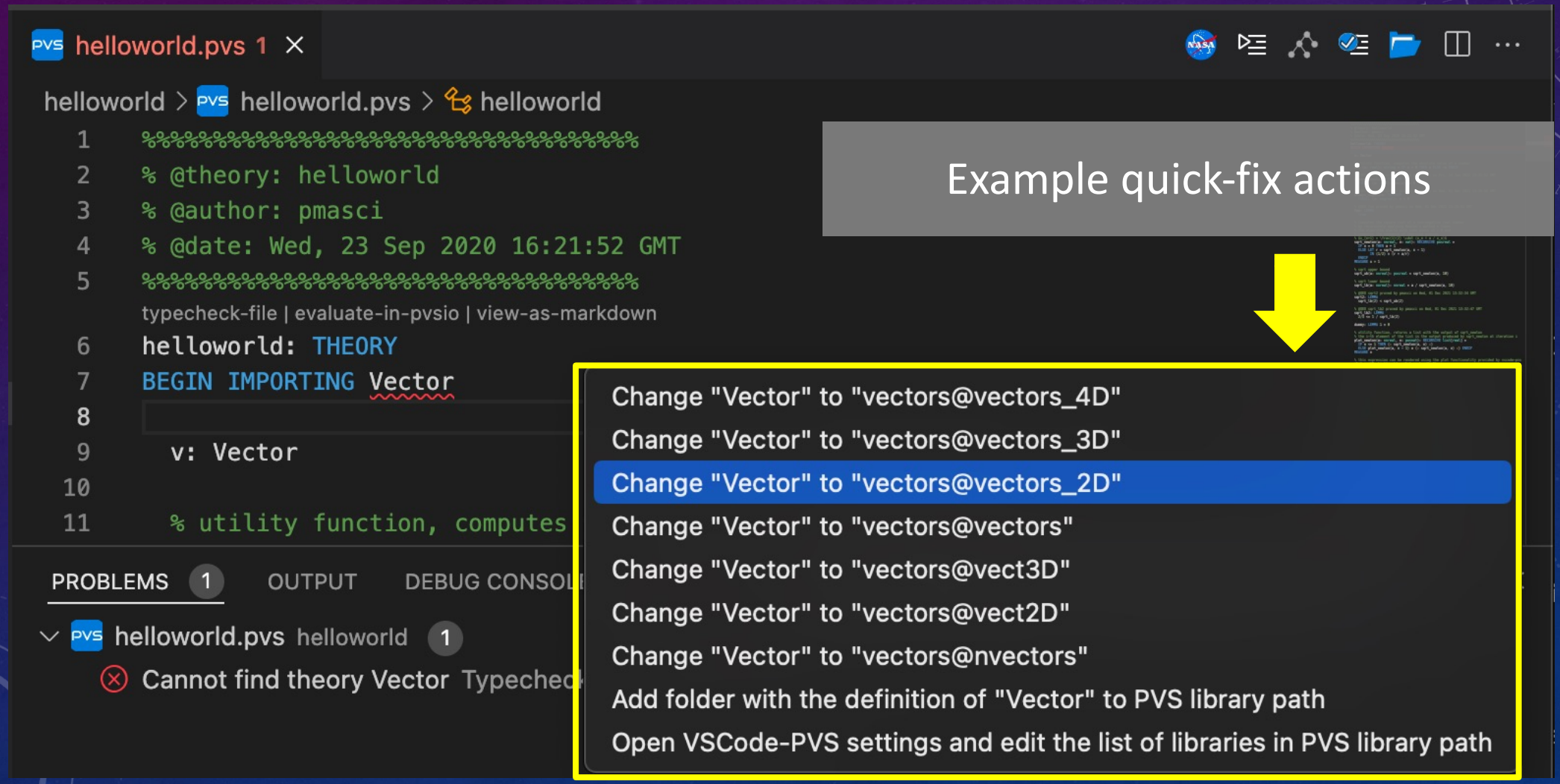
Importing errors are detected when you typecheck the .pvs file

QUICK-FIX ACTIONS



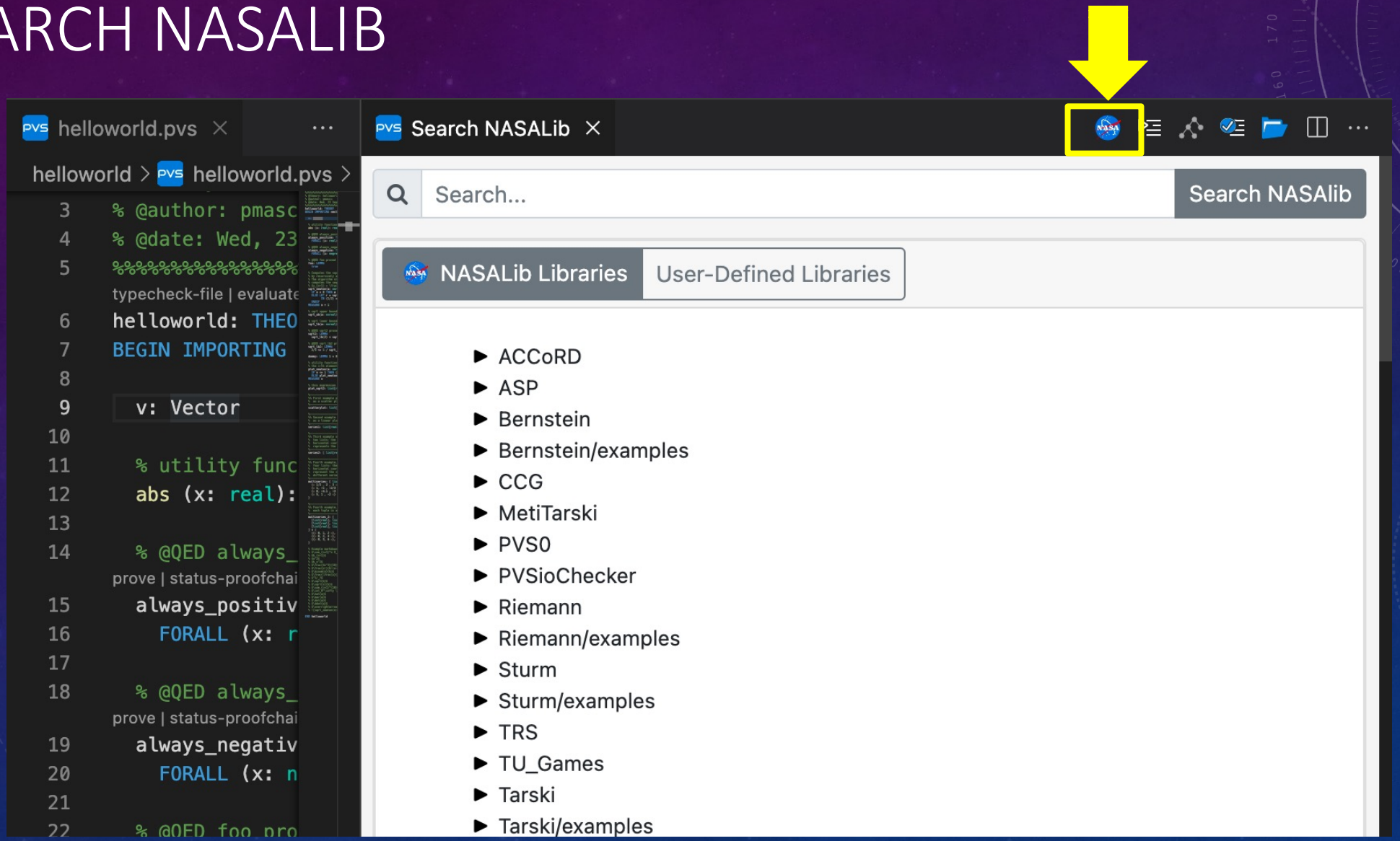
VSCode-PVS provides quick-fix actions that can resolve importing errors

QUICK-FIX ACTIONS

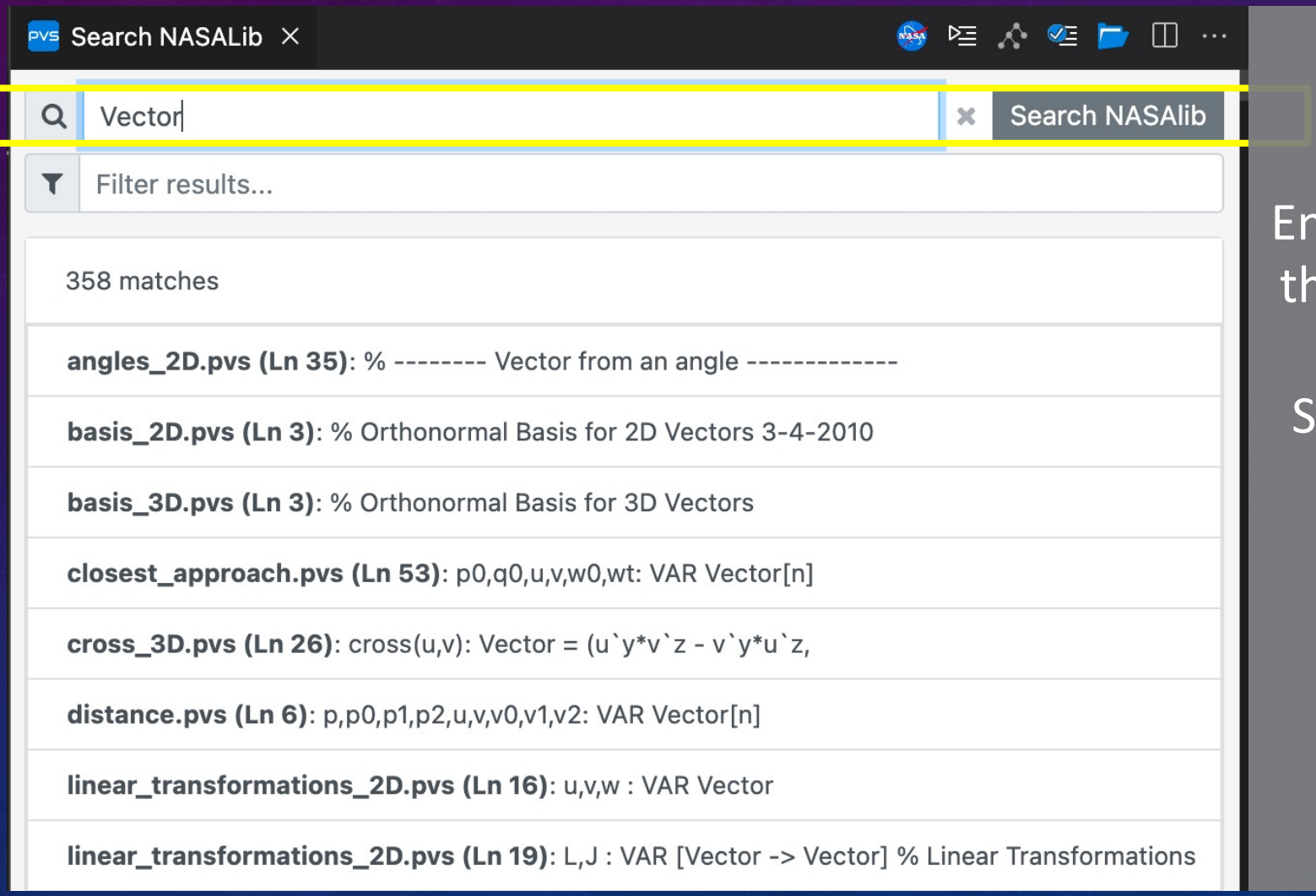


Click the NASA meatball logo to search definitions and lemmas in NASALib

SEARCH NASALIB

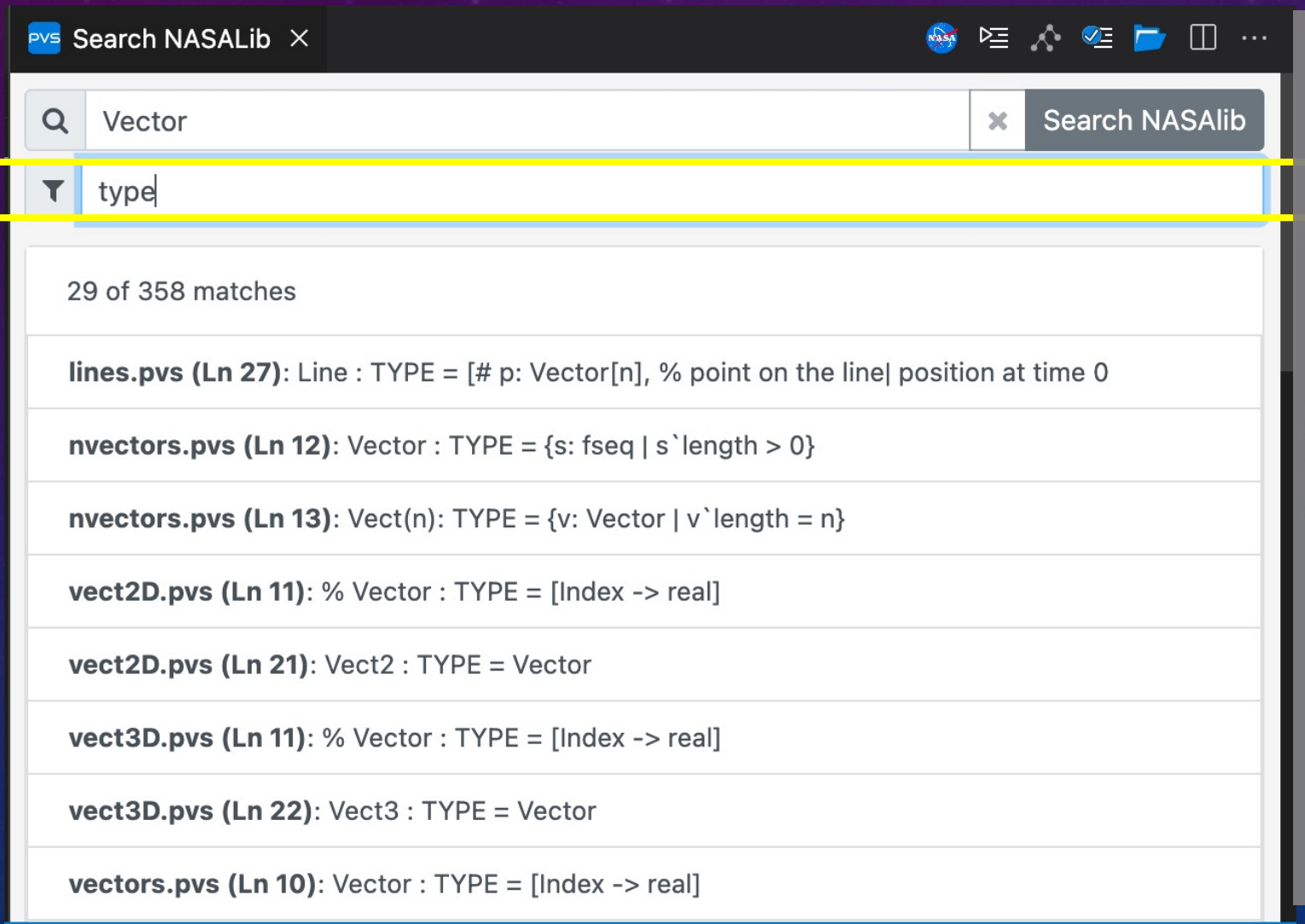


SEARCH NASALIB



Enter the search string in the corresponding input field and press the Search NASALib button

SEARCH NASALIB



Search results can be filtered, e.g., to show only type definitions

PROVING A THEOREM

helloworld.pvs

helloworld > helloworld.pvs > helloworld

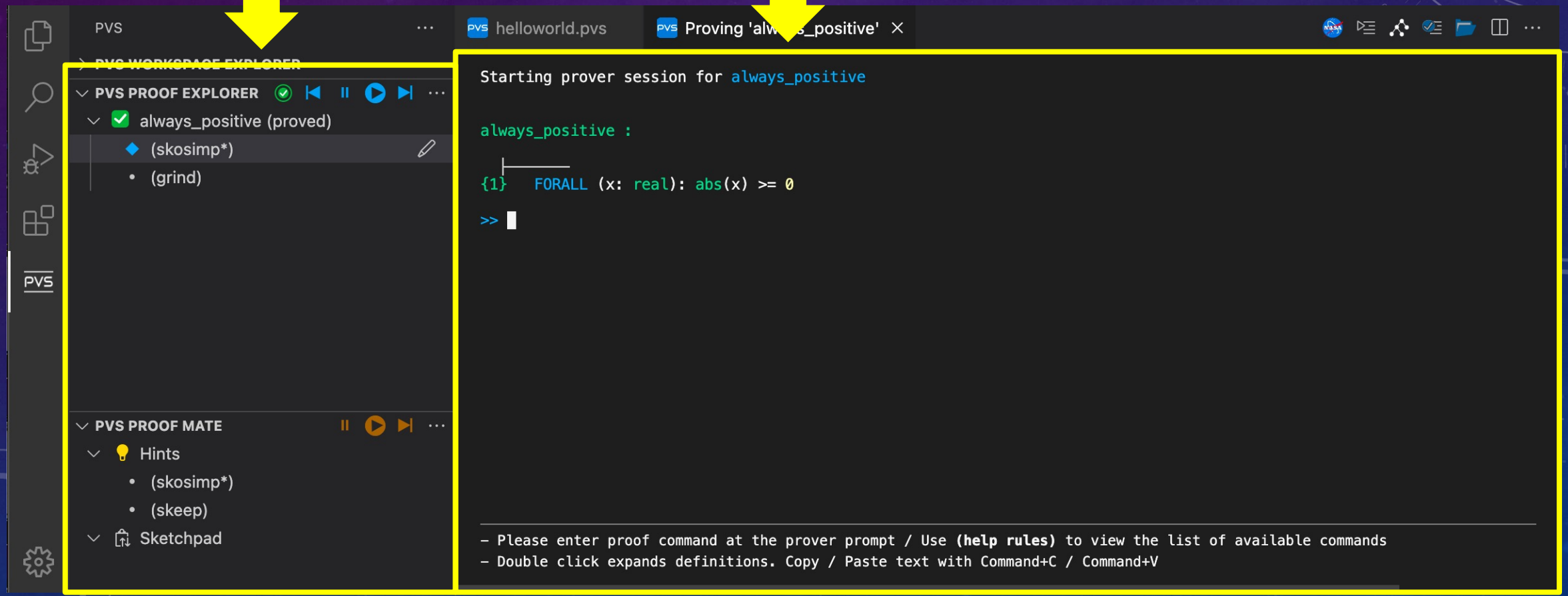
```
3    % @author: pmasci
4    % @date: Wed, 23 Sep 2020 16:21:52 GMT
5    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     typecheck-file | evaluate-in-pvsio | view-as-markdown
6    helloworld: THEORY
7    BEGIN
8
9      % utility function, computes the absolute value of a number
10     abs (x: real): real = IF x > 0 THEN x ELSE -x ENDIF
11
12     % @QED always_positive proved by pmasci on Fri, 14 Jan 2022 13:31:52 GMT
       prove | status-proofchain | show-prooflite
13     always_positive: THEOREM
14       FORALL (x: real): abs(x) >= 0
15
```

Click the inline command 'prove' to start a proof

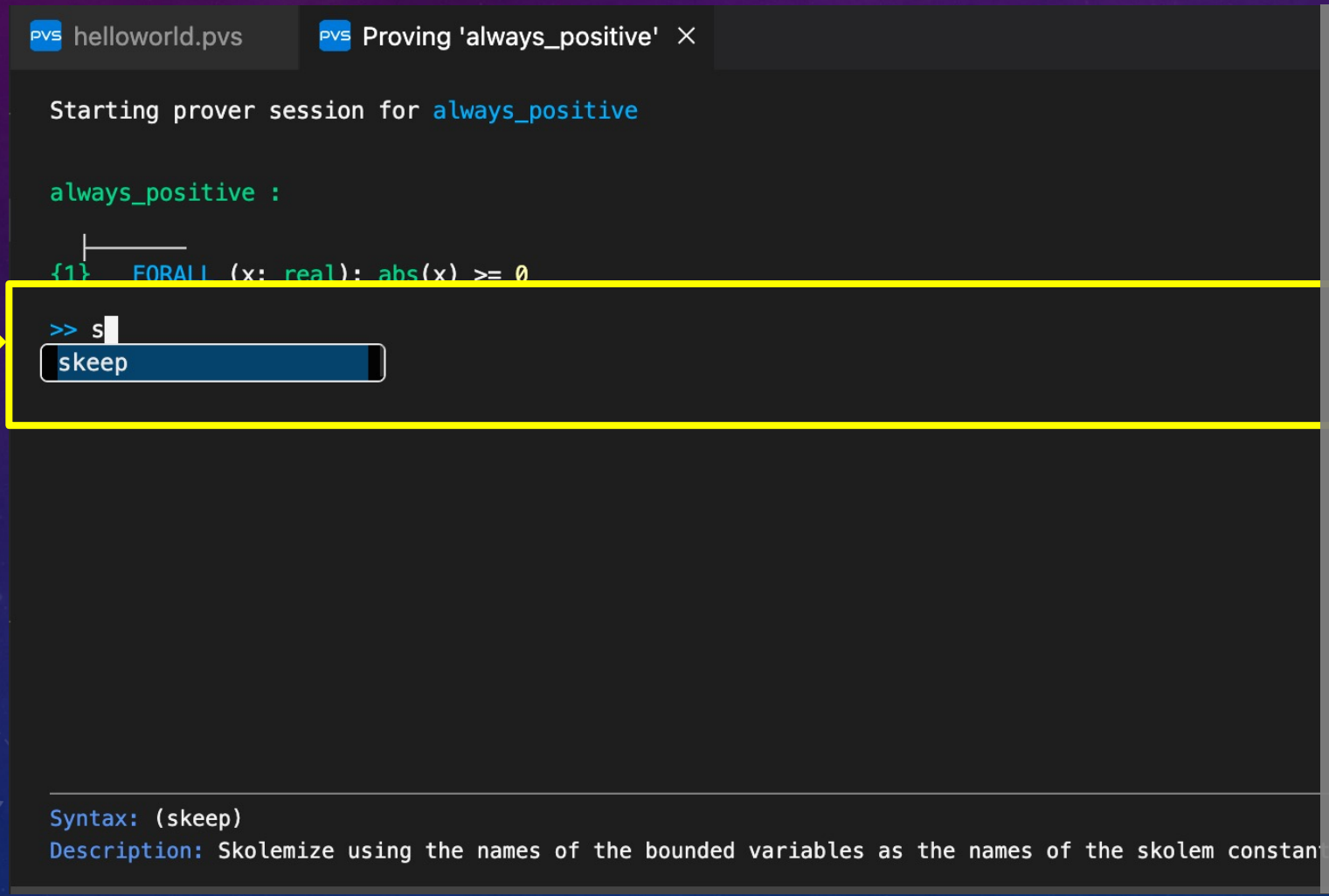
PROVING A THEOREM

Proof Explorer + Proof Mate

Prover Console



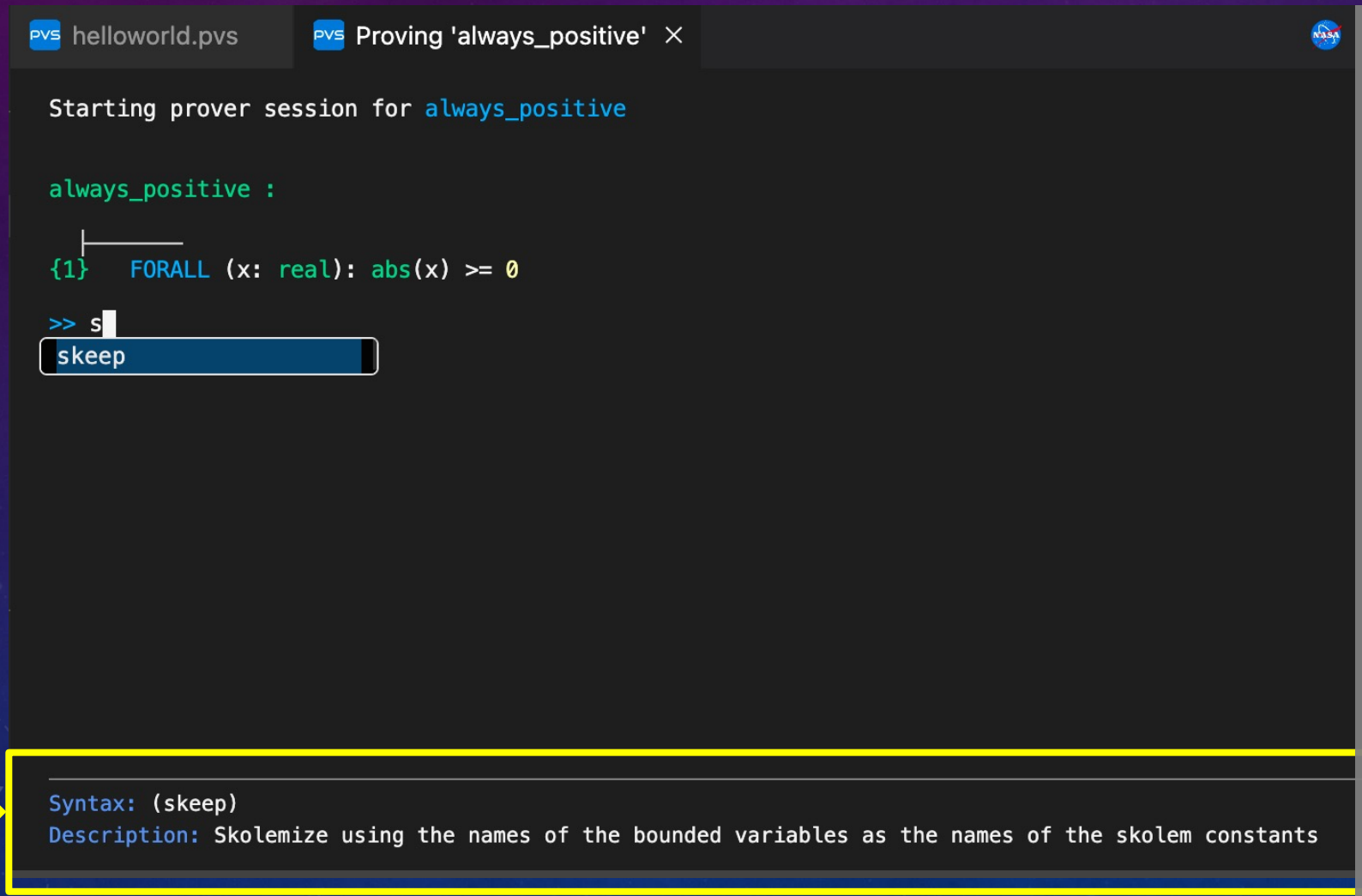
ENTERING PROOF COMMANDS



Proof commands are entered in the Prover Console

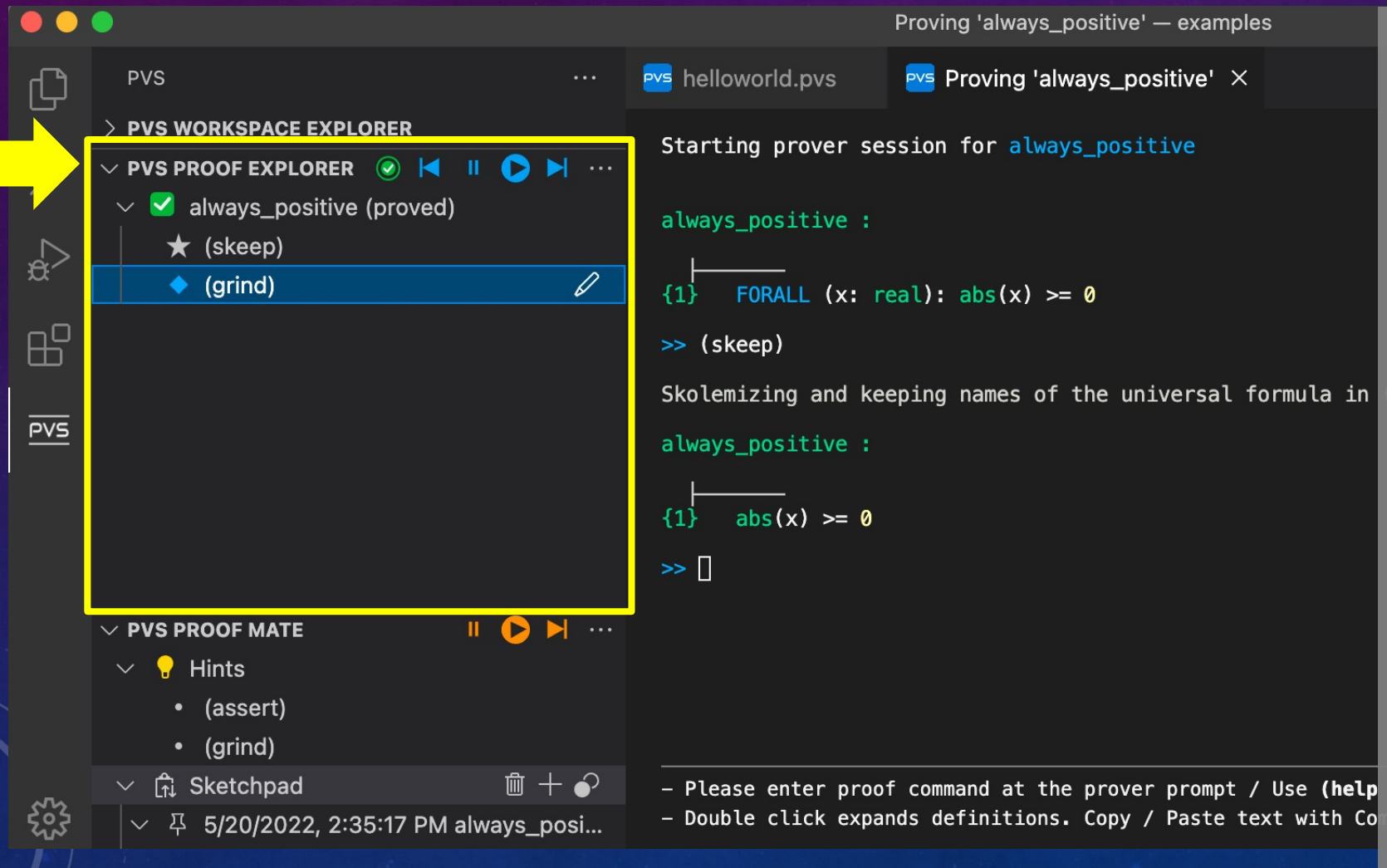
Use TAB to autocomplete proof commands

INTEGRATED HELP



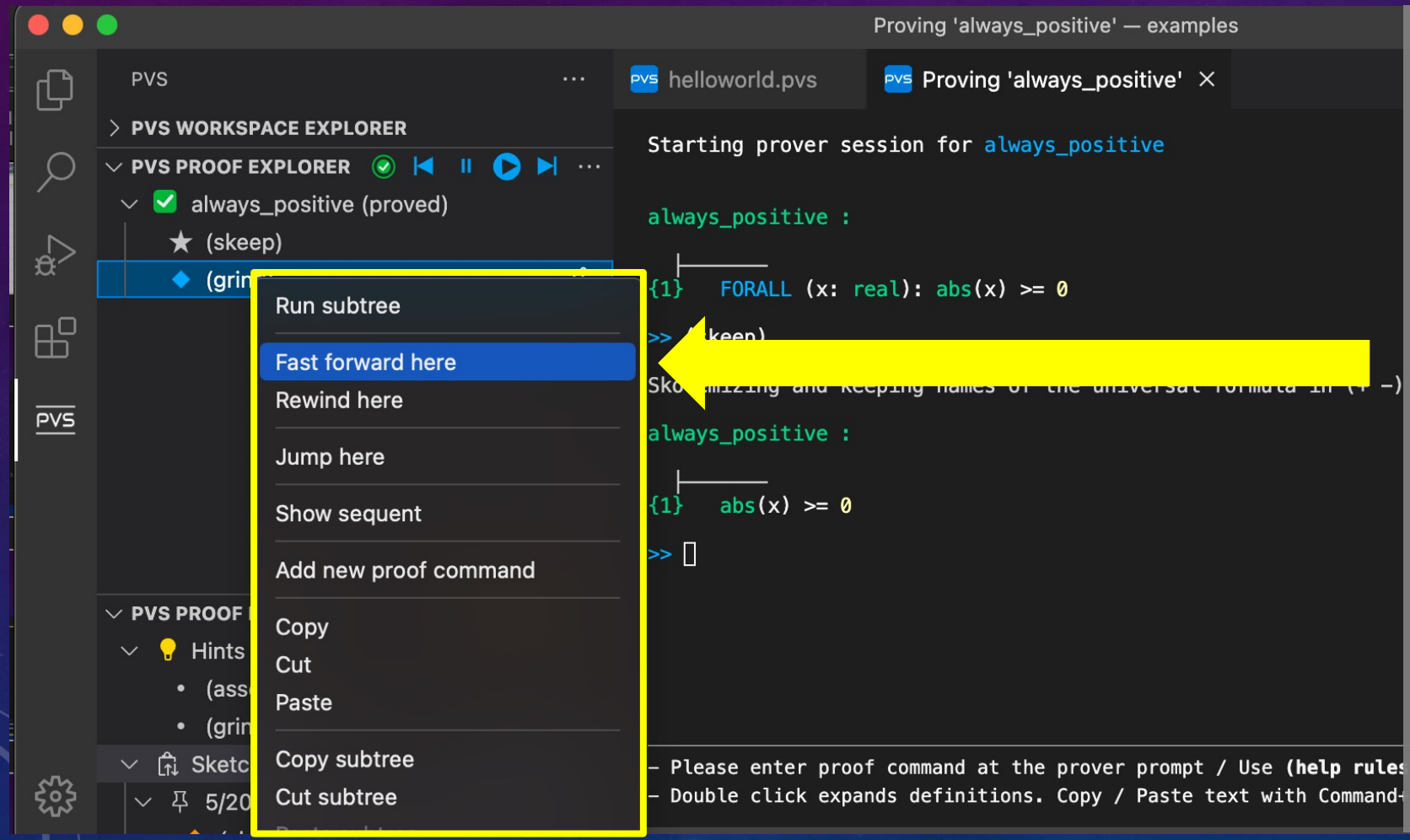
Command syntax and a brief description of the command is shown at the bottom of the Prover Console

PROOF EXPLORER



Proof Explorer shows the proof tree

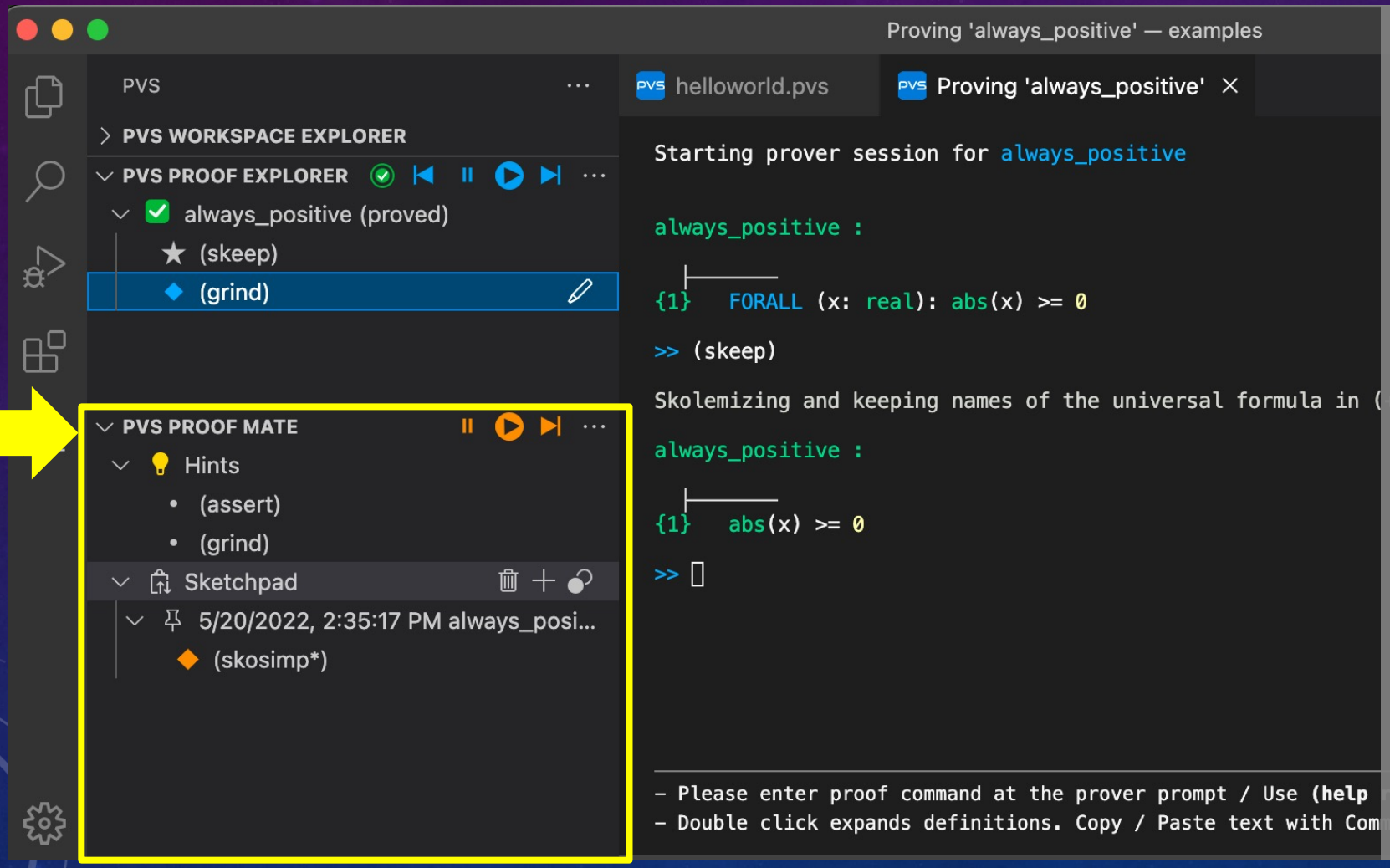
PROOF EXPLORER / CONTEXT MENU



Point-and-click actions can be used for undo/redo, fast-forward, re-play

Edit and copy/paste operations are restricted

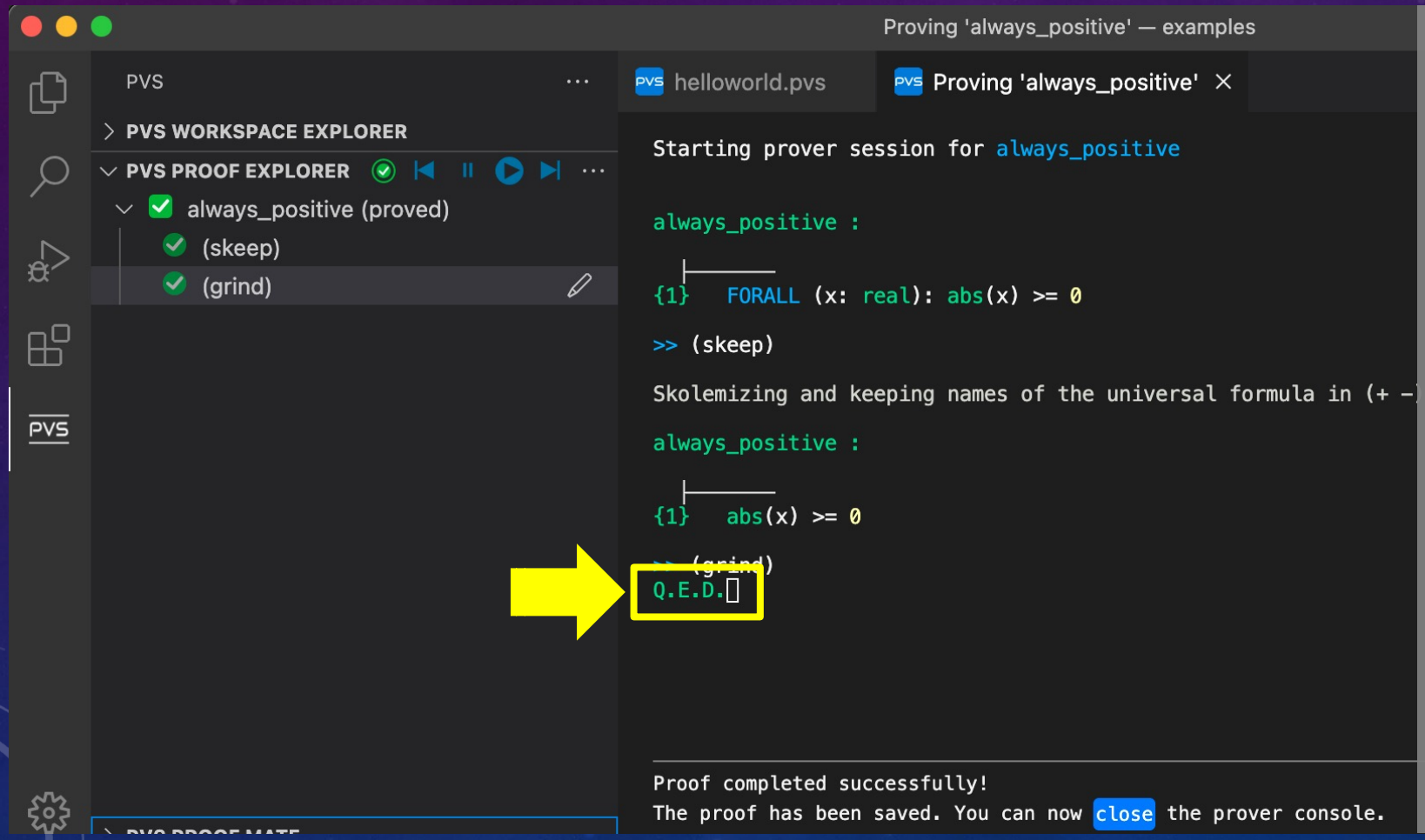
PROOF MATE



Proof Mate supports unrestricted editing and copy/paste

Hints suggest proof commands

PROOF COMPLETE!



A message 'Q.E.D.' in the Prover Console indicates proof complete

A @QED comment is automatically added to the corresponding theorem in the .pvs file

DOCUMENTING YOU PVS FILES

VSCode-PVS provides functionalities to support documentation of theories

- @QED annotation for proved theorems
- Header annotations for new theories
- Pretty-printing of comments written in the markdown language

USING MARKDOWN SYNTAX IN THE COMMENTS

```
typecheck-file | evaluate-in-pvsio | view-as-markdown
helloworld: THEORY
BEGIN

  % utility function, computes the absolute value of a number
  abs (x: real): real = IF x > 0 THEN x ELSE -x ENDIF

  % @QED always_positive proved by pmasci on Fri, 14 Jan 2022 13:31:5
  prove | status-proofchain | show-prooflite
  always_positive: THEOREM
    FORALL (x: real): abs(x) >= 0

  % Example markdown expressions that can be used in the documentatio
  % $\sum_{i=1}^n X_i$
  % $k_{n+1}$
  % $n^2$
  % $k_n^2$
  % $\frac{4z^3}{16}$
  % $\frac{n!}{k!(n-k)!}$
  % $\binom{n}{k}$
  % $\frac{\frac{x}{1}}{x - y}$
  % $^3/_7$
  % $\sqrt{k}$
  % $\sqrt[n]{k}$
  % $\sum_{i=1}^{10} t_i$
  % $\int_0^\infty \mathrm{e}^{-x}\,\mathrm{d}x$
  % $\hat{a}$
  % $\bar{a}$
  % $\dot{a}$
  % $\ddot{a}$
  % $\overrightarrow{AB}$
  open(sqrt_newton(n)) | view-as-markdown
  % ![sqrt_newton(n)](sqrt_newton_n.png)

END helloworld
```

Example markdown expressions that can be used in the theory:

- Math equations
- Inline links
- Figures

MARKDOWN SYNTAX / LIVE PREVIEW

typecheck-file | evaluate-in-pvs | view-as-markdown

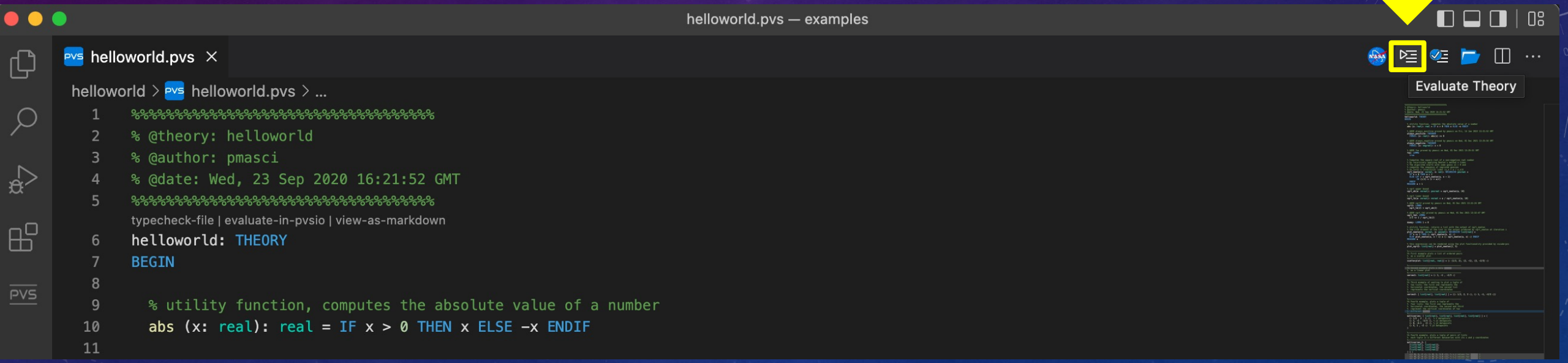
```
6   helloworld: THEORY
7   BEGIN
8
9     % utility function, compute the absolute value of a number
10    abs (x: real): real = IF x > 0 THEN x ELSE -x ENDIF
11
12    % @QED always_positive proved by pmasci on Fri, 14 Jan 2022 13:31:52 GMT
13    always_positive: THEOREM
14    FORALL (x: real): abs(x) >= 0
15
16    % Example markdown expressions that can be used in the documentation
17    % $\sum_{i=1}^n X_i$
18    % $k_{n+1}$
19    % $n^2$
20    % $k_n^2$
21    % $\frac{4z^3}{16}$
22    % $\frac{n!}{k!(n-k)!}$
23    % $\binom{n}{k}$
24    % $\frac{\frac{x}{1}}{x - y}$
25    % $^3/_7$
26    % $\sqrt{k}$
27    % $\sqrt[n]{k}$
28    % $\sum_{i=1}^{10} t_i$
29    % $\int_0^\infty \mathrm{e}^{-x}\,\mathrm{d}x$
30    % $\hat{a}$
31    % $\bar{a}$
32    % $\dot{a}$
33    % $\ddot{a}$
34    % $\overrightarrow{AB}$
      open(sqrt_newton(n)) | view-as-markdown
35    % ![sqrt_newton(n)](sqrt_newton_n.png)
36
```

Click 'view-as-markdown' to view the pretty-printed version

```
helloworld: THEORY
BEGIN

% utility function, computes the absolute value of a number
abs (x: real): real = IF x > 0 THEN x ELSE -x ENDIF

% @QED always_positive proved by pmasci on Fri, 14 Jan 2022 13:31:52 GMT
always_positive: THEOREM
FORALL (x: real): abs(x) >= 0

% Example markdown expressions that can be used in the documentation
% $\sum_{i=1}^n X_i$
% $k_{n+1}$
% $n^2$
% $k_n^2$
% $\frac{4z^3}{16}$
% $\frac{n!}{k!(n-k)!}$
% $\binom{n}{k}$
% $\frac{\frac{x}{1}}{x - y}$
% $^3/_7$
% $\sqrt{k}$
% $\sqrt[n]{k}$
% $\sum_{i=1}^{10} t_i$
% $\int_0^\infty \mathrm{e}^{-x}\,\mathrm{d}x$
% $\hat{a}$
% $\bar{a}$
% $\dot{a}$
% $\ddot{a}$
% $\overrightarrow{AB}$
```

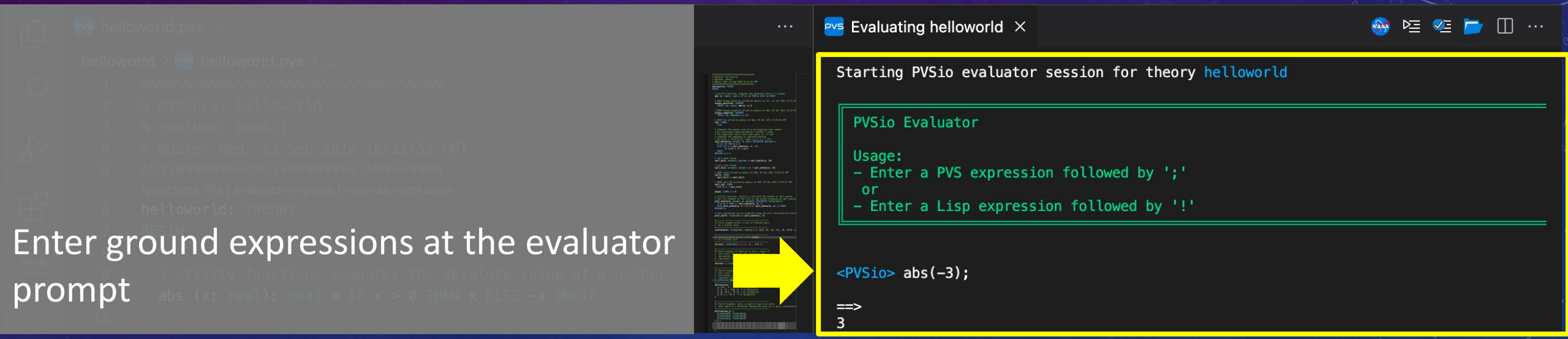
EVALUATING PVS EXPRESSIONS

Click the 'play' icon in the editor toolbar



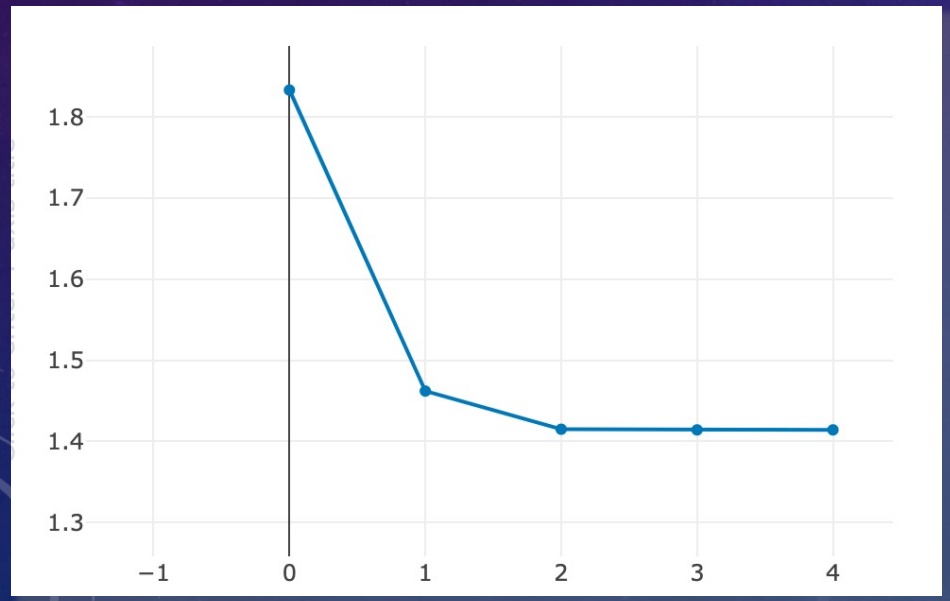
EVALUATING PVS EXPRESSIONS

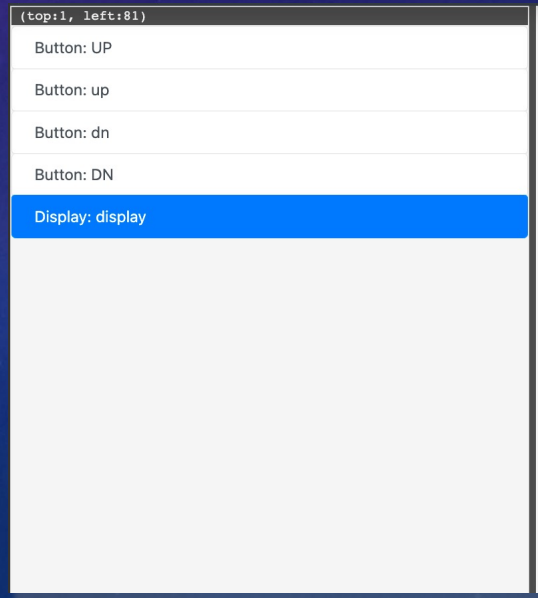
Enter ground expressions at the evaluator prompt



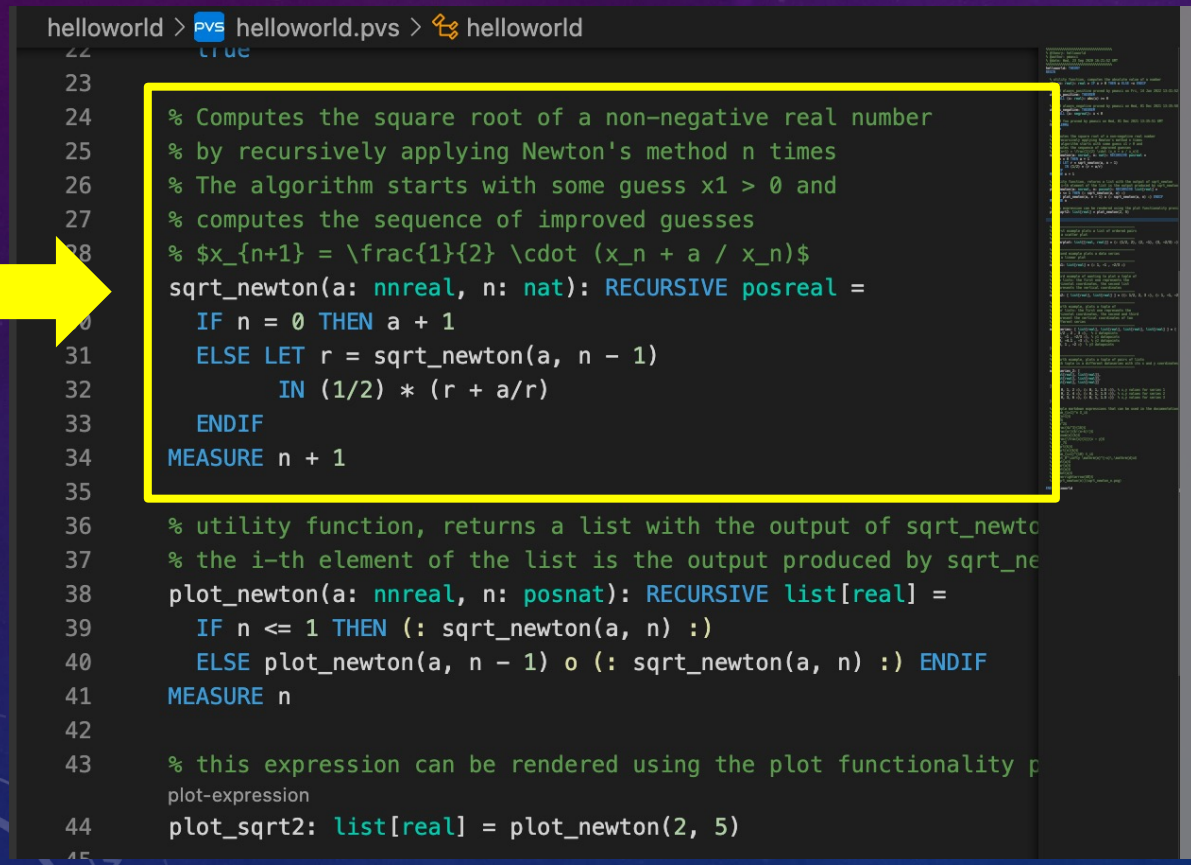
PLOTTING + RAPID PROTOTYPING

VSCode-PVS extends the evaluation mechanism of PVS with front-ends for plotting functions and creating interactive prototypes





PLOTTING FUNCTIONS



Define a function, e.g., 'sqrt_newton'

PLOTTING FUNCTIONS

helloworld > helloworld.pvs > helloworld

```
23
24      % Computes the square root of a non-negative real number
25      % by recursively applying Newton's method n times
26      % The algorithm starts with some guess x1 > 0 and
27      % computes the sequence of improved guesses
28      % $x_{n+1} = \frac{1}{2} \cdot (x_n + a / x_n)$
29      sqrt_newton(a: nnreal, n: nat): RECURSIVE posreal =
30        IF n = 0 THEN a + 1
31        ELSE LET r = sqrt_newton(a, n - 1)
32             IN (1/2) * (r + a/r)
33        ENDIF
34      MEASURE n + 1
35
36      % utility function, returns a list with the output of sqrt_newto
37      % the i-th element of the list is the output produced by sqrt_ne
38      plot_newton(a: nnreal, n: posnat): RECURSIVE list[real] =
39        IF n <= 1 THEN (: sqrt_newton(a, n) :)
40        ELSE plot_newton(a, n - 1) o (: sqrt_newton(a, n) :) ENDIF
41      MEASURE n
42
43      % this expression can be rendered using the plot functionality p
        plot-expression
44      plot_sqrt2: list[real] = plot_newton(2, 5)
```

Create a function that evalutes sqrt_newton and stores the results into a list of reals

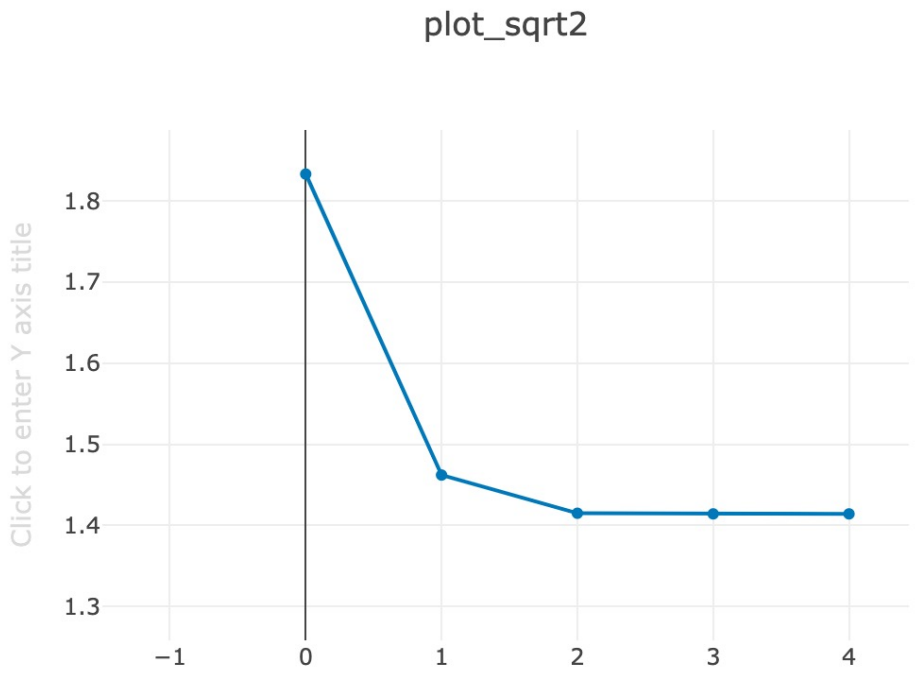
PLOTTING FUNCTIONS

Plot the saved list of reals by clicking the inline command 'plot-expression'

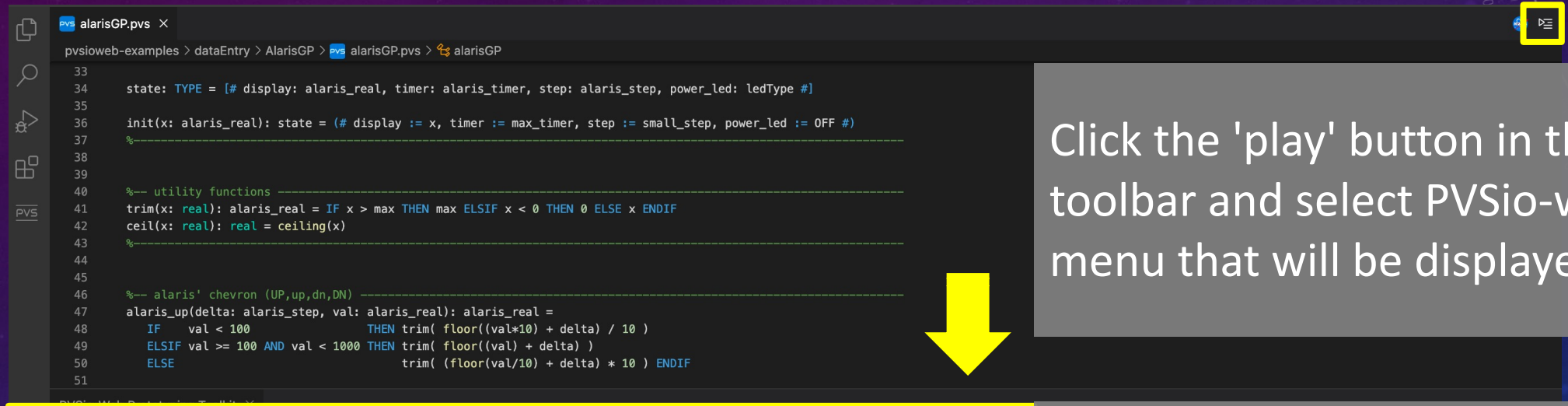
```
% this expression can be rendered using the plot functionality p
plot-expression
plot_sqrt2: list[real] = plot_newton(2, 5)
```

(: 11/6, 193/132, 72097/50952, 10390190017/7346972688, 215912063945802350977/152672884556058511392 :)

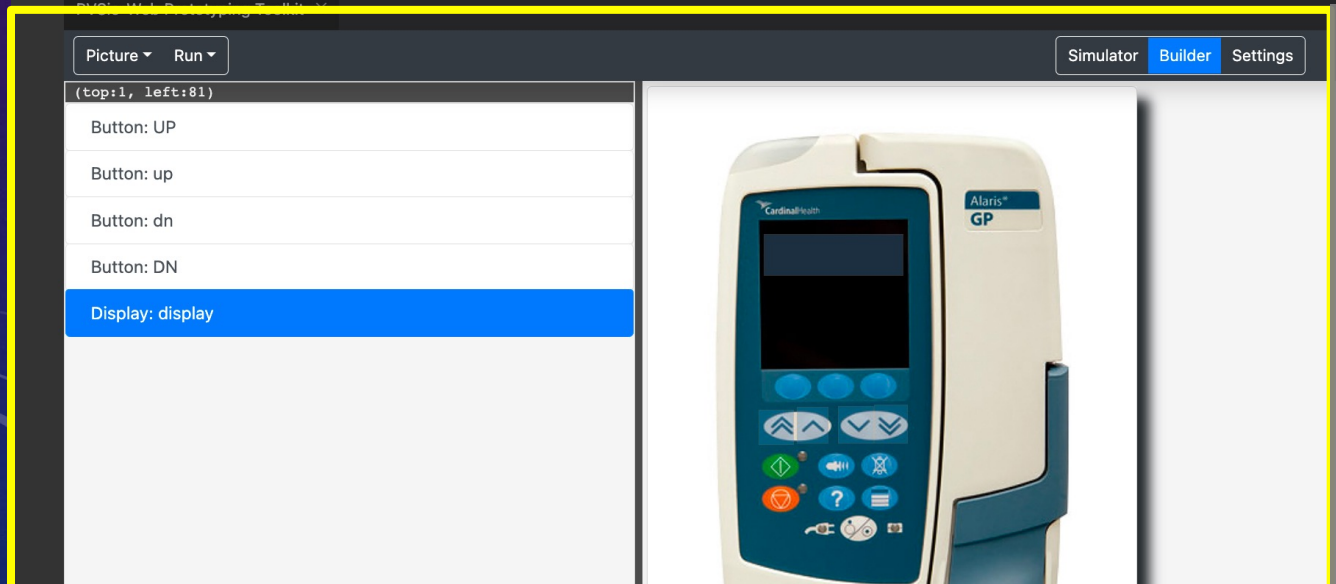
Linear SemiLog LogLog



CREATING INTERACTIVE PROTOTYPES



Click the 'play' button in the editor toolbar and select PVSio-web from the menu that will be displayed



A new panel will be opened where you can build and simulate a prototype driven by the PVS theory

Examples: https://github.com/pvsioweb/examples/

COMMAND SHORTCUTS

Commands can be triggered with keyboard shortcuts initiated with the sequence M-x, where M is the META key (Alt on Linux, Option (⌥) on Mac)

Frequent Commands

M-x tc *(typecheck file)*
M-x tcp *(typecheck file and re-run all proofs)*
M-x show-tccs *(show proof obligations)*
M-x parse *(parse file)*
M-x pr *(prove formula at the cursor location)*
M-x prt *(prove theory, i.e., re-run all proofs)*
M-x pri *(prove importchain, i.e., re-run all proofs including those in imported theories)*
M-x pvsio *(start PVSio)*
M-x x-show-proof *(shows proof tree)*
M-x show-proof-summary *(show proof summary)*
M-x show-prooflite *(show prooflite script)*
M-x insert-prooflite-script *(insert prooflite script at cursor location)*
M-x status-proof-chain *(status proof chain)*
M-x vpf *(view prelude file)*

Additional Commands

M-x add-pvs-library *(add a folder to the vscode-pvs library path)*
M-x pvs-library-path *(show pvs library)*
M-x reset-pvs-library-path *(resets the vscode-pvs library path to empty)*
M-x reboot-pvs *(reboot pvs-server)*
M-x clean-bin *(remove pvsbin files created by pvs)*
M-x clean-tccs *(remove .tccs files created by pvs)*
M-x clean-all *(remove temporary files, including .tccs and pvsbin)*
M-x install-pvs *(install or update PVS)*
M-x install-nasalib *(install NASALib)*
M-x update-nasalib *(update the installed version of NASALib)*
M-x set-pvs-path *(sets the path to the PVS executables)*
M-x settings *(shows vscode-pvs settings)*
M-x welcome *(shows vscode-pvs welcome screen)*

KEY RESOURCES

Examples presented in this tutorial

- https://github.com/nasa/vscode-pvs/tree/master/vscode-pvs/examples
- https://github.com/pvsioweb/examples

Tools

- VSCode-PVS (source code, user manual, tutorials): https://github.com/nasa/vscode-pvs
- PVS Allegro (pvs language reference, documentation): https://pvs.csl.sri.com/
- Visual Studio Code (user interface guide): https://code.visualstudio.com/docs/getstarted/userinterface
- PVS Google Group: https://groups.google.com/g/pvs-group

Publications

- Paolo Masci and César Muñoz, An Integrated Development Environment for the Prototype Verification System, F-IDE Workshop, Electronic Proceedings in Theoretical Computer Science (EPTCS), Vol. 310, pp. 35-49, 2019
- Paolo Masci and Aaron Dutle, Proof Mate: an Interactive Proof Helper for PVS, NASA Formal Methods Symposium (NFM2022), Lecture Notes in Computer Science, Springer, 2022 (to appear)