INTRODUCING FRIENDLY

[LONG VERSION]

PyCon Education Summit - May 12, 2021

ANDRÉ ROBERGE

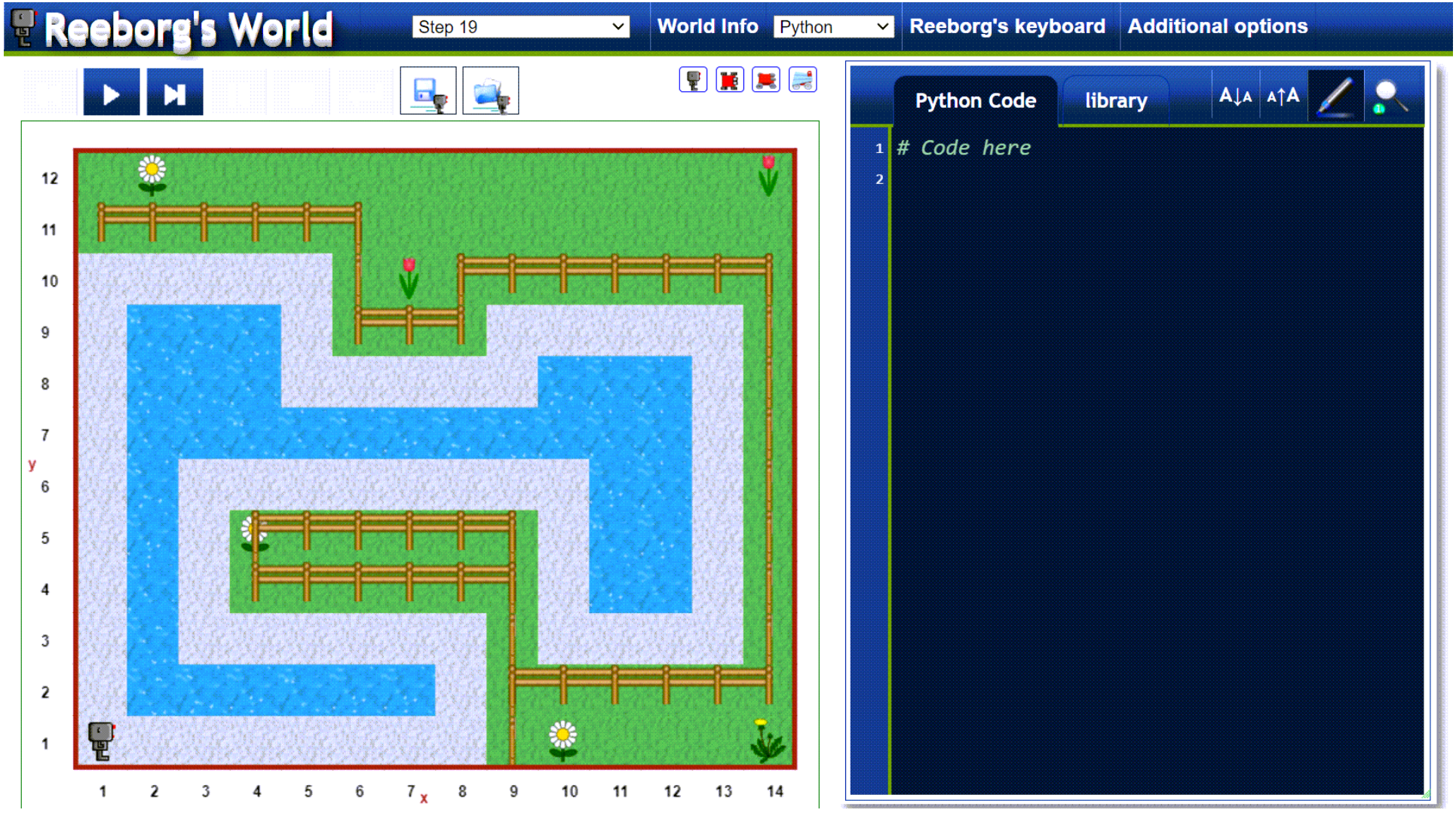
He/him

https://github.com/aroberge/friendly
https://github.com/aroberge/talks

who(am I?)

Python hobbyist trying since 2004 to help other people learn Python through various projects:

- RUR-PLE
- Reeborg's World
- Crunchy
- docpicture
- AvantPy
- **friendly**
- *etc.*


Reeborg's World
Step 19
World Info
Python
Reeborg's keyboard
Additional options
Python Code
library
Code here
y
x

`who( are you? )`

I assume that most of you have experience and/or interest in helping beginners learn Python.

`what( is friendly? )`

A tool that enhances Python tracebacks to make them easier to understand.

`what( is friendly? )`

Initial idea: given a traceback with an error message

- Add explanation in "simple English" about what the error message means.
- Make it posssible to translate this explanation into other languages (e.g. French).

friendly now does much more than that.

friendly used to be called **friendly-traceback.**

```
>>> import this
...
Explicit is better than implicit.
...
```

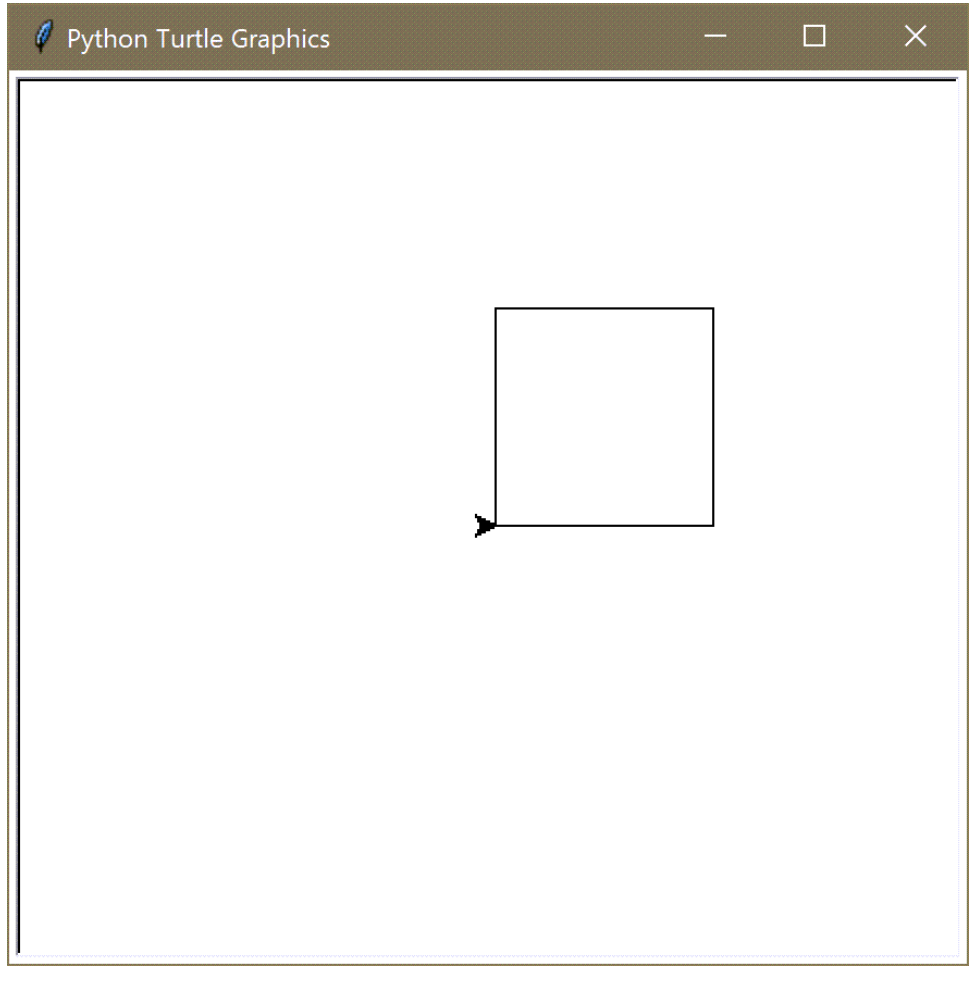
friendly-traceback is more explicit than **friendly**.

Why was the name changed?

"DEMO" 1

```
import turtle as t

for i in range(4):
    t.forward(100)
    t.left(90)
```

"DEMO" 1: EXPECTED

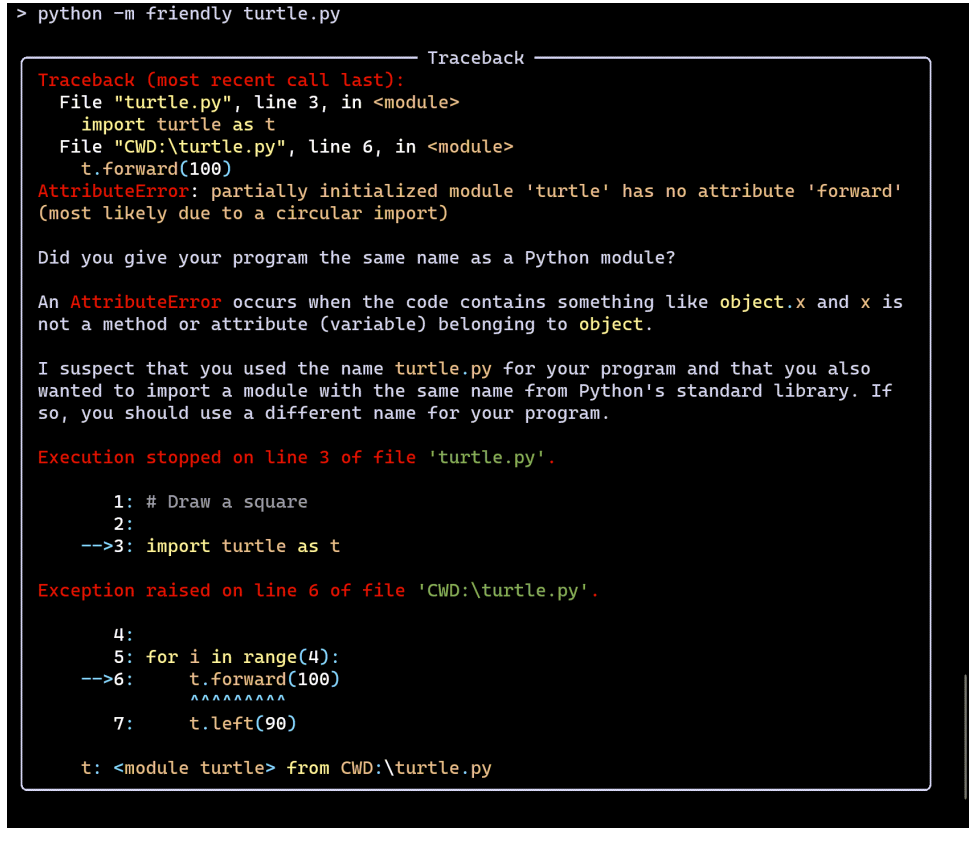


"DEMO" 1: RESULT

```
> python turtle.py

Traceback (most recent call last):
  File "turtle.py", line 3, in <module>
    import turtle as t
  File "...\turtle.py", line 6, in <module>
    t.forward(100)
AttributeError: partially initialized module
 'turtle' has no attribute 'forward'
(most likely due to a circular import)
```

"DEMO" 1: RUNNING WITH `friendly`

```
python -m friendly turtle.py
```



ASIDE: OBSERVATION

Running a program using an editor such as
IDLE, Mu, or Thonny,
is equivalent to doing

```
python -i my_program.py
```

in a terminal.

```
# my_program.py

answer = 42
print('Hello World!')
```

"Run" my_program.py

```
Hello World!
>>> answer
42
>>>
```

BACK TO "DEMO" 1

```
python -im friendly turtle.py
...

Friendly Console ...

>>>
```

"DEMO" 1: TRACEBACK AND hint()

```
> python -im friendly turtle.py

Traceback (most recent call last):
  File "turtle.py", line 3, in <module>
    import turtle as t
  File "CWD:\turtle.py", line 6, in <module>
    t.forward(100)
AttributeError: partially initialized module 'turtle' has no
attribute 'forward' (most likely due to a circular import)

Did you give your program the same name as a Python module?

Friendly Console version 0.3.45. [Python version: 3.8.4]

>>>
```

"DEMO" 1: what()

```
>>> what()

AttributeError: partially initialized module 'turtle' has no
attribute 'forward' (most likely due to a circular import)
```

An AttributeError occurs when the code contains something like object.x and x is not a method or attribute (variable) belonging to object.

"DEMO" 1: why()

```
>>> why()

I suspect that you used the name turtle.py for your program and
that you also wanted to import a module with the same name from
Python's standard library. If so, you should use a different name
for your program.
```

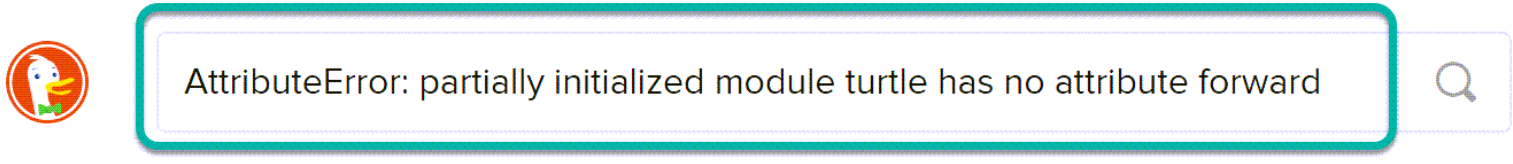
"DEMO" 1: where()

```
>>> where()

Execution stopped on line 3 of file 'turtle.py'.

      1: # Draw a square
      2:
   -->3: import turtle as t

Exception raised on line 6 of file 'CWD:\turtle.py'.

      4:
      5: for i in range(4):
   -->6:     t.forward(100)
             ^^^^^^^^^^
      7:     t.left(90)

   t: <module turtle> from CWD:\turtle.py
```

Only one or two frames are shown.

"DEMO" 1: www()

```
>>> www()
```



"DEMO" 1: what() IN FRENCH

```
>>> set_lang('fr')
>>> what()

AttributeError: partially initialized module 'turtle' has no
attribute 'forward' (most likely due to a circular import)

Une exception AttributeError se produit lorsque le code contient
quelque chose comme object.x et x n'est pas une méthode ou un
attribut (variable) appartenant à objet.
```

"DEMO" 1: what(...)

```
>>> set_lang('en')
>>> what(UnboundLocalError)

In Python, variables that are used inside a function are known as
local variables. Before they are used, they must be assigned a
value. A variable that is used before it is assigned a value is
assumed to be defined outside that function; it is known as a
global (or sometimes nonlocal) variable. You cannot assign a
value to such a global variable inside a function without first
indicating to Python that this is a global variable, otherwise
you will see an UnboundLocalError.
```

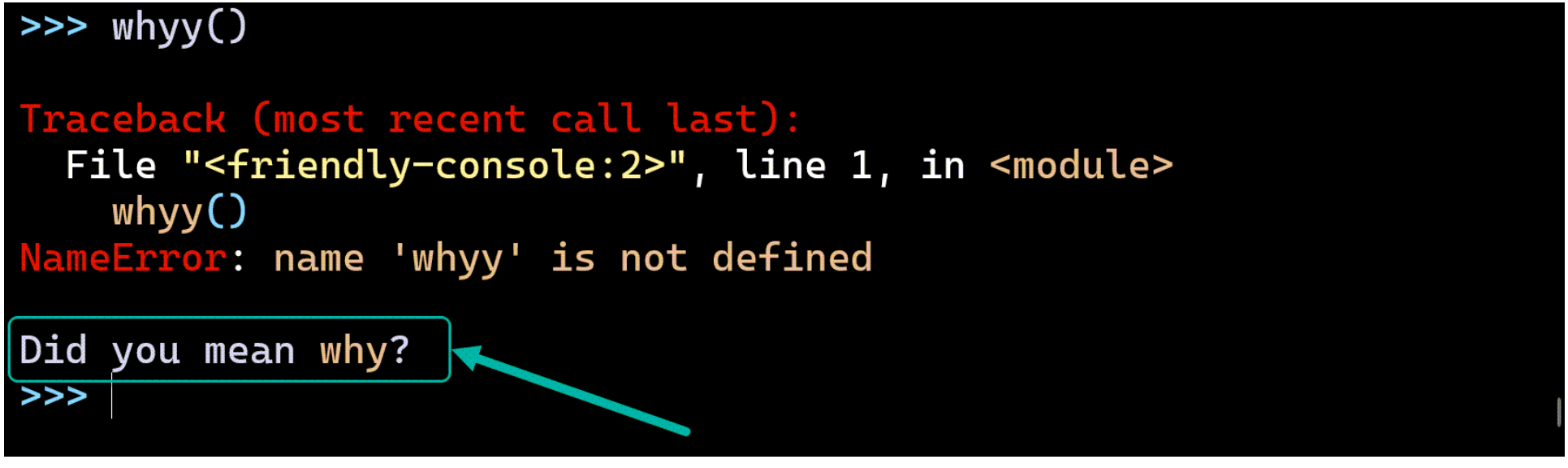
"DEMO 1": SUMMARY

We can use **friendly** to ask questions and obtain answers helping us understand what caused a given traceback.

`how_to( deal with typos? )`

Can a REPL be friendly?

```
Friendly Console version 0.3.45. [Python version: 3.10.0b1]

>>> if "word" := True:

Traceback (most recent call last):
  File "<friendly-console:1>", line 1
    if "word" := True:
       ^
SyntaxError: cannot use assignment expressions with literal

You can only assign objects to identifiers (variable names).
```


>>> whyy()
Traceback (most recent call last):
File "<friendly-console:2>", line 1, in <module>
whyy()
NameError: name 'whyy' is not defined
Did you mean why?
>>>

```
Did you mean why?
>>> why()

In your program, whyy is an unknown name. The similar name why
was found in the local scope.
>>>
```

```
>>> history()
SyntaxError: cannot use assignment expressions with literal
NameError: name 'whyy' is not defined
>>>
```

```
>>> back()
>>> history()
SyntaxError: cannot use assignment expressions with literal
>>>
```

```
>>> why()

You cannot use the augmented assignment operator :=, sometimes
called the walrus operator, with literals like "word". You can
only assign objects to identifiers (variable names).
```

how_to(install?)

```
Friendly Console version 0.3.45. [Python version: 3.8.4]

>>> python -m pip install friendly

Traceback (most recent call last):
  File "<friendly-console:1>", line 1
    python -m pip install friendly
               ^
SyntaxError: invalid syntax

Pip cannot be used in a Python interpreter.
>>> why()

It looks as if you are attempting to use pip to install a module.
pip is a command that needs to run in a terminal, not from a
Python interpreter.
```

Add support for IPython #126

Closed **aroberge** opened this issue on 30 Nov 2020 · 9 comments

Carreau commented on 30 Nov 2020

Great ! Thanks; if you ever get to it I would be ok to have friendly TB be automatically enabled in IPython if installed.

1

Next: "demos" with IDLE, Mu, and Jupyter Lab.

But first, an aside.

how_to(...)

Use **friendly** as an exception hook

```
from friendly[.X] import *

install(...)
```

or, in some cases, the following is enough

```
from friendly[.X] import *
```

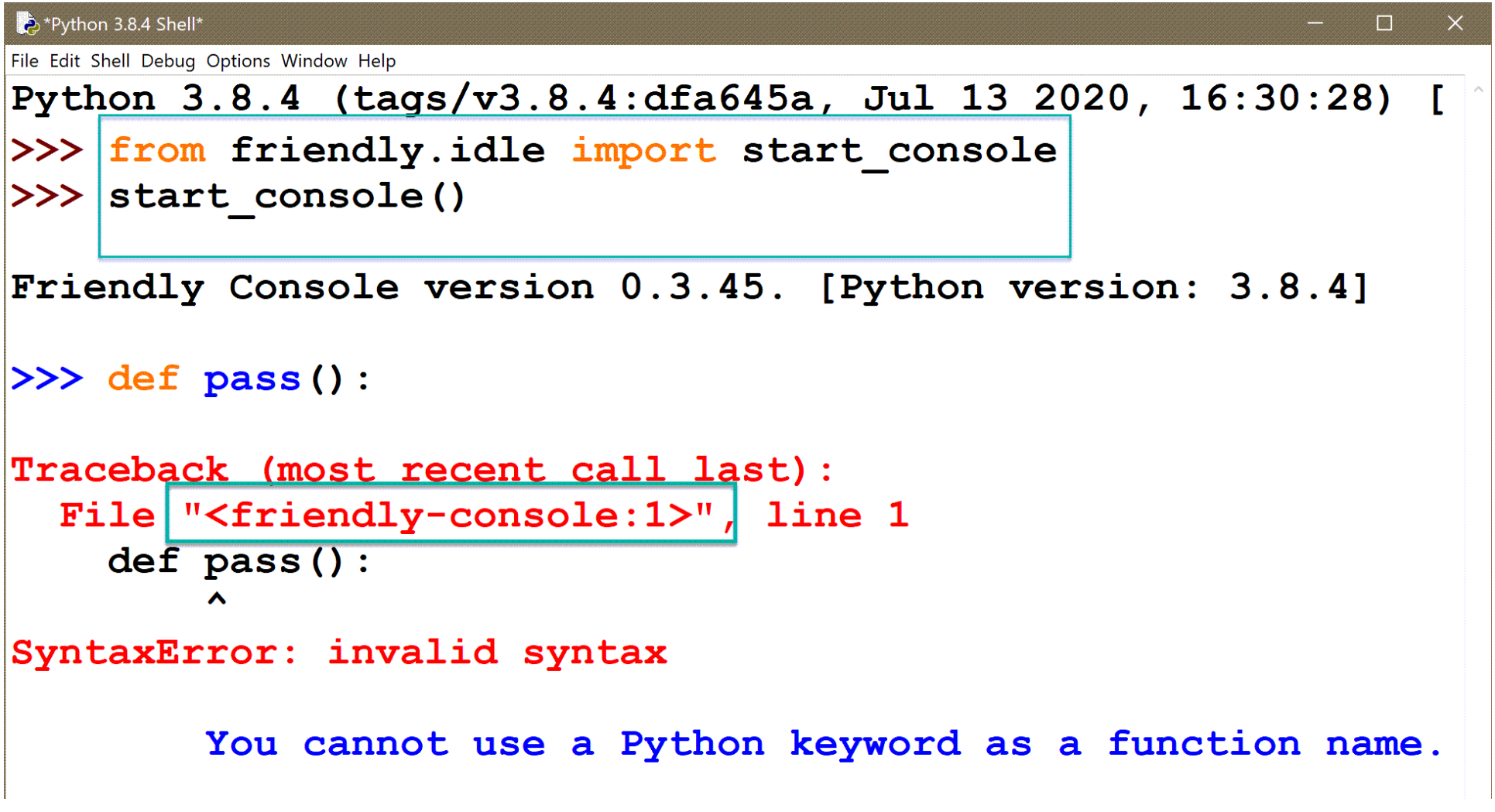
Replace a REPL

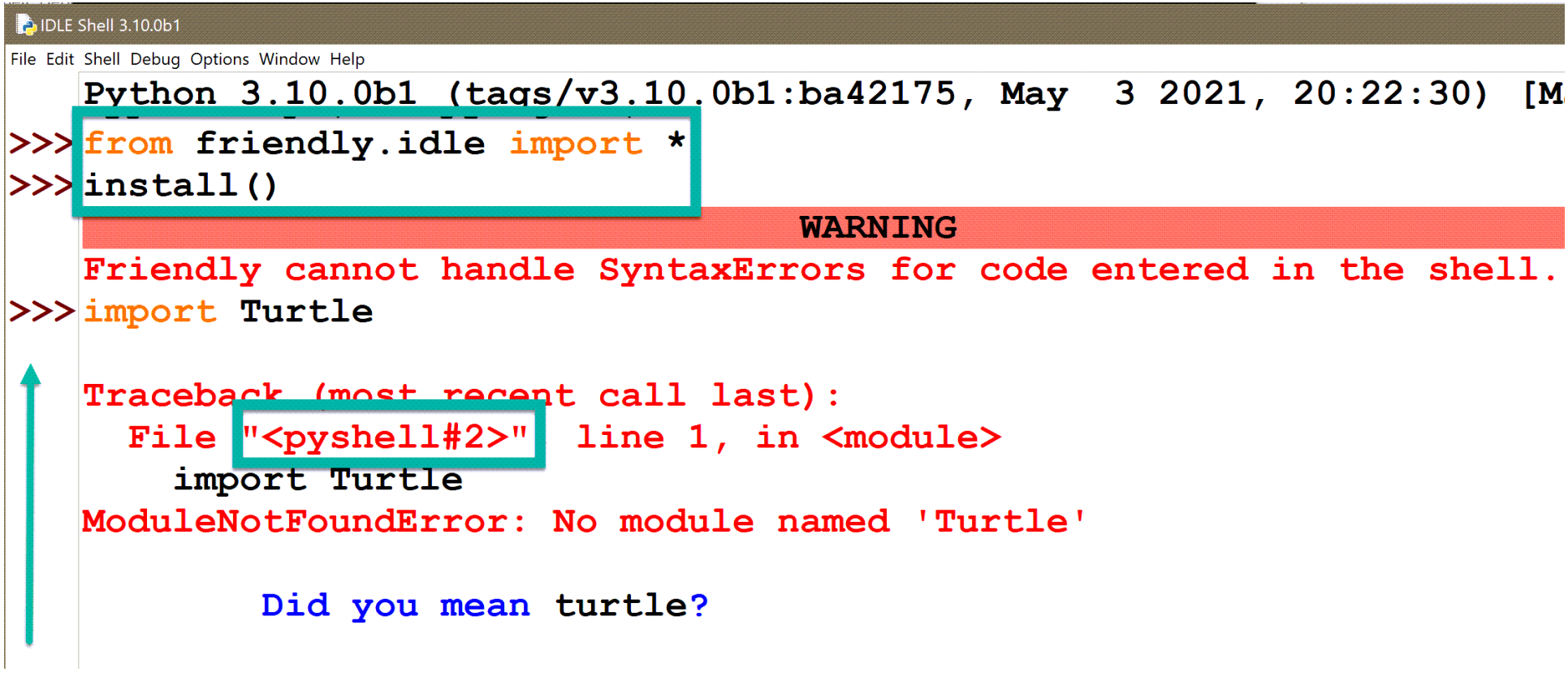
```
from friendly[.X] import start_console

start_console(...)
```

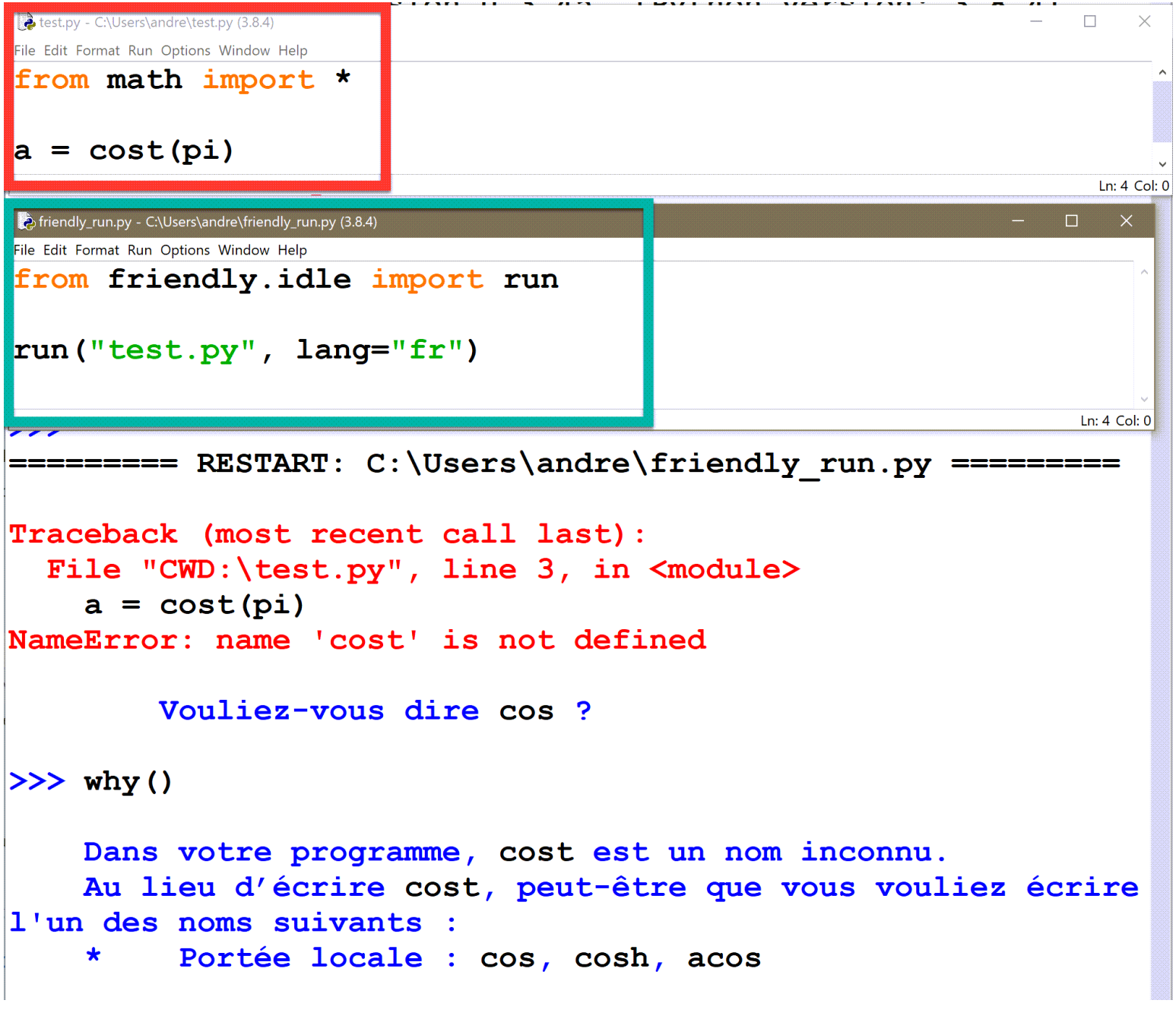
Run a program

```
# friendly_runner.py

from friendly[.X] import run

run("my_program.py", ...)
```

"DEMO"

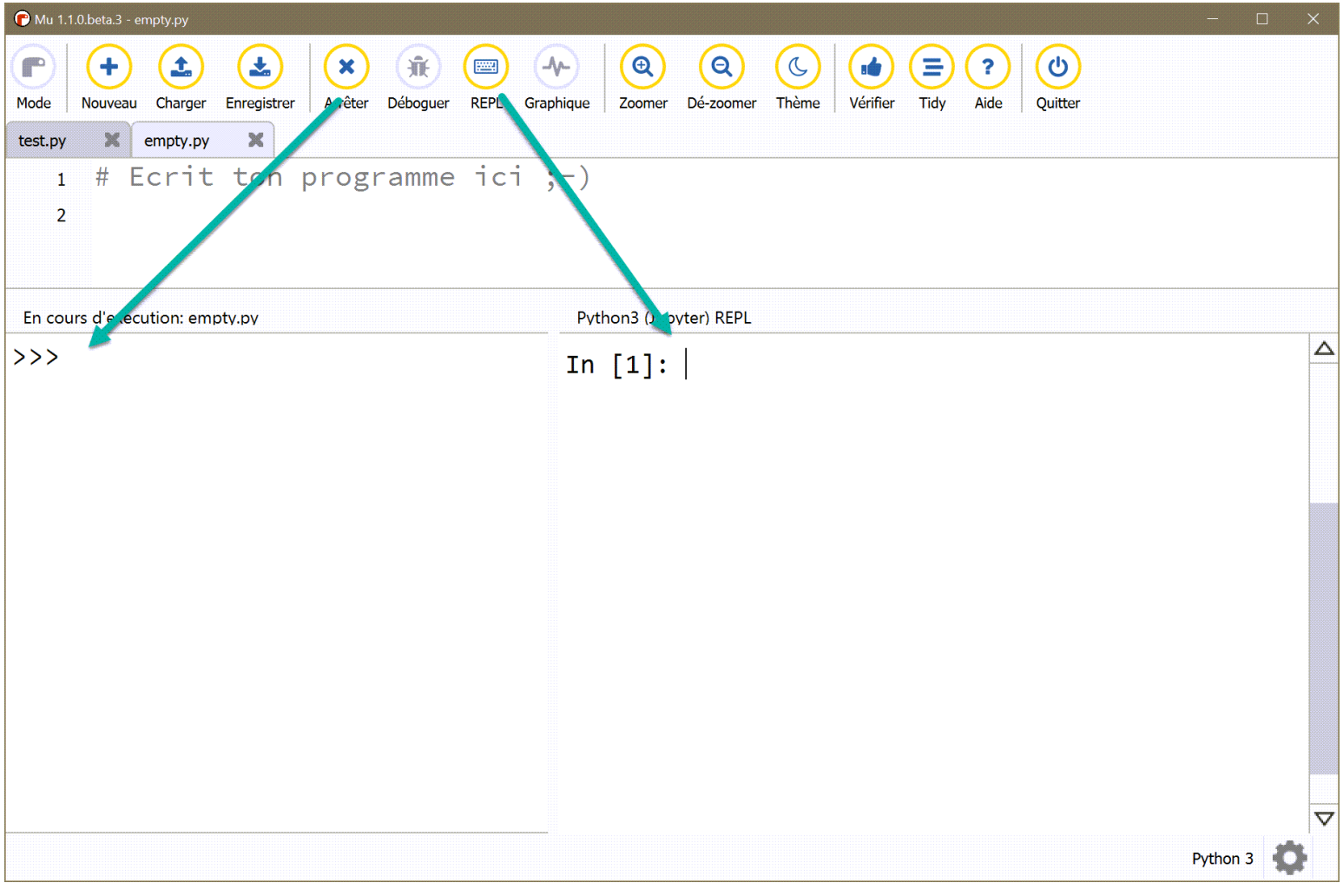

Python 3.8.4 Shell
File Edit Shell Debug Options Window Help
Python 3.8.4 (tags/v3.8.4:dfa645a, Jul 13 2020, 16:30:28) [
>>> from friendly.idle import start_console
>>> start_console()
Friendly Console version 0.3.45. [Python version: 3.8.4]
>>> def pass():
Traceback (most recent call last):
File "<friendly-console:1>", line 1
def pass():
^
SyntaxError: invalid syntax
You cannot use a Python keyword as a function name.

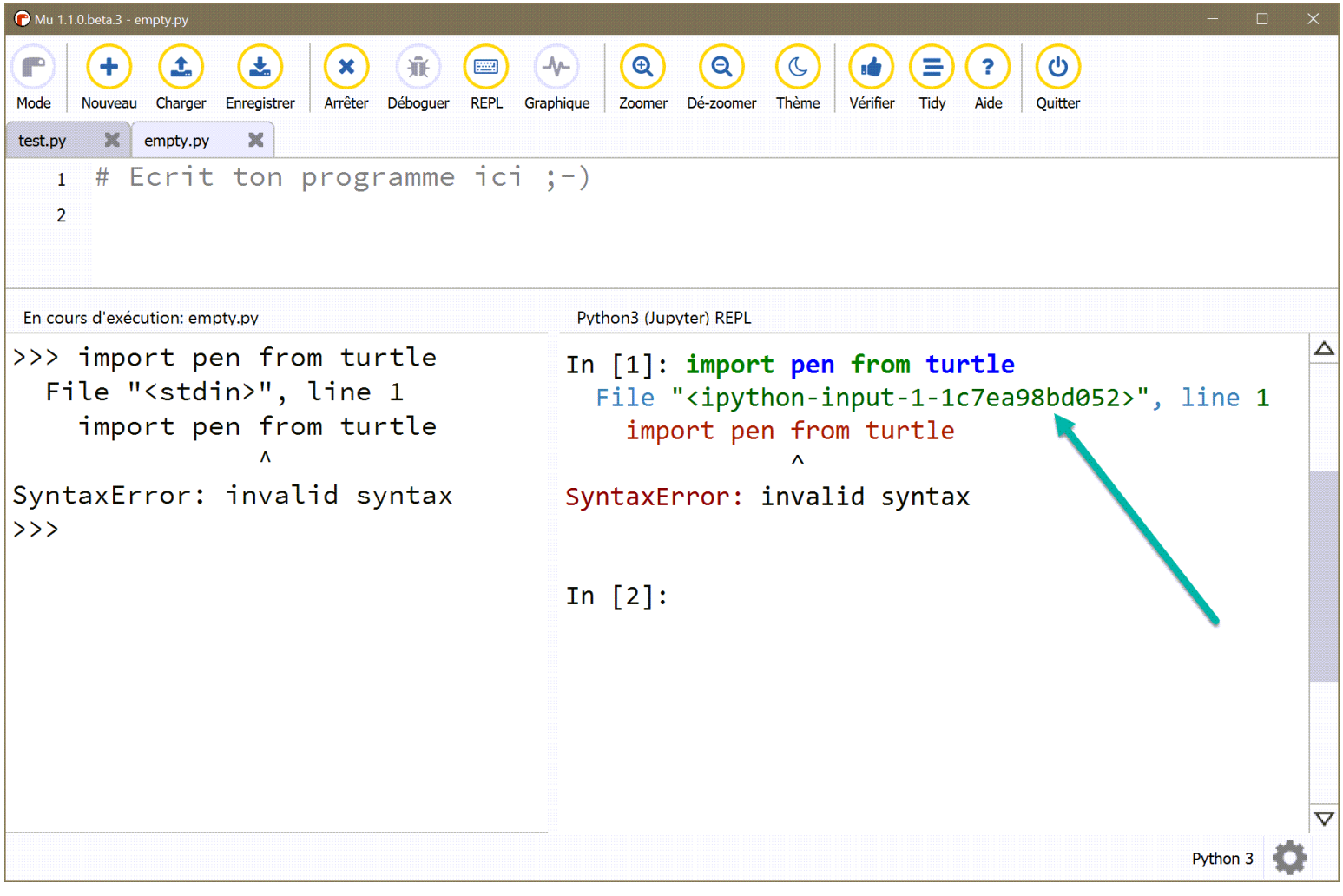

IDLE Shell 3.10.0b1
File Edit Shell Debug Options Window Help
Python 3.10.0b1 (tags/v3.10.0b1:ba42175, May 3 2021, 20:22:30) [M
>>> from friendly.idle import *
>>> install()
WARNING
Friendly cannot handle SyntaxErrors for code entered in the shell.
>>> import Turtle
Traceback (most recent call last):
File "<pyshell#2>" line 1, in <module>
import Turtle
ModuleNotFoundError: No module named 'Turtle'
Did you mean turtle?

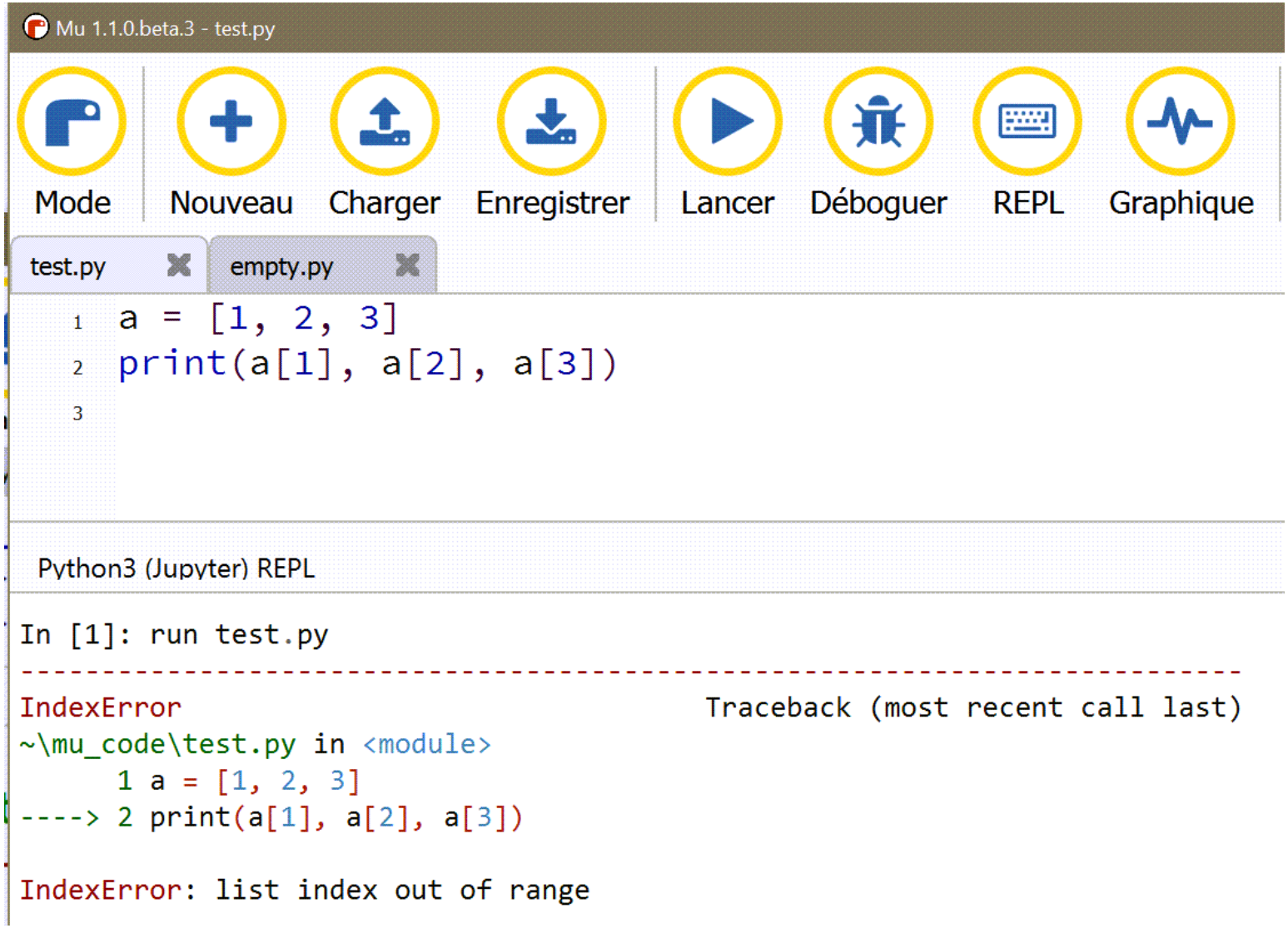

test.py - C:\Users\andre\test.py (3.8.4)
File Edit Format Run Options Window Help
from math import *
a = cost(pi)
Ln: 4 Col: 0
friendly_run.py - C:\Users\andre\friendly_run.py (3.8.4)
File Edit Format Run Options Window Help
from friendly.idle import run
run("test.py", lang="fr")
Ln: 4 Col: 0
>>>
========== RESTART: C:\Users\andre\friendly_run.py ==========
Traceback (most recent call last):
File "CWD:\test.py", line 3, in <module>
a = cost(pi)
NameError: name 'cost' is not defined
Vouliez-vous dire cos ?
>>> why()
Dans votre programme, cost est un nom inconnu.
Au lieu d'écrire cost, peut-être que vous vouliez écrire
l'un des noms suivants :
* Portée locale : cos, cosh, acos

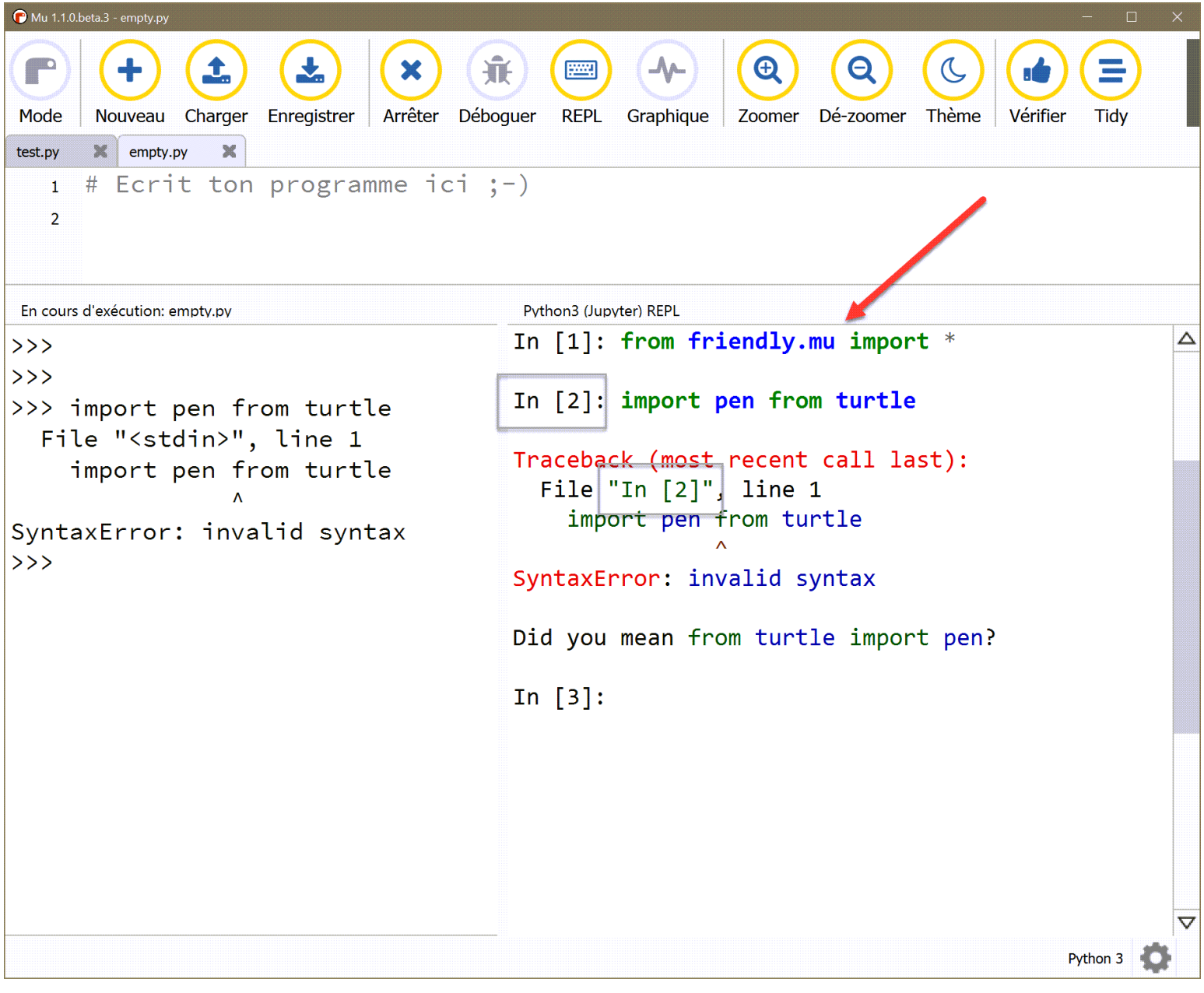
MU

"DEMO"

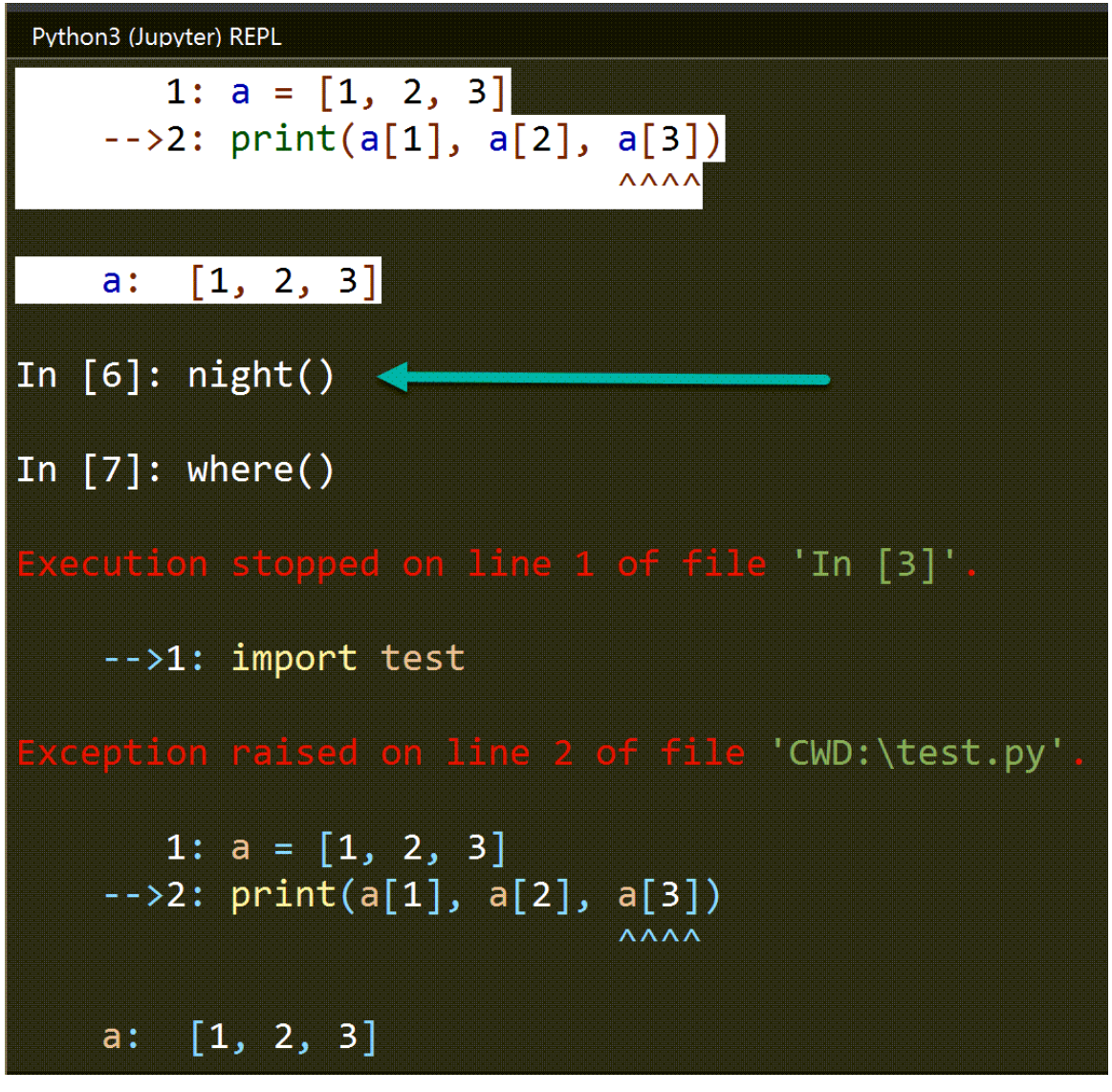

Mu 1.1.0.beta.3 - empty.py
Mode
Nouveau
Charger
Enregistrer
Arrêter
Déboguer
REPL
Graphique
Zoomer
Dé-zoomer
Thème
Vérifier
Tidy
Aide
Quitter
test.py
empty.py
1 # Ecrit ton programme ici ;-)
2
En cours d'exécution: empty.py
>>>
Python3 (Jupyter) REPL
In [1]:
Python 3

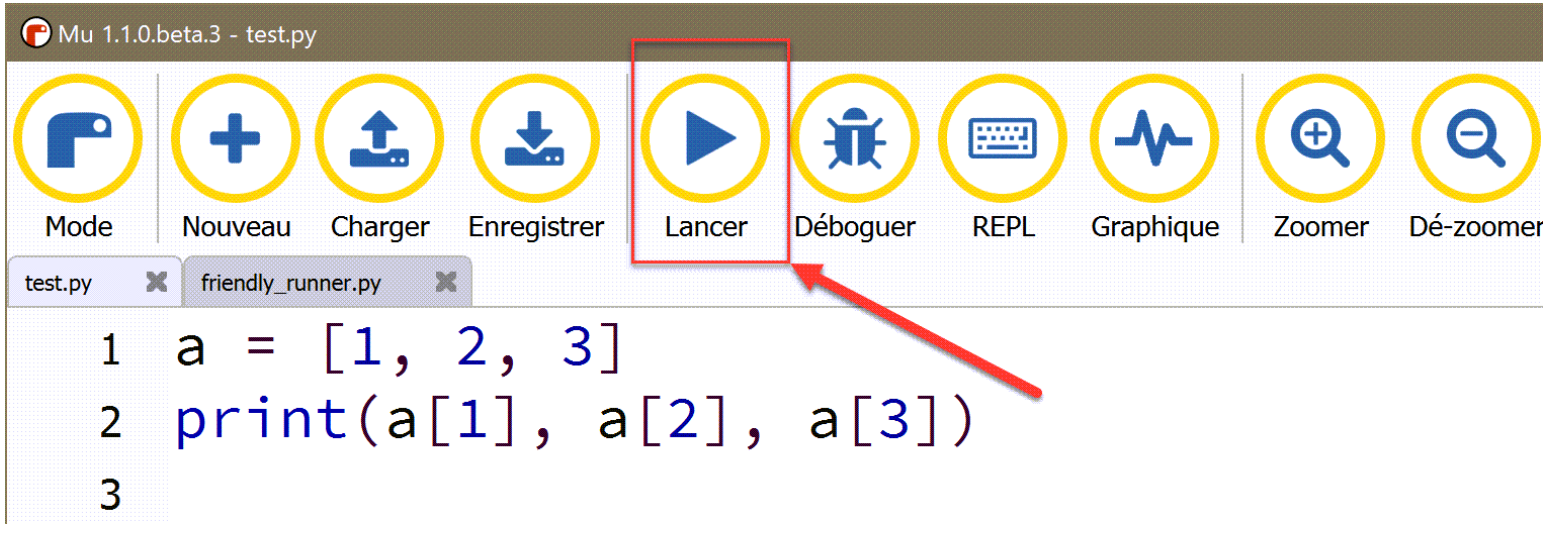

Mu 1.1.0.beta.3 - empty.py
Mode
Nouveau
Charger
Enregistrer
Arrêter
Déboguer
REPL
Graphique
Zoomer
Dé-zoomer
Thème
Vérifier
Tidy
Aide
Quitter
test.py
empty.py
1 # Ecrit ton programme ici ;-)
2
En cours d'exécution: empty.py
>>> import pen from turtle
File "<stdin>", line 1
import pen from turtle
^
SyntaxError: invalid syntax
>>>
Python3 (Jupyter) REPL
In [1]: import pen from turtle
File "<ipython-input-1-1c7ea98bd052>", line 1
import pen from turtle
^
SyntaxError: invalid syntax
In [2]:
Python 3

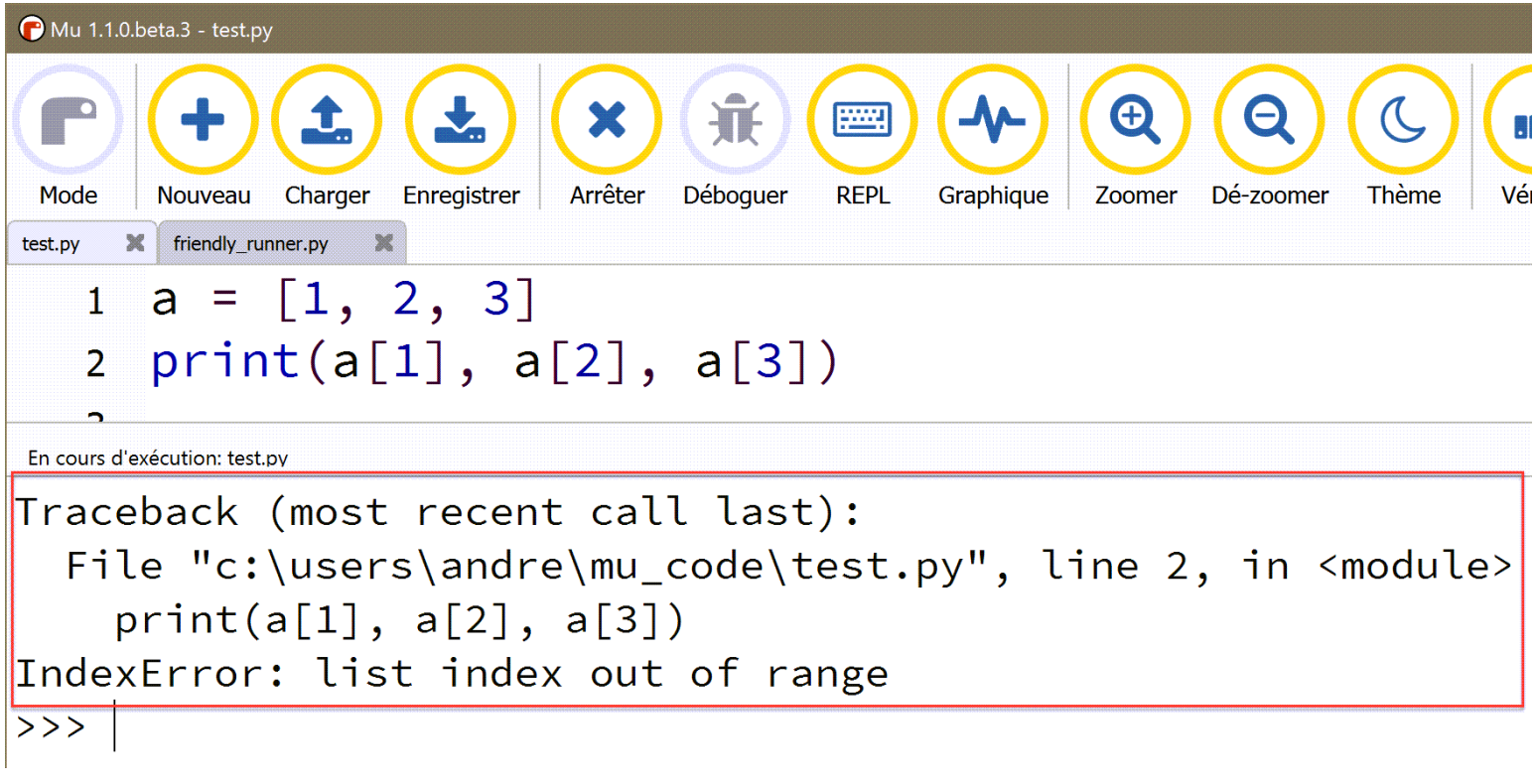

Mu 1.1.0.beta.3 - test.py
Mode
Nouveau
Charger
Enregistrer
Lancer
Déboguer
REPL
Graphique
test.py
empty.py
1 a = [1, 2, 3]
2 print(a[1], a[2], a[3])
3
Python3 (Jupyter) REPL
In [1]: run test.py
IndexError Traceback (most recent call last)
~\mu_code\test.py in <module>
1 a = [1, 2, 3]
----> 2 print(a[1], a[2], a[3])
IndexError: list index out of range

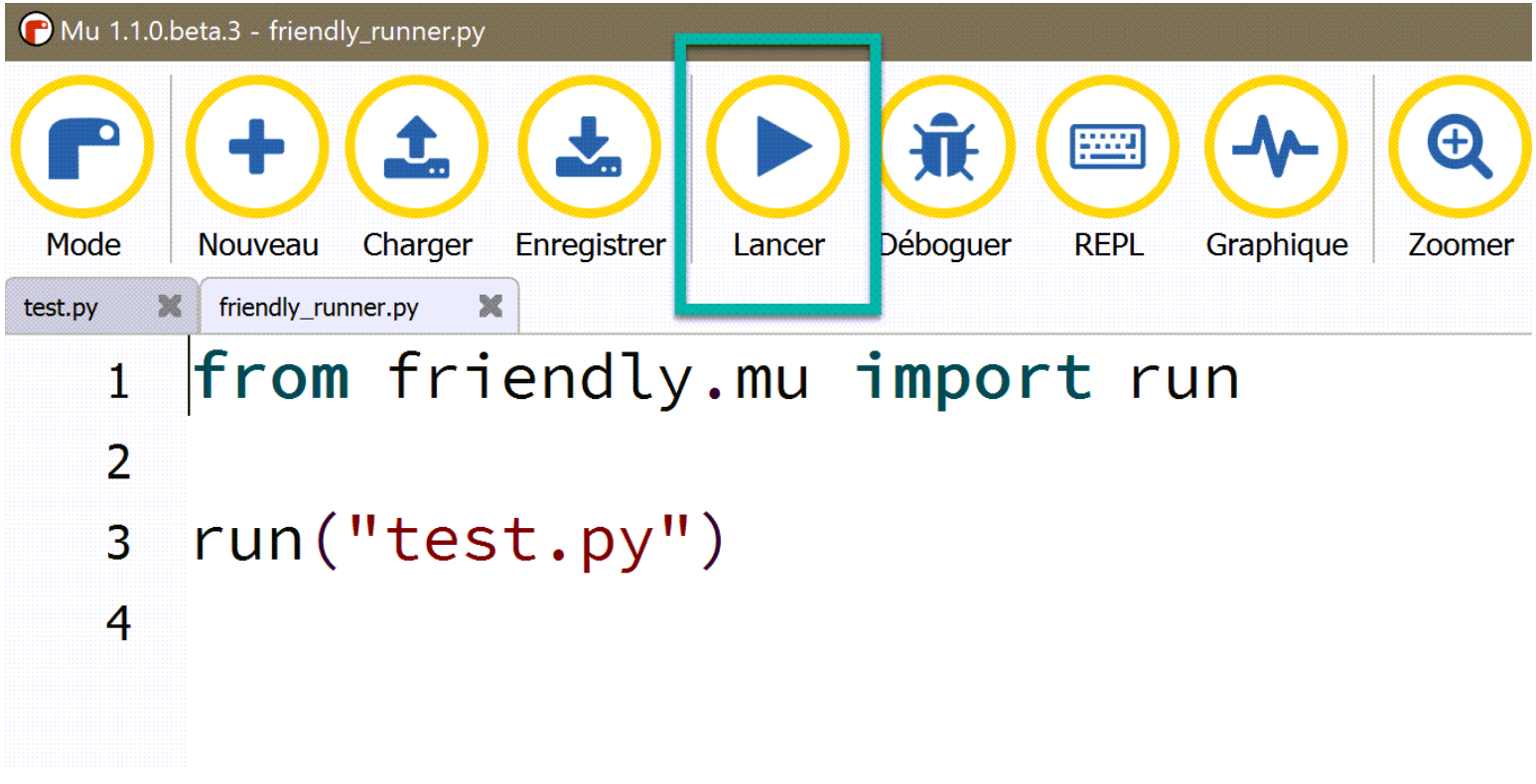

Mu 1.1.0.beta.3 - empty.py
Mode
Nouveau
Charger
Enregistrer
Arrêter
Déboguer
REPL
Graphique
Zoomer
Dé-zoomer
Thème
Vérifier
Tidy
test.py
empty.py
1 # Ecrit ton programme ici ;-)
2
En cours d'exécution: empty.py
>>>
>>>
>>> import pen from turtle
File "<stdin>", line 1
import pen from turtle
^
SyntaxError: invalid syntax
>>>
Python3 (Jupyter) REPL
In [1]: from friendly.mu import *
In [2]: import pen from turtle
Traceback (most recent call last):
File "In [2]", line 1
import pen from turtle
^
SyntaxError: invalid syntax
Did you mean from turtle import pen?
In [3]:
Python 3

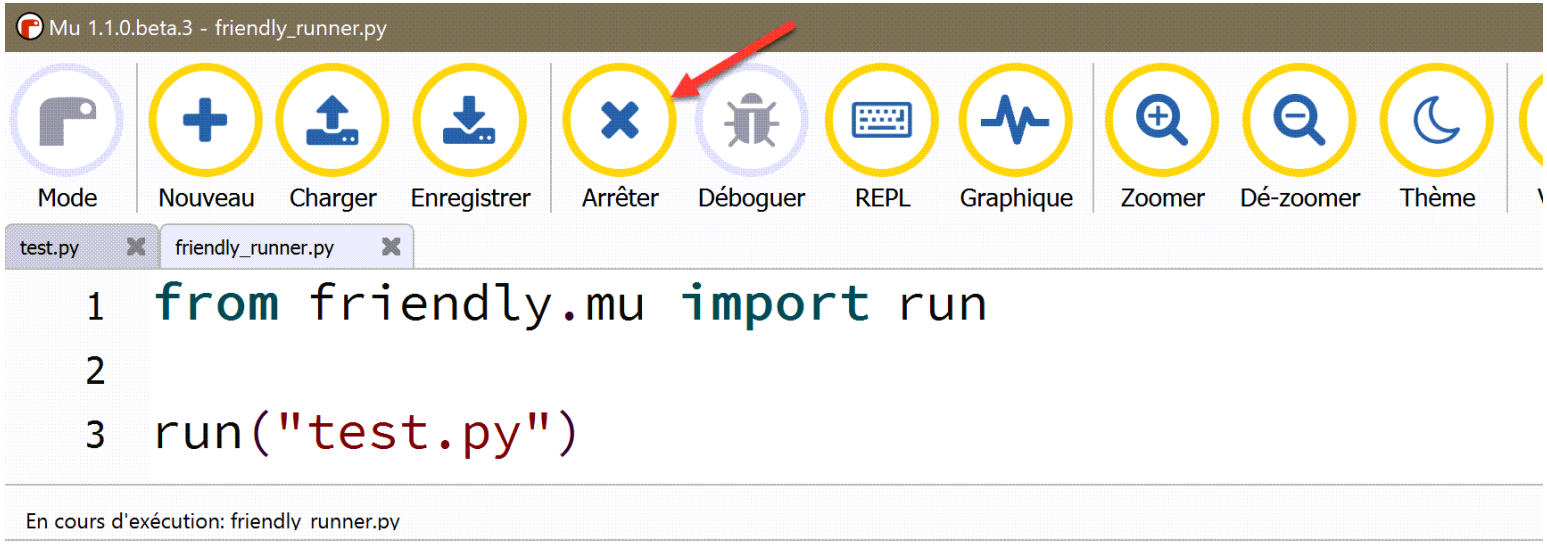
```
In [4]: why()

You have tried to get the item with index 3 of a, a list of length 3. The
largest valid index of a is 2.

In [5]: where()

Execution stopped on line 1 of file 'In [3]'.

    -->1: import test

Exception raised on line 2 of file 'CWD:\test.py'.

       1: a = [1, 2, 3]
    -->2: print(a[1], a[2], a[3])
                            ^^^^


    a:  [1, 2, 3]
```


Python3 (Jupyter) REPL
1: a = [1, 2, 3]
-->2: print(a[1], a[2], a[3])
^^^^
a: [1, 2, 3]
In [6]: night()
In [7]: where()
Execution stopped on line 1 of file 'In [3]'.
-->1: import test
Exception raised on line 2 of file 'CWD:\test.py'.
1: a = [1, 2, 3]
-->2: print(a[1], a[2], a[3])
^^^^
a: [1, 2, 3]


Mu 1.1.0.beta.3 - test.py
Mode
Nouveau
Charger
Enregistrer
Lancer
Déboguer
REPL
Graphique
Zoomer
Dé-zoomer
test.py
friendly_runner.py
1 a = [1, 2, 3]
2 print(a[1], a[2], a[3])
3


Mu 1.1.0.beta.3 - test.py
Mode
Nouveau
Charger
Enregistrer
Arrêter
Déboguer
REPL
Graphique
Zoomer
Dé-zoomer
Thème
test.py
friendly_runner.py
1 a = [1, 2, 3]
2 print(a[1], a[2], a[3])
En cours d'exécution: test.py
Traceback (most recent call last):
File "c:\users\andre\mu_code\test.py", line 2, in <module>
print(a[1], a[2], a[3])
IndexError: list index out of range
>>>

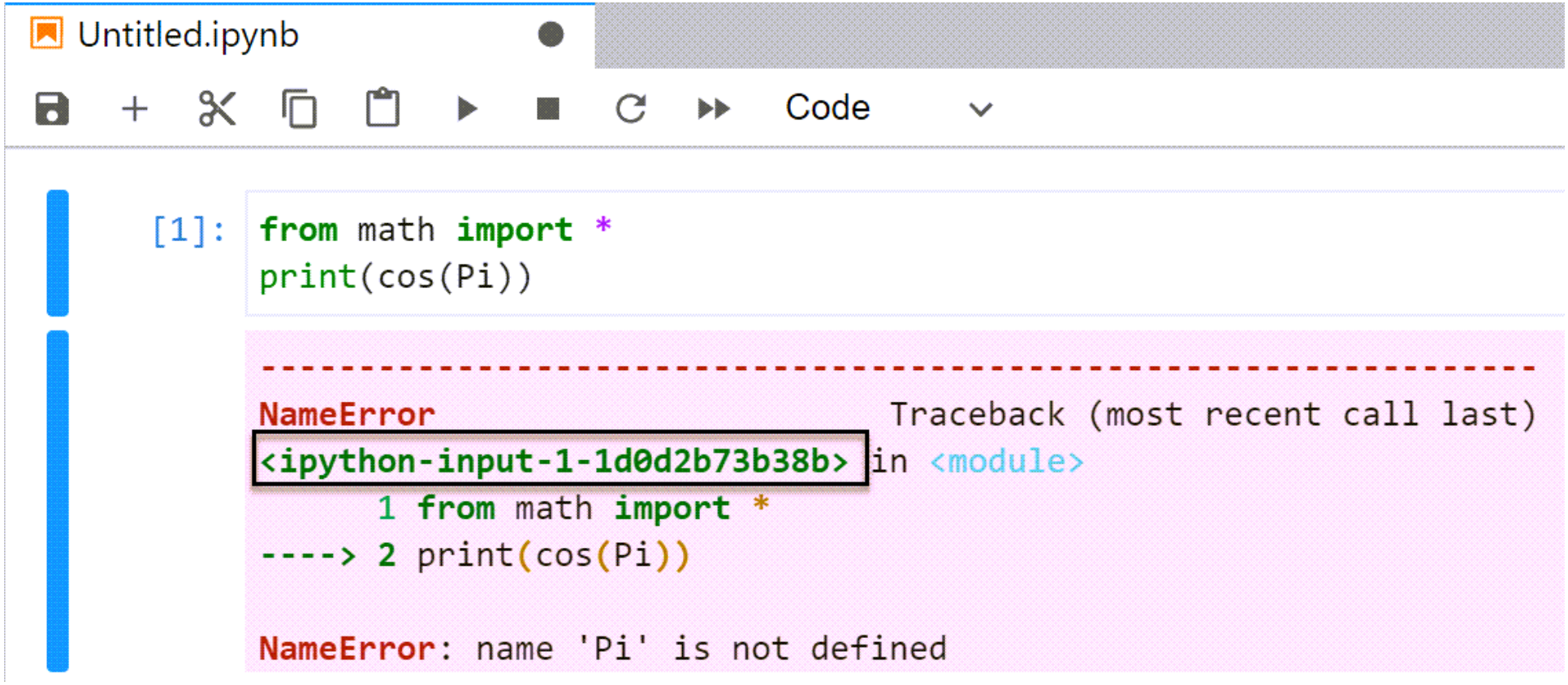

Mu 1.1.0.beta.3 - friendly_runner.py
Mode
Nouveau
Charger
Enregistrer
Lancer
Déboguer
REPL
Graphique
Zoomer
test.py
friendly_runner.py
1 from friendly.mu import run
2
3 run("test.py")
4



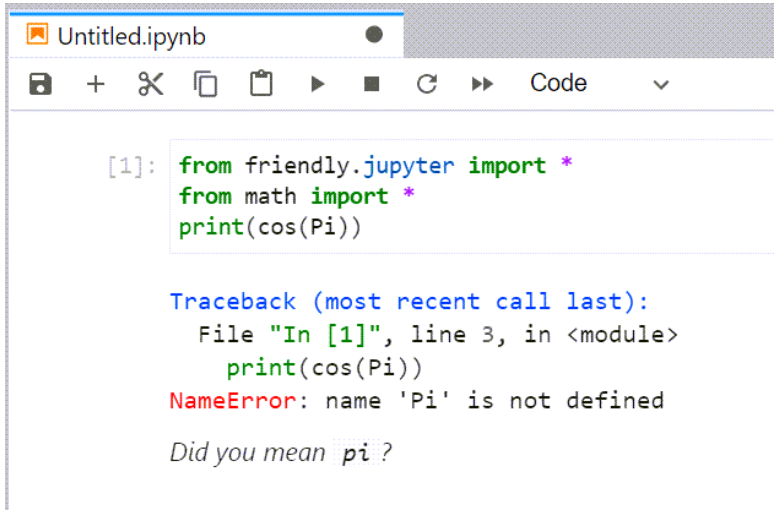

```
Traceback (most recent call last):
  File "CWD:\test.py", line 2, in <module>
    print(a[1], a[2], a[3])
IndexError: list index out of range

        Remember: the first item of a `list` is at index 0.

>>> why()

    You have tried to get the item with index `3` of `a`,
    a `list` of length `3`.
    The largest valid index of `a` is `2`.

>>>
```

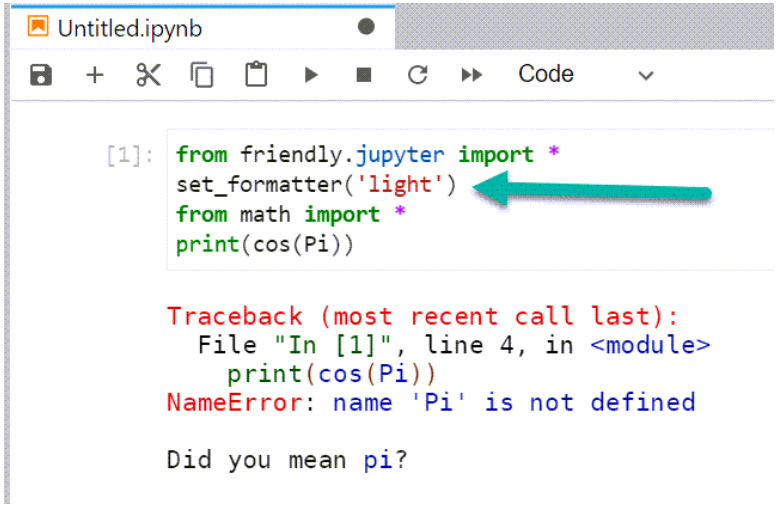
jupyter

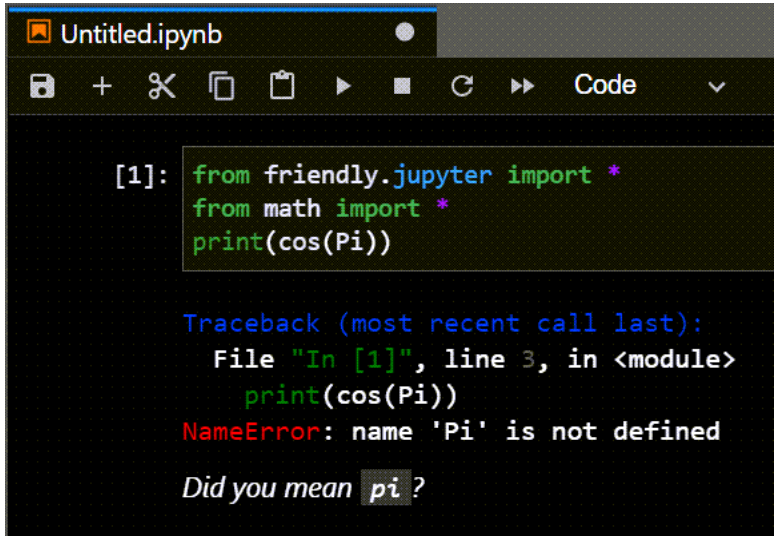
"DEMO"

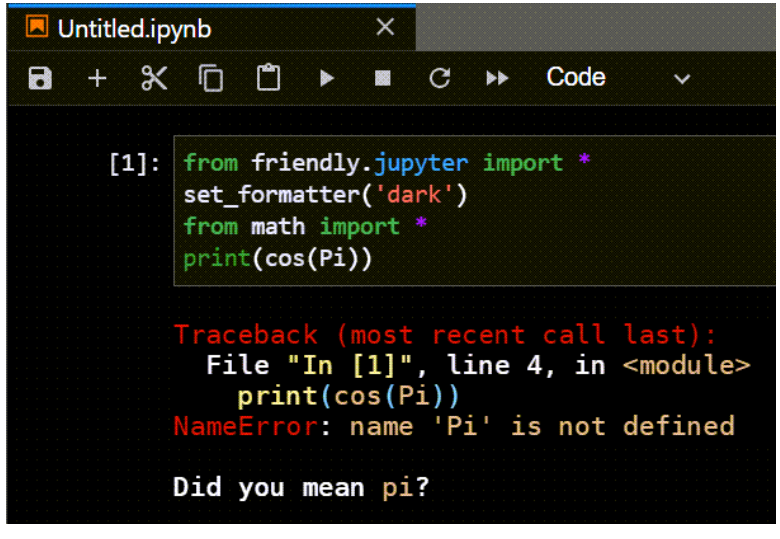

Untitled.ipynb
Code
[1]:
from math import *
print(cos(Pi))
NameError Traceback (most recent call last)
<ipython-input-1-1d0d2b73b38b> in <module>
1 from math import *
----> 2 print(cos(Pi))
NameError: name 'Pi' is not defined

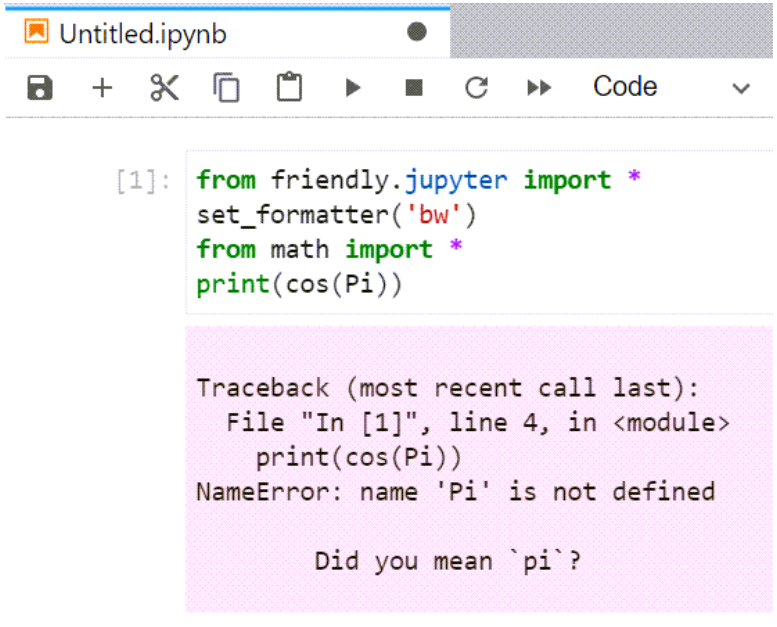
```
from friendly.jupyter import *
```

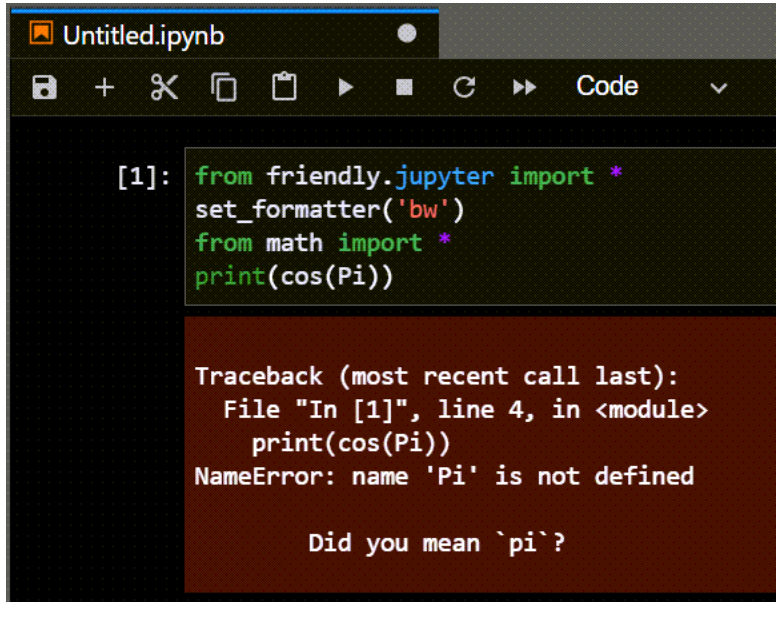





Untitled.ipynb
Code
[1]:
from friendly.jupyter import *
from math import *
print(cos(Pi))
Traceback (most recent call last):
File "In [1]", line 3, in <module>
print(cos(Pi))
NameError: name 'Pi' is not defined
Did you mean pi ?


Untitled.ipynb
Code
[1]:
from friendly.jupyter import *
set_formatter('dark')
from math import *
print(cos(Pi))
Traceback (most recent call last):
File "In [1]", line 4, in <module>
print(cos(Pi))
NameError: name 'Pi' is not defined
Did you mean pi?


Untitled.ipynb
Code
[1]:
from friendly.jupyter import *
set_formatter('bw')
from math import *
print(cos(Pi))
Traceback (most recent call last):
File "In [1]", line 4, in <module>
print(cos(Pi))
NameError: name 'Pi' is not defined
Did you mean `pi`?


Untitled.ipynb
Code
[1]:
from friendly.jupyter import *
set_formatter('bw')
from math import *
print(cos(Pi))
Traceback (most recent call last):
File "In [1]", line 4, in <module>
print(cos(Pi))
NameError: name 'Pi' is not defined
Did you mean `pi`?

PYTHON 3.10 - NameError

```
> python
Python 3.10.0b1 (tags/v3.10.0b1:ba42175, May  3 2021, 20:22:30) [
>>> from math import *
>>> a = cost(x)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'cost' is not defined. Did you mean: 'cos'?
>>>
>>> b = babs(3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'babs' is not defined. Did you mean: 'fabs'?
>>>
```

FRIENDLY - NameError

```
Friendly Console version 0.3.45. [Python version: 3.10.0b1]

>>> from math import *
>>> a = cost(x)

Traceback (most recent call last):
  File "<friendly-console:2>", line 1, in <module>
    a = cost(x)
NameError: name 'cost' is not defined

Did you mean cos?
>>> why()

In your program, cost is an unknown name. Instead of writing
cost, perhaps you meant one of the following:

 • Local scope: cos, cosh, acos
>>>
```

FRIENDLY - NameError

```
>>> b = babs(3)

Traceback (most recent call last):
  File "<friendly-console:4>", line 1, in <module>
    b = babs(3)
NameError: name 'babs' is not defined

Did you mean abs?
>>> why()

In your program, babs is an unknown name. Instead of writing
babs, perhaps you meant one of the following:

 • Local scope: fabs
 • Python builtins: abs
>>>
```

PYTHON 3.10 - NameError

```
> python -c "fabs"
Traceback (most recent call last):
  File "<string>", line 1, in <module>
NameError: name 'fabs' is not defined. Did you mean: 'abs'?
```

PEP 657
Include Fine Grained Error Locations in Tracebacks

May 8 2021

```
Traceback (most recent call last):
  File "test.py", line 2, in <module>
    x['a']['b']['c']['d'] = 1
    ^^^^^^^^^^^^^^^^^^^^^
TypeError: 'NoneType' object is not subscriptable
```

```
Traceback (most recent call last):
  File "pep657.py", line 10, in <module>
    print(x[42][1][2][3][4])
TypeError: 'NoneType' object is not subscriptable

>>> where()

Exception raised on line 10 of file 'pep657.py'.

        8: # The following will raise an exception
        9:
    -->10: print(x[42][1][2][3][4])
                  ^^^^^^^^^^^^^^

    x:  {0: {0: {0: 0}, 1: {1: 0}, 2: {2: 0}, 3: {3: 0}, 4: {4: 0}, ...}
        len(x): 50

    x[42]:  {0: {0: 0}, 1: {1: 42, 2: None}, 2: {2: 84}, 3: {3: 126}, ...}
            len(x[42]): 12

    x[42][1]:  {1: 42, 2: None}
    x[42][1][2]:  None
```

Thanks to Alex Hall's executing which uses ASTTokens

```
>>> why()

Subscriptable objects are typically containers from which you can
retrieve item using the notation [...]. Using this notation, you
attempted to retrieve an item from x, an object of type NoneType.
This is not allowed.

Note: NoneType means that the object has a value of None.
>>>
```

why(did the name change?)

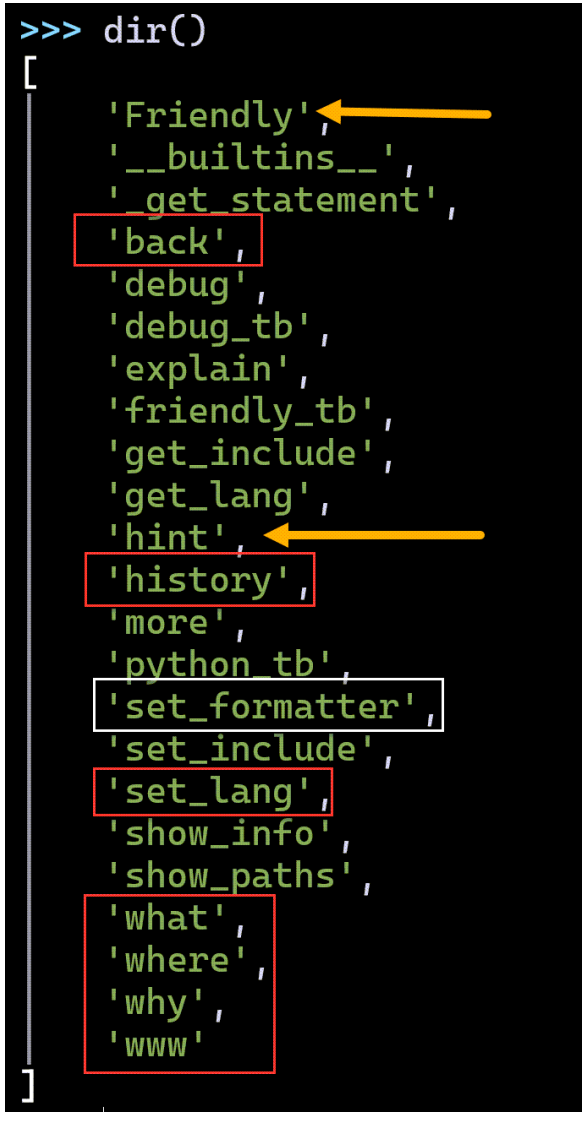
```
from friendly          import ...
from friendly.idle     import ...
from friendly.mu       import ...
from friendly.ipython  import ...
from friendly.jupyter  import ...

# are better than

from friendly_traceback.idle import ...
```

Besides, *traceback* is not exactly a beginner-friendly term and, in the future, **friendly** might do more than simply helping with tracebacks ...

what(are warnings?)

```
Friendly Console version 0.2.38. [Python version: 3.8.4]

>>> a : int
Warning: you used a type hint for a variable without assigning it a value.
   Do you find these warnings useful?
   Comment at https://github.com/aroberge/friendly-traceback/issues/112
>>> list = [1, 2, 3]
Warning: you have redefined the python builtin list.
   Do you find these warnings useful?
   Comment at https://github.com/aroberge/friendly-traceback/issues/112
>>>
```

why(import * ?)



Friendly.why() == why(), etc

`who( influenced the design of friendly? )`

- Nicholas Tollervey (Mu)
- Aivar Annamaa (Thonny)
- Julien Palard (HackInScience)
- Alex Hall (futurecoder + many packages)
- Sylvain Desodt (DidYouMean-Python inspired by Raymond Hettinger)
- Michael Kennedy, Brian Okken, and Hannah Stepanek in PythonBytes podcast #220
- *plus many others who filed issues, etc.*

friendly is currently at version 0.3.47
Many features not originally thought of when I started this project have been incorporated in its design.

Can **YOU** think of possible improvements to include in the road to version 1.0?

`where( is the code? )`

https://github.com/aroberge/friendly

You will find a link to the documentation.

But perhaps there is a more pythonic way ...

where(is the documentation?)😉

```
> python -m pip install friendly
> python -m friendly

Friendly Console version ...

>>> www()
```

Feel free to contact me with your suggestions as to how to improve **friendly**.