ABOUT friendly

PyCon Lightning Talk - May 14, 2021
ANDRÉ ROBERGE

See PyCon's Education Summit

<ASIDE>

What's new in Python 3.10?

IMPROVED TRACEBACKS!

Thanks to many people, but mostly Pablo Galindo Salgado.

May 8: PEP 657

Include Fine Grained Error Locations in Tracebacks

```
Traceback ...
    print(x[42][1][2][3][4])
          ^^^^^^^^^^^^^^

TypeError: 'NoneType' object is not subscriptable
```

</ASIDE>

WHAT IS friendly?

Library providing help with tracebacks.

```
> python -im friendly pep657.py

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

Friendly Console version 0.3.47. [Python version: 3.8.4]


>>> |
```

why()

```
>>> 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.
>>>
```

where()

```
>>> where()

Exception raised on line 10 of file 'pep657.py'.          PEP 657

      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
>>>
```

Python 3.6+

7

what()

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

TypeError: 'NoneType' object is not subscriptable

Une exception TypeError est généralement causée une
tentative de combiner deux types d'objets incompatibles,
en invoquant une fonction avec le mauvais type d'objet, ou
en tentant d'effectuer une opération non permise sur un
type d'objet donné.
```

SyntaxError: invalid syntax

```
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.
```

www()

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

AttributeError: partially initialized module turtle has no attribute forward

Tout   Images   Vidéos   Actualités   Carte   Préférences ▾

◯ Canada (fr) ▾   Filtre parental : modérée ▾   À tout moment ▾

AttributeError: partially initialized module 'turtle' has ...

https://stackoverflow.com/questions/60480328/attributeerror-partially-initialized-mod...

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

attributeerror: partially initialized module 'turtle' has ...

https://stackoverflow.com/questions/65962607/attributeerror-partially-initialized-mod...

AttributeError: partially initialized module 'turtle' has no attribute 'Turtle' (most likely due

Python 3.10: NameError, AttributeError


```
Python 3.10.0b1
>>> from math import *
>>> a = tabs(3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'tabs' is not defined. Did you mean: 'fabs'?
>>>
```

Python 3.10: Jupyter

```
In [1]:  ▶| import sys
            print(sys.version)
            from math import *
```

```
3.10.0b1
```

```
In [2]:  ▶| a = tabs(3)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-89980eda1018> in <module>
----> 1 a = tabs(3)

NameError: name 'tabs' is not defined
```

```
In [3]:  ▶| from friendly.jupyter import *

         set_formatter('light')  # Nicer display

In [4]:  ▶| a = tabs(3)


         Traceback (most recent call last):
           File "In [4]", line 1, in <module>
             a = tabs(3)
         NameError: name 'tabs' is not defined

         Did you mean abs?
```

```
▶| why()
```

In your program, `tabs` is an unknown name. Instead of wr
the following:

- Local scope: `fabs`
- Python builtins: `abs`

```python
# terminal, VS Code, etc.
from friendly            import ...

# Special cases
from friendly.idle    import ...
from friendly.mu      import ...
from friendly.ipython import ...
from friendly.jupyter import ...
```

```
>>> dir()
[
    'Friendly',
    '__builtins__',
    '_get_statement',
    'back',
    'debug',
    'debug_tb',
    'explain',
    'friendly_tb',
    'get_include',
    'get_lang',
    'hint',
    'history',
    'more',
    'python_tb',
    'set_formatter',
    'set_include',
    'set_lang',
    'show_info',
    'show_paths',
    'what',
    'where',
    'why',
    'www'
]
```

If you need help with tracebacks

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

```
> python -m friendly

Friendly Console version ...

>>>
```

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

Code on Github ⧉

Bienvenue 환영합니다 Bienvenido - ようこそ Welcome 歡迎光臨

friendly helps users understand what caused a given exception in their Python program, and how to fix it.

There exists many Python projects whose **primary** goal is to supplement the information given by Python traceback to make them more useful for **advanced** programmers. To my knowledge, of all those projects, friendly is the only one designed with beginners in mind.

To be more specific, while friendly can be useful for advanced programmers, it strives to present the information in a way that is easily understood by beginners and/or by users who would like to get information about traceback in their own language. friendly can give more detailed information as to **where** an exception occurred, **what** a given exception means and **why** it might have occurred (sometimes adding suggestions as to how to fix it.)

Even though friendly can be used on its own with a specially designed console, a better option might be to use it together with GUI-based editors/IDE including JupyterLab and Jupyter notebooks. I explain how to do so later. For now, I'll just show one quick example.

> **Using screenshots**
>
> In this documentation, I most often use some screenshots, as I believe that they are more easily understandable by most users. Occasionally, I write some examples as text so that they can provide the required information to people relying on screen readers. If you are such a user and need help because the use of screenshots prevents you from understanding the explanations, please do not hesitate to reach out to me:
>
> andre.roberge@gmail.com ⧉

A quick look

Consider the following program:

```python
# example.py
def get_last(seq):
    last_index = len(seq)
    return seq[last_index]


print(get_last([1, 2, 3]))
```

First, here is what happens when I use Python to run this program:

```
> python example.py
```

back (most recent call last):

Try friendly!