Week 4: debugging | testing

NRSC 7657 Workshop in Advanced Programming for Neuroscientists

course business

- Projects: have you started? A couple of pushes to repositories - if you have started, push. If you haven’t…

Debugging


FAST DELIVERY
PREMIUM
BEST PRICE 100% GUARANTEED
DISCOUNT 15%
6 STAGES OF DEBUGGING
1 THAT CAN'T HAPPEN
2 THAT DOESN'T HAPPEN ON MY MACHINE
3 THAT SHOULDN'T HAPPEN
4 WHY DOES THAT HAPPEN?
5 OH I SEE
6 HOW DID THAT EVER WORK?
OUR GLOBAL BRAND SINCE 2010
Baxboxshirt.com
shirtnation.net
leesilk.com
utrustshop.com
moteefe.com
teechip.com
Kybershop.com
Buy our products with 100% guaranteed, high quality items, fast shipping via tracking code in your email.
THANK YOU VERY MUCH !

Debugging

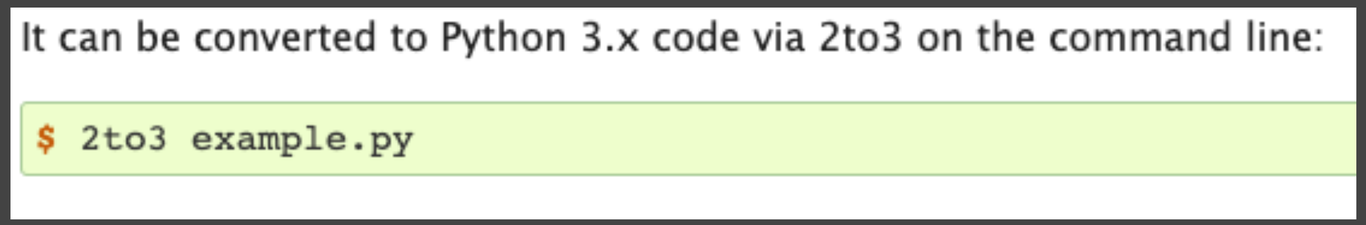
- The first step in debugging: `print("something")`

Debugging

- The first step in debugging: `print("something")`

  - Aside: you may see `print "something"` sometimes in older code, this is a relic of python 2. Change it to `print("something")`

    - https://docs.python.org/3/library/2to3.html


It can be converted to Python 3.x code via 2to3 on the command line:

```
$ 2to3 example.py
```

Debugging

Types of bugs: syntax errors

- Tracebacks

```
a = 2
print(2 + a)
b = a + c
```

```
4
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-0b5043da75c2> in <module>
      1 a = 2
      2 print(2 + a)
----> 3 b = a + c

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

Debugging

Types of bugs: syntax errors

- Tracebacks

```
[116]: rez = loadmat('rez.mat')
```

```
---------------------------------------------------------------------------
NotImplementedError                       Traceback (most recent call last)
<ipython-input-116-852d7d6a9435> in <module>
----> 1 rez = loadmat('rez.mat')

~/opt/anaconda3/envs/NRSC7657/lib/python3.8/site-packages/scipy/io/matlab/mio.py in loadmat(file_name, mdict,
appendmat, **kwargs)
    223     variable_names = kwargs.pop('variable_names', None)
    224     with _open_file_context(file_name, appendmat) as f:
--> 225         MR, _ = mat_reader_factory(f, **kwargs)
    226         matfile_dict = MR.get_variables(variable_names)
    227

~/opt/anaconda3/envs/NRSC7657/lib/python3.8/site-packages/scipy/io/matlab/mio.py in mat_reader_factory(file_n
ame, appendmat, **kwargs)
     78         return MatFile5Reader(byte_stream, **kwargs), file_opened
     79     elif mjv == 2:
---> 80         raise NotImplementedError('Please use HDF reader for matlab v7.3 files')
     81     else:
     82         raise TypeError('Did not recognize version %s' % mjv)

NotImplementedError: Please use HDF reader for matlab v7.3 files
```

Debugging

Types of bugs: semantic errors

- No `Traceback`, but…something didn’t work like you thought it would.

Debugging

Types of bugs: syntax and semantic errors

- The first step in debugging: `print(“something”)`

Debugging

Types of bugs: syntax and semantic errors

- The first step in debugging: `print("something")`
- The first step in debugging: `print("something")`
- The first step in debugging: `print("something")`
- The first step in debugging: `print("something")`

Debugging

Types of bugs: you can fix both with print statements

- The first step in debugging: `print("something")`
  - Aside: you may see `print "something"` sometimes in older code, this is a relic of python 2. Change it to `print("something")`
    - https://docs.python.org/3/library/2to3.html

Debugging
Python standard debugger

import pdb

```
import pdb

x = 3
y = 4
pdb.set_trace()

total = x + y
pdb.set_trace()
```

We have inserted a few breakpoints in this program. The program will pause at each breakpoint (**pdb.set_trace()**). To view a variables contents simply type the variable name:

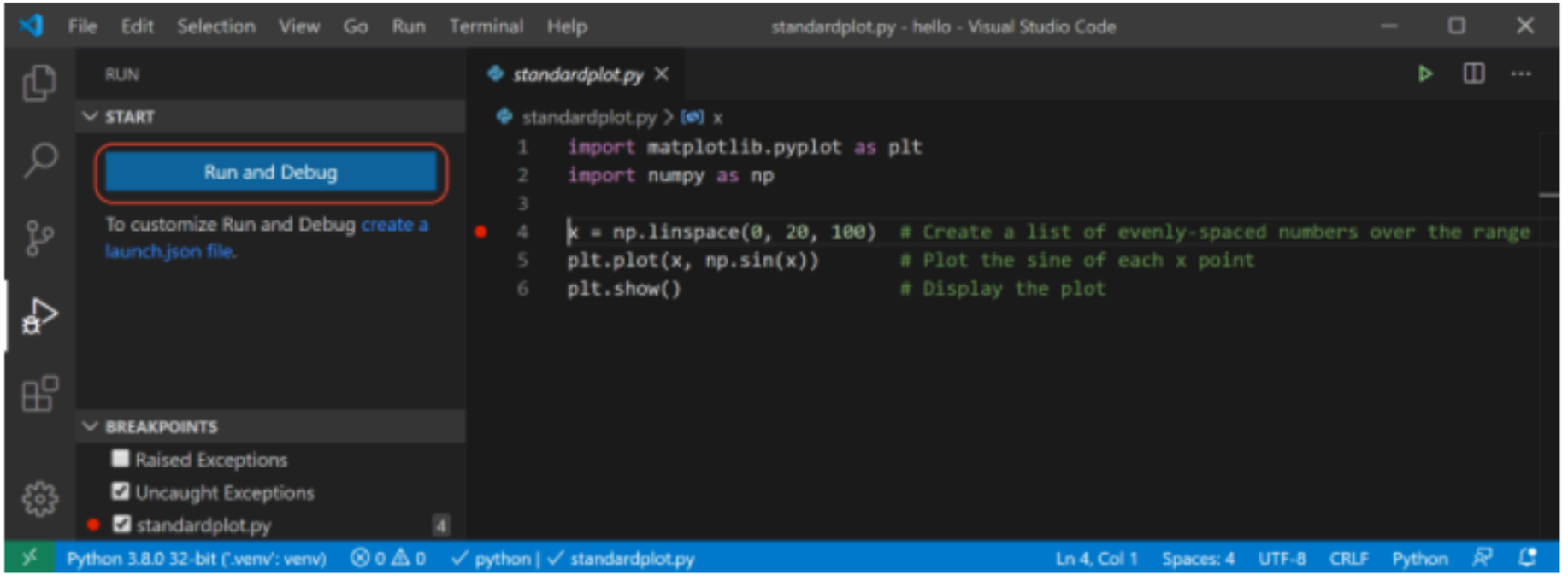
```
$ python3 program.py
(Pdb) x
3
(Pdb) y
4
(Pdb) total
*** NameError: name 'total' is not defined
(Pdb)
```

Debugging

VSCode Run and Debug mode, for scripts

Basic debugging

The simplest way to begin debugging a Python file is to use the **Run** view and click the **Run and Debug** button. When no configuration has been previously set, you will be presented with a list of debugging options. Select the appropriate option to quickly begin debugging your code.



Debugging

VSCode Run and Debug mode, for Jupyter notebooks

- The first step in debugging: `print("something")`
  - If you are looking at your code block, and thinking `print("something")` is going to be too complicated…maybe your code block is too long and you should be breaking it up or moving part to a function.

Testing

- You have written useful code.
- You (or someone else) wants to write some more code.
  - How do you make sure you don’t break it?

Unit testing

- General idea:
  - a series of tests
  - fixed inputs and known outputs
  - Some that test the end-to-end function of the code
  - Some that isolate small parts, in case the end to end is broken so you can figure out where the break is

Unit testing

- Python built in: `unittest`
- A third-party package that has more features, and is simpler: `pytest`

1. Arrange, or set up, the conditions for the test
2. Act by calling some function or method
3. Assert that some end condition is true