Next Steps in Python: Troubleshooting Python Code

with Colby Witherup Wood

```
This workshop will begin
at 12:02 pm Central
```

Question:
When it comes to coding in Python, what are you most insecure about?

If you're thinking "Everything", try to be more specific. Is there something you feel like you don't understand well enough? Something you always mess up or forget?

Feel free to share your answers in the chat!

This workshop is brought to you by

NUIT Research Computing Services

Have a programming or data question about your research?

We're here to help.

Request a free consultation at bit.ly/rcsconsult

The workshop is being recorded.

Please ask any questions in the Zoom chat.

If my internet goes out during the workshop, everyone gets a 10-minute break. We will meet back at the same Zoom link.

Materials are at www.github.com/aGitHasNoName/trouble. Click the green Code button and choose Download ZIP.

This workshop is designed for beginning and intermediate Python coders. We will not be covering advanced packages for debugging.

Errors are normal. Debugging is normal.

I just wanted to get that out of the way.

The takeaway of this workshop:

Search for the problem,
not the solution.

The takeaway of this workshop:

Search for the problem, not the solution.

- It will save you time *over the course of a project*
- Introducing solutions can introduce new problems
- You will actually learn

Use logic to identify the problem.

Follow your instincts, but don't trust them.

Use logic to identify the problem.

Follow your instincts, but don't trust them.

- Don't start trying solutions to what you think is the problem without confirming that it is the problem
- Start narrowing down *where* the problem is

What is instinct?

What is instinct?

"an innate, typically fixed pattern of behavior in animals in response to certain stimuli."

What is instinct?

Our "instinct" can be led by our insecurities. I'm bad at X, so X must be causing the problem.

What is instinct?

The more Python code I see, the more types of data, the more types of errors, the better my "instinct" becomes.

First division in troubleshooting

It used to work,
but now it doesn’t.

I just wrote it
and it doesn’t work.

First division in troubleshooting

It used to work,
but now it doesn’t.

I just wrote it
and it doesn’t work.

These are two very different things.

First division in troubleshooting

It used to work,
but now it doesn’t.

I just wrote it
and it doesn’t work.

I just made changes,
and now it doesn’t work.

I just wrote it and it doesn’t work.

I’m getting an
error message.

syntactic error

It’s not doing what
I want it to.

semantic error

Error messages

Python 10 now has better error messages! 3.10 is not the default Python in Anaconda yet, but you can create a new conda environment

1. Look up the error message online. Does anything seem right?
2. Start debugging (we'll talk about what this means).

It’s not doing what I want it to.

It's a code problem.

It’s a data problem.

It’s not doing what I want it to.

It's a code problem.

It’s a data problem.

Both will probably require debugging, but you might do it a bit differently if you have data.

Data

1. Think about it. Could something about your data be causing the problem? Could there be differences in your data? (some rows aren’t formatted the same way as others, you have missing data in some places that you didn’t notice, distributions vary, etc.)
2. If you've identified a possible way that your data could be involved:
    a. Look at the raw data (briefly), or
    b. Write Python code to search through the data for the anticipated problem.
3. Try creating a shorter version of your dataset with only a few clean rows – if that works the way it should, but the full dataset fails, the problem lies with your data.

Debugging – find *where* the problem is

1. Work in a copy of your script or notebook
2. Start at the top of the script or notebook (or code cell, if you're working in a Jupyter notebook and you know where the trouble begins)
3. Run the code in chunks (you can comment out code below inside a function) until you get the error, OR
4. Write print statements to check that each piece of the code is doing what you expect, from top to bottom, removing each print statement once that chunk of code is ruled out.

It used to work, but now it doesn't.

Did you change anything in the code recently?

Are you using a different data set?

Are you running it on a different system?

Has it been a while since the last time you ran it?

It used to work, but now it doesn't.

Did you change anything in the code recently?

Start debugging just above the changed code.

It used to work, but now it doesn't.

Are you using a different data set?

How is the new dataset different from the old?

It used to work, but now it doesn't.

Can you test it again on the old system to confirm?

Rewrite the code OR Create a conda environment to replicate the old system

Are you running it on a different system?

Local vs. server, script vs. Jupyter notebook, local Jupyter vs. colab, new laptop, etc.

It used to work, but now it doesn't.

Python updated? Package updated? (might be a dependency)

Rewrite the code OR Create a conda environment to replicate the old versions

Has it been a while since the last time you ran it?

Once you've identified where the problem is

1. Look it up online – are there solutions?
2. Spend time thinking about how to fix it
3. Work in a copy of your script or notebook
4. Try one solution at a time and revert to the original version before trying each new solution
5. Ask for help if you can't figure it out within a reasonable time

If you have a data problem:

Should you clean the data OR change the code to accommodate the messiness?

- Do you plan on running the same code on a different dataset in the future, or is this a one-time thing?
- Which will take less time?

Questions?