Welcome.

Everyone:

- Pull the updates from the course GitHub repo:
  - `cd <46120-PiWE repo>`
  - `git pull upstream main` ← you might have “upstream2” instead

Physical students:

- Sit WHEREVER you want. 😎
- Turn off laptop volume (mute). ←IMPORTANT!
- Log into the Zoom meeting.
  - Microphone muted. Camera off.

46120: Scientific Programming for Wind Energy

Function handles

Jenni Rinker

Agenda for today.

- Pull new course material ✔

- Round robin.
- Function handles.

- Your homework for next week.
  - And preview of what you'll hand in for codecamp.

Round robin

Share solutions with your peers and give feedback.

Time to review and collaborate.

- 1 round of 30 minutes.
- 5 minutes: chaos.
- 25 minutes: present/discuss homework. Today's feedback focus:
  1. How "clean" do you feel the team's code is? How easy to understand?
  2. How is collaborating with git going? Any changes since Week 1?

- Afterwards: plenum discussion.
  - Be ready with questions!

```
WEEK 4
========================================
  BOR 0: WindyWizards, Los Programadores, Team Team
  BOR 1: Git Happens, CodeTeam, Mouxtarides tou Mahalla
  BOR 2: WindCoders, CryptoMania, CodeGust
  BOR 3: Push & Pray, Stop Fucking Spiders, brunchy
  BOR 4: BreezeTech, FatalError, BugBusters
  BOR 5: BugHunters, CodeFusion, BladePYrunners
  BOR 6: Power-Fire, ¿Qué? ¿Como Qué?, A4 Highway, La Bombas
  BOR 7: Lightning McTeam, NetZero, PowerPuff Girls, SIF
```

Notes in plenum.

- Add here.

5-minute survey, all together.

Function handles

Ya gotta grab things.

Function handles.

- We can “handle” functions by calling the name of the function without parentheses.
  - A function with parentheses tells Python to execute (a.k.a., “invoke”) the function.

```
>>> def mysum(x):
>>>     return sum(x)

>>> mysum
<function mysum at 0x000001B1030CF0D0>

>>> type(mysum)
<class 'function'>

>>> mysum([4, 2])
6
```

Here we define a symbolic variable `mysum`, which is a function, and associate some code with that variable. Note that we do not execute the code when we define the function!

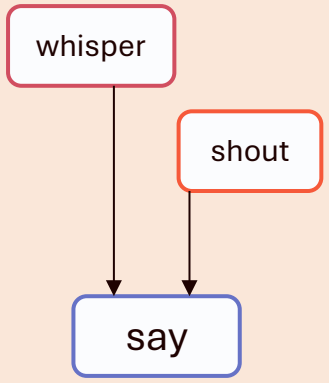
This returns some metadata information of the variable `mysum`, indicating that it is a function stored at a certain place in memory. Because we did not include parentheses after `mysum`, the function is NOT invoked.

It’s a function (shock).

Now I have used parentheses, so the function is invoked.

Passing functions into functions.

- In some cases, it is useful to pass a function into a function and invoke it there.




```
def whisper(s):  # print a string in lowercase
      print(s.lower())

def shout(s):  # print a string in uppercase
      print(s.upper())

def say(fun, s):  # say something
      fun(s)
```

No parentheses here. We are passing in the function handle, we don't want to invoke it here.

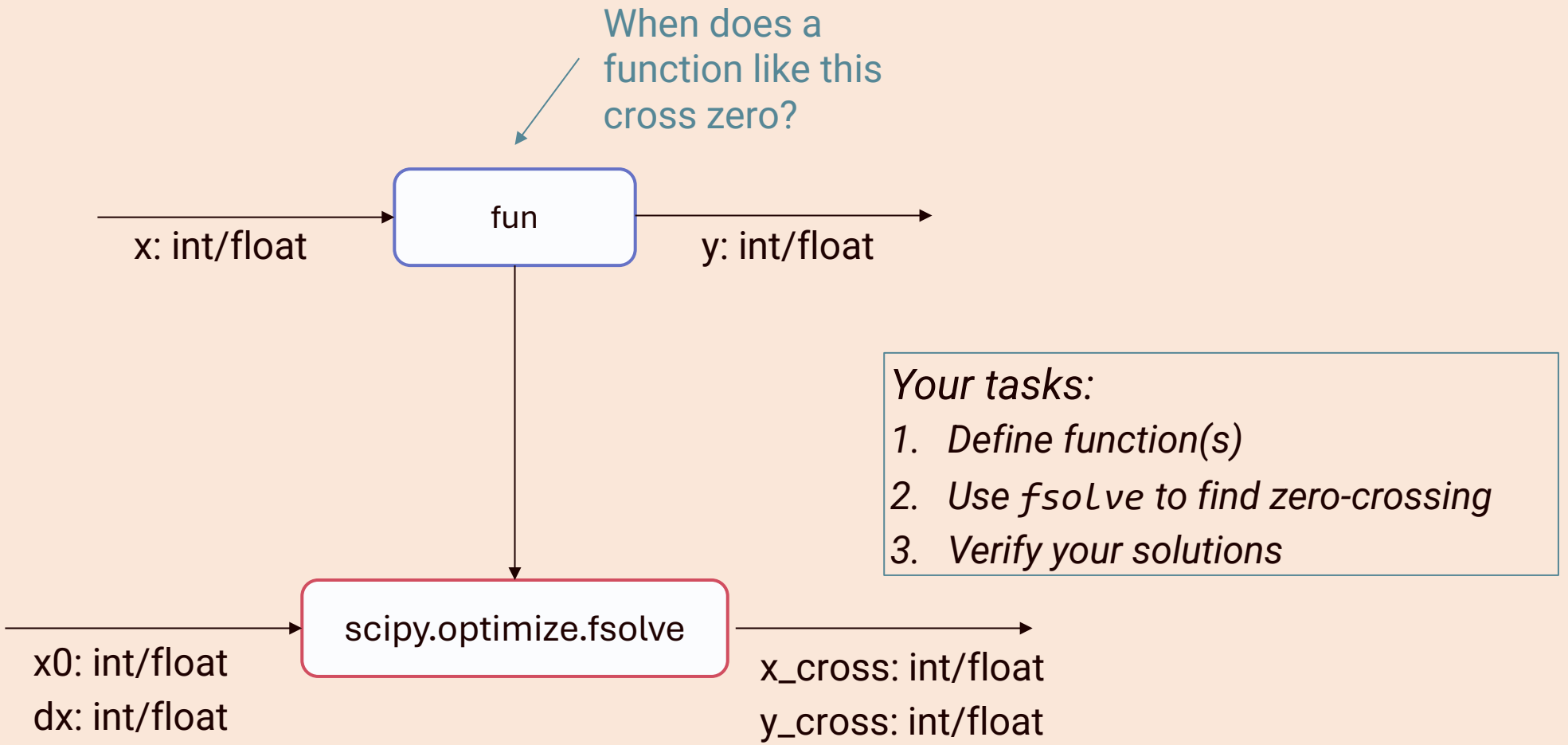
Here is where we want to call the function, so we add parentheses.

```
>>> say(whisper, 'HEY YOU')  # whisper something
'hey you'

>>> say(shout, 'you are awesome')  # shout something
'YOU ARE AWESOME'
```

No parentheses – passing in a handle that the `say` function will invoke.

Exercise. Pairs preferred, alone ok.



Your tasks:

1. *Define function(s)*
2. *Use `fsolve` to find zero-crossing*
3. *Verify your solutions*

Exercise. Pairs preferred, alone ok.

1. Open `demo_fsolve.py` in VS Code.
2. Given the default values of the parabola coefficients, what are the expected roots? *(Hint: Remember how to factor quadratic functions…)*
3. Define initial guess `X0` and add it to plot as black “x”.
4. Look up the `fsolve` documentation from `scipy`.
   - What required does it take? What are the inputs/outputs of the passed-in function?
   - What can you do if your function has more inputs than `fsolve` expects? *(Hint: check out `args` keyword argument.)*
   - What does `fsolve` return?
   - Look at the example at the bottom of the docs. Does it make sense?
5. Add code that calls fsolve, then adds the solution to plot as a red ring.
6. (Extra credit) Think of a combination of a, b, c with no root and try your code. What happens?

Exercise. Pairs preferred, alone ok.

- Live-code the solution together.

Questions?

Homework for this week

What better way to get better at something than to practice?

First: some information.

“Final” codecamp project, due before Week 6.

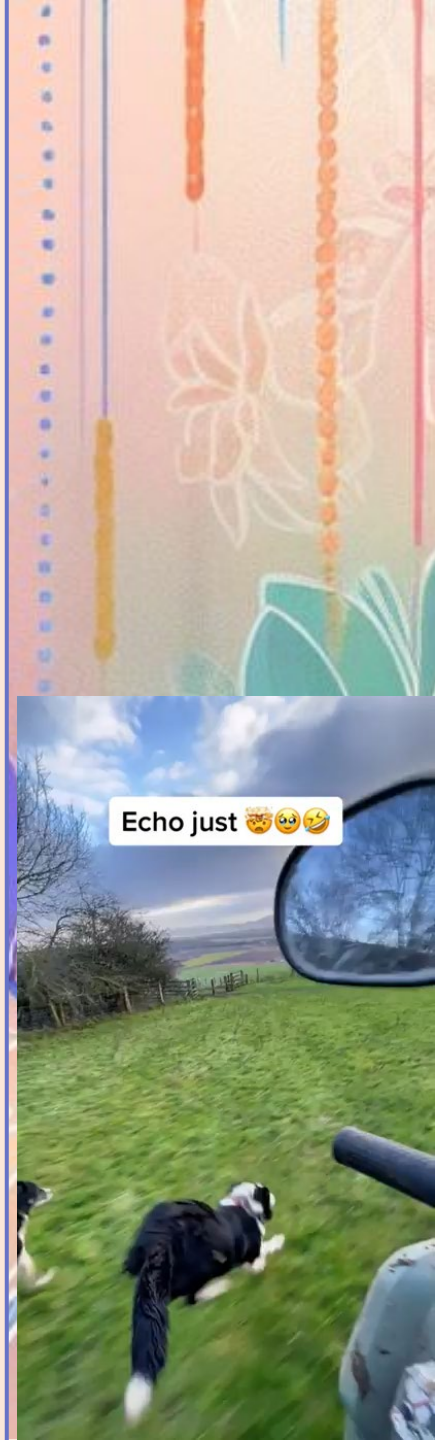
- Draft of details/peer-feedback rubric in week06 subfolder.
- Let’s go through it together.
- If you finish this week’s homework quickly and want to move onto final project:
  - DO NOT WRITE ANY CODE.
  - But, can make a clear outline of what code you want to write. Bullet list of steps, description of new functions (black-box diagrams!), etc.
  - Ideal: meet with no laptops, just a whiteboard.

*Remember, you're expected to work **about 6 hours outside of class**. Schedule accordingly.*

Homework.

- Detailed on the course GitHub repo.
  - **Short summary**: make functions to simulate time-marching response to turbulence. If you want, start designing code for final project.
- We'll open BORs in a minute. Enter room corresponding to your Team ID (on team excel sheet).
- Complete **Part 0** of the weekly assignment in class, then move on as agreed with your team.
- **To get help during class**: Post in Slack / #debugging if you want a TA to enter your BOR or come find your group.

Any questions?



Tutorials.

1. Functions and passing functions [1.11. Defining Functions of your Own — Hands-on Python Tutorial for Python 3 (luc.edu)]