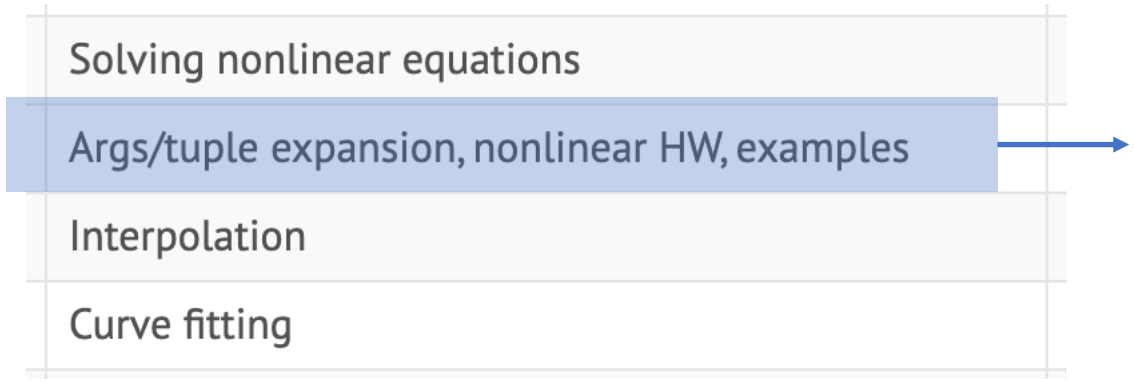
# Exercise

- Make a list of all python functions/variables/code we have seen.
- Discuss the items in the list:
  - What is the basic idea?
  - What further details?
  - What are some “gotchas” or caveats?

# Excercise

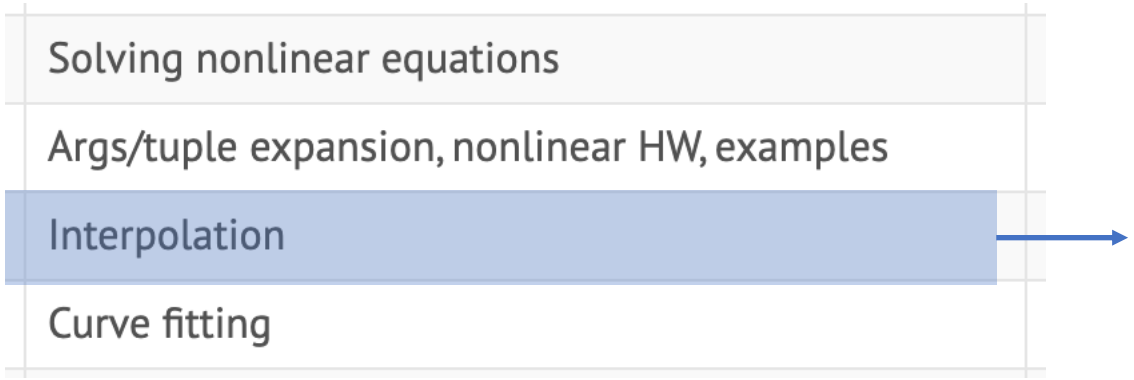
- Write down the main topic/title of the last five classes on Python
- For each class, what were the main ideas
  - Outline each class.
  - Put in details
  - What examples were done?
- Do this alone, then with your neighbor.
- Create your own review notes (like these slides).
  - First “recall” then look up to fill in.

# Class 20: Examples, nonlinear solvers




- Discussed Newton Fractal
- Discussed turbulent problem: coupling nonlinear solve
- Discussed homework
  - Adiabatic flame temperature solution
- Approaches for solving systems of equations
  - Use intermediate variables
    - Re = rDv/m normally doesn't need to be a separate equation/variable
  - Often, the solution in Python can be written as it appears in text
  - Carefully consider how many equations/variables you really need.
    - Trivial equations can be used directly: like $x_1 + x_2 = 1 \rightarrow x_2 = (1-x_1)$
    - In such cases, final values can be recovered after solution.
  - Functions can be defined inside functions if convenient
  - Can mix solution approaches: quad, interp1d, etc.

# Class 21: Interpolation




- Given a list of xg, yg points.
- Find values of y at intermediate x
- Linear interpolation exercise
  - Pass in xg, yg, xw → find yw
  - Find location (two bounding points).
  - Form and evaluate equation for line between two points.
- Python interpolator

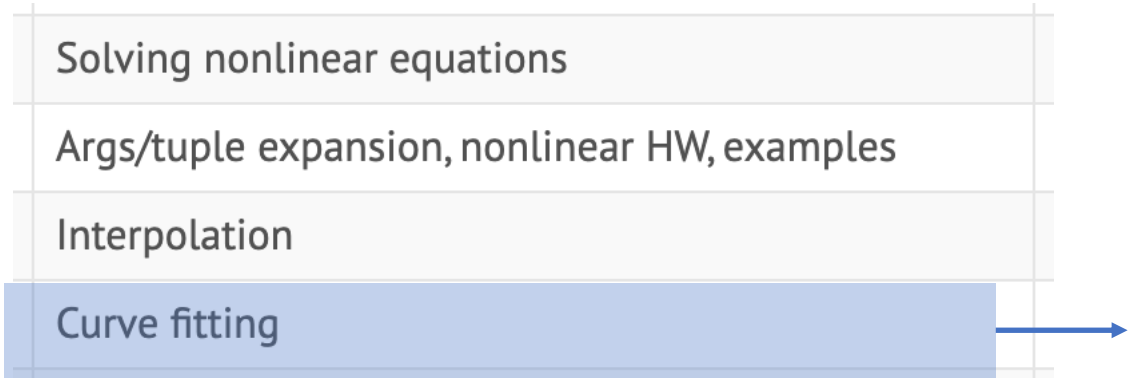
```
from scipy.interpolate import interp1d

xg = np.array([0,1,2,3,4,5,6,7,8,9,10])
yg = np.cos(xg**2.0/8.0)+1

f_interp = interp1d(xg, yg)

xw = 2.5
yw = f_interp(xw)
```

# Class 22: Curve Fitting




- Given a list of xg, yg points.
- Given a model function with unknown parameters
- Find the parameters
- `polyfit, polyval` functions

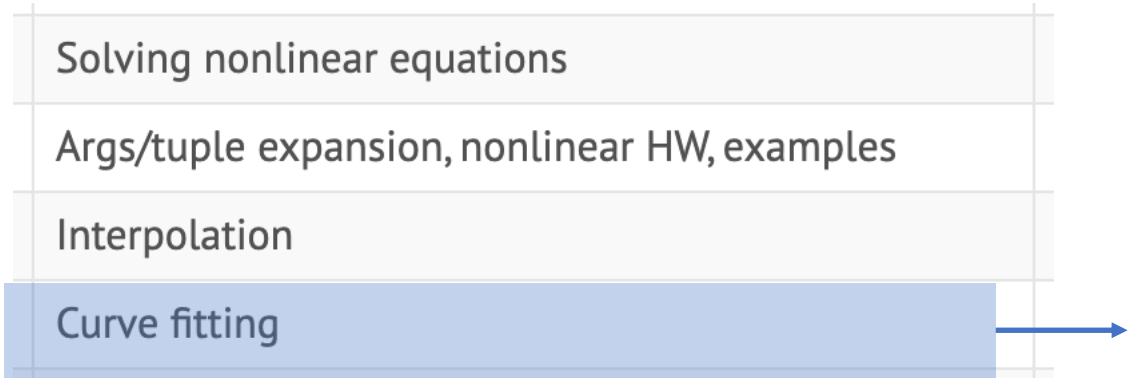
```
import numpy as np

xg = np.array([0., 1., 2., 3., 4., 5.])
yg = np.array([0, 0.8, 0.9, 0.1, -0.8, -1.0])

p3 = np.polyfit(xg, yg, 3)

xw = np.linspace(0,5,1000)
yw = np.polyval(p3, xw)
```

# Class 22: Curve Fitting




- Given a list of xg, yg points.
- Given a model function with unknown parameters
- Find the parameters
- `curve_fit` function

```
import numpy as np
from scipy.optimize import curve_fit

#-------- given data

xg = np.linspace(0,4,50)
yg = 2.5*np.exp(-1.3*xg)+0.5 + 0.2*np.random.normal(size=len(xg))

#-------- Define the function with parameters: x comes first

def f(x, a, b, c) :
    return a*np.exp(-b*x) + c

#-------- Do the curve fit

abc, extras = curve_fit(f, xg, yg)

a = abc[0]
b = abc[1]
c = abc[2]

xx = xg
yy = f(xx,a,b,c)
```

# Class 23: Rate Equations

- Rate equations
- Symbolic math
- Widgets
- Python-Excel interface

- from scipy.integrate import odeint

$$\frac{dy}{dt} = f(y,t) \qquad y(0) = y_0$$

- f(y,t) is the “right hand side function” or the rate function.
- f(y,t) depends on y and t in general, but the actual expression often doesn’t include t.
- For multiple equations y is a vector of “variables”, f is a vector of functions
- Solve for y(t). Solution will be an array of t and an array of y

$$\frac{dy}{dt} = -2y + 3$$

$$y(0) = 1$$

```
def f(y,t):
    return -2*y + 3

y0 = 1
t  = np.linspace(0,5,100)
y  = odeint(f, y0, t)
```

$$\frac{dv}{dt} = g$$

$$\frac{dx}{dt} = v$$

$$y(0) = x(0) = 0$$

```
def f(vx,t):
    v = vx[0]
    x = vx[1]
    dvdt = 9.81
    dxdt = v
    return np.array([dvdt, dxdt])

xy0 = np.array([0, 0])
t  = np.linspace(0,5,100)
y  = odeint(f, xy0, t)
```