Using Python in an Introductory ODE Course

Patrick Davis

MAA MathFest 2016

6 August 2016

CENTRAL MICHIGAN UNIVERSITY

General Disclaimer: What follows is ancedotal evidence from a particular teaching experience. It is not meant to be taken as mathematics education research, and the course was not run as a part of any formal study.

- **Bulletin Description:**
  Definition and solution of first, second, and higher order differential equations.
- **Prerequisites:**
  Calculus II, Linear Algebra
- **Course Rationale:**
  This is traditionally a sophomore-junior level course that prepares students for the mathematical sciences in such areas as applied mathematics, engineering, physics and chemistry.

- **Student Learning Objectives:**
  1. Solve first order differential equations using appropriate techniques.
  2. Demonstrate familiarity with the Existence and Uniqueness Theorem.
  3. Apply appropriate techniques to solve homogeneous and non-homogeneous differential equations.
  4. Solve differential equations with solutions represented by power series.
  5. Find the Laplace transform of a given function and use the Laplace transform to solve initial value problems.
  6. Solve systems of differential equations.
  7. Use numerical methods to solve differential equations.

- **Student Learning Objectives:**
  1. **Solve** first order differential equations using appropriate techniques.
  2. Demonstrate familiarity with the Existence and Uniqueness Theorem.
  3. Apply appropriate techniques to **solve** homogeneous and non-homogeneous differential equations.
  4. **Solve** differential equations with solutions represented by power series.
  5. Find the Laplace transform of a given function and use the Laplace transform to **solve** initial value problems.
  6. **Solve** systems of differential equations.
  7. Use numerical methods to **solve** differential equations.

- **Student Learning Objectives:**
  1. **Solve** first order differential equations using appropriate techniques.
  2. Demonstrate familiarity with the Existence and Uniqueness Theorem.
  3. Apply appropriate techniques to **solve** homogeneous and non-homogeneous differential equations.
  4. **Solve** differential equations with solutions represented by power series.
  5. Find the Laplace transform of a given function and use the Laplace transform to **solve** initial value problems.
  6. **Solve** systems of differential equations.
  7. Use numerical methods to **solve** differential equations.

MY GOAL: And have students understand that solution in a broader context!

Pedagogical Method

Tactic #1: “Conversational lecture”:

- Lecture and guided discovery hybrid.
- Driven highly by instructor and student questions.
- Holistic and motivating examples.
- Emphasis on creating a narrative of the mathematics and expressing formal concepts in natural language.

Pedagogical Method

Tactic #1: “Conversational lecture”:

- Lecture and guided discovery hybrid.
- Driven highly by instructor and student questions.
- Holistic and motivating examples.
- Emphasis on creating a narrative of the mathematics and expressing formal concepts in natural language.

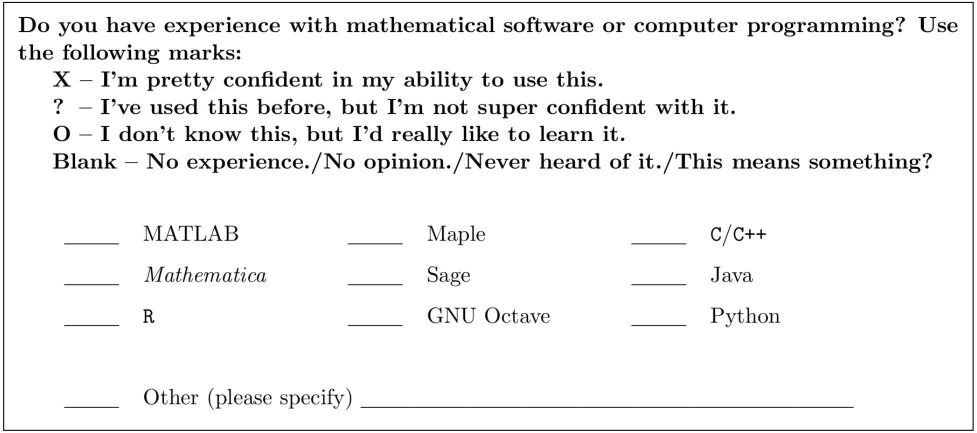
Tactic #2: Project-Based Learning

Why Use Mathematical Software?

- Allows for quick computations, thereby enabling a greater focus on system analysis
- Facilitates in-class instruction
- Gives a tool for students to explore material on their own
- Builds students' resumes

Why Not Use Mathematical Software?

- Takes time away from...
  - students (less time to learn traditional techniques and mathematical theory)
  - instructors (must create supporting material and projects)
- Many students (and instructors!) are intimidated and/or quickly frustrated by it
- Can change the focus of the course
- Difficult to include in syllabus (what has to go?)

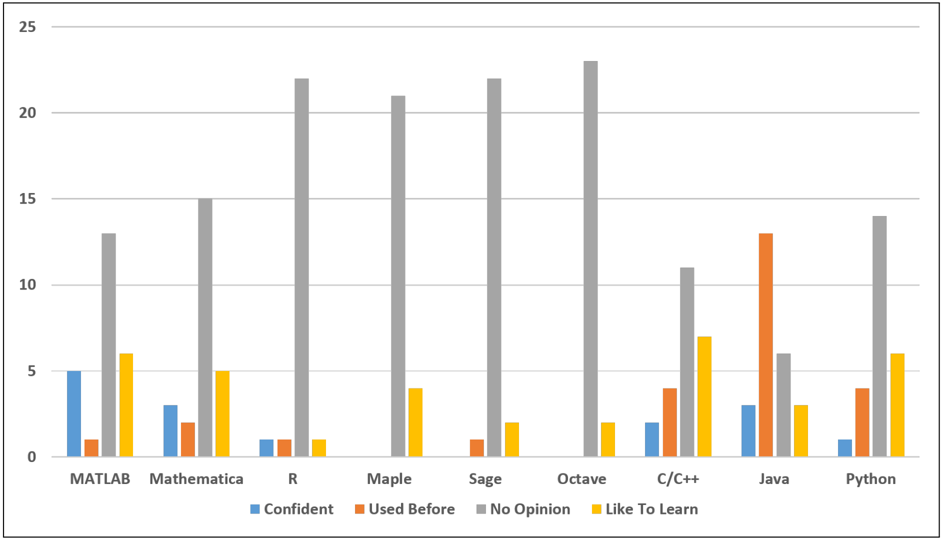


Do you have experience with mathematical software or computer programming? Use the following marks:

X – I'm pretty confident in my ability to use this.
? – I've used this before, but I'm not super confident with it.
O – I don't know this, but I'd really like to learn it.
Blank – No experience./No opinion./Never heard of it./This means something?

_____	MATLAB	_____	Maple	_____	`C/C++`
_____	*Mathematica*	_____	Sage	_____	Java
_____	`R`	_____	GNU Octave	_____	Python

_____ Other (please specify) ______________________________________


25
20
15
10
5
0
MATLAB
Mathematica
R
Maple
Sage
Octave
C/C++
Java
Python
Confident
Used Before
No Opinion
Like To Learn

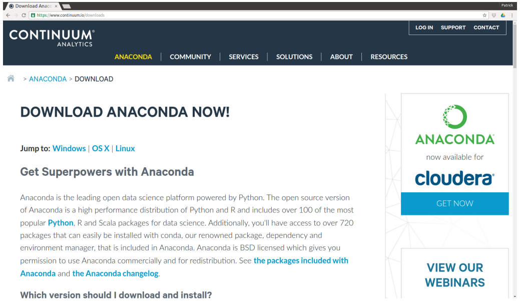
Why Use Python?

- Versatility (used for both computer programming and scientific computing)
- Trendy (looks *particularly* good on students' resumes)
- Similarity to computer programming and mathematical software
- Cross-platform (Windows, MacOS, and Linux)
- It's FREE!!

Why Not Use Python?

- Rugged (not as "put together" as proriety mathematical software)
- Less user support
- Not already available to students on university maintained computers

The Anaconda Distribution



DOWNLOAD ANACONDA NOW!

Jump to: Windows | OS X | Linux

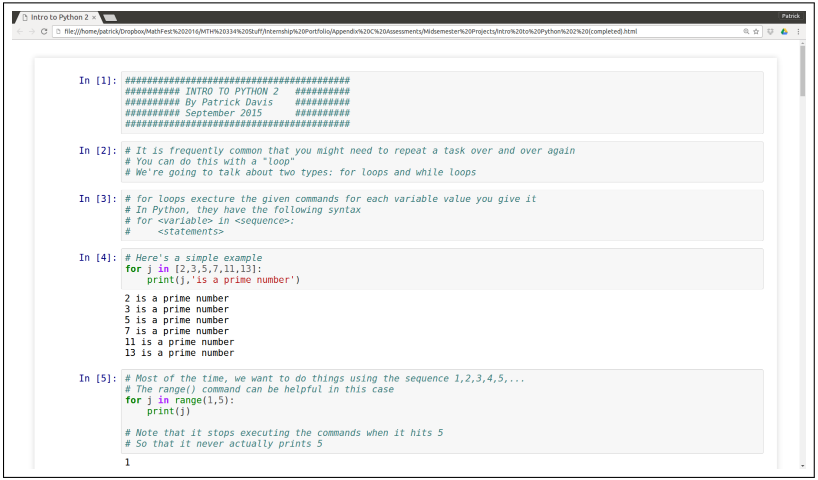
Get Superpowers with Anaconda

Anaconda is the leading open data science platform powered by Python. The open source version of Anaconda is a high performance distribution of Python and R and includes over 100 of the most popular Python, R and Scala packages for data science. Additionally, you'll have access to over 720 packages that can easily be installed with conda, our renowned package, dependency and environment manager, that is included in Anaconda. Anaconda is BSD licensed which gives you permission to use Anaconda commercially and for redistribution. See the packages included with Anaconda and the Anaconda changelog.

Which version should I download and install?

- Bundled Python packages (over 100 packages, including NumPy and matplotlib)
- Comes with its own package manager called conda.
- Includes Jupyter/IPython (an in-browser notebook style front end to write and execute Python code)
- Cross-platform (Windows, MacOS, and Linux)
- Doesn't require admin privledges to install

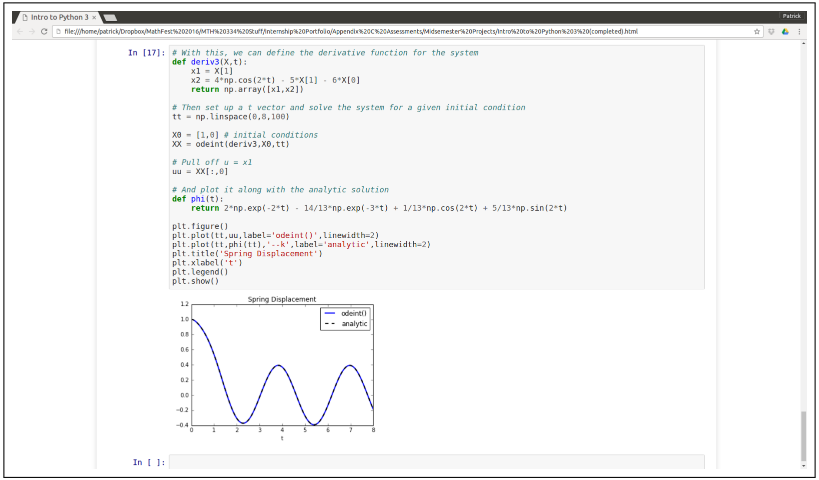
The Anaconda Distribution



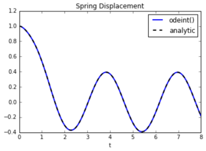
Intro to Python 2
Patrick
file:///home/patrick/Dropbox/MathFest%202016/MTH%20334%20Stuff/Internship%20Portfolio/Appendix%20C%20Assessments/Midsemester%20Projects/Intro%20to%20Python%202%20(completed).html

```
In [1]: ##########################################
        ########## INTRO TO PYTHON 2   ##########
        ########## By Patrick Davis    ##########
        ########## September 2015      ##########
        ##########################################

In [2]: # It is frequently common that you might need to repeat a task over and over again
        # You can do this with a "loop"
        # We're going to talk about two types: for loops and while loops

In [3]: # for loops execture the given commands for each variable value you give it
        # In Python, they have the following syntax
        # for <variable> in <sequence>:
        #     <statements>

In [4]: # Here's a simple example
        for j in [2,3,5,7,11,13]:
            print(j,'is a prime number')

        2 is a prime number
        3 is a prime number
        5 is a prime number
        7 is a prime number
        11 is a prime number
        13 is a prime number

In [5]: # Most of the time, we want to do things using the sequence 1,2,3,4,5,...
        # The range() command can be helpful in this case
        for j in range(1,5):
            print(j)

        # Note that it stops executing the commands when it hits 5
        # So that it never actually prints 5

        1
```

The Anaconda Distribution


Intro to Python 3
Patrick
file:///home/patrick/Dropbox/MathFest%202016/MTH%20334%20Stuff/Internship%20Portfolio/Appendix%20C%20Assessments/Midsemester%20Projects/Intro%20to%20Python%203%20(completed).html

```
In [17]: # With this, we can define the derivative function for the system
         def deriv3(X,t):
             x1 = X[1]
             x2 = 4*np.cos(2*t) - 5*X[1] - 6*X[0]
             return np.array([x1,x2])

         # Then set up a t vector and solve the system for a given initial condition
         tt = np.linspace(0,8,100)

         X0 = [1,0] # initial conditions
         XX = odeint(deriv3,X0,tt)

         # Pull off u = x1
         uu = XX[:,0]

         # And plot it along with the analytic solution
         def phi(t):
             return 2*np.exp(-2*t) - 14/13*np.exp(-3*t) + 1/13*np.cos(2*t) + 5/13*np.sin(2*t)

         plt.figure()
         plt.plot(tt,uu,label='odeint()',linewidth=2)
         plt.plot(tt,phi(tt),'--k',label='analytic',linewidth=2)
         plt.title('Spring Displacement')
         plt.xlabel('t')
         plt.legend()
         plt.show()
```



```
In [ ]:
```

Course Materials:

- Online homework system (WebAssign)
- Textbook (*Elementary Differential Equations* by Boyce & DiPrima)
- Mathematical Software (Python)

Assessment:

- Homework 15%
- Mid-Semester Projects 15%
- Quizzes 15%
- Final Project 15%
- Midterm 20%
- Final Exam (cumulative) 20%

Mid-Semester Projects

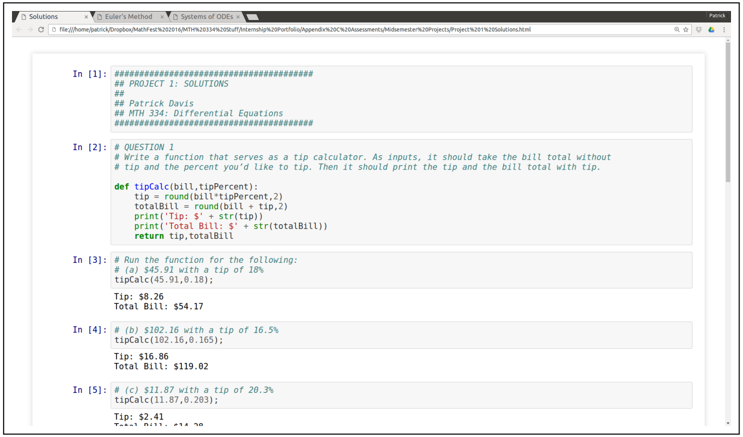
Instructions:

- Goals ("By the end of this project, you should...")
- Background information (linked to in-class material)
- Exercises
  - To become familiar with code implementation
  - To make connections with deeper questions

Python Tutorial:

- Guided-discovery style how-to on things students needed
- Lots of explanation

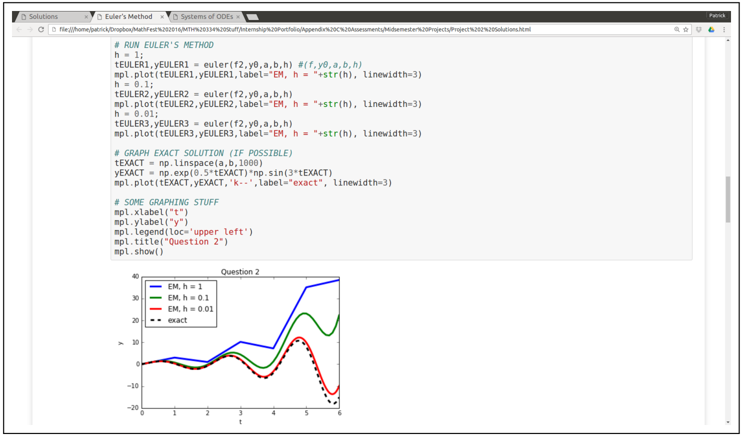
Project #1: Introduction to Python


Solutions
Euler's Method
Systems of ODEs
Patrick
file:///home/patrick/Dropbox/MathFest%202016/MTH%20334%20Stuff/Internship%20Portfolio/Appendix%20C%20Assessments/Midsemester%20Projects/Project%201%20Solutions.html

```
In [1]: #########################################
        ## PROJECT 1: SOLUTIONS
        ##
        ## Patrick Davis
        ## MTH 334: Differential Equations
        #########################################

In [2]: # QUESTION 1
        # Write a function that serves as a tip calculator. As inputs, it should take the bill total without
        # tip and the percent you'd like to tip. Then it should print the tip and the bill total with tip.

        def tipCalc(bill,tipPercent):
            tip = round(bill*tipPercent,2)
            totalBill = round(bill + tip,2)
            print('Tip: $' + str(tip))
            print('Total Bill: $' + str(totalBill))
            return tip,totalBill

In [3]: # Run the function for the following:
        # (a) $45.91 with a tip of 18%
        tipCalc(45.91,0.18);

        Tip: $8.26
        Total Bill: $54.17

In [4]: # (b) $102.16 with a tip of 16.5%
        tipCalc(102.16,0.165);

        Tip: $16.86
        Total Bill: $119.02

In [5]: # (c) $11.87 with a tip of 20.3%
        tipCalc(11.87,0.203);

        Tip: $2.41
```

- Get comfortable with Jupyter/IPython
- Learn the Python basics (operations, variables, data structures, functions)
- Graph mathematical functions

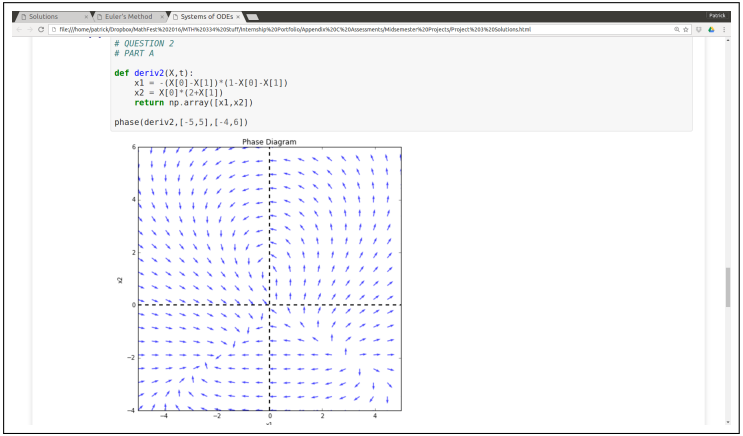
Project #2: Numerical Methods


Solutions
Euler's Method
Systems of ODEs
Patrick
file:///home/patrick/Dropbox/MathFest%202016/MTH%20334%20Stuff/Internship%20Portfolio/Appendix%20C%20Assessments/Midsemester%20Projects/Project%202%20Solutions.html
Question 2
EM, h = 1
EM, h = 0.1
EM, h = 0.01
exact
y
t
40
30
20
10
0
−10
−20
0
1
2
3
4
5
6

```
# RUN EULER'S METHOD
h = 1;
tEULER1,yEULER1 = euler(f2,y0,a,b,h) #(f,y0,a,b,h)
mpl.plot(tEULER1,yEULER1,label="EM, h = "+str(h), linewidth=3)
h = 0.1;
tEULER2,yEULER2 = euler(f2,y0,a,b,h)
mpl.plot(tEULER2,yEULER2,label="EM, h = "+str(h), linewidth=3)
h = 0.01;
tEULER3,yEULER3 = euler(f2,y0,a,b,h)
mpl.plot(tEULER3,yEULER3,label="EM, h = "+str(h), linewidth=3)

# GRAPH EXACT SOLUTION (IF POSSIBLE)
tEXACT = np.linspace(a,b,1000)
yEXACT = np.exp(0.5*tEXACT)*np.sin(3*tEXACT)
mpl.plot(tEXACT,yEXACT,'k--',label="exact", linewidth=3)

# SOME GRAPHING STUFF
mpl.xlabel("t")
mpl.ylabel("y")
mpl.legend(loc='upper left')
mpl.title("Question 2")
mpl.show()
```

- Write code to implement Euler's Method and learn to use Python's `odeint()` to solve a single ODE
- Solve and analyze linear and nonlinear ODEs

Project #3: Systems of ODEs


Solutions
Euler's Method
Systems of ODEs
Patrick
file:///home/patrick/Dropbox/MathFest%202016/MTH%20334%20Stuff/Internship%20Portfolio/Appendix%20C%20Assessments/Midsemester%20Projects/Project%203%20Solutions.html
Phase Diagram
x2
x1

```
# QUESTION 2
# PART A

def deriv2(X,t):
    x1 = -(X[0]-X[1])*(1-X[0]-X[1])
    x2 = X[0]*(2+X[1])
    return np.array([x1,x2])

phase(deriv2,[-5,5],[-4,6])
```

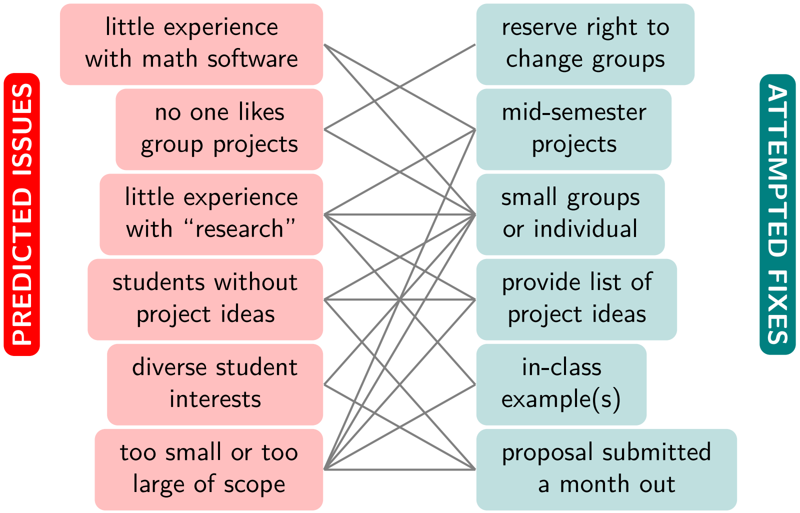
- Learn to use Python's `odeint()` to solve systems of ODEs
- Draw phase planes and use them to analyze linear and nonlinear systems of ODEs

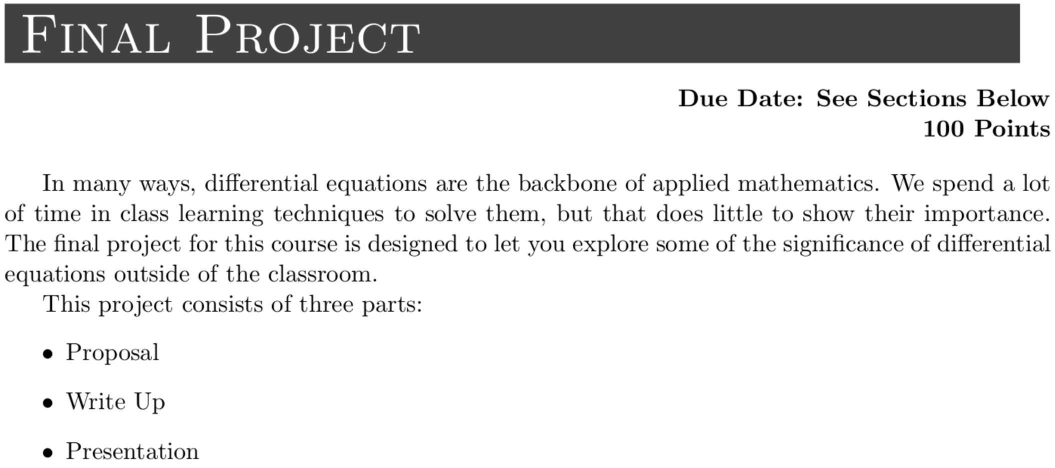
Final Project

Goals

By the end of this project, you should...

- appreciate differential equations in a larger context.
- be capable of thinking about a topic outside of the classroom setting.
- have demonstrated your ability to communicate ideas through good academic writing and public speaking.
- be glad its over.
- have had a lot of fun.


PREDICTED ISSUES
little experience with math software
no one likes group projects
little experience with “research”
students without project ideas
diverse student interests
too small or too large of scope
reserve right to change groups
mid-semester projects
small groups or individual
provide list of project ideas
in-class example(s)
proposal submitted a month out
ATTEMPTED FIXES



Final Project

Due Date: See Sections Below
100 Points

In many ways, differential equations are the backbone of applied mathematics. We spend a lot of time in class learning techniques to solve them, but that does little to show their importance. The final project for this course is designed to let you explore some of the significance of differential equations outside of the classroom.

This project consists of three parts:

- Proposal
- Write Up
- Presentation

Choosing a Topic

Finding a suitable topic is perhaps the most difficult part of this project. You may choose any topic that has to deal with ordinary differential equations, and I have provided some ideas on Blackboard in the Final Project folder. When you're looking for a project idea, try to draw from your own interests; it will be much easier to motivate yourself if you have some enthusiasm for the project.

Final Project Ideas

Below is a list of project ideas. Some of them are of an appropriate scope for a whole project; some are just a starting point. They're roughly sorted into categories, but there is quite a bit of overlap for some fields (especially physics and engineering). So if you don't see something you like under your "preferred" section, then look under the other ones as well.

As long as your project has to do with differential equations, there are no restrictions on what you may do. If you don't see something you like here, feel free come up with your own idea, draw ideas from other courses, or search the Internet for something you are interested in.

If you need some help picking a project or finding a place to start with an idea you've chosen, I'm always available to chat before/after class, during office hours, or via email.

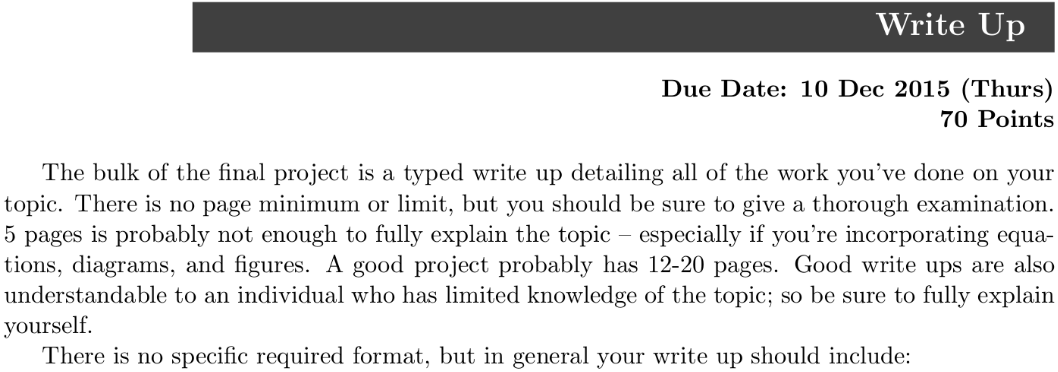
Proposal

Due Date: 29 Oct 2015 (Thurs)
15 Points

In short, your proposal should summarize the game plan for this project. It should be 1-2 pages typed and include a brief overview of your chosen topic. If you have a differential equation (or a system of differential equations) already formulated, then you should include this as well.

Your proposal should indicate specific questions you plan to answer with your project. These are not set in stone. So feel free to change them, but the proposal should point you in a direction. This is especially important if you're working in a group because it should get all of the group members on the same page.

At the end, please list some references (2-3+). Be aware that I will look up your references so that I can understand your perspective on this project; so make sure they're worthwhile. (I should note: in my opinion, Wikipedia is a worthwhile initial source. They do a decent job of summarizing topics, but it should not be your only reference.)

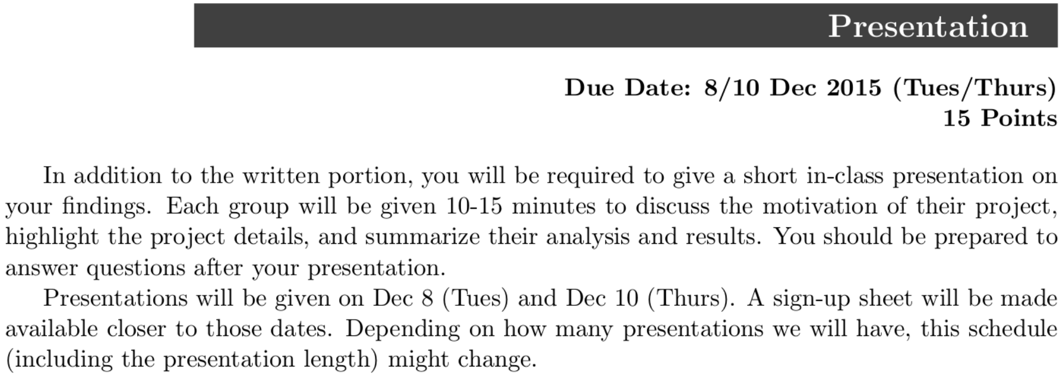


Write Up

Due Date: 10 Dec 2015 (Thurs)
70 Points

The bulk of the final project is a typed write up detailing all of the work you've done on your topic. There is no page minimum or limit, but you should be sure to give a thorough examination. 5 pages is probably not enough to fully explain the topic – especially if you're incorporating equations, diagrams, and figures. A good project probably has 12-20 pages. Good write ups are also understandable to an individual who has limited knowledge of the topic; so be sure to fully explain yourself.

There is no specific required format, but in general your write up should include:



Presentation

Due Date: 8/10 Dec 2015 (Tues/Thurs)
15 Points

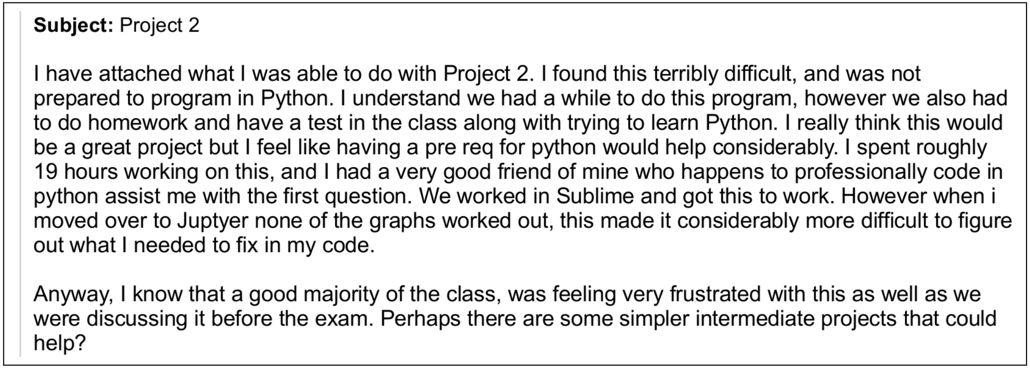
In addition to the written portion, you will be required to give a short in-class presentation on your findings. Each group will be given 10-15 minutes to discuss the motivation of their project, highlight the project details, and summarize their analysis and results. You should be prepared to answer questions after your presentation.

Presentations will be given on Dec 8 (Tues) and Dec 10 (Thurs). A sign-up sheet will be made available closer to those dates. Depending on how many presentations we will have, this schedule (including the presentation length) might change.

Examples of Student Final Projects:

- The Two Body Problem
- Space Colonization
- Zebra Mussel Populations
- The Lorenz System
- Improved Eulers Methods and Adams Method
- Cooling Fin
- Drugs in the Bloodstream

Student Email



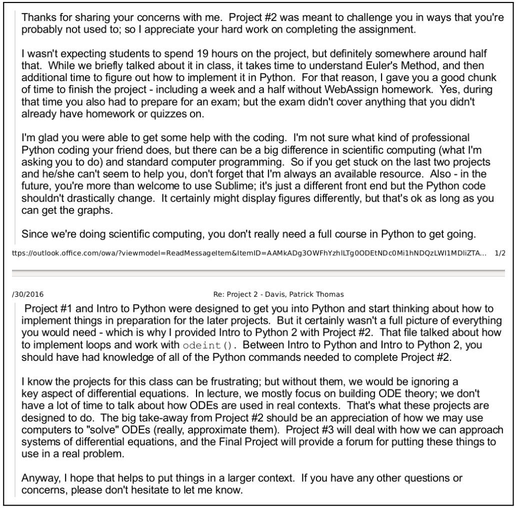
Subject: Project 2

I have attached what I was able to do with Project 2. I found this terribly difficult, and was not prepared to program in Python. I understand we had a while to do this program, however we also had to do homework and have a test in the class along with trying to learn Python. I really think this would be a great project but I feel like having a pre req for python would help considerably. I spent roughly 19 hours working on this, and I had a very good friend of mine who happens to professionally code in python assist me with the first question. We worked in Sublime and got this to work. However when i moved over to Juptyer none of the graphs worked out, this made it considerably more difficult to figure out what I needed to fix in my code.

Anyway, I know that a good majority of the class, was feeling very frustrated with this as well as we were discussing it before the exam. Perhaps there are some simpler intermediate projects that could help?

- No Python prerequisite, difficult to balance with homework and exam
- Worked with a friend who is a professional computer programmer, using a different Python front end
- “Intermediate” projects might help

Student Email



Thanks for sharing your concerns with me. Project #2 was meant to challenge you in ways that you're probably not used to; so I appreciate your hard work on completing the assignment.

I wasn't expecting students to spend 19 hours on the project, but definitely somewhere around half that. While we briefly talked about it in class, it takes time to understand Euler's Method, and then additional time to figure out how to implement it in Python. For that reason, I gave you a good chunk of time to finish the project - including a week and a half without WebAssign homework. Yes, during that time you also had to prepare for an exam; but the exam didn't cover anything that you didn't already have homework or quizzes on.

I'm glad you were able to get some help with the coding. I'm not sure what kind of professional Python coding your friend does, but there can be a big difference in scientific computing (what I'm asking you to do) and standard computer programming. So if you get stuck on the last two projects and he/she can't seem to help you, don't forget that I'm always an available resource. Also - in the future, you're more than welcome to use Sublime; it's just a different front end but the Python code shouldn't drastically change. It certainly might display figures differently, but that's ok as long as you can get the graphs.

Since we're doing scientific computing, you don't really need a full course in Python to get going.

ttps://outlook.office.com/owa/?viewmodel=ReadMessageItem&ItemID=AAMkADg3OWFhYzhlLTg0ODEtNDc0Mi1hNDQzLWI1MDliZTA... 1/2

/30/2016 Re: Project 2 - Davis, Patrick Thomas

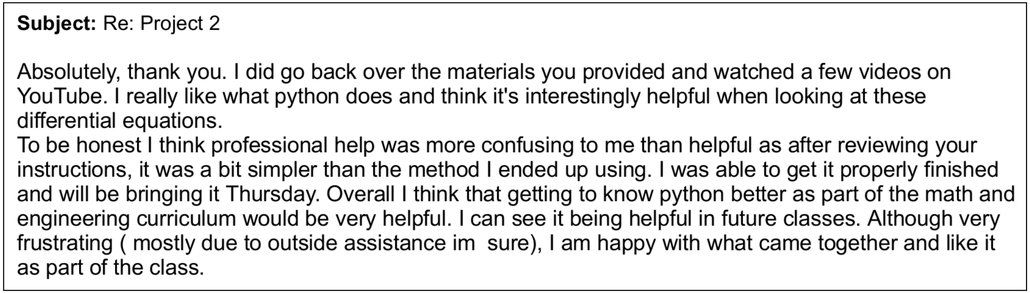
Project #1 and Intro to Python were designed to get you into Python and start thinking about how to implement things in preparation for the later projects. But it certainly wasn't a full picture of everything you would need - which is why I provided Intro to Python 2 with Project #2. That file talked about how to implement loops and work with `odeint()`. Between Intro to Python and Intro to Python 2, you should have had knowledge of all of the Python commands needed to complete Project #2.

I know the projects for this class can be frustrating; but without them, we would be ignoring a key aspect of differential equations. In lecture, we mostly focus on building ODE theory; we don't have a lot of time to talk about how ODEs are used in real contexts. That's what these projects are designed to do. The big take-away from Project #2 should be an appreciation of how we may use computers to "solve" ODEs (really, approximate them). Project #3 will deal with how we can approach systems of differential equations, and the Final Project will provide a forum for putting these things to use in a real problem.

Anyway, I hope that helps to put things in a larger context. If you have any other questions or concerns, please don't hesitate to let me know.

- Time is required to learn the numerical technique, and then implement it in Python
- Scientific computing is not the same as computer programming
- Mid-semester project #1 covered the Python basics, the accompanying `.ipynb` file covered everything else needed
- Projects help to put ODEs in a larger context (how they are typically “solved”, used in real life examples, etc.)

Student Email

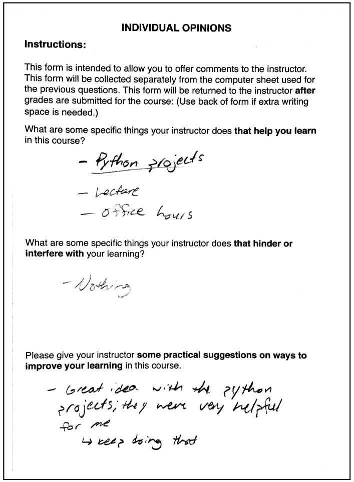


Subject: Re: Project 2

Absolutely, thank you. I did go back over the materials you provided and watched a few videos on YouTube. I really like what python does and think it's interestingly helpful when looking at these differential equations.
To be honest I think professional help was more confusing to me than helpful as after reviewing your instructions, it was a bit simpler than the method I ended up using. I was able to get it properly finished and will be bringing it Thursday. Overall I think that getting to know python better as part of the math and engineering curriculum would be very helpful. I can see it being helpful in future classes. Although very frustrating (mostly due to outside assistance im sure), I am happy with what came together and like it as part of the class.

- Looked at accompanying material and some YouTube videos
- The friend's help ended up being "more confusing"
- *"Getting to know Python [...] would be very helpful. I can see it being helpful in future classes [...] and like it as part of the class."*

Student Opinion Surveys



INDIVIDUAL OPINIONS

Instructions:

This form is intended to allow you to offer comments to the instructor. This form will be collected separately from the computer sheet used for the previous questions. This form will be returned to the instructor **after** grades are submitted for the course: (Use back of form if extra writing space is needed.)

What are some specific things your instructor does **that help you learn** in this course?

- Python projects
- Lecture
- office hours

What are some specific things your instructor does **that hinder or interfere with** your learning?

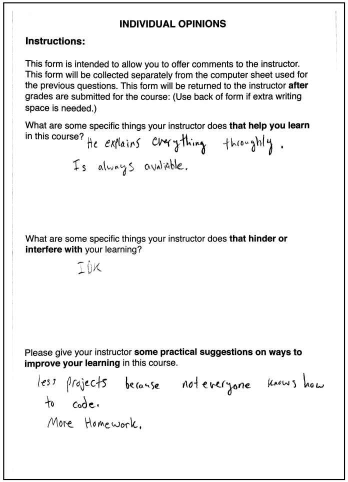
- Nothing

Please give your instructor **some practical suggestions on ways to improve your learning** in this course.

- Great idea with the python projects; they were very helpful for me
  ↳ keep doing that

- *"Great idea with the Python projects; they were very helpful for me $\hookrightarrow$ keep doing that"*
- *"I really liked the projects in the class"*

Student Opinion Surveys



INDIVIDUAL OPINIONS

Instructions:

This form is intended to allow you to offer comments to the instructor. This form will be collected separately from the computer sheet used for the previous questions. This form will be returned to the instructor **after** grades are submitted for the course: (Use back of form if extra writing space is needed.)

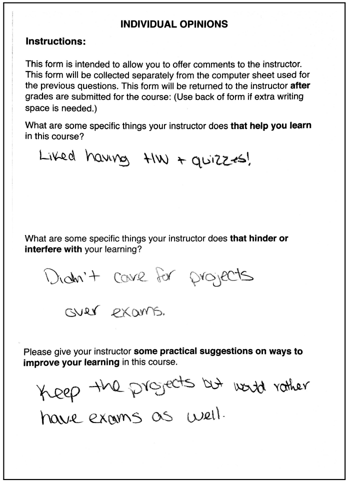
What are some specific things your instructor does **that help you learn** in this course? He explains everything throughly.
Is always avaliable.

What are some specific things your instructor does **that hinder or interfere with** your learning?
IDK

Please give your instructor **some practical suggestions on ways to improve your learning** in this course.
less projects because not everyone knows how to code.
More Homework.

- *"Less projects because not everyone knows how to code"*
- *"Python sucks, looks cool, but sucks"*
- *"Less projects, didn't learn much with no programming background"*
- *"I would have liked to have had a Python course before this I think I could have gotten a lot more from the class"*

Student Opinion Surveys



INDIVIDUAL OPINIONS

Instructions:

This form is intended to allow you to offer comments to the instructor. This form will be collected separately from the computer sheet used for the previous questions. This form will be returned to the instructor **after** grades are submitted for the course: (Use back of form if extra writing space is needed.)

What are some specific things your instructor does **that help you learn** in this course?

Liked having HW + quizzes!

What are some specific things your instructor does **that hinder or interfere with** your learning?

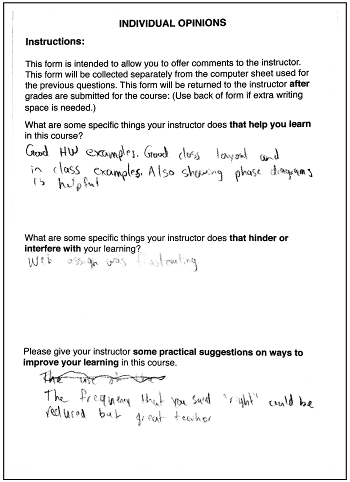
Didn't care for projects over exams.

Please give your instructor **some practical suggestions on ways to improve your learning** in this course.

Keep the projects but would rather have exams as well.

- *"Didn't care for projects over exams [...] Keep the projects but would rather have exams as well"*
- *"More exams, less projects"*
- *"Too many pts awarded to projects"*

Student Opinion Surveys



INDIVIDUAL OPINIONS

Instructions:

This form is intended to allow you to offer comments to the instructor. This form will be collected separately from the computer sheet used for the previous questions. This form will be returned to the instructor **after** grades are submitted for the course: (Use back of form if extra writing space is needed.)

What are some specific things your instructor does **that help you learn** in this course?

Good HW examples. Good class layout and in class examples. Also showing phase diagrams is helpful

What are some specific things your instructor does **that hinder or interfere with** your learning?

Web assign was frustrating

Please give your instructor **some practical suggestions on ways to improve your learning** in this course.

~~The use of the~~
The frequency that you said "right" could be reduced but great teacher

- *"Also showing phase diagrams is helpful"*

Victories:

- Highly annotated Intro to Python files
- Well-structured projects and other assessments
- Accessibility to students (office hours, emails, etc) for project help
- Culture of learning (project presentations provoked discussions on deeper topics, etc)
- Good for students interested in grad school

Challenges:

- Not enough course-project integration
- Attitude against the projects (learning scientific computing, place in the syllabus, etc)
- Too broad of a final project
- Difficult to grade projects
- Getting through all of the prescribed material

Changes I Would Make...

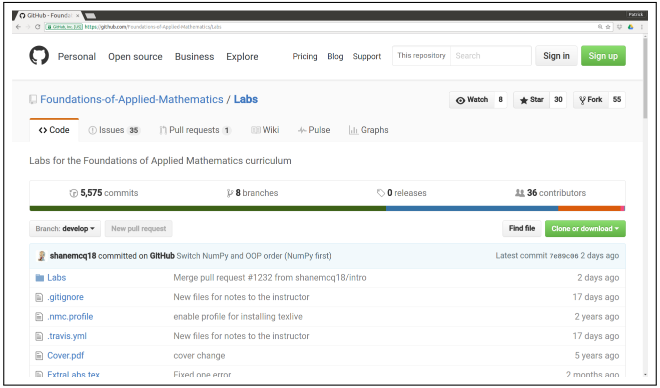
To Non-Python Aspects:

- Hybrid Homework-Quizzes (take home, spaced around project timing)
- Two Midterm Exams
- Other things...

To Python Aspects:

- Split Mid-Semester Project #1 (into two more detailed projects)
- Python Workshop (2-3 sessions, outside of class time for insterested students)
- More Structured Final Project (with 3-4 specific problems for students to choose between)
- Utilize Project Rubrics (as a guideline for student write ups and to help me while grading)
- Address Computer Programming vs. Scientific Computing Early On

Foundations of Applied Mathematics Curriculum



Foundations-of-Applied-Mathematics / Labs

Labs for the Foundations of Applied Mathematics curriculum

5,575 commits · 8 branches · 0 releases · 36 contributors

shanemcq18 committed on GitHub Switch NumPy and OOP order (NumPy first) — Latest commit 7e89c06 2 days ago

Labs	Merge pull request #1232 from shanemcq18/intro	2 days ago
.gitignore	New files for notes to the instructor	17 days ago
.nmc.profile	enable profile for installing texlive	2 years ago
.travis.yml	New files for notes to the instructor	17 days ago
Cover.pdf	cover change	5 years ago
Extral abs tex	Fixed one error	2 months ago

Developled for the Applied and Computational Mathematics degree program at Brigham Young University
By Jeffrey Humpherys and Tyler J. Jarvis, Department of Mathematics
Creative Commons Attribution 3.0 United States License, available at:
`https://github.com/Foundations-of-Applied-Mathematics/Labs/blob/develop/cc-by-license.txt`

References

[1] Continuum Analytics.
Anaconda Software Distribution, Vers. 2-2.4.0., 2015.
`https://continuum.io`.

[2] William E. Boyce and Richard C. DiPrima.
Elementary Differential Equations.
Wiley, 10th edition.

[3] Jeffrey Humpherys and Tyler J. Jarvis.
Foundations of Applied Mathematics.
Brigham Young University, 2016 (accessed July 2016).
`https://github.com/Foundations-of-Applied-Mathematics`.

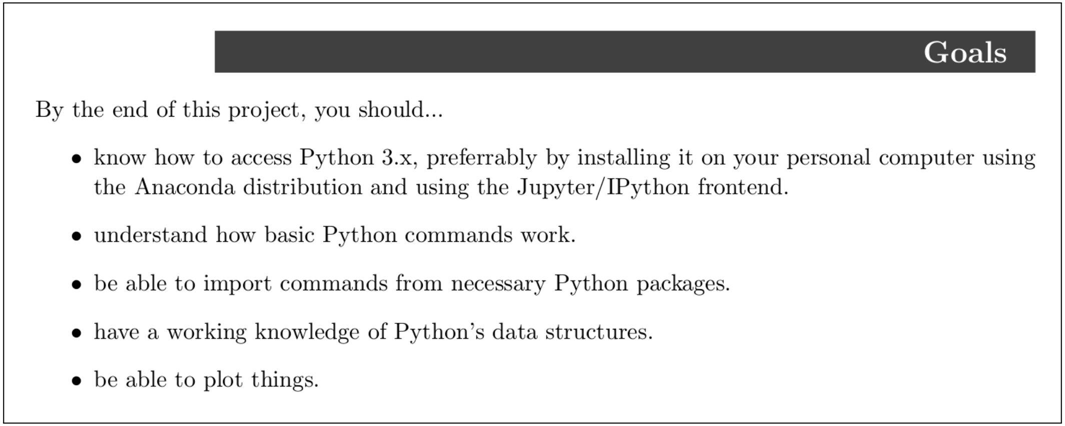
Thank You!

(**Patrick** – `davis1pt@cmich.edu`)

CMU

CENTRAL MICHIGAN UNIVERSITY

Project 1: Intro to Python



Goals

By the end of this project, you should...

- know how to access Python 3.x, preferrably by installing it on your personal computer using the Anaconda distribution and using the Jupyter/IPython frontend.
- understand how basic Python commands work.
- be able to import commands from necessary Python packages.
- have a working knowledge of Python's data structures.
- be able to plot things.

Project 1: Intro to Python

Background:

- Screencast on how to install Anaconda
- Screencast showing key parts of Jupyter/IPython

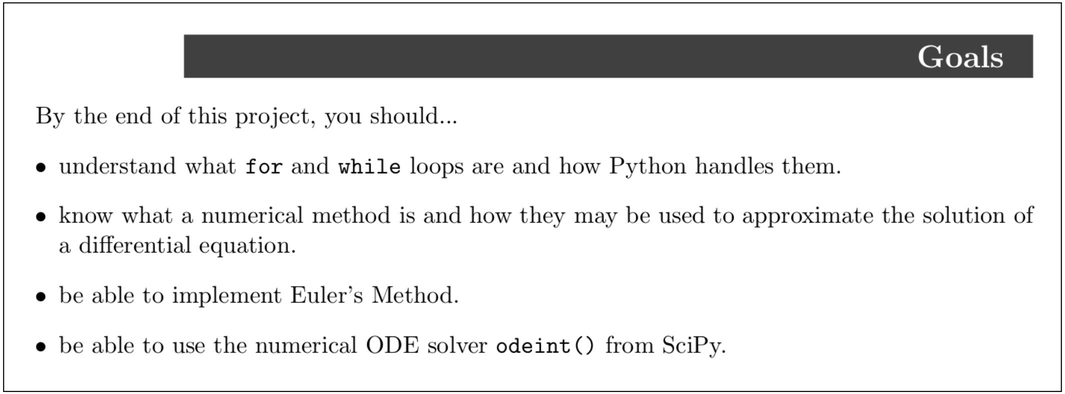
Intro to Python 1:

- Using Python as a calculator
- Storing values in variables and arrays
- Defining Python functions
- Importing NumPy and matplotlib

Exercises:

- Go through `Intro to Python.ipynb`
- Write a simple (Python) function
- Graph some (mathematical) functions

Project 2: Numerical Methods



Goals

By the end of this project, you should...

- understand what `for` and `while` loops are and how Python handles them.
- know what a numerical method is and how they may be used to approximate the solution of a differential equation.
- be able to implement Euler's Method.
- be able to use the numerical ODE solver `odeint()` from SciPy.

Project 2: Numerical Methods

Background:

- Review of Euler's Method (including an example with computed values)
- Psuedo-algorithm of Euler's Method

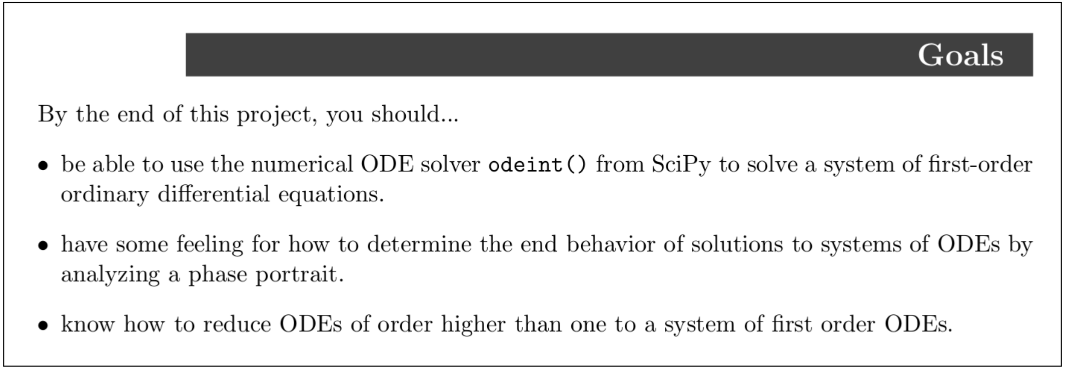
Intro to Python 2:

- How to implement (`for` and `while`)
- How to use NumPy's `odeint()` for a single first-order ODE

Exercises:

- Three initial value problems, solved with Euler's Method and `odeint()`
- Pros and cons of Euler's Method and `odeint()`

Project 3: Systems of ODEs



Goals

By the end of this project, you should...

- be able to use the numerical ODE solver `odeint()` from SciPy to solve a system of first-order ordinary differential equations.
- have some feeling for how to determine the end behavior of solutions to systems of ODEs by analyzing a phase portrait.
- know how to reduce ODEs of order higher than one to a system of first order ODEs.

Project 3: Systems of ODEs

Background:

- Review of how to reduce higher-order ODEs to a system of first-order ODEs

Intro to Python 3:

- How to use `odeint()` to solve systems of first-order ODEs
- Finding eigenvalues and eigenvectors using NumPy
- Code to generate phase planes of autonomous system of two first-order ODEs

Exercises:

- Solve systems of linear and nonlinear ODEs
- Analyze the respective phase plots and determine long-term behavior
- Solve and determine long-term behavior of a second-order nonlinear ODE