

## Lesson Plan

# Introductions to Class

---

## Summary

1. Introduction and Attendance
2. Go over blackboard site (Show how to find PAL Notes and request office hours)
3. POLL: How many have:
  - a. Used a computer
  - b. Know what a web browser is
  - c. Knows how to type
  - d. Can type what they see on the projector screen
  - e. Have programmed before
4. What is Python?
5. How to be successful for CS

Time Allotment: 50 Minutes

## Implementation


### Procedure

- a. Introduction

Hello Everyone, As I'm sure you know, since I'm standing up here, I am the PAL for this class.

My name is Alex Dasneves, and I am a junior here at Bridgewater State. I too have sat through this course in your position before, and I know some days may feel very boring. BUT if you are a CS student, this class will set you up for success. If you can understand the basics of programming, and the reasoning behind it, you can go far.

Over my time at Bridgewater, I have learned 5-7 programming languages (C, C++, Python, Java, X86, RISCy, Verilog). I have taken 10 CS Classes, the whole Calculus series of math classes (161, 162, 261), Physics I, and a couple others.



Please do not feel bad for booking office hours. They are there to help you, and I can submit for payment on them :)

If at any point you feel that I am going too fast, please let me know, and I can slow down.

So, lets see who you are! No need for the 3-ish paragraphs I had, just something basic, Name, Major(s), Minor(s), and Grade.

b. Blackboard

On the blackboard, there is a website that has all the information required for this course, including a basic Syllabus.

On this site there is a form to request office hours. While I do prefer this form, since it sends all the information nice and neatly to my email, and I can track the data from it easily, a regular email can suffice.

- c. Do the Poll on different categories above
- d. What is Python?

Python is a 'High Level' programming language.

Python implements a variety of useful features, such as Dynamic typing, Implicit Typing, and 'Duck' typing.


A python source code file is symbolized by the extension '.py'.

Python has a variety of different tools that come included (If you ask for them to be used), such as Trigonometric (and other math) Functions, mathematical pre-defined constants, such as e and pi. It also included a variety of powerful data science and data visualization tools.

For a while, MatLab was the 'Go-To' math and science programming language, because of its superior numeric computation, and symbolic math engine (Basically allowing you to type in a function with variables and have it spit out possible solutions), however, Python is becoming more and more mainstream, as all basic functionality is there, and some of the more advanced things are simple(ish) to make after learning the language.

Python is useful as an introductory language because of its 'near english' syntax styling. While that might not make sense now, it will when we start to get into programming.

When you eventually take CS2, (COMP 152), you will be learning a language called Java. Java implements a lot of the same features as python, however, some are notably absent



from the language. I won't harp on that topic for too long, since that is a topic for a future class.

If you are wondering what Python can do, the answer is a lot. Personally I have used it for:

- Basic Game Development (Minesweeper)
- Mathematical Simulations (Some of which might be given to you as projects, or worked through in the PAL class)
- Physics Simulations (Applying Newton's Laws, and the kinematic equations on objects in a virtual universe)
- Creating a programming language (COMP 340)
- Creating an 'Assembler' (A program that takes somewhat human-readable code, and translates it into what your computer can actually run)
- Modeling a basic computer

As you can see, these are more complex ideas, but it's all stuff that can be done in this language.

Not all of this will be done this semester. Some of it takes a few years of learning the intricacies of the language, and a good bit of determination and perseverance.

Now, with that said, here are a few tips I've learned from taking CS Classes here:

- Start your projects early.
  - Sometimes it might seem fairly straightforward, but there may be a time that you can't seem to figure out what to do.
  - It's very beneficial to take a step back, do something for a bit, (Either going outside, or some other classes work) then coming back and giving it another try.
  - There have been times where I will run into a wall (metaphorically), and I have taken a step back, and came back with the solution.
- Don't overthink the solutions.
  - This is a CS 1 class. All the stuff you need to complete your assignments is in the slides.
  - When you overthink the solution, you are bound to have some issue with your code.
- Keep your code clean and readable
  - Not super applicable in this course, but in the future, there are classes with projects spanning hundreds to thousands of lines.
  - By keeping your code clean and readable, you ensure that you, as well as your professor, can read your code, and understand what is happening.
  - There are standards online for clean code. Look up **PEP 8** if you are interested.
- Comments!



- Comments are going to be the best thing ever.
- Comments are ways to note what is happening in your code, in your own words.