Chapter 1

Intro

I took this course many moons ago and so now I'm making notes based on what I remember from the course and my own experience playing around with programming languages. That being said, I take a pretty holistic approach to this course. That is, I assume that you have a good understanding of the previous classes you needed to get here and we will talk about how what you learn here relates to what you have learned previously.

1.1 What is a language?

After pulling out my HESP140 notes, I can say with some confidence that a formal definition for language can be simply put as a system of communication. However, after pulling out my philosophy notes, I want to say that language is anything that transfers what goes on 'in our mind' to 'out of our mind.' I'm sure more important people would disagree, but eh.

What I am trying to get at is the fact that we all have thoughts and feelings, and ultimately no one knows what goes on in our heads until we express or share what we are thinking and feeling. To be put even more succinctly: a language is a medium used to express ourselves. And in my experience, programming languages are no different. However it's not that simple.

1.2 How do we use language?

So now that we answered what a language is, we can now focus on one part of what this course wants to make sure you understand: how to express ourselves with a language. To answer this question, we need to learn some new words: semantics and syntax. Semantics refers to the meaning of sentences/languages while Syntax refers to the structure of the language.

Consider the following:

• The snow is white
• Schnee ist weiß
• Precipitation comprised of ice cystrals under the temperature of 0°C reflects all wavelengths of light from 400 nm to 700 nm.

I would argue that all 3 sentences have the same meaning. That is, the semantics of the sentences are the same. Programming languages are the same. Consider the following:

---

1I would recommend taking philosophy of language with Alexander Williams
2Thomas Nagel has a fun little paper called "What is it like to be a bat" that says no matter how much we know about bats and no matter how hard we imagine what echolocation would be like, we could not experience how a bat navigates.
3Some would say this sounds like grammar. This is partially true since grammar is a subset of syntax
The semantics of these two programs are the same, despite them looking different. This is where syntax comes into play. Since syntax deals with the rules of what is valid or not, let us take an even smaller example:

```
\java
System.out.println("Hello, World!");
```

```
*C*
printf("Hello, World!");
```

Syntax deals with what is valid rules to create a sentence in a language. If we tried to write the first line in a C program, the compiler will yell at us, and had we tried to write some C code in Java, the compiler will be yell at us again. That is to express the same thing, in one language requires one thing, and in another something else. We will eventually talk about grammar, but for now just keep in mind the idea of syntax and semantics. An array will always be an array, but the code you use to make one will differ from language to language. That said, most languages are Turing complete (which we will also discuss later), so basically any program you make in one language, can be made in a different one which raises the following question(s).

### 1.3 Why so many languages? Why use one over another?

As we hopefully all know, computers only really know is machine code. But we as programmers don't really know or play around with bytecode. We use other languages which are easier to write with because they have shortcuts or macros that cover the hard and tedious stuff. Consider the following assembly code:

```
func:
    push   ebp
    mov    ebp,esp
    ; code
    mov    esp, ebp
    pop    ebp
```

This particular example represents the stack frame that is added to the stack whenever a function is called. It would be terrible if we had to do this everytime we wanted to call a function, so if we can replace the aforementioned assembly with something nice:

```
func();
```

That is most languages have some way to implement or represent a function call (typically means adding parentheses after the function name). The idea of having special shortcuts in a language is the basis for the second point this course finds important: language features. Different languages features is why there are so many languages and why you way want to use one language instead of another.

### 1.4 Language Features

Let us consider the following Java code:
1.5 Conclusion

This course focuses on a few things

- Recognizing language features and analyzing their effects on problem solving
- Imperative and functional programming
- Computational abilities of regular, context-free and Turing-complete languages
- Deriving meaning from a language
- Creating proofs about the correctness of a program
- Designing languages and implementing their evaluation