# C

# Why learn C?

- help develop better understanding of what higher (and lower) level languages are doing
- help understand memory management
- system programming - interact with the OS
- to pass this class :)

# Compiled vs Interpreted Languages

- Compiled:
  - C ( `gcc program_file.c` )
  - Java ( `javac program_file.java` )
- Interpreted: can simply run without explicitly compiling
  - Bash ( `bash name_of_program.sh` )
  - Python ( `python name_of_program.py` )
  - Awk ( `gawk name_of_program.awk` )

# C vs Other Languages

- Portability:
  - C = less portable, compiled code is architecture dependent
  - Java = more portable, compiled byte code runs the same on same JVM version
  - Python = more portable, mostly depends on python version, though underlying
    packages may be compiled underneath

# C vs Other Languages

- Speed
  - C is fast
  - Java is slower, but can be reasonably fast (JIT can make optimizations)
  - Python is known for being slow, but can be fast
    - Speed comes when using libraries implemented underneath in C

# Getting Started with C

## Hello World Program

```
#include <stdio.h>
int main(void) {
    printf("Hello World\n");
    return 0;
}
```

# C Preprocessor

- Performs source code substitution
- `#` indicates preprocessing directive
- `include` - replaces line with contents of file
  - typically use to include header files (we'll talk about what these mean later)
  - kind of like "importing"
- `define PI 3.14159`
  - everywhere `PI` occurs, replace with value
  - typically use all caps

# Running C Code

## Compiling

```
gcc program_file.c
```

```
gcc -o output_name program_file.c
```

## Running

```
./a.out
```

```
./output_name
```

# :( don't do this

```
gcc -o project.c project.c
```