CS 103000
Prof. Madeline Blount

Week 9:
FUNCTIONS

attendance link:

https://cs103-3proton.glitch.me

Dall-E 2: cats learning C++ in the forest on '90's technology

🤖 Mid-term grades + notes

- Autograder = part 1
- I will post adjusted grade before next class (should be Tues. evening), Blackboard column
- Also adjusted HW/labs grade, and total grade so far
- For those with < 70% (total grade):
  - There will be extra credit opportunities

🤖 Mid-term: some observations

- 👍🎉 comments! + creativity, individuality
- Watch out for:
  - Indentations, variable names 👀
  - When can you use loops?
  - Always, always try
- GOAL != TO FOOL AUTOGRADER!
  - Hardcoded outputs (--)
  - Case-sensitivity question, coin toss (if statement vs. lower/upper)

USER-DEFINED FUNCTIONS!

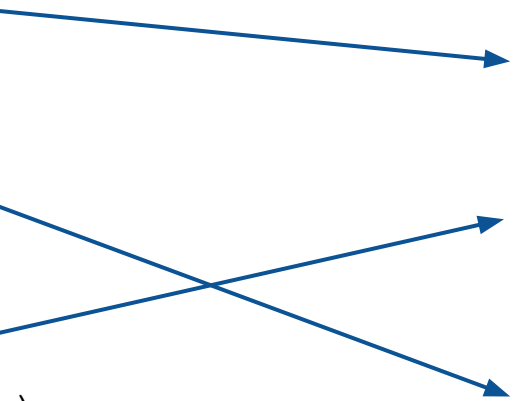
- Building blocks of code
- Set of executable statements (runs when called)

WHY?

- Easier to read (style)
- Easier to debug
- Easier to reuse, not repeat code
- Modular

Function structure:

- **1. Declare**
  (above main)
- **2. Define**
  (below main)
- **3. Call**
  (inside main)



```cpp
#include <iostream>

int sum(int a, int b);

int main() {
  int r = sum(10, 20);
  std::cout << r;
}

int sum(int a, int b) {
  return(a + b);
}
```

📣 Function declaration anatomy:

return type name(parameter list);

```
int myFunction(int var1, int var2);
```

📣 Function definition anatomy:

return type name(parameter list)

```
int myFunction(int var1, int var2)
{
// my code!
   return something
}
```

📣 Function call anatomy:

name(arguments);

```
myFunction(num1, num2);
```

📣 VOID function anatomy:

return type name(parameter list)

```
void sayHello(string name) {
// my code!
    cout << “hello, ” << name;
}
```

```cpp
#include <iostream>

// Declaring a function
void print();

int main() {
  print();
}

// Defining a function
void print() {
  std::cout << "Hello World!";
}
```

📣 MAIN function anatomy:

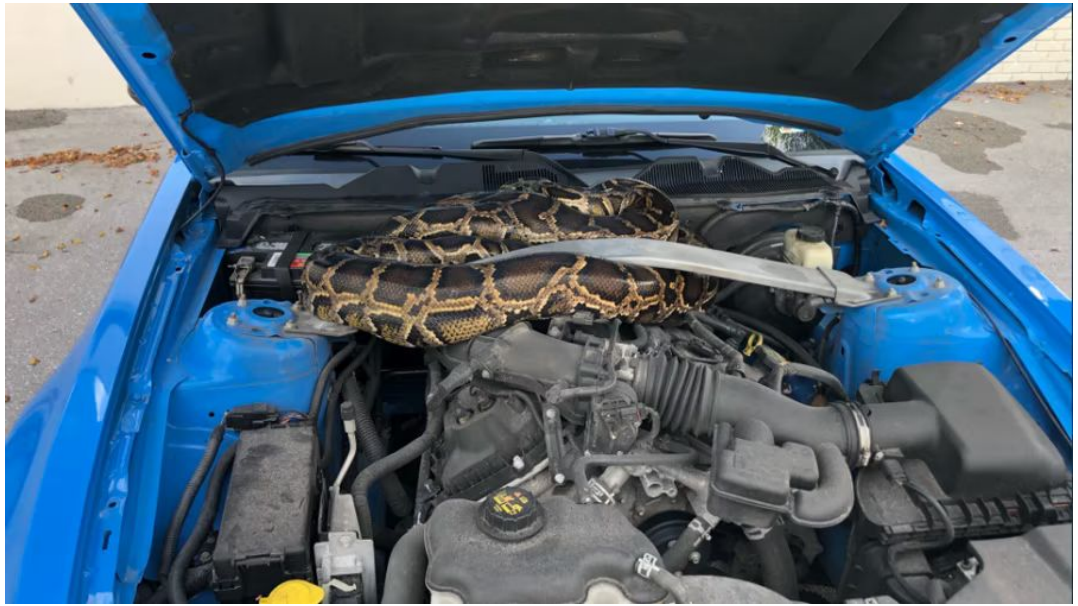
return type name(parameter list)
Return type = int!
0 = exit code, “success”

```
int main() {
// my code!
   return 0
}
```

https://cs103-3proton.glitch.me

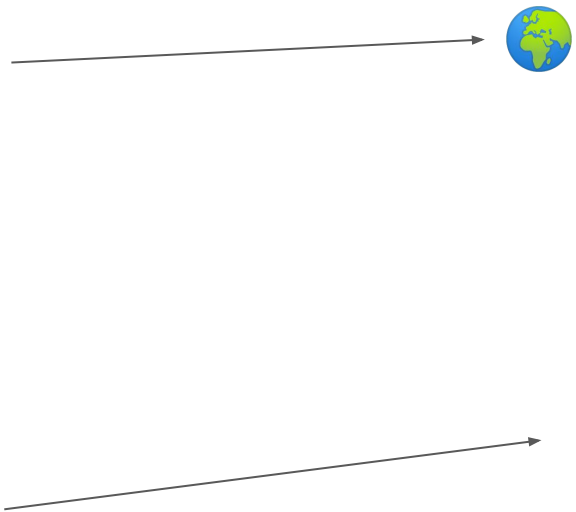
FUNCTIONS: abstraction

farther **away** from the guts, computation, data, hardware, etc.



🔭 SCOPE

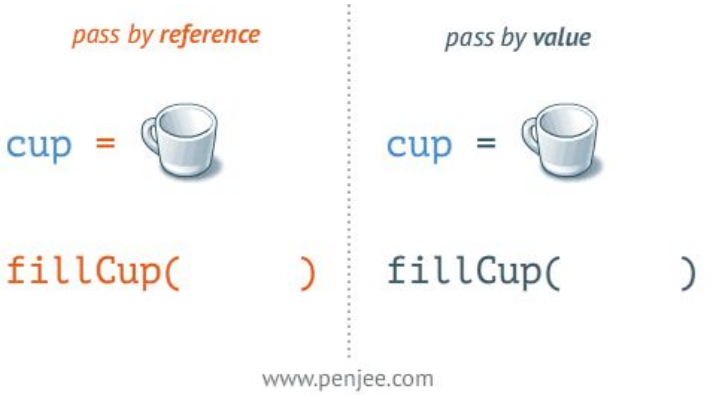

GLOBAL
everywhere!
local
only accessible in the function where they were created



```cpp
#include <iostream>

void print();

int i = 10;       // global variable

int main() {
  std::cout << i << "\n";
}

void print() {
  int j = 0;      // local variable
  i = 20;
  std::cout << i << "\n";
  std::cout << j << "\n";
}
```

"PASS BY VALUE":
everything we have done so far, local variables stay local, because every parameter is a "copy"

"PASS BY REFERENCE":
can treat parameter like a global variable, changes outside of scope



"Function overload":

Functions can have the same name but handle different data types

```
int cubeNumber(int x);
double cubeNumber(double x);
```