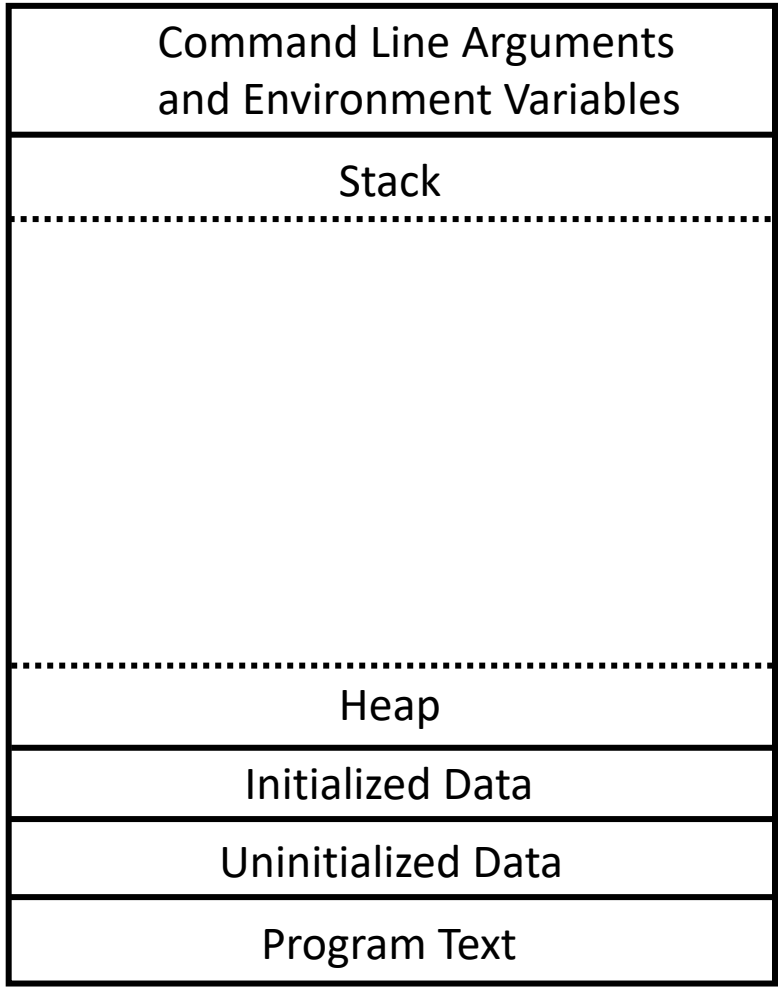
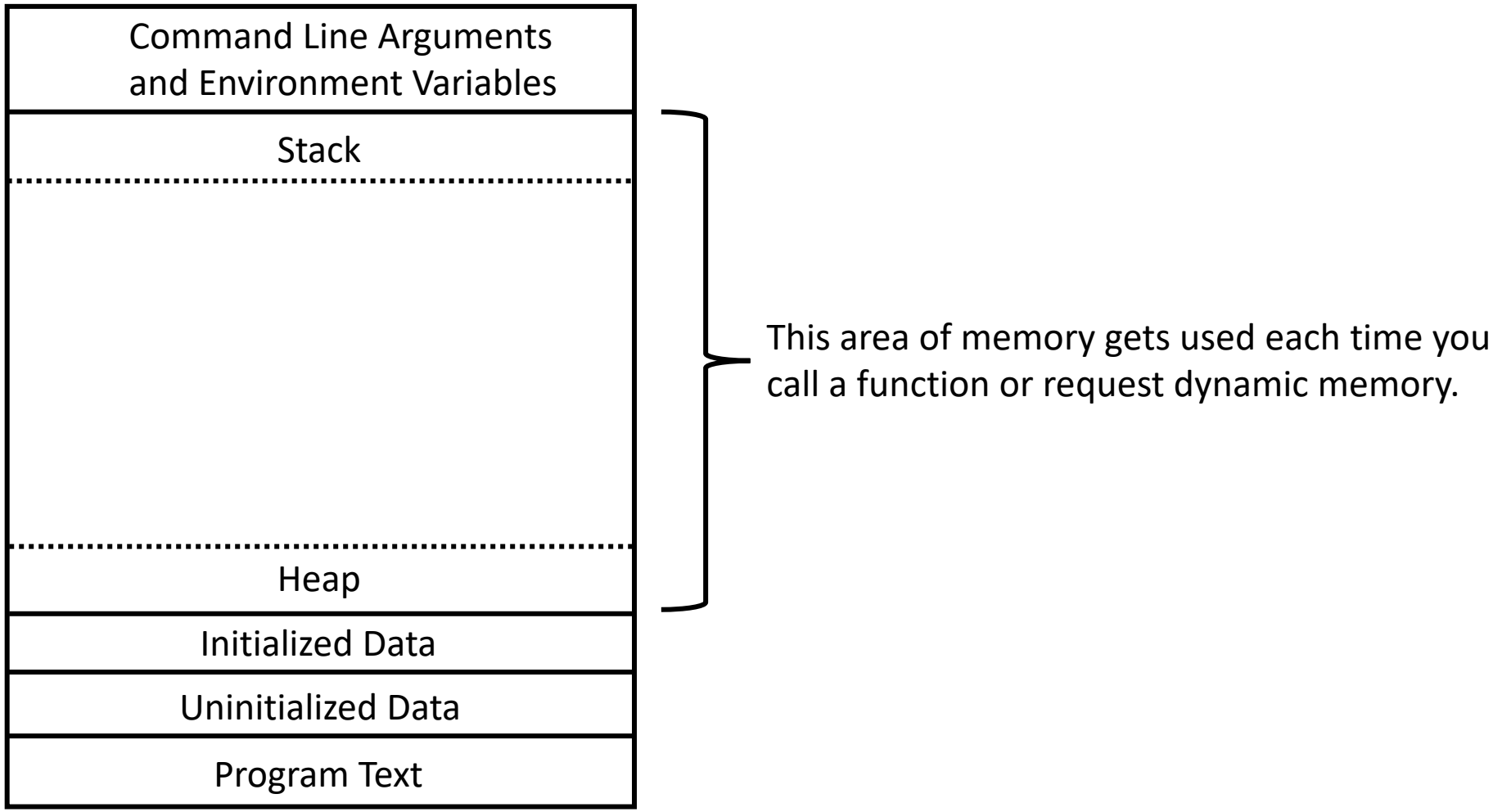
The Call Stack

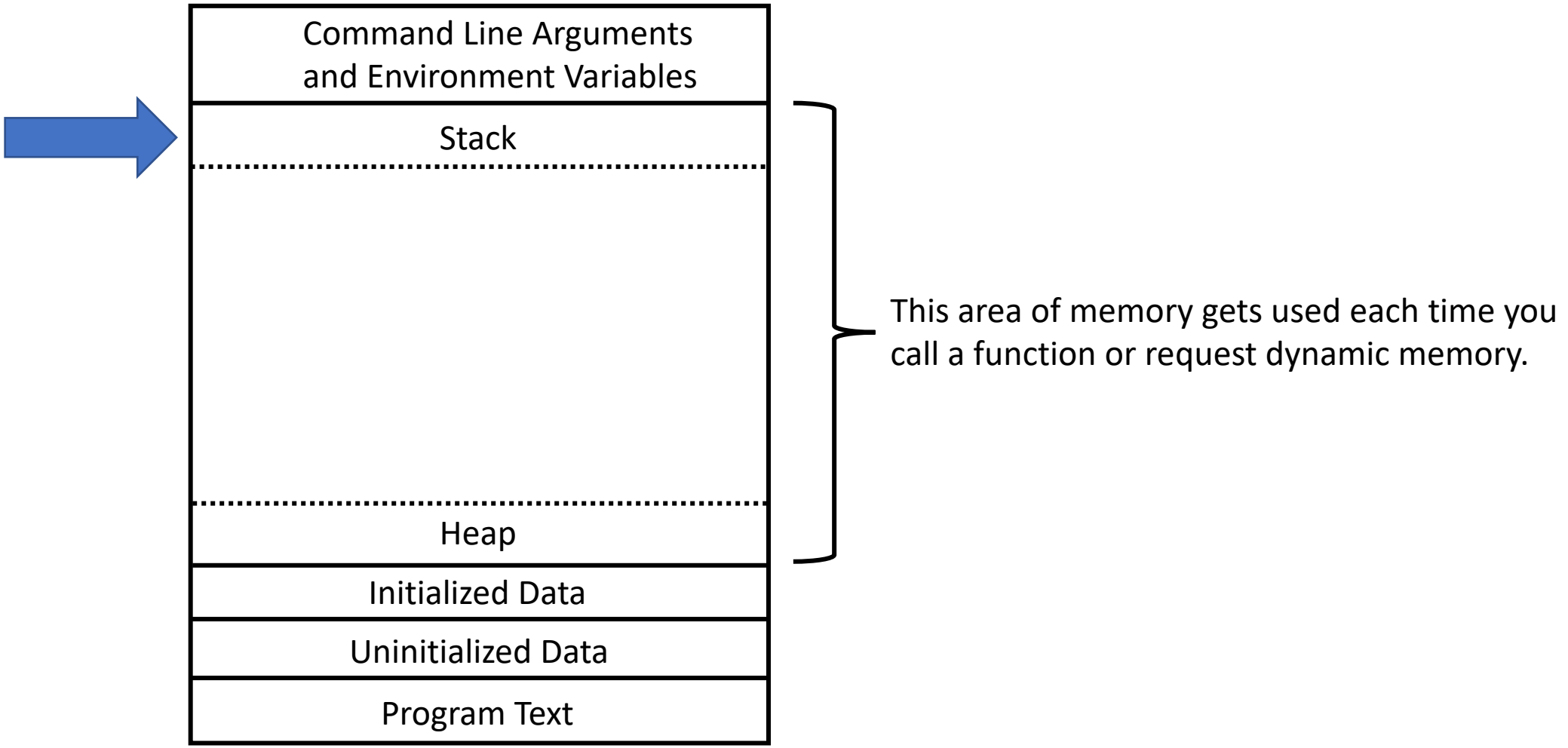
Programs in Memory



Programs in Memory



Programs in Memory



What is a Stack?

- A stack is a type of data structure
- Think of it like a stack of papers or dishes
- We can:
  - add items to the top with a **PUSH**
  - remove items from the top with a **POP**

The Call Stack

- Often just called “the stack”.
- Every running program has its own stack
- Each time a function is called a stack frame is pushed onto the stack
  - The function at the top of the stack is the active function
- A stack frame consists of:
  - Return address (where in the code the function was called)
  - Automatic variables used by the function

Automatic Variables

- All the variables we have been using so far have been automatic variables.
- When a function is called, memory (RAM) is allocated for local variables and function parameters.
- The memory is automatically released when the function returns (or reaches the end in the case of a void function).

How does this work?

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


int main() {
  int value = sum(1, 2);
  printf("%i\n", value);
  return 0;
}
```

The Stack

How does this work?

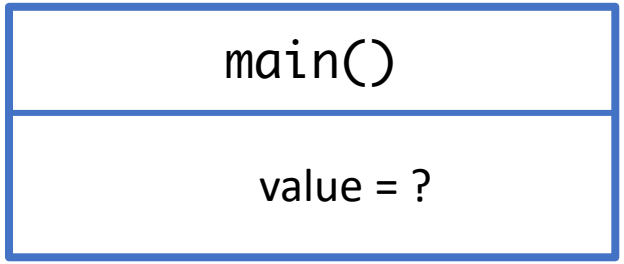
```
int sum(int a, int b) {
  return a + b;
}
```

```
int main() {
  int value = sum(1, 2);
  printf("%i\n", value);
  return 0;
}
```

The main function is pushed to the stack with its variables when the program starts.

The Stack

Active Function



How does this work?

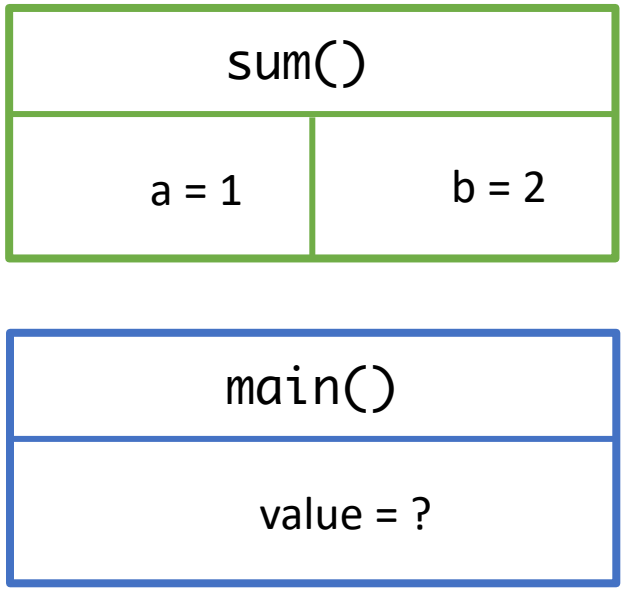
```
int sum(int a, int b) {
  return a + b;
}
```

```
int main() {
  int value = sum(1, 2);
  printf("%i\n", value);
  return 0;
}
```

Once we reach the call to sum, we need to push that to the stack.

The Stack

Active Function



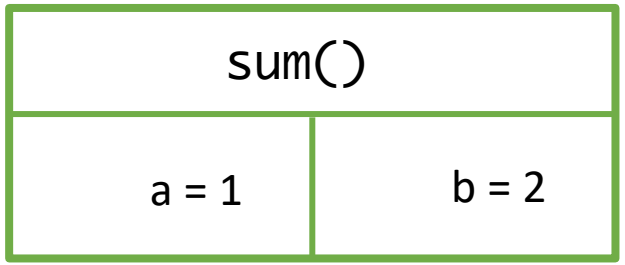
How does this work?

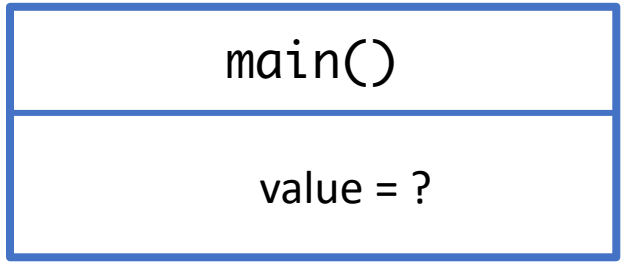
```
int sum(int a, int b) {
  return a + b;
}

int main() {
  int value = sum(1, 2);
  printf("%i\n", value);
  return 0;
}
```

When we return from sum we pop sum off the stack and go back to the main function.

The Stack

Active Function





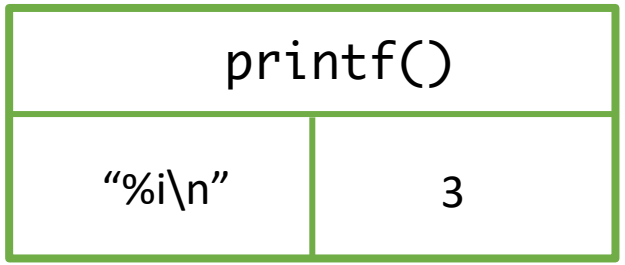
How does this work?

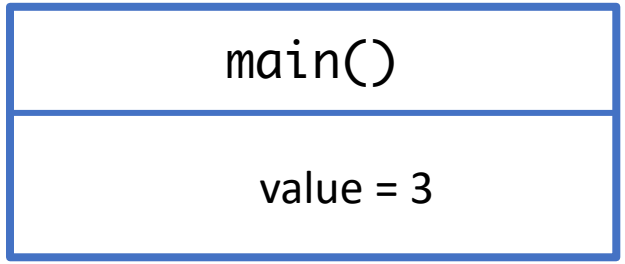
```
int sum(int a, int b) {
  return a + b;
}

int main() {
  int value = sum(1, 2);
  printf("%i\n", value);
  return 0;
}
```

Printf is also a function so that will have to be pushed on the stack.

The Stack

Active Function





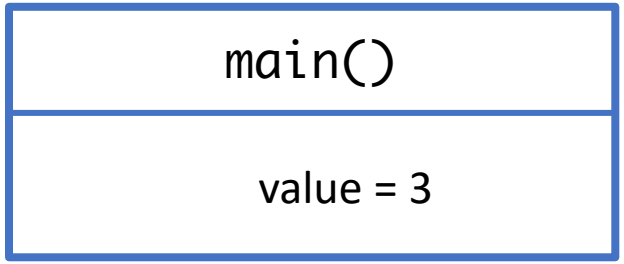
How does this work?

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

int main() {
  int value = sum(1, 2);
  printf("%i\n", value);
  return 0;
}
```

When printf is complete, we can remove the function from the stack and resume the main.

The Stack

Active Function



How does this work?

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

int main() {
  int value = sum(1, 2);
  printf("%i\n", value);
  return 0;
}
```

When main returns it is also removed from the the stack and the program quits.

The Stack

You can try this out for yourself!

- Python tutor can visualize the running of a single file C program and show the call stack

- http://pythontutor.com/c.html#mode=edit