

CS Summer Challenge

Day 0x2 / 0x3

How does a code work?

How does code work?

Variable Scopes

Variable **Scope**

- When a source code file is translated into binary code, **both instructions and data are translated**. Each are allocated their own memory slots in the final executable binary file.
 - Instructions: *function calls, function definitions, and assignment operations with the form ' $x = y +/- * z$ '.*
 - Data: *all declared variables.*
- An instruction will be able to use data if the data's location in the binary is known to it.
- Data can have 1) a **global** location or 2) a **local** location.
 - Global data can be located by all instructions of the binary file.
 - Local data can be located by the instructions in the block of code that created it.

```
x1 = 0
x2 = 0
result = 0
```

```
def get_input():
    x1 = int(input("Enter x1: "))
    x2 = int(input("Enter x2: "))
```

```
def add(x, y):
    sum = x + y
    return sum
```

} *add()
Code
Block*

```
def subtract(x, y):
    diff = x - y
    return diff
```

} *subtract()
Code
Block*

```
...
get_input()
result = add(x1, x2)
```

Global Code Block

Global Variables

```
x1 = 0  
x2 = 0  
result = 0
```

←
declared
here

```
def get_input():  
    x1 = int(input("Enter x1: "))  
    x2 = int(input("Enter x2: "))
```

```
def add(x, y):  
    sum = x + y  
    return sum
```

} *add()
Code
Block*

```
def subtract(x, y):  
    diff = x - y  
    return diff
```

} *subtract()
Code
Block*

```
...  
get_input()  
result = add(x1, x2)
```

Global Code Block

```
x1 = 0
x2 = 0
result = 0
```

Global Variables

```
def get_input():
    x1 = int(input("Enter x1: "))
    x2 = int(input("Enter x2: "))
```

```
def add(x, y):
    sum = x + y
    return sum
```

*add()
Code
Block*

```
def subtract(x, y):
    diff = x - y
    return diff
```

*subtract()
Code
Block*

```
...
get_input()
result = add(x1, x2)
```

used here

Global Code Block

Global Variables

```
x1 = 0
x2 = 0
result = 0
```

```
def get_input():
    x1 = int(input("Enter x1: "))
    x2 = int(input("Enter x2: "))
```

```
def add(x, y):
    sum = x + y
    return sum
```

} *add()
Code
Block*

```
def subtract(x, y):
    diff = x - y
    return diff
```

```
...
get_input()
result = add(x1, x2)
```

*Local Variables - visible by
instructions of add function*

Global Code Block

Global Variables

```
x1 = 0  
x2 = 0  
result = 0
```

```
def get_input():  
    x1 = int(input("Enter x1: "))  
    x2 = int(input("Enter x2: "))
```

```
def add(x, y):  
    sum = x + y  
    return sum
```

} *add()
Code
Block*

*Local Variables - visible by
instructions of add function*

```
def subtract(x, y):  
    diff = x - y  
    return diff
```

} *subtract()
Code
Block*

*Local Variables - visible by
instructions of add function*

```
...  
get_input()  
result = add(x1, x2)
```

Global Code Block


```
x1 = 0
x2 = 0
result = 0
```

```
def get_input():
    x1 = int(input("Enter x1: "))
    x2 = int(input("Enter x2: "))
```

```
def add(x, y):
    sum = x + y
    return sum

def subtract(x, y):
    diff = x - y
    return diff
```

```
...
get_input()
result = add(x1, x2)
```

Global Variables

Local Variables - visible by instructions of add function

declared
here

Local Variables - visible by instructions of subtract function

Global Code Block

```
x1 = 0
x2 = 0
result = 0
```

Global Variables

```
def get_input():
    x1 = int(input("Enter x1: "))
    x2 = int(input("Enter x2: "))
```

```
def add(x, y):
    sum = x + y
    return sum
```

*add()
Code
Block*

*Local Variables - visible by
instructions of add function*

used here

```
def subtract(x, y):
    diff = x - y
    return diff
```

*subtract()
Code
Block*

*Local Variables - visible by
instructions of add function*

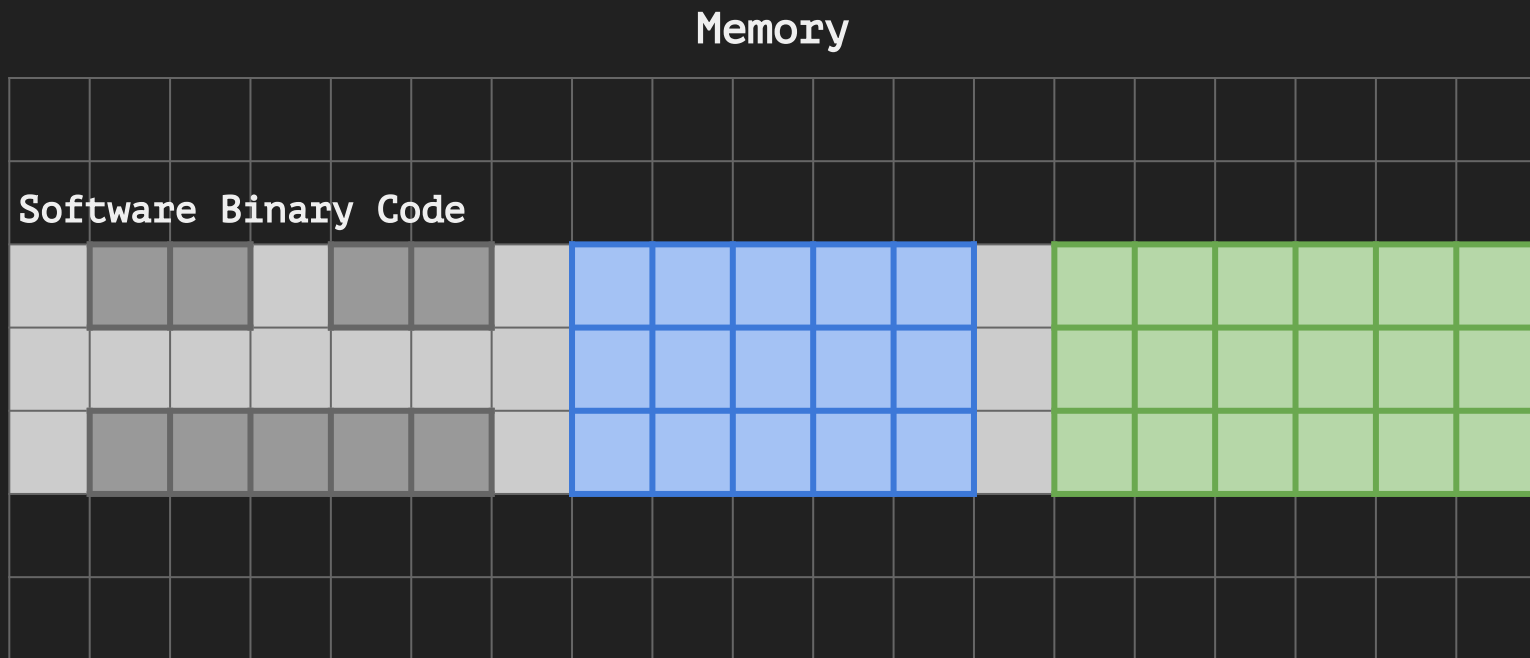
```
...
get_input()
result = add(x1, x2)
```

Global Code Block

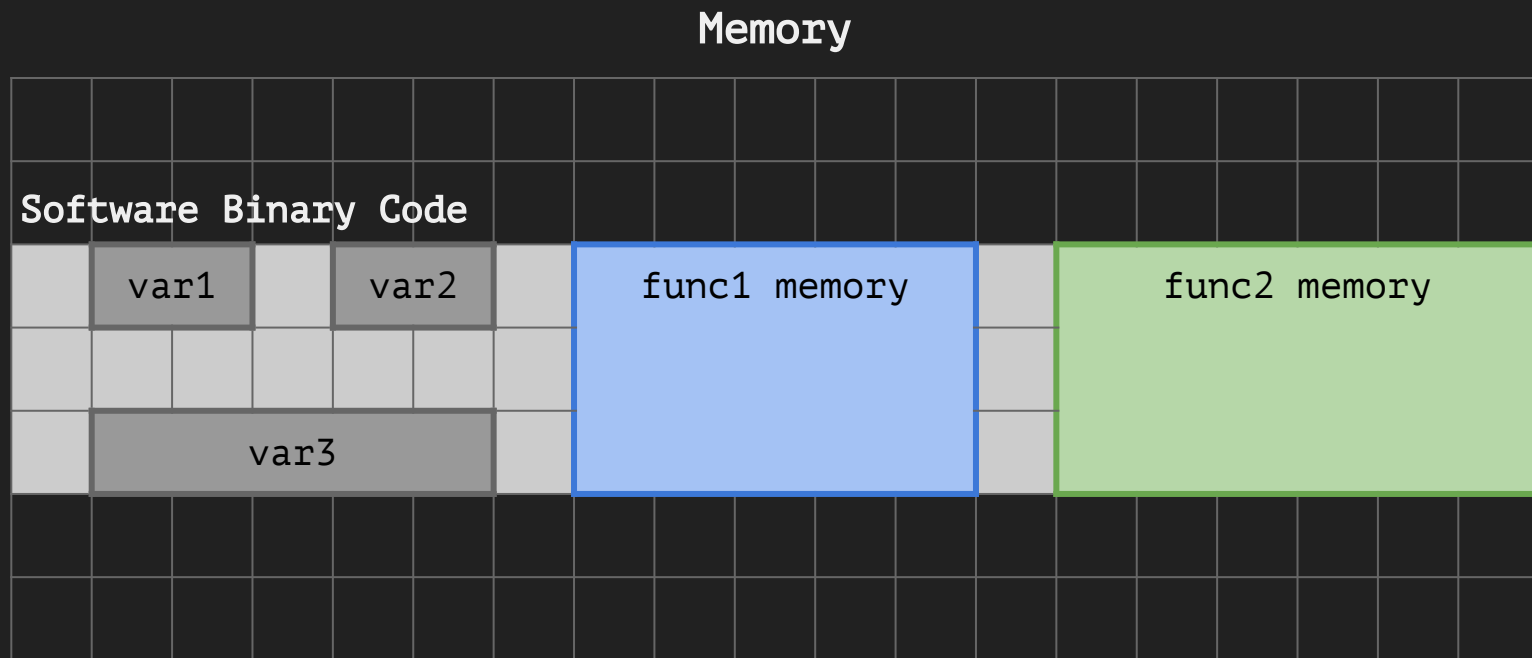
How does code work?

Variables in Action

Variables: *are translated into their binary encoding and added to the binary code of the software.*



Variables: *values change in the memory locations allocated for the variables*

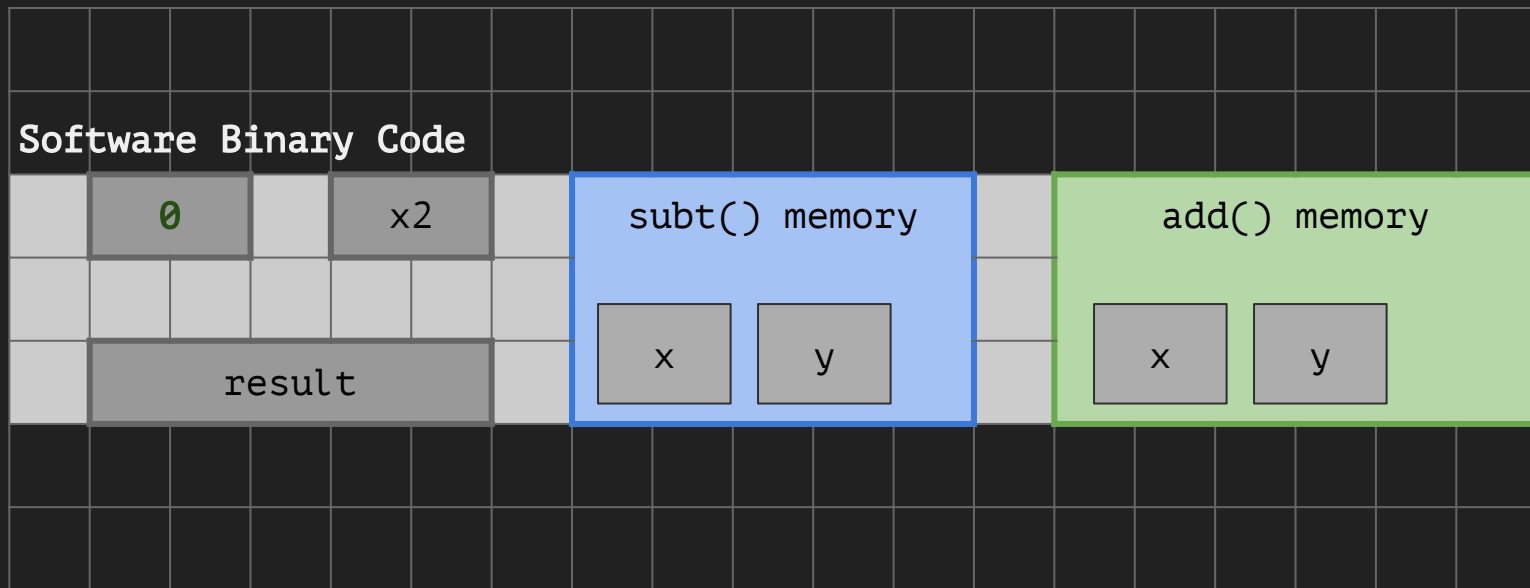


```
x1 = 0
```

```
...
```

```
def get_input():  
    x1 = int(input())
```

```
...
```



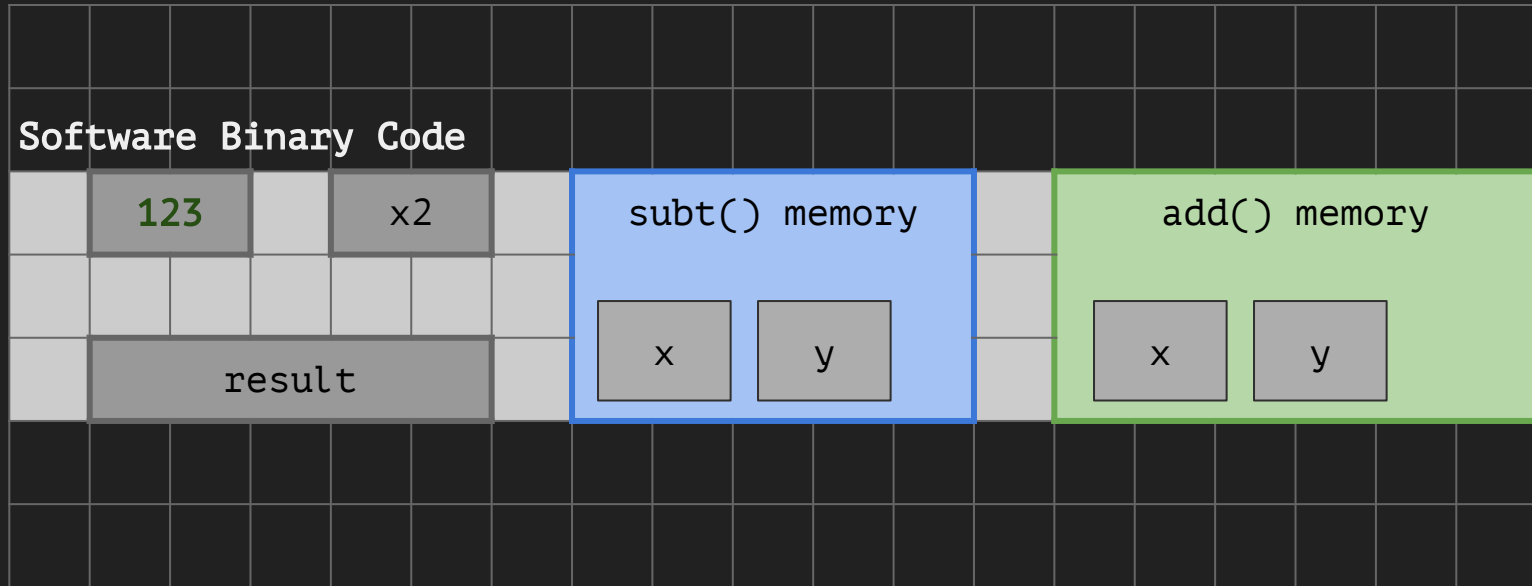
```
x1 = 0
```

```
...
```

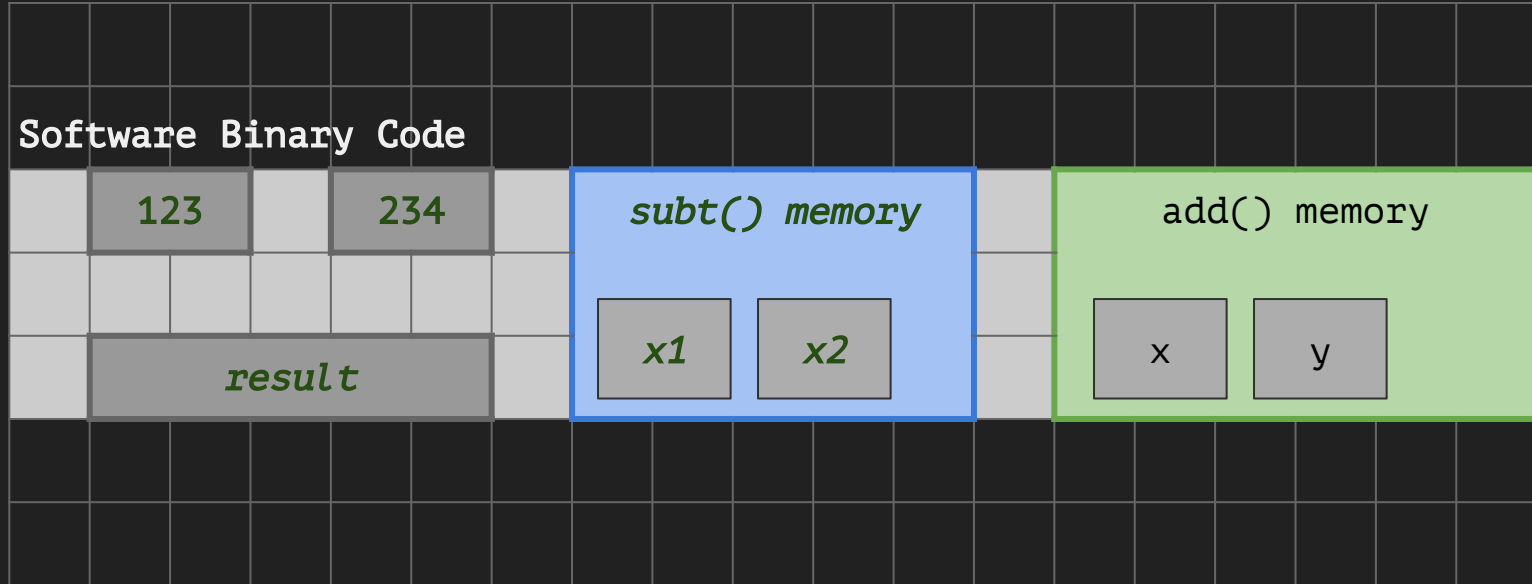
```
def get_input():
```

```
    x1 = int(input())
```

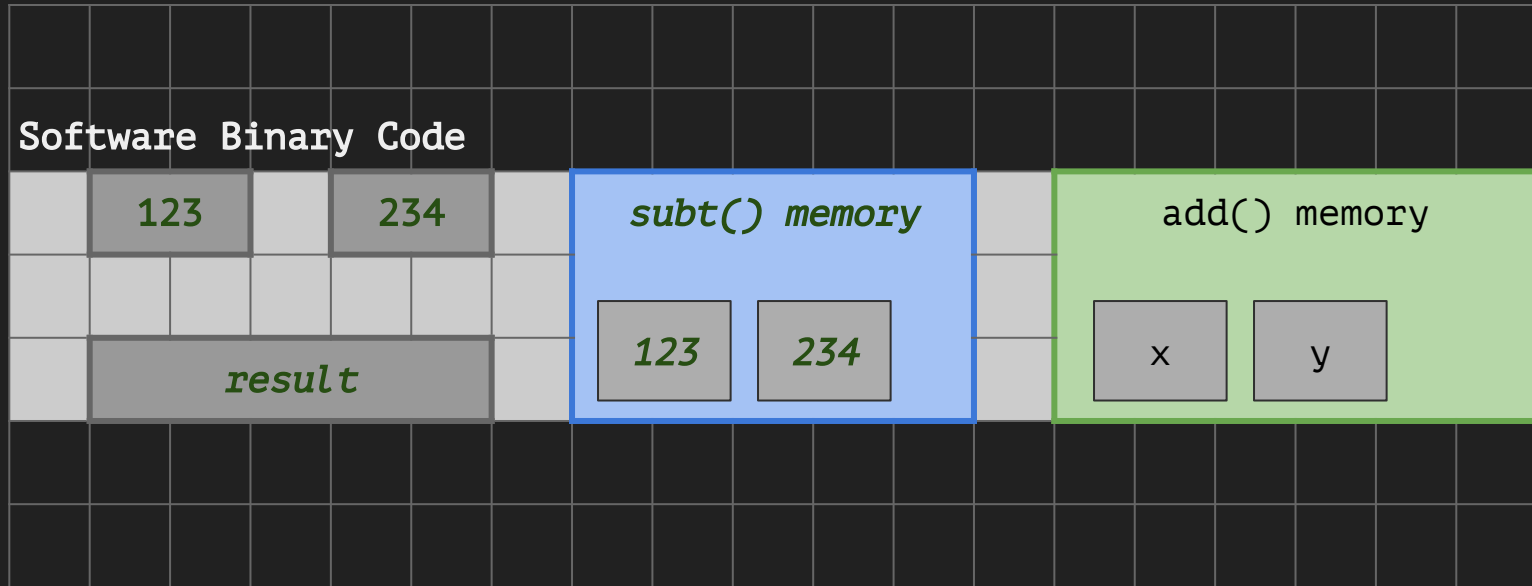
```
    ...
```



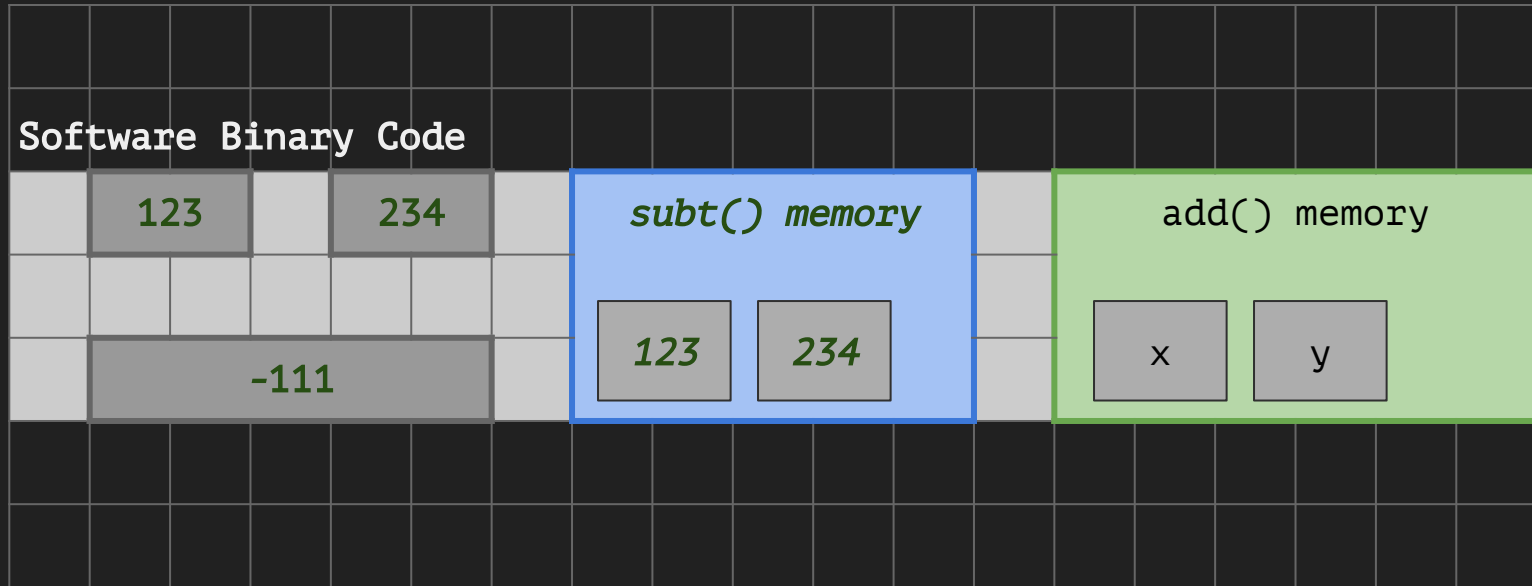
```
...  
result = sub(x1, x2)  
def sub(x, y):  
    return x - y  
...
```



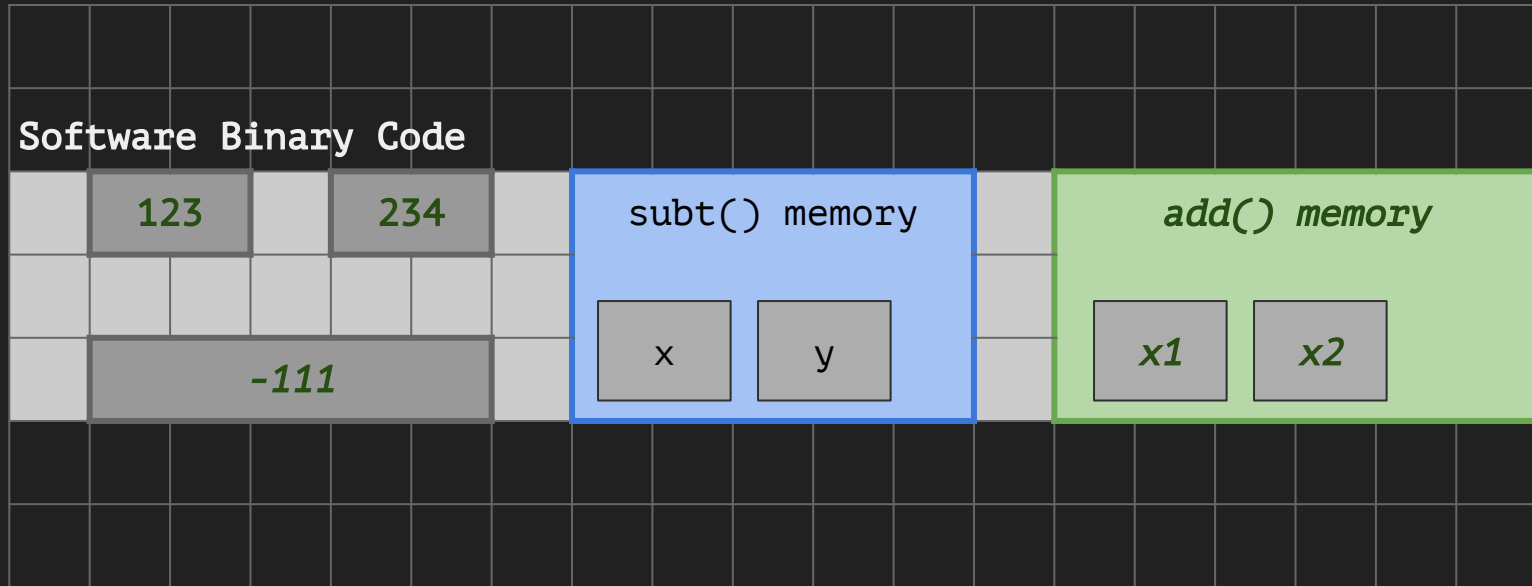

```
...  
result = sub(x1, x2)  
def sub(x, y):  
    return x - y  
...
```



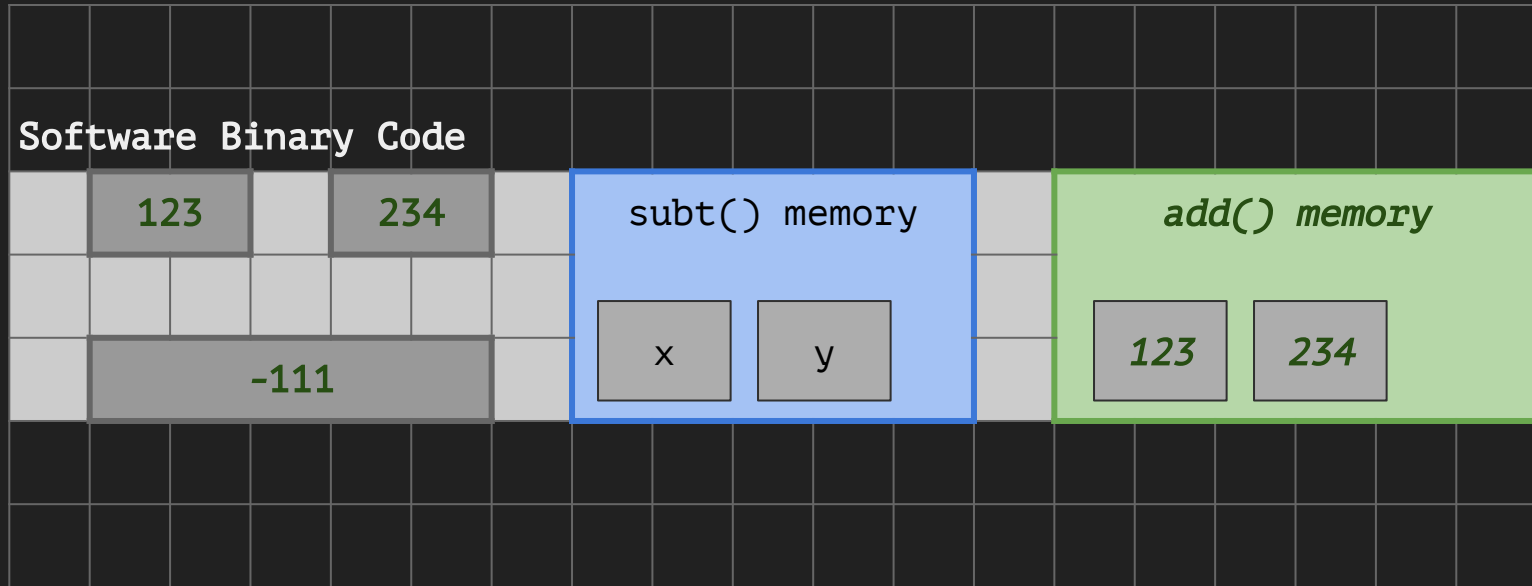
```
...  
result = sub(x1, x2)  
def sub(x, y):  
    return x - y  
...
```



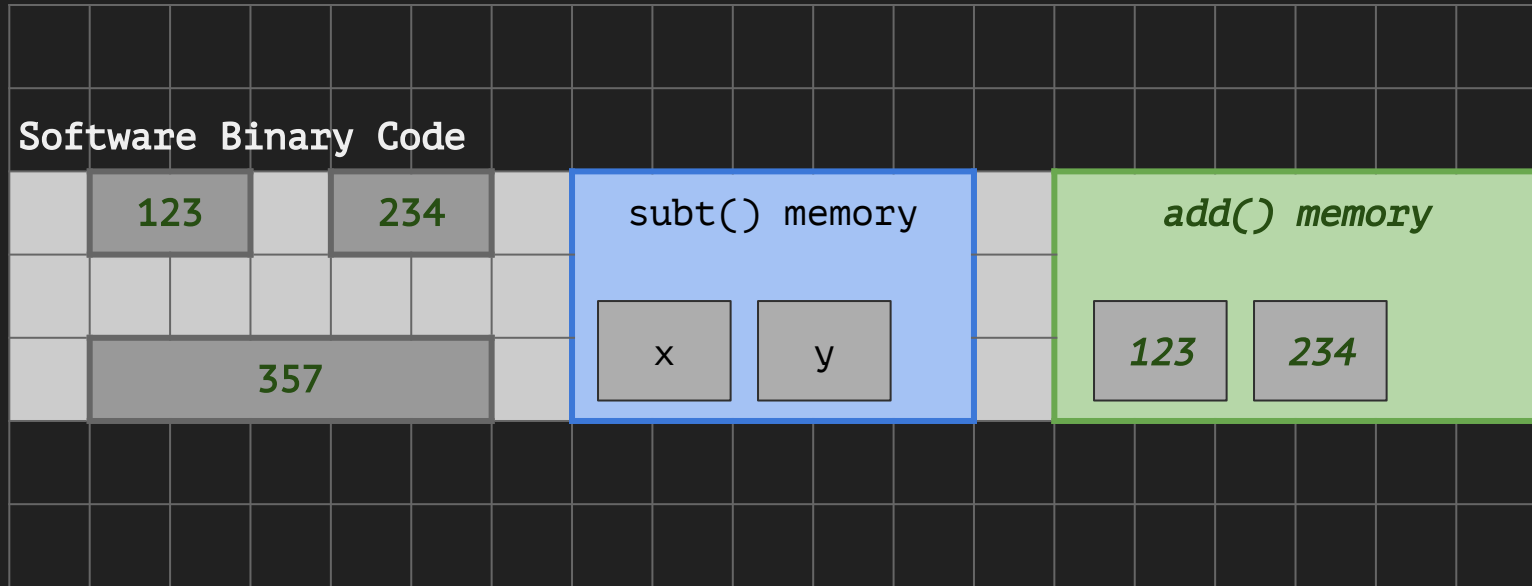
```
...  
result = add(x1, x2)  
def add(x, y):  
    return x + y  
...
```



```
...  
result = add(x1, x2)  
def add(x, y):  
    return x + y  
...
```



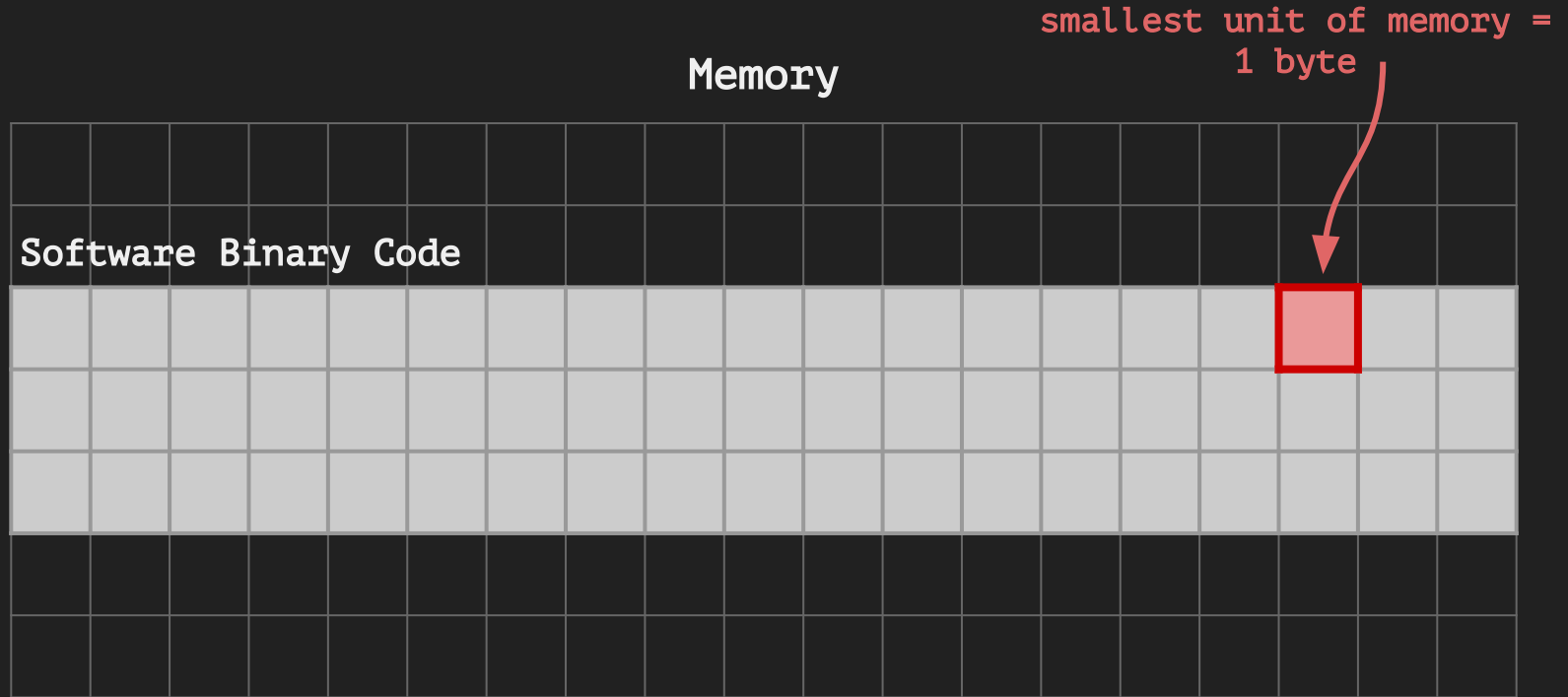
```
...  
result = add(x1, x2)  
def add(x, y):  
    return x + y  
...
```



How does code work?

Variable Types

Variables: *consume different amounts of memory based on their data type*



Variables: *consume different amounts of memory based on their data type*

Memory

integer				character															
26						c				decimal		number							
	2	6	\n		h	e	l	l	o		w	o	r	l	d	!	!\n		
string				string															

Variables: *have different binary encodings depending on their data type*

Memory

The diagram illustrates memory layout for a C program. It shows a grid representing memory cells. The first row contains 'integer' (green) and '01100011' (red). The second row contains '0...011110' (green), 'c' (pink), and 'decimal number' (blue). The third row contains a long binary string '001100100011011000010000' (brown/green/blue) and 'hello world!!\n' (orange). The fourth row contains 'string' (orange) and 'string' (orange).