COP 3223H: Introduction to C Programming

Fall 2023

University of Central Florida

Dr. Kevin Moran

Week 2- Class 1: Executable Statements

Administrivia

- Eustis assignment has been posted
  - Due on Sunday (Sept 3rd)
- Syllabus Quiz has been posted to Webcourses
  - Due on Friday (Sept 3rd) - should only take a few mins

Today's Agenda

1. Discuss Executable Statements
2. Demo and Activity for Connecting to Eustis

Executable Statements

Assignment Statements

- Assignment statements stores a value or a computational result in a variable and is used to perform most arithmetic operations in a program.

- = is called the assignment operator

```
int var;
var = 32;
```

- Syntax:
  - `variable = expression;`

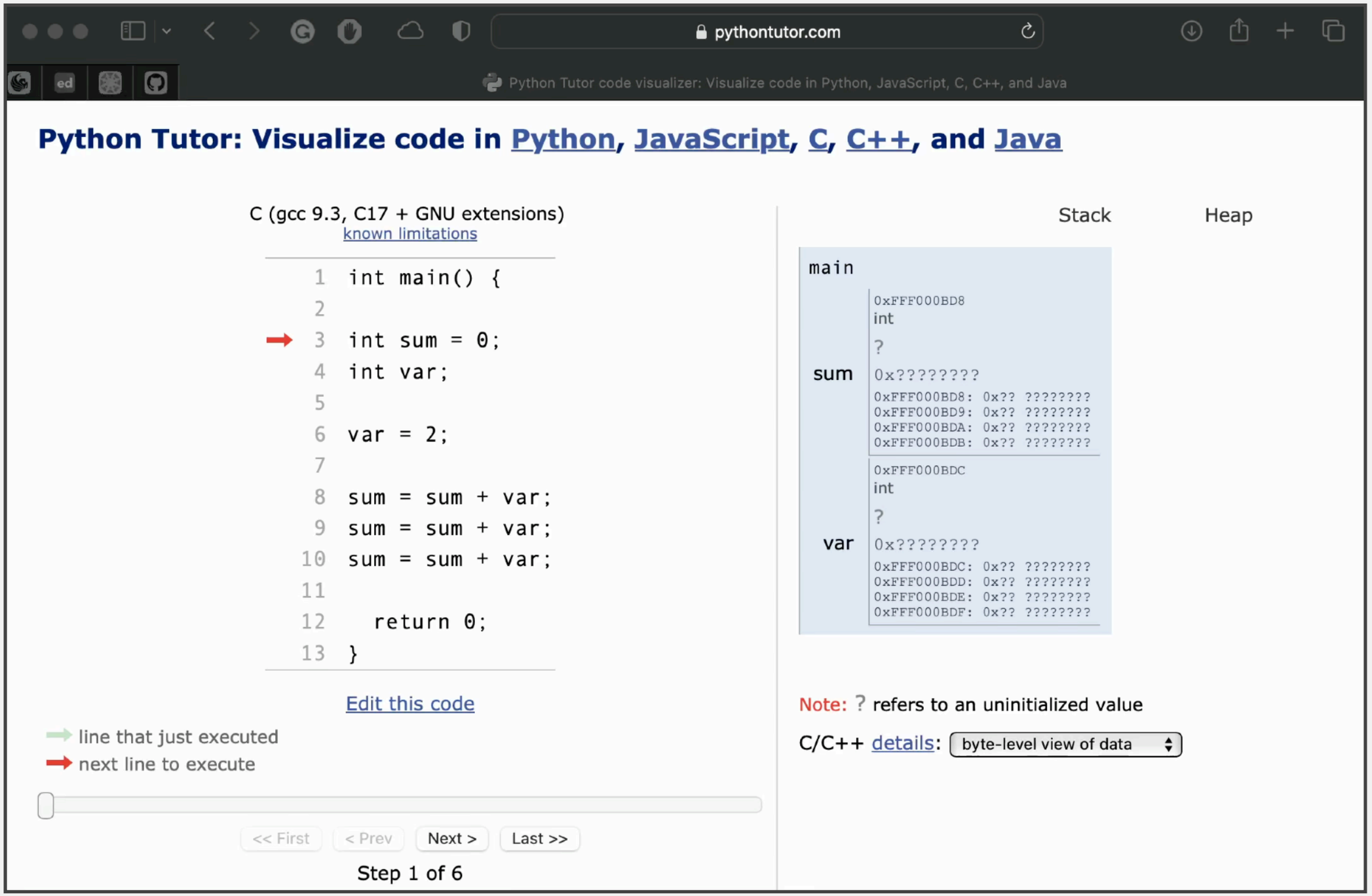
Compound Assignment Statements

- In C, you can create *compound assignment statements* in the form of:

```
sum = sum + var;
```

Yes! You are seeing double! Let's take a look at what is happening in a statement like this!

Compound Assignment Statements



Input/Output Operations and Functions

- Input operation is an instruction that copies data from an input device into memory. (Ex. Keyboard)
- Output operation is an instruction that displays information stored in memory.
- Input/Output functions are C functions that performs an input and output operation (comes from the `stdio.h`)
  - `printf() //display to screen`
  - `scanf() //collect input`

Printing Variables

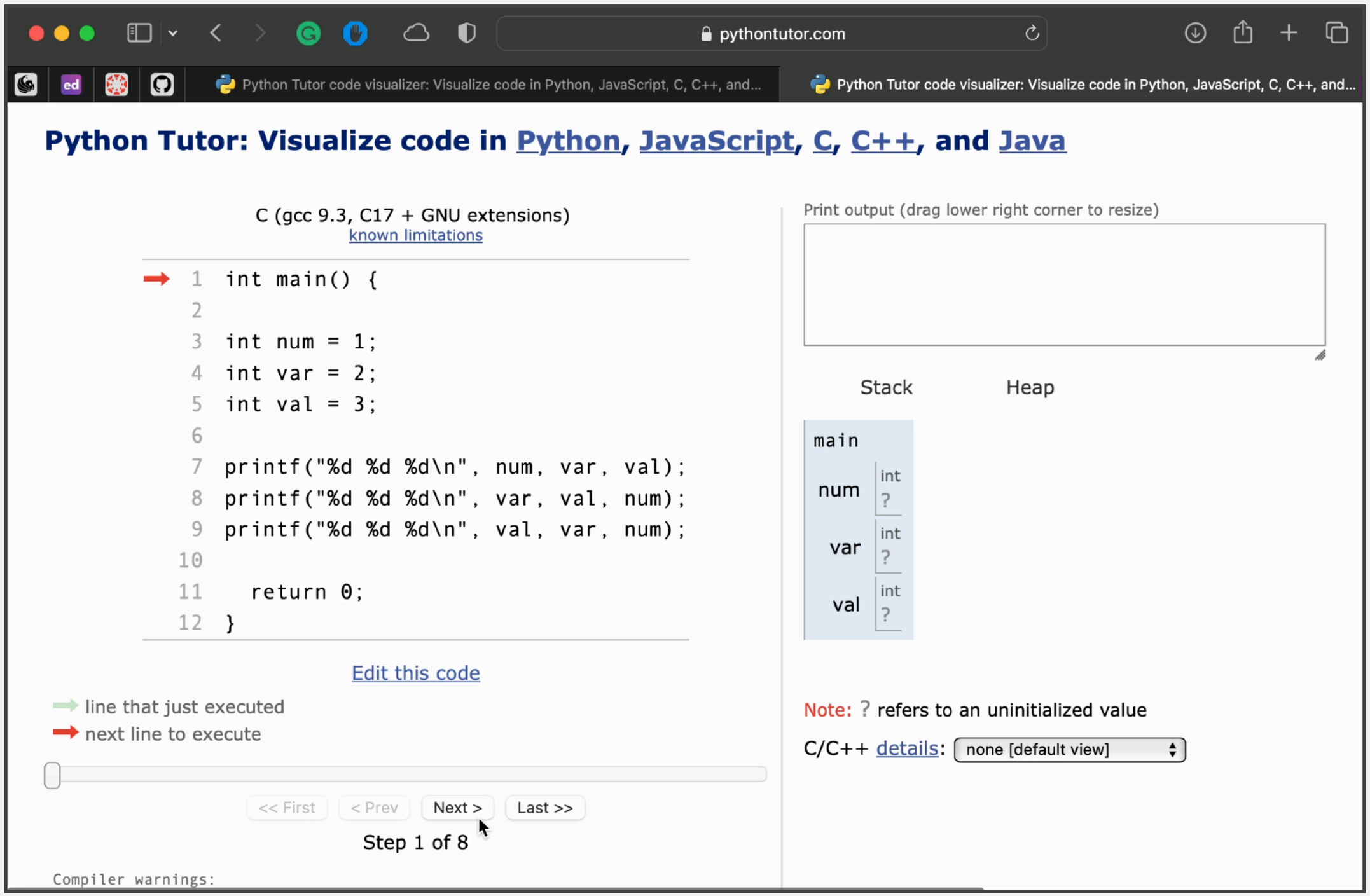
- In order to print a variable value, we must instruct the printf function on how to do this:
  - 1. Specify the format of the variable
  - 2. The variable name to print

```
printf("The final values are %d and %lf \n",var,y);
```

Printing Variables

Format Specifier	Data Type	description	Syntax
%d	int	To print the integer value	printf("%d",<int_variable>);
%f	float	To print the floating number	printf("%f",<float_variable>);
%lf	double	To print the double precision floating number or long float	printf("%lf",<double_variable>);
%c	char	To print the character value	printf("%c",<char_variable>);

Printing Variables



Printing Special Characters

Escape Sequence	Meaning
\a	Alert
\b	Backspace
\n	Newline
\t	Horizontal Tab
\v	Vertical Tab
\\	Backslash
\’	Single Quote
\”	Double Quote
\?	Question Mark
%%	Percent Symbol

Accepting User Input with `scanf()`

- Copies data into a variable stored in memory
- Collects user input through the keyboard and stores the value into the respective address of the variable in memory

```
scanf("%d", &var);
```

function name

placeholder with data type delimiter

reference to memory address of the `var` variable

Example `scanf()`

```c
// Header file for input output functions
#include <stdio.h>

// main function -
// where the execution of program begins
int main()
{

int num;
int var;
int val;
printf("Enter 3 values");

scanf("%d", &num);
scanf("%d", &var);
scanf("%d", &val);

printf("%d, %d, %d", num, var, val);

    return 0;
}
```

Example `scanf()`

```c
// Header file for input output functions
#include <stdio.h>

// main function -
// where the execution of program begins
int main()
{

int num;
int var;
int val;
printf("Enter 3 values");

scanf(“%d%d%d”, &num, &var, &val);

printf("%d, %d, %d", num, var, val);

    return 0;
}
```

Accepting User Input with `scanf()`

- When collecting any type of information from the user, there are 2 important details to consider.

    1. **Data Type:** Each data type (int, double , float, and char) has different size requirements. The placeholder specification allows C to properly store the specific value in memory.

    2. **Memory Location (Address):** Each data value needs to be properly stored somewhere in memory. The scanf function requires the programmer to specify the EXACT location in memory of where to store the value.

Return Statements

- Return terminates the function and transfers control from a function back to the activator of the function. For the main function, the control is transferred back to the operating system.
- A value is sent back to the operating system.
  - 0 means code executed successfully
  - 1 means code executed with run time error (code crash).

```
return 0; // function terminator
```

Constant Macro

- A name that is replaced by a particular constant value before program is sent to compiler
- Always seen at the top of a program file.
- Syntax:

```
#define MILES PER KM 0.62137
```

Macro Constants and Constant Variables

- You are probably thinking that macro constants and constant variables are the same.
- THEY ARE NOT!!!!!
- Constant Variables
  - The keyword const is handled by the compiler
- Macro Constant
  - The macro is handled by the preprocessor directive. It replaces the text in the C source file.

Eustis Demo

Acknowledgements

Slides adapted from Dr. Andrew Steinberg's COP 3223H course