Intro
Intended audience: Student who has working knowledge of Python

Target compiler: I’ll try to center the discussion on C99 using gcc 7

- Code examples might have an accompanying link
  - Follow link to step through example at pythontutor.com
  - Does a nice job of graphically showing variables in memory, the heap, and the stack
Functions
Functions

- Can *not* be defined inside another function (must be global)¹
- Return type part of function declaration
  - Can be `void`
  - `void` is a place-holder type
  - As a return type, it indicates that the function doesn’t return a value
- Arguments are passed *by value*²
- Must be declared before use
  - A *prototype* serves
- May not be overloaded
- C supports recursion

¹GCC will allow it, but, it’s not C
²Except for arrays
```c
#include <string.h>
#include <stdio.h>

/* prototype */
int foo( int k ) ;

int main( void ) {
    int j = 12, k = 13, r ;
    r = foo( j ) ;
    printf( "foo returned %d\n", r ) ;
    if( j != 12 )
        printf( "j changed\n" ) ;
    if( k != 13 )
        printf( "k changed\n" ) ;
    return 0 ;
}

int foo( int k )
{
    /* j is local, and 
    k behaves so */
    int j = 23 ;
    int rv = k * j ;
    k = 928357 ;
    return rv ;
}
```

• [https://goo.gl/v4rZgR](https://goo.gl/v4rZgR)
Function – Separate Files

main.c

```c
#include <string.h>
#include <stdio.h>
#include "foo.h"

int main( void ) {
    int j = 12,
        k = 13,
        r;
    r = foo( j ) ;
    printf( "foo returned %d\n", r ) ;
    return 0 ;
}
```

foo.h

```c
#ifndef __MY_FOO_H_
#define __MY_FOO_H_
int foo( int k ) ;
#endif /* __MY_FOO_H_ */
```

foo.c

```c
#include "foo.h"

int foo( int k )
{
    int j = 23 ;
    int rv = k * j ;
    k = 928357 ;
    return rv ;
}
```

```
$ gcc main.c foo.c -o foo
```

Kurt Schmidt (Skipjack Solutions)  C Functions  November 1, 2021  7/12
Scoping
Scoping in C

- C uses *lexical scoping* (or *static binding*)
- A symbol (variable name) is bound to a location at compile time
- A variable is either
  - Local to the function it appears in (if it’s declared there, or is a formal parameter), or
  - Global (declared outside of all functions, possibly in another file)
- Pretty straightforward in C, since we can’t define functions inside of other functions
Array Arguments
Passing Arrays to Functions

- Arrays are passed by reference
  - The name of the array is a pointer
  - The pointer is copied in
  - Semantics are preserved
- Array can be modified by the function
- Size (number of elements) must also be passed to function
  - Unless a suitable sentinel value exists (e.g., ‘\0’)

Kurt Schmidt  (Skipjack Solutions)
```c
#include <stdio.h>

void arrPrint( float a[], int n, FILE* ofile )
{
    for( int i=0; i<n; ++i )
        fprintf( ofile, "%f ", a[i] ) ; /* access by index */
}

int main()
{
    int a[20] = { 5, 12, 13, 23, 42, 67 } ;

    arrPrint( a, 6, stdout ) ;

    return 0 ;
}
```