As you arrive:

1. Start up your computer and plug it in
2. ***Log into Angel*** and go to CSSE 120
3. Do the ***Attendance Widget*** – the PIN is on the board
4. Go to the course ***Schedule Page***
5. Open the ***Slides*** for today if you wish
6. Check out today's project: `Session25_Pointers`

Plus in-class time working on these concepts AND practicing previous concepts, continued as homework.

Pointers

- What they are. Why they are useful.
- Their notation in C: & * *
- Using pointers to get data back from a function. scanf as example.
- Next time: Using pointers to send a reference to lots of data to a function

Outline

- Pointers
  - What they are.
    Why they are useful.
  - Their notation in C
    * & *
  - Pointers vs Pointee's – deferencing
  - Using pointers to get data back from a function
    - *scanf* as an example
  - Next time: Using pointers to send a reference to lots of data to a function

Parameter Passing in Python

- In Python, parameters are passed two ways:
  - For numbers, a copy of the number is passed to the function
  - For mutable objects (like lists), a **reference** to the object is passed to the function

```
def swapInts(x,  y):
    x,y = y,x
```

2 5

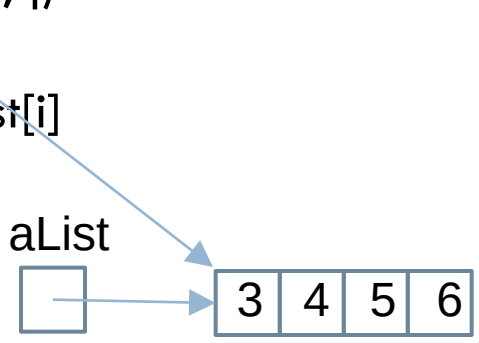
```
x,y = 2, 5
swapInts (x, y)
```

x y

2 5

```
def swapListElements(alist, i, j):
    alist[i], alist[j] = alist[j], alist[i]

alist = [3, 4, 5, 6]
swapListElements(alist, 1, 3)
```



Variables in C

- Variables are stored in memory
  - We call the place in memory the variable's *address*

```
int num;
```

memory:

num:

			???

```
num = 10;
```

memory:

num:

			10

- C has several types of variables:
  - Integers – their bits are interpreted as a whole number
  - Doubles – their bits are interpreted as a floating point number
  - …
  - ***Pointers*** – their bits are interpreted as an ***address in memory***
    - As such, they are *references* to other data

Q1

Visualizing pointers

pNum is a ***pointer*** to an **int**

pNum is set to the ***address*** of **num**

The ***thing pNum points to*** is set to **99**

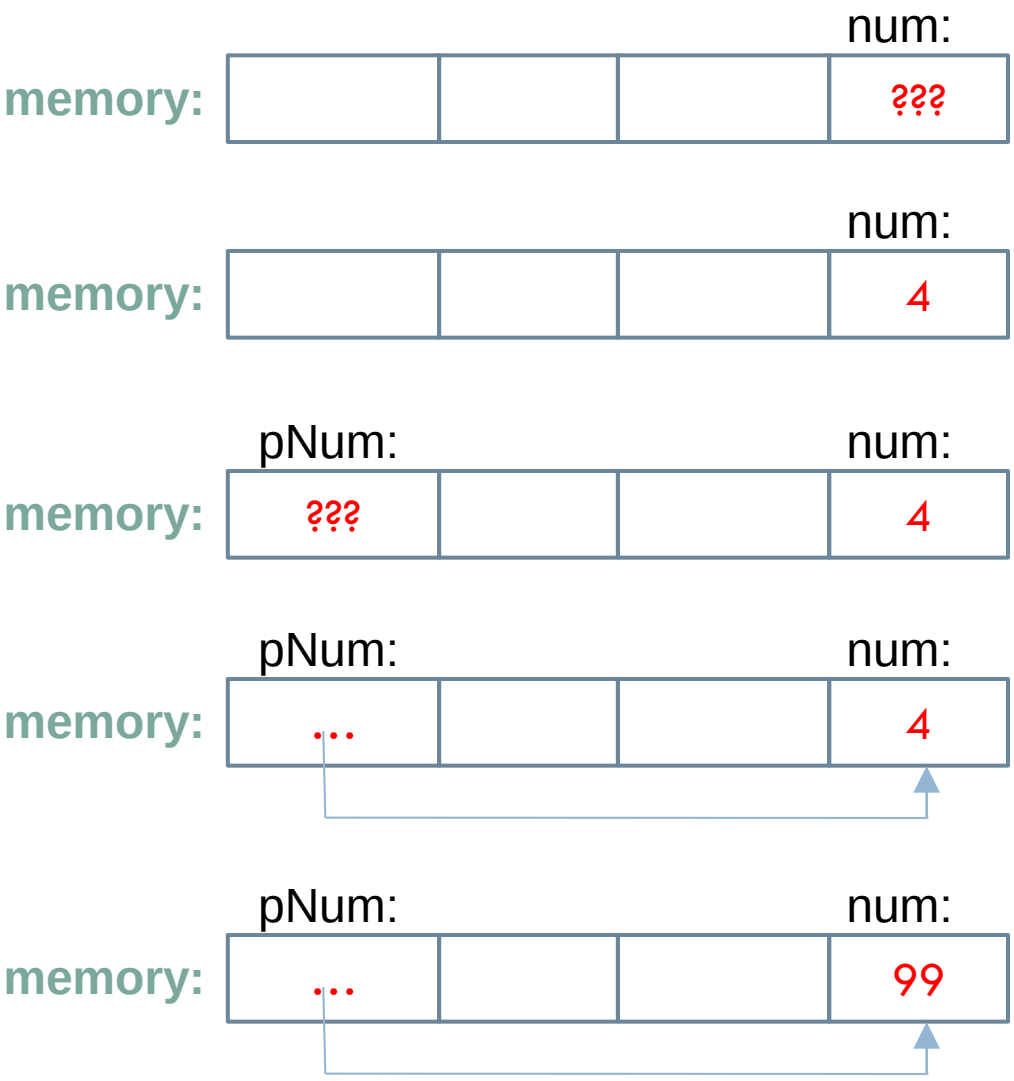
```
int num;

num = 4;

int *pNum;

pNum = &num;

*pNum = 99;
```



pNum is the *pointer* and **num** is the *pointee*.

***pNum** *deferences* the pointer, which means that it obtains the pointee.

Q2-5

Here's Binky!

- Ignore *malloc* in the video for now
- Vocabulary
  - *Pointee*: the thing referenced by a pointer
  - *Dereference*: obtain the pointee
- See http://cslibrary.stanford.edu/104/
- What name did we give pointer "sharing" in Python?
  - Answer: *aliasing*

Proof that pointers store addresses

- Checkout today's exercise:
  Session25_Pointers

- Run it in the debugger
  - The console is a separate window
  - It automatically inserts a breakpoint at the start of main()

- Let's start quiz questions 6-8 together

Q6-8

Box-and-pointer diagrams

- Together, let's draw Box-and-Pointer Diagrams for the variables in **simplePointers()**.
  - Such diagrams help you understand pointers and are critical for tracing-pointers-by-hand problems.
- Let us follow **change** and **pChange** through the execution of the function.

Box and Pointer diagrams

pChange is a ***pointer*** to a **double**

pChange is set to the ***address*** of **change**

The ***thing pChange points to*** is set to **0.62**

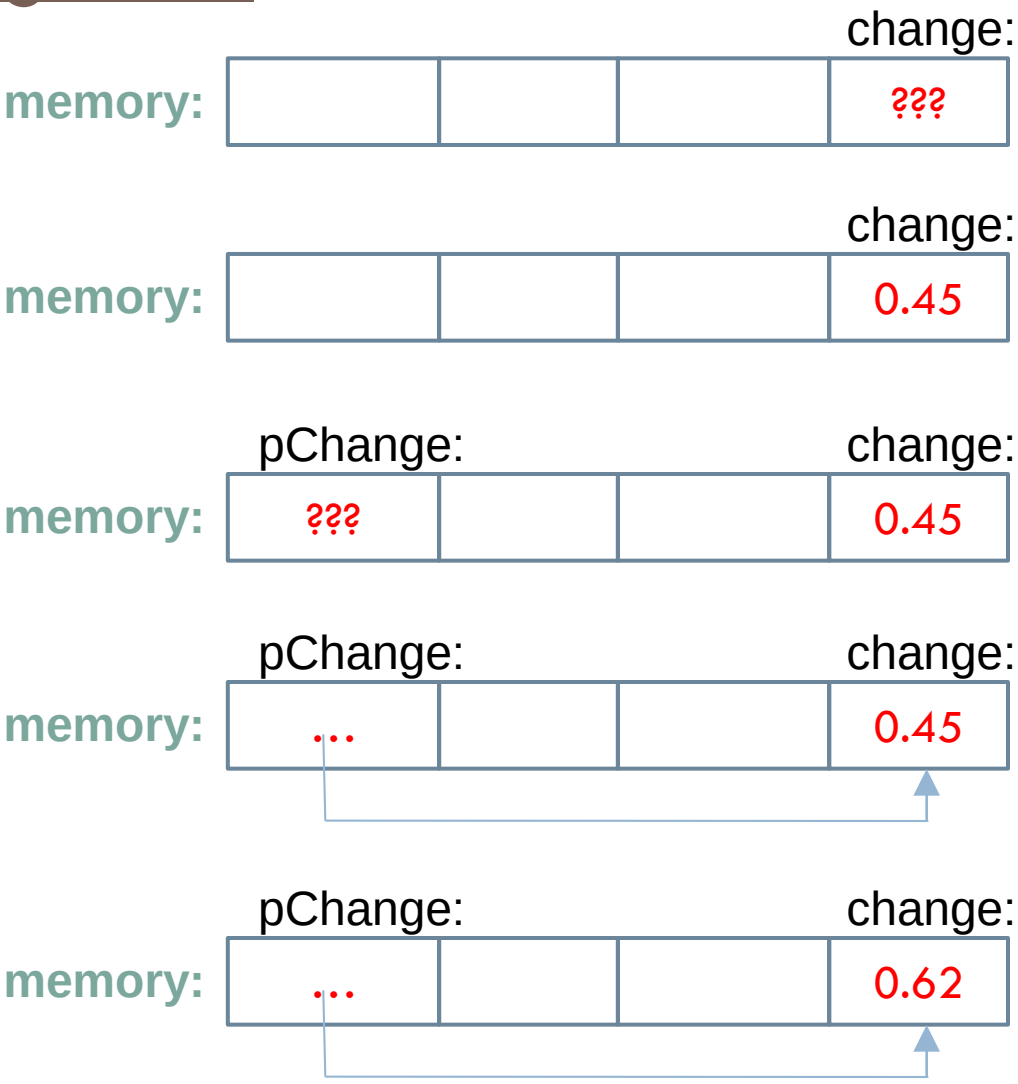
```
double change;

change = 0.45;

double *pchange;

pChange = &change;

*pChange = 0.62;
```



pChange is the *pointer* and **change** is the *pointee*.
***pChange** *deferences* the pointer, which means that it obtains the pointee.

Using pointers with functions

- We claimed earlier that if we passed a variable's reference as a parameter to a function, the function could change that variable.
- Reminder:
  - To get an address, use &
  - To get a variable referenced by a pointer, use *
  - To declare a pointer variable, use *

upAndDown, wrong version

- Do box-and-pointer diagrams to illustrates why this does NOT change the arguments.

```
//This function attempts to modify its parameters
void upAndDown(int takeMeHigher,
               int putMeDown) {
    takeMeHigher = takeMeHigher + 1;
    putMeDown = putMeDown - 1;
}
```

Using pointers as parameters
Box and Pointer Diagrams

```
int b;
foo(&b);

void foo(int *a) {
    ...
    *a = 7;
}
```

Send the address of b

Receive an address

Modify value at address

Now `b` has the value 7 that was established in `foo`!

This is useful for:

- sending data back from a function via the parameters, and for
- passing large amounts of data to a function.

Thus pointers in C give us the same advantages as references-to-objects in Python.

upAndDown, A version that works

- Change the function and how it's called so that it works!
- When you are done, please answer the quiz question.
- Modifying the previous box-and-pointer diagrams to show what is happening.

Q9

Practice with Pointers

```
1.     int x = 3, y = 5;
2.     int *px = &x;
3.     int *py = &y;
4.     printf("%d %d\n", x, y);
5.     *px = 10;
6.     printf("%d %d\n", x, y);  /* x is changed  */
7.     px = py;
8.     printf("%d %d\n", x, y);  /* x not changed */
9.     *px = 12;
10.    printf("%d %d\n", x, y);  /* y is changed  */
```

Pointer Pitfalls

- Don't try to dereference an unassigned pointer:
  - ```
    int *p;
    *p = 5;
    ```
  - ```
    /* immediate crash! */
    ```
- Pointer variables must be assigned *address* values.
  - ```
    int x = 3;
    int *p;
    p = x;
    ```
  - ```
    /* eventual crash */
    ```
- Be careful how you increment
  - ```
    *p += 1;      /* is not the same as … */
    ```
  - ```
    *p++;
    ```

In-class exercise on pointer pitfalls

- Turn in part 1 of the quiz.
- The rest of today's quiz lets you see some pointer pitfalls in action. These make great exam questions!
  - Do it now
- When you are done, start the homework:
  - A **written portion** (box and pointer diagrams)
  - More pointer output
  - Writing functions to change variables
    - doubleMe
    - Swap
    - minAndMax
  - scanf revisited

Part 2 Q1-4

Rest of today

- Work through the TODO's, as numbered.
  - We'll do the first few together
- Ask questions as needed!
  - Don't merely make the code "work". Make sure you understand the C notation and how to use it.
- If you:
  - don't finish in class, then ***finish the exercises for homework***
    - ***Get help from the assistants in F-217 in the evenings as needed!***