C Programming
A Historical Perspective

Dr. Charles R. Severance

www.cc4e.com

code.cc4e.com (sample code)

online.dr-chuck.com

Learning Path: online.dr-chuck.com

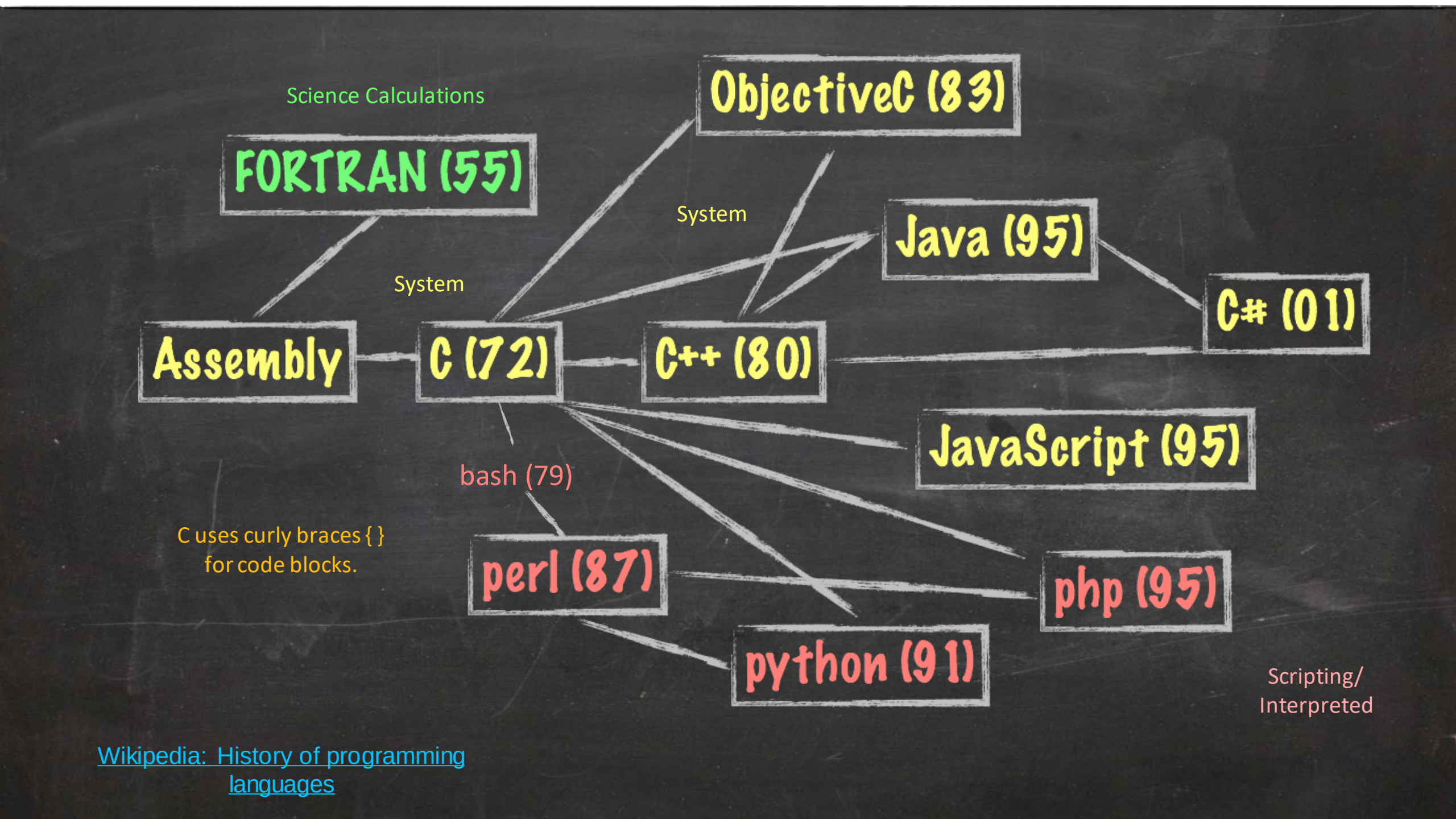
- Internet History, Technology, and Security – iht s.pr4e.com
- Python for Everybody – www.py4e.com
- Django (Python, HTML, CSS, SQL, JavaScript) – www.dj4e.com
- Web Applications (PHP, HTML, CSS, SQL, JavaScript) – www.wa4e.com
- PostgreSQL (SQL) – www.pg4e.com
- C Programming – www.cc4e.com ← We are here ☺
- Computer Architecture
- Java Enterprise Application Development

History of C

- 1969 – B Language – Word oriented (i.e. not byte oriented)
- 1972 – C Multiple types including (byte / character)
- 1972 – 1978 – C and UNIX co-evolved with a goal of increasingly less assembly language in UNIX
- **1978 – K&R C**
- 1989 – C89 / ANSI – void type, C++ declarations, character sets, locales
- 1990 - C90 / ISO C
- 1999 – C99 – complex type, // comments, Unicode
- 2011 – C11 – Library improvements
- 2018 – C17 - Cleanup of C11

Modern C / Future of C / post-C

- Challenges to use C as general purpose languages
  - No dynamic memory support in the core types / libraries
  - No “safe” string type
- C++ is best thought of as a more powerful and flexible C for professional programmers and systems applications
- Java / JavaScript / C# / Python – Types are usually objects – not ”close to the metal” – Not as well suited for an operating system Kernel
- The likely follow on to C in systems applications is Rust
  - Stays close to the metal while providing simple and safe core data types
  - Becoming the second official language in “Linux”



Wikipedia: History of programming languages

A Brief History of Computers

- 1940's – Top Secret / Military / WWII (ihts.dr-chuck.com)
- Early 1950's – Custom built
- Late 1950's – Companies like IBM, DEC, etc. begin selling computers
- 1960's – More companies, less expensive, wider range of options
- Late 1960's – Many kinds of computers old/new/fast/slow
- **1970's – Searching for "the one" solution for software**
- 1980's – Microprocessors and Personal Computers – performance++
- 1990's – The network is the computer – performance++
- 2000's – Amazon AWS founded in 2002 – computing as commodity

History of UNIX

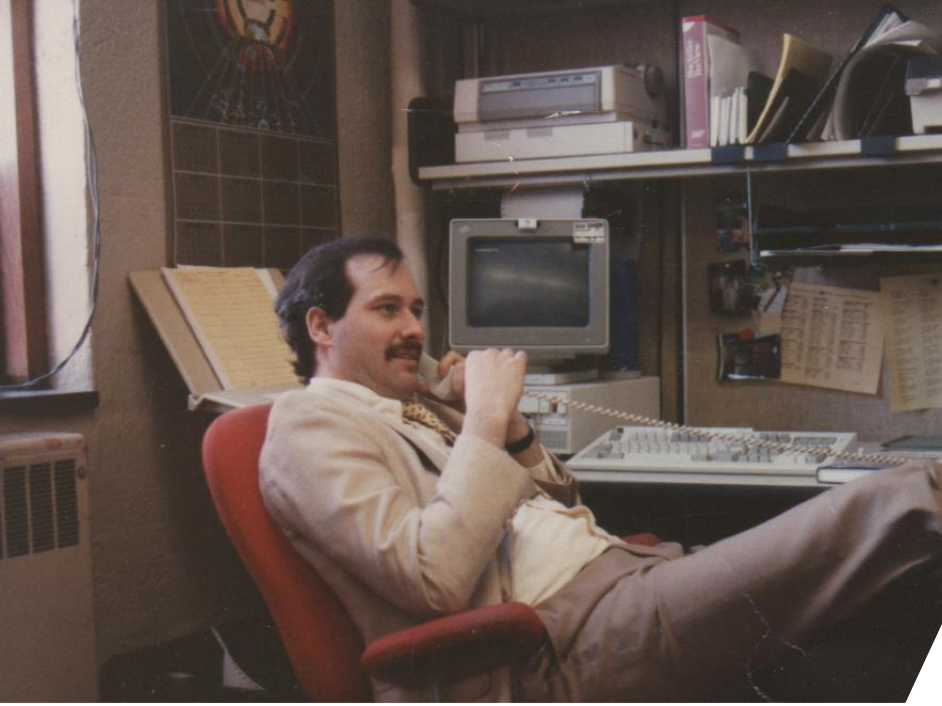
- 1960s – Multics
- 1970 – UNIX on a DEC PDP 11/20
- 1973 – UNIX Rewritten in C – Ran only on the PDP 11
- 1978 – UNIX ran on the Interdata 8/32 - C Evolved as well to support portability so UNIX could be ported
- **1978 – Unix version 7 ran on DEC VAX systems**
- **1978 – 1BSD Unix Released from Berkeley Software Distribution**
- 1982 - Sun Microsystems Founded – UNIX Workstation
- Late 1980's Intellectual Property became complex

The post-UNIX world

- Late 1980's UNIX was very popular – AT&T saw an opportunity to commercialize their work. Many variations of UNIX had bits and pieces taken from AT&T UNIX – it got complex quickly
- 1987 – Minux was developed as a fresh ground-up implementation by Andrew S. Tannenbaum to teach operating system concepts – it was free but modification and redistribution were restricted.
- 1991 – Linus Torvalds wanted to build a fresh ground up implementation of the "UNIX" kernel that was 100% free – some of the utility code came from the GPL-Licensed GNU project
- 1992 – Linux adopted the GPL license

A Brief History of Dr. Chuck

- **1970's – CDC 6500 / SCOPE/Hustler / FORTRAN / Pascal / Assembly**
- 1980's
  - HP21MX – Assembly
  - Burroughs B4900 / COBOL
  - Fortune 32:13 / UNIX / C
  - IBM PC / DOS / DBase / Turbo Pascal
  - IBM 360 / Assembly
  - DEC VAX / VMS / Fortran
  - AT&T 3B2 / UNIX / C / FORTRAN
- 1990's – UNIX / Sun / Ardent / Stellar / IBM RS-6000 / Convex C2400 / NeXT – C - Also TCP/IP, HTTP – Windows / MacOS
- 2000's – Linux / MacOS – Java / PHP / JavaScript
- 2010's – Linux / MacOS – Python / Java / PHP / JavaScript



SEVERANCE CHARLES R

STUDENT:

NAME, STUDENT NUMBER AND LOCAL ADDRESS MUST BE COMPLETED TO INSURE PROPER CREDIT. PLEASE COMPLETE IF BLANK.

TERM/YR: SP89

FEE DIVISION: 3

CREDITS: 1

CLASS: 6

STUDENT NUMBER

LAST NAME FIRST NAME MIDDLE I

4352 DONCASTER
HOLT, MI

LOCAL ADDRESS

FOR REFERENCE CODES
SEE BACK OF CARD

AMT PER CR

FEE PAYMENT CARD

MICHIGAN STATE UNIVERSITY

REGISTRAR 47

CONTROLLER 23

VISA

ITEMIZATION OF CHECKS

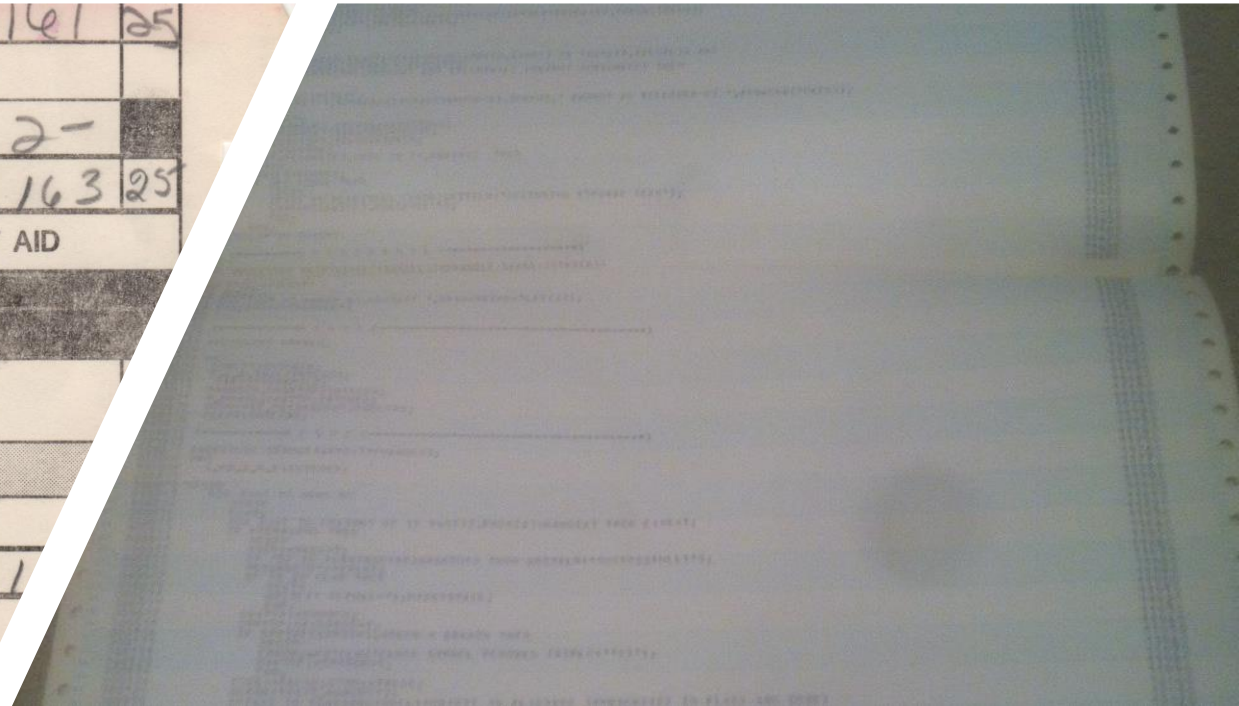
FEES	161	25
UNIVERSITY HOUSING		
LATE FEE	2	-
TOTAL	163	25
LESS STUDENT AID		
LUMP SUM		
SUB TOTAL		
HOUSING DEFERRED		
PAY THIS AMOUNT	1	

002 MSU 4/04/89#3470 $163.25FE

MSU 101 REV. 5/85

PRESERVE THIS STUDENT RECEIPT

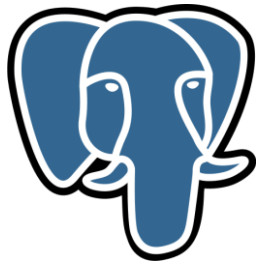
MUST BE MACHINE RECEIPTED ABOVE TO BE V

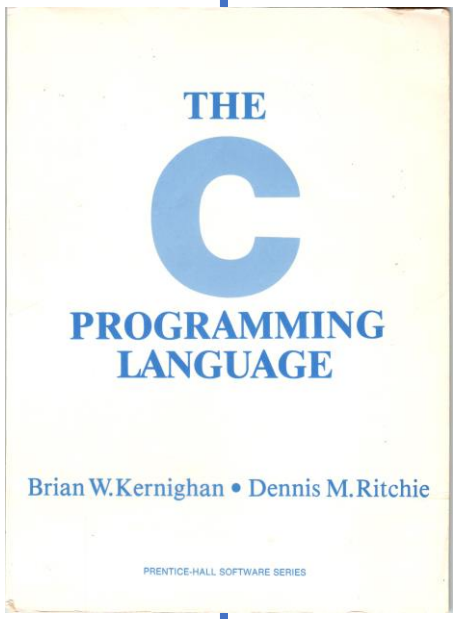


C Programming for Everybody

- C is the most important programming language you will ever learn
- C should not be the first programming language we teach to students
- You might never write a “professional” line of C during your career
- Learning C at the right time in your path, is necessary for you to become a master programmer
- Be patient – do not rush – **do not** search for solutions to programming
- Each exercise is teaching you something and preparing you to learn something much more challenging later in the course

Unlocking Advanced Topics





Data Structures

Computer Architecture

Object Oriented Design

Hardware

Service Oriented Architecture

Summary

- History of the C Language
- History of Computer Hardware
- History of the UNIX operating system
- History of “Dr. Chuck” and Computing
- Gender balance in computing

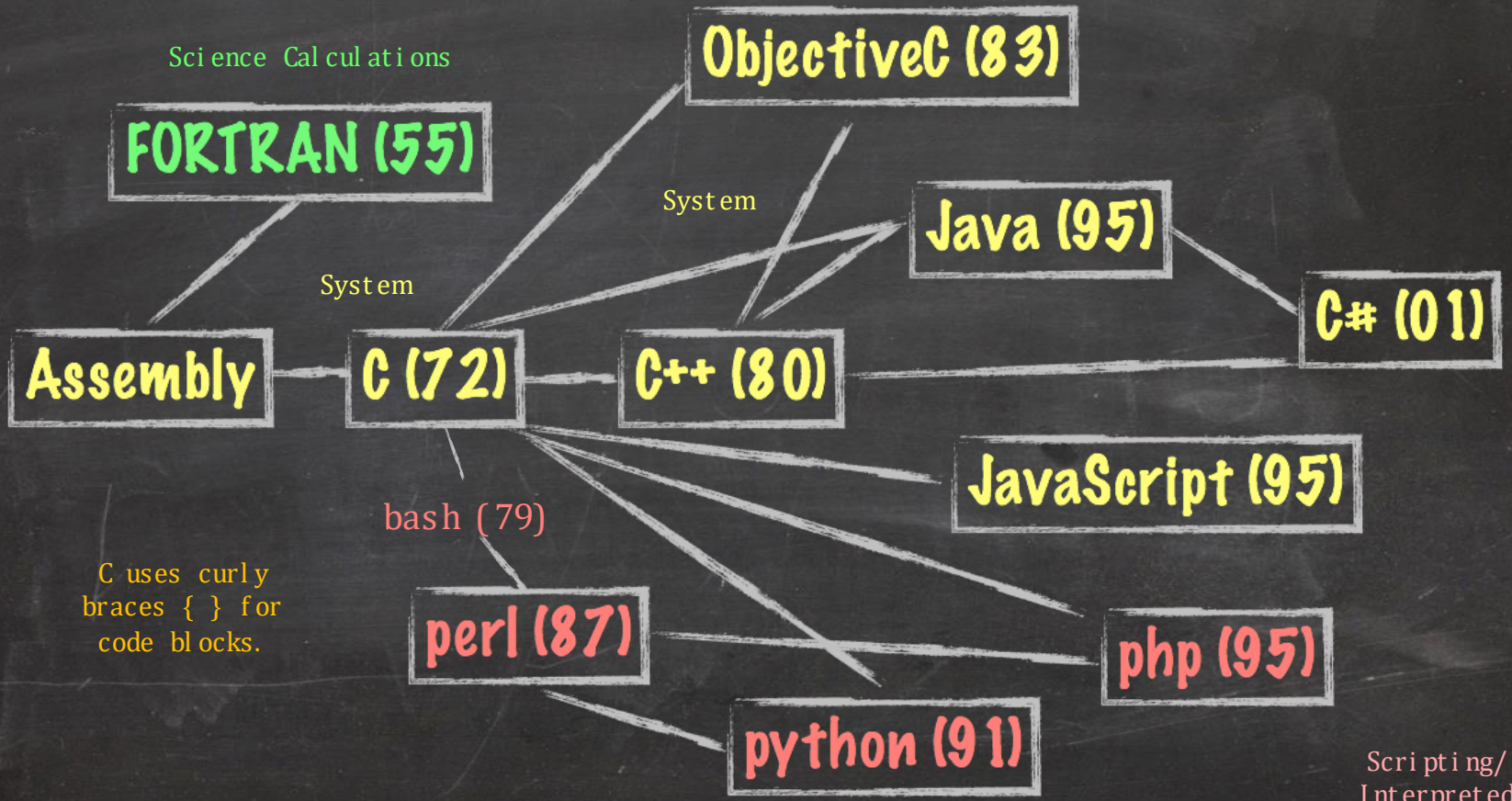
From Python to C

Dr. Charles R. Severance

www.cc4e.com

code.cc4e.com (sample code)

online.dr-chuck.com



Wikipedia: History of programming languages

Learning Path: online.dr-chuck.com

- History – ihts.pr4e.com
- Python – www.py4e.com
- Django (Python, HTML, CSS, SQL, JavaScript) – www.dj4e.com
- Web Applications (PHP, HTML, CSS, SQL, JavaScript) – www.wa4e.com
- PostgreSQL (SQL) – www.pg4e.com
- C Programming – www.cc4e.com ← We are here ☺
- Computer Architecture
- Java Enterprise Application Development

Python and C

- White space is essential
- Very object oriented
- Convenient data structures
  - list
  - dict
- Auto memory management
- 1980's

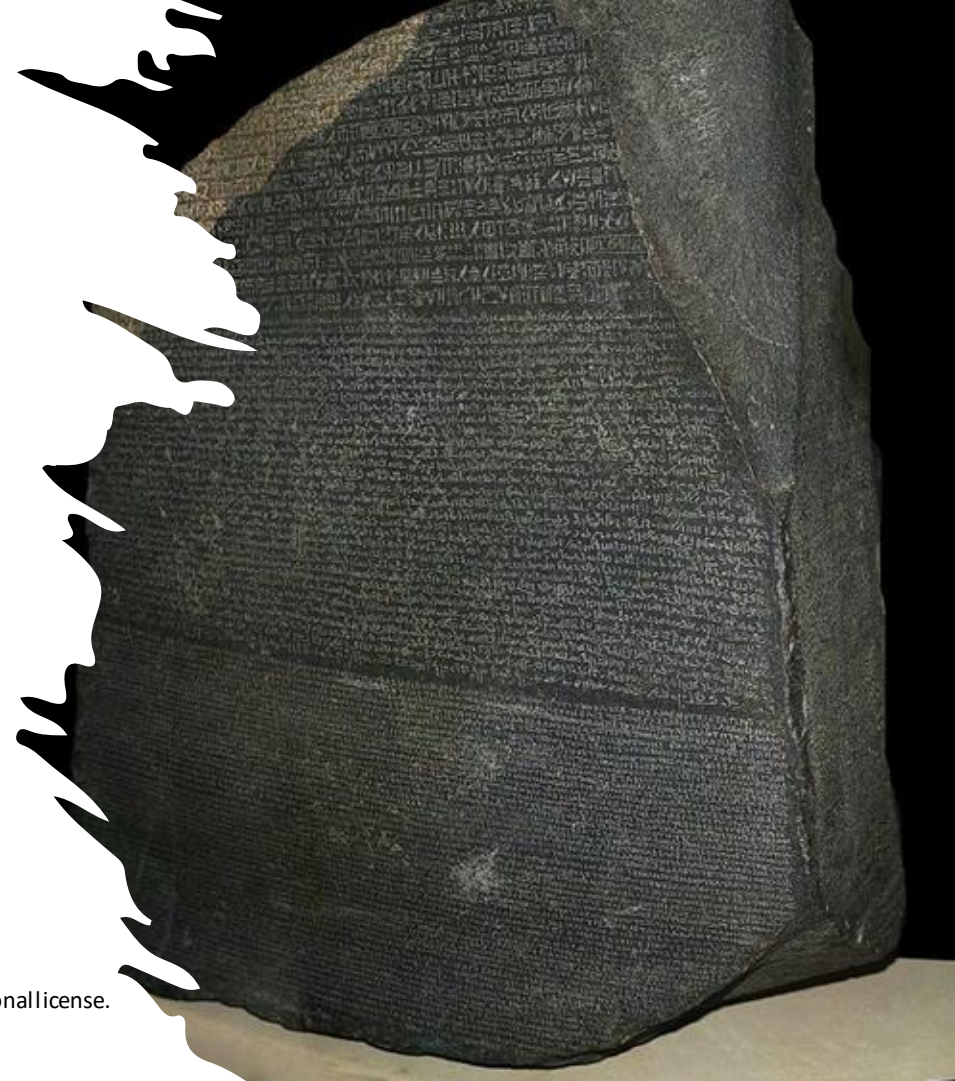
- Whitespace ignored
- Not object oriented at all
- Fast efficient powerful
  - struct
  - Pointers
- Manual memory management
- 1970's

In many ways, Python is a convenience layer built on top of C to make it so users could write code without worrying about the complex details.

C Through A Python Lens

Learning by example

Rosetta Stone



These code examples

- Most of these examples are also programming exercises
- It is my intention that you watch these lectures and work on the exercises at the same time
- These exercises are not trying to assess what you learned
- Watching and listening to the lecture and then typing this code in to make it work *is* the learning objective of this lecture
- After this section – you should do the assignments yourself (i.e. don't search) to gain maximum benefit

Similarities

- Arithmetic Operators: + - * / %
- Comparison Operators: == != < > <= the same
- Variable naming rules – letter/underscore + numbers/letters/underscores – also case matters
- While loops – also break and continue in loops
- Constants similar except for strings and characters and booleans
- Both have int / float, and char / byte
  - C has no str, list, or dict
  - Python has no struct or double

Differences

- Boolean operators
  - and / not / or versus && ! ||
- C for loops are indeterminant (i.e. no for ... in in C)
- C has no pre-defined True or False
- None and NULL are similar concepts but quite different
- Strings and character arrays are similar concepts but *very* different
- C has no list, or dict
- Python has no struct - float in Python is a C double

Output

```
# I am a comment
print('Hello world')
print('Answer', 42)
print('Name', 'Sarah')
print('x',3.5,'i',10)
```

```
#include <stdio.h>
/* I am a comment */
int main() {
    printf("Hello world\n");
    printf("Answer %d\n", 42);
    printf("Name %s\n", "Sarah");
    printf("x %.1f i %d\n", 3.5, 100);
}
```

Hello world
Answer 42
Name Sarah
x 3.5 i 100

cc_02_01.c

Number Input

```
print('Enter US
Floor')
usf = int(input())
euf = usf - 1
print('EU Floor', euf)
```

```
#include <stdio.h>
int main() {
    int usf, euf;
    printf("Enter US Floor\n");
    scanf("%d", &usf);
    euf = usf - 1;
    printf("EU Floor %d\n", euf);
}
```

Enter US Floor
2
EU Floor 1

For those of us who learned Python 2, recall the difference between **input()** and **raw_input()**. In Python 3 there is only **input()** which is the same as Python 2's **raw_input()**. In C, **scanf("%d", ...)** is more like Python 2's **input().**

cc_02_02.c

String Input

```
print('Enter name')
name = input()
print('Hello', name)
```

```
#include <stdio.h>
int main() {
    char name[100];
    printf("Enter
name\n");
    scanf("%100s", name);
    printf("Hello %s\n",
name);
}
```

Enter name
Sarah
Hello Sarah

cc_02_03.c

Line Input

```
print('Enter line')
line = input()
print('Line:', line)
```

```
#include <stdio.h>
int main() {
    char line[1000];
    printf("Enter line\n");
    scanf("%[^\n]1000s", line);
    printf("Line: %s\n", line);
}
```

Enter line
Hello world - have a nice day
Line: Hello world - have a nice day

cc_02_04.c

Line Input (safe)

```
print('Enter line')
line = input()
print('Line:', line)
```

```
#include <stdio.h>
int main() {
    char line[1000];
    printf("Enter line\n");
    fgets(line, 1000, stdin);
    printf("Line: %s\n", line);
}
```

Enter line
Hello world - have a nice day
Line: Hello world - have a nice day

cc_02_09.c

Read A File

```
hand =
open('romeo.txt')
for line in hand:
    print(line.strip())
```

```
#include <stdio.h>
int main() {
    char line[1000];
    FILE *hand;
    hand = fopen("romeo.txt", "r");
    while( fgets(line, 1000, hand) != NULL ) {
        printf("%s", line);
    }
}
```

But soft what light through yonder window breaks
It is the east and Juliet is the sun
Arise fair sun and kill the envious moon
Who is already sick and pale with grief

cc_02_06.c

Counted Loop

```
for i in range(5) :
    print(i)
```

```
0
1
2
3
4
```

```
#include <stdio.h>
int main() {
    int i;
    for(i=0; i<5; i++) {
        printf("%d\n",i);
    }
}
```

cc_02_07.c

Max / Min Python

```
maxval = None
minval = None
while True:
    line = input()
    line = line.strip()
    if line == "done" : break
    ival = int(line)
    if ( maxval is None or ival > maxval)
:
        maxval = ival
    if ( minval is None or ival < minval)
:
        minval = ival

print('Maximum', maxval)
print('Minimum', minval)
```

5
2
9
done
Maximum 9
Minimum 2

cc_02_08.py

Max / Min C

5
2
9
(EOF)
Maximum 9
Minimum 2

5 2 9
(EOF)
Maximum 9
Minimum 2

```
#include <stdio.h>
int main() {
    int first = 1;
    int val, maxval, minval;

    while(scanf("%d",&val) != EOF ) {
        if ( first || val > maxval ) maxval =
val;
        if ( first || val < minval ) minval =
val;
        first = 0;
    }

    printf("Maximum %d\n", maxval);
    printf("Minimum %d\n", minval);
}
```

cc_02_08.c

Guessing

```
while True:
    try:
        line = input()
    except:  # If we get EOF
        break
    line = line.strip()
    guess = int(line)
    if guess == 42:
        print('Nice work!')
        break
    elif guess < 42 :
        print('Too low - guess again')
    else :
        print('Too high - guess again')
```

```
#include <stdio.h>
int main() {
    int guess;
    while(scanf("%d",&guess) != EOF ) {
        if ( guess == 42 ) {
            printf("Nice work!\n");
            break;
        }
        else if ( guess < 42 )
            printf("Too low - guess again\n");
        else
            printf("Too high - guess again\n");
    }
}
```

5
Too low - guess again
50
Too high - guess again
42
Nice work!

cc_02_09.c

Functions (call by value)

```
def mymult(a,b):
    c = a * b
    return c

retval = mymult(6, 7)
print('Answer:',
retval)
```

Answer: 42

```
#include <stdio.h>
int main() {
    int mymult();
    int retval;

    retval = mymult(6,7);
    printf("Answer: %d\n",retval);
}

int mymult(a, b)
    int a,b;
{
    int c = a * b;
    return c;
}
```

cc_02_10.c

Shouting

```
hand = open('romeo.txt')
for line in hand:

print(line.strip().upper())
```

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int main() {
    char line[1000];
    FILE *hand;
    int i;
    hand = fopen("romeo.txt", "r");
    while( fgets(line, 1000, hand) != NULL
)
        for(i=0; i<strlen(line); i++)
            putchar(toupper(line[i]));
}
```

BUT SOFT WHAT LIGHT THROUGH YONDER WINDOW
BREAKS
IT IS THE EAST AND JULIET IS THE SUN
ARISE FAIR SUN AND KILL THE ENVIOUS MOON
WHO IS ALREADY SICK AND PALE WITH GRIEF

cc_02_11.c

Summary

- Input/Output
- Looping
- Reading a file
- Strings
- Float

- Way too complex to implement in C (for now)
  - Python str()
  - Python list()
  - Python dict()
- We will revisit these as the end of the course

K&R Chapter 1
A Tutorial Introduction

Dr. Charles R. Severance

www.cc4e.com

code.cc4e.com (sample code)

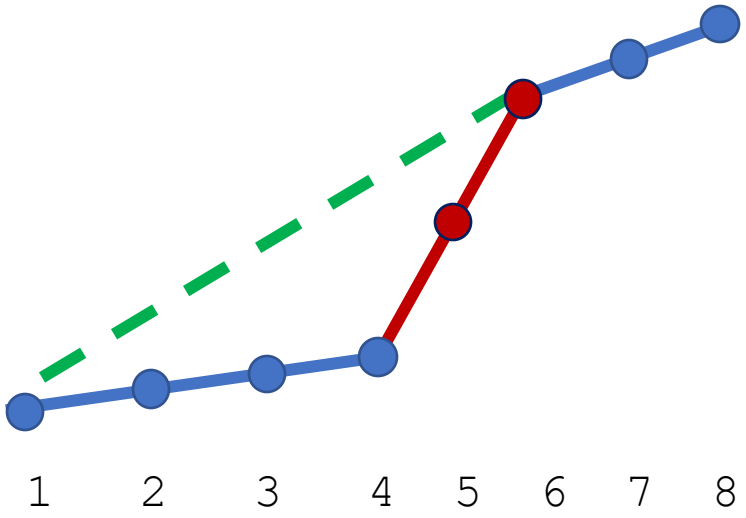
online.dr-chuck.com

Learning Path: online.dr-chuck.com

- Internet History, Technology, and Security – iths.pr4e.com
- Python – www.py4e.com
- Django (Python, HTML, CSS, SQL, JavaScript) – www.dj4e.com
- Web Applications (PHP, HTML, CSS, SQL, JavaScript) – www.wa4e.com
- PostgreSQL (SQL) – www.pg4e.com
- C Programming – www.cc4e.com ← We are here ☺
- Computer Architecture
- Java Enterprise Application Development

Outline of the book

- Chapters 1-4 – Mostly syntax
  - Just another programming language
  - Arrays, strings and the weird fact that strings are character arrays
- Chapter 5 – Pointers and Arrays
- Chapter 6 – Structures
- Chapter 7 – 8 – Detailed C Features
  - The detail we skipped



Chapter 1

- Sections 1.1 – 1.5 – “Just another programming language”
- Section 1.6 – Arrays
  - Static allocation – cannot be resized until Chapter 5 ☺
- Section 1.7 – 1.8 – Functions and Parameters
  - Call by value is simple – call by reference is in Chapter 5 ☺
- Section 1.9 – Character Arrays
  - This is important because there ***is no string object*** in C
- Section 1.10 – Variable scoping between functions

Character Arrays

- We must carefully understand the “size” of the character array and not exceed it – in C nothing is “auto-extended”

```
x = ""
for i in range(1000) :
    x += '*'
print(x)

$ python3 cc_01_01.py
*********…**********
```

```
#include <stdio.h>
int main() {
    char x[10];
    int i;

    for(i=0; i<1000; i++ ) x[i] = '*';
    printf("%s\n",x);
}

$ a.out
Segmentation fault: 11
```

kr_01_01.c

String / Character Constants

- Languages like PHP, Python, and JavaScript treat single and double quotes nearly the same. Both create *string* constants.
- In C single quotes are a character and double quotes are a character array (neither are a string)
- In C, a character is a byte – a short (typically 8-bit) integer

```
#include <stdio.h>
int main() {
    char x[3] = "Hi";
    char y[3] = { 'H', 'i' };
    printf("x %s\n", x);
    printf("y %s\n", y);
    printf("%s\n", "Hi");
    printf("%c%c\n", 'H', 'i');
}
```

```
$ a.out
x Hi
y Hi
Hi
Hi
```

kr_01_02.c

Character Sets

- The C **char** type is just a number – character representations depend on the character set.
- Modern characters including 😊 are represented in multi-byte sequences using Unicode and UTF-8 – but in 1978 we used ASCII and other character sets.

```
print('A', ord('A'))
print('😊',
ord('😊'))
```

```
$ python3 kr_01_03.py
A 65
😊 128522
```

```
#include <stdio.h>
int main()
{
    printf("%c %d\n", 'A', 'A');
}
```

```
$ a.out
A 65
```

kr_01_03.c

ASCII

- American Standard Code for Information Interchange

Dec	Hex	Oct	Bin	Char	Dec	Hex	Oct	Bin	Char	Dec	Hex	Oct	Bin	Char	Dec	Hex	Oct	Bin	Char
0	0x00	000	0000000	NUL	32	0x20	040	0100000	space	64	0x40	100	1000000	@	96	0x60	140	1100000	`
1	0x01	001	0000001	SOH	33	0x21	041	0100001	!	65	0x41	101	1000001	A	97	0x61	141	1100001	a
2	0x02	002	0000010	STX	34	0x22	042	0100010	"	66	0x42	102	1000010	B	98	0x62	142	1100010	b
3	0x03	003	0000011	ETX	35	0x23	043	0100011	#	67	0x43	103	1000011	C	99	0x63	143	1100011	c
4	0x04	004	0000100	EOT	36	0x24	044	0100100	$	68	0x44	104	1000100	D	100	0x64	144	1100100	d
5	0x05	005	0000101	ENQ	37	0x25	045	0100101	%	69	0x45	105	1000101	E	101	0x65	145	1100101	e
6	0x06	006	0000110	ACK	38	0x26	046	0100110	&	70	0x46	106	1000110	F	102	0x66	146	1100110	f
7	0x07	007	0000111	BEL	39	0x27	047	0100111	'	71	0x47	107	1000111	G	103	0x67	147	1100111	g
8	0x08	010	0001000	BS	40	0x28	050	0101000	(	72	0x48	110	1001000	H	104	0x68	150	1101000	h
9	0x09	011	0001001	TAB	41	0x29	051	0101001	)	73	0x49	111	1001001	I	105	0x69	151	1101001	i
10	0x0A	012	0001010	LF	42	0x2A	052	0101010	*	74	0x4A	112	1001010	J	106	0x6A	152	1101010	j
11	0x0B	013	0001011	VT	43	0x2B	053	0101011	+	75	0x4B	113	1001011	K	107	0x6B	153	1101011	k
12	0x0C	014	0001100	FF	44	0x2C	054	0101100	,	76	0x4C	114	1001100	L	108	0x6C	154	1101100	l
13	0x0D	015	0001101	CR	45	0x2D	055	0101101	-	77	0x4D	115	1001101	M	109	0x6D	155	1101101	m
14	0x0E	016	0001110	SO	46	0x2E	056	0101110	.	78	0x4E	116	1001110	N	110	0x6E	156	1101110	n
15	0x0F	017	0001111	SI	47	0x2F	057	0101111	/	79	0x4F	117	1001111	O	111	0x6F	157	1101111	o
16	0x10	020	0010000	DLE	48	0x30	060	0110000	0	80	0x50	120	1010000	P	112	0x70	160	1110000	p
17	0x11	021	0010001	DC1	49	0x31	061	0110001	1	81	0x51	121	1010001	Q	113	0x71	161	1110001	q
18	0x12	022	0010010	DC2	50	0x32	062	0110010	2	82	0x52	122	1010010	R	114	0x72	162	1110010	r
19	0x13	023	0010011	DC3	51	0x33	063	0110011	3	83	0x53	123	1010011	S	115	0x73	163	1110011	s
20	0x14	024	0010100	DC4	52	0x34	064	0110100	4	84	0x54	124	1010100	T	116	0x74	164	1110100	t
21	0x15	025	0010101	NAK	53	0x35	065	0110101	5	85	0x55	125	1010101	U	117	0x75	165	1110101	u
22	0x16	026	0010110	SYN	54	0x36	066	0110110	6	86	0x56	126	1010110	V	118	0x76	166	1110110	v
23	0x17	027	0010111	ETB	55	0x37	067	0110111	7	87	0x57	127	1010111	W	119	0x77	167	1110111	w
24	0x18	030	0011000	CAN	56	0x38	070	0111000	8	88	0x58	130	1011000	X	120	0x78	170	1111000	x
25	0x19	031	0011001	EM	57	0x39	071	0111001	9	89	0x59	131	1011001	Y	121	0x79	171	1111001	y
26	0x1A	032	0011010	SUB	58	0x3A	072	0111010	:	90	0x5A	132	1011010	Z	122	0x7A	172	1111010	z
27	0x1B	033	0011011	ESC	59	0x3B	073	0111011	;	91	0x5B	133	1011011	[	123	0x7B	173	1111011	{
28	0x1C	034	0011100	FS	60	0x3C	074	0111100	<	92	0x5C	134	1011100	\	124	0x7C	174	1111100	|
29	0x1D	035	0011101	GS	61	0x3D	075	0111101	=	93	0x5D	135	1011101	]	125	0x7D	175	1111101	}
30	0x1E	036	0011110	RS	62	0x3E	076	0111110	>	94	0x5E	136	1011110	^	126	0x7E	176	1111110	~
31	0x1F	037	0011111	US	63	0x3F	077	0111111	?	95	0x5F	137	1011111	_	127	0x7F	177	1111111	DEL

Wikipedia: ASCII

Image source: catonmat.net

Terminating a String

- The size of a “string” stored in a C array is not the length of the array
- C uses a special character ' \0 ' that *marks* the string end by convention
- Character arrays need to allocate an extra byte to store the line-end character

```
#include <stdio.h>
int main() {
    char x[6];
    x[0] = 'H';
    x[1] = 'e';
    x[2] = 'l';
    x[3] = 'l';
    x[4] = 'o';
    x[5] = '\0';
    printf("%s\n",x);

    x[2] = 'L';
    printf("%s\n",x);

    x[3] = '\0';
    printf("%s\n",x);
}
```

```
$ a.out
Hello
HeLlo
HeL
```

kr_01_04.c

String Length

- In C string “length” must be computed in a loop that scans for a zero character
- There the **strlen()** function in **string.h** computes string length

```
x = 'Hello'
print(x, len(x))
```

```
$ python3 kr_01_05.py
Hello 5
```

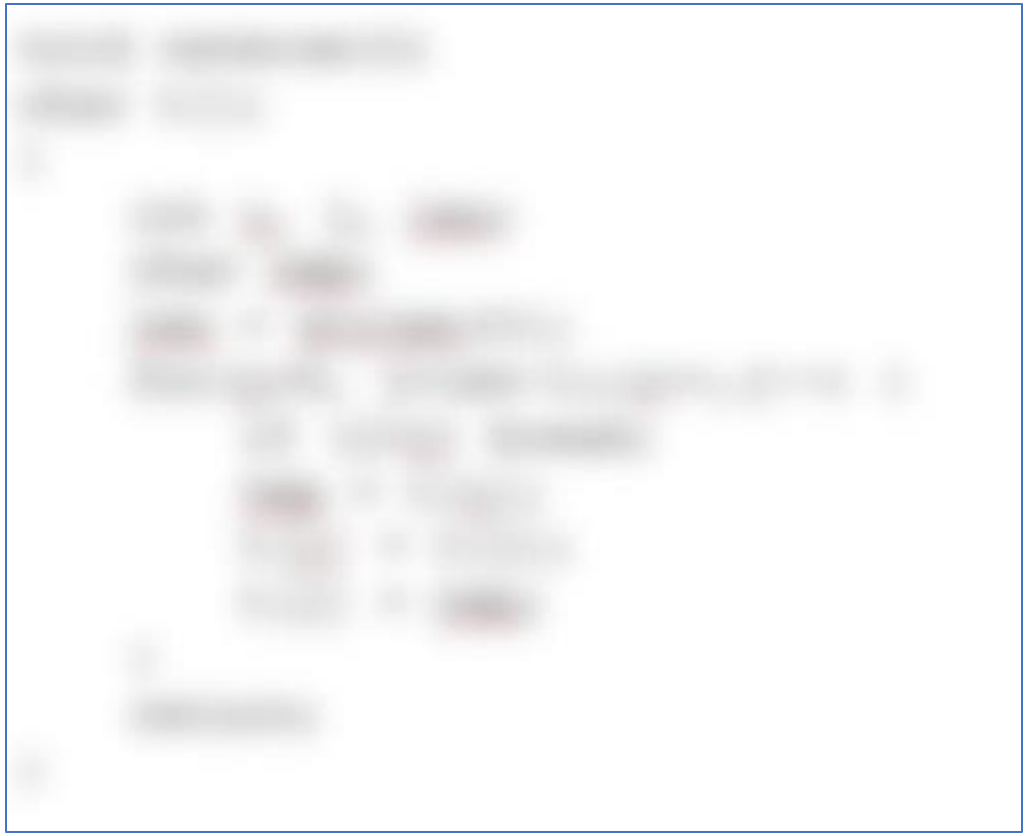
```
#include <stdio.h>
int main() {
    char x[] = "Hello";
    int py_len();
    printf("%s %d\n",x, py_len(x));
}
int py_len(self)
    char self[];
{
    int i;
    for(i=0; self[i]; i++);
    return i;
}
```

```
$ a.out
Hello 5
```

kr_01_05.c

Reverse a String in C

- A beloved / terrible coding interview question
- Exercise 1-17 in K&R
- Do *not* cheat, do not look for the answer (1000's are out there)
- Your struggle is very valuable for you
- The reversal must be done in-place
- Even length strings, odd length strings, empty strings, single character strings – think about them all



Summary

- Overview and approach for the book
- Representing ”strings” in C character arrays
- Actually doing your homework because it is good for you

Acknowledgements / Contributions

Initial Development: Charles Severance, University of Michigan School of Information

Insert new Contributors and Translators here including names and dates

Continue new Contributors and Translators here