Midterm Review
More on Strings

CS 8: Introduction to Computer Science, Winter 2018
Lecture #8

Ziad Matni
Dept. of Computer Science, UCSB

Administrative

- Homework #3 due today
- Homework #4 is due next **Wednesday (2/21)**
- Project #1 is up on the website

- Midterm is on **Wednesday!**

Lecture Outline

- About the Midterm Exam
- Character manipulation in strings
- Review questions

MIDTERM IS COMING!

- Material: ***Everything*** we've done**, incl. up to Mon. 2/12**
  - Homework, Labs, Lectures, Textbook
- **Wednesday, 2/14** in this classroom
- **Starts at 9:30pm **SHARP****
- Duration: **1 hour 15 minutes long**

- **Closed book: no calculators, no phones, no computers**
- **You will write your answers on the exam sheet itself.**
- Attention DSP students: Please follow up with DSP or you will not be accommodated!

Bring your UCSB IDs to the exam!!!

What's on the Midterm#1?
All Lecture Materials, Including…

- What is CS? What are computers? Brief history
- What is programming? How does abstraction fit in?
- Numbers and Arithmetic in Python
- Variables in Python
- Modules in Python including **`turtle`**
- Loops using **`for`**
  - Different uses of **`range`**
  - Implementing accumulations
- Conditional statements using **`if/elif/else`**
- Boolean Logic
- Random Number Generation
- Functions – how to define them, how to call them
- Strings in Python

What's on the Midterm#1?
Textbook Readings

- Ch. 1 (all)
  - Intro to Python
- Ch. 2 (all)
  - Finding Pi:
    a context to learn/use loops, functions, random numbers
- Ch. 3 (sections 3.1 and 3.2)
  - Strings and their manipulations

What's on the Midterm#1?
Homework and Labs

- Review them and understand what you did
  - The lab processes and experiences, especially

What Will it Look Like?

- Multiple Choice
- Fill in the Blanks
- Write code

- We will do some questions from the review sheet later today

Functions `chr(n)` and `ord(c)`

- **Characters** are stored as **numbers** in computer memory
  - There are standard codes for characters, e.g. **ASCII**, **UTF-8**, etc…
- For example, `'A'` has code `65` in ASCII
  - Use the `ord` function to verify this: `ord('A')` is `65`
  - Notice `'A'` is not same as `'a'`: `ord('a')` is `97`
- Every character, **seen** (e.g. %, !, G, =, space, tab,…) and **unseen** (e.g. CONTROL-X, newline…) has an ASCII code

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	*[NULL]*	32	20	*[SPACE]*	64	40	@	96	60	`
1	1	*[START OF HEADING]*	33	21	!	65	41	A	97	61	a
2	2	*[START OF TEXT]*	34	22	"	66	42	B	98	62	b
3	3	*[END OF TEXT]*	35	23	#	67	43	C	99	63	c
4	4	*[END OF TRANSMISSION]*	36	24	$	68	44	D	100	64	d
5	5	*[ENQUIRY]*	37	25	%	69	45	E	101	65	e
6	6	*[ACKNOWLEDGE]*	38	26	&	70	46	F	102	66	f
7	7	*[BELL]*	39	27	'	71	47	G	103	67	g
8	8	*[BACKSPACE]*	40	28	(	72	48	H	104	68	h
9	9	*[HORIZONTAL TAB]*	41	29	)	73	49	I	105	69	i
10	A	*[LINE FEED]*	42	2A	*	74	4A	J	106	6A	j
11	B	*[VERTICAL TAB]*	43	2B	+	75	4B	K	107	6B	k
12	C	*[FORM FEED]*	44	2C	,	76	4C	L	108	6C	l
13	D	*[CARRIAGE RETURN]*	45	2D	-	77	4D	M	109	6D	m
14	E	*[SHIFT OUT]*	46	2E	.	78	4E	N	110	6E	n
15	F	*[SHIFT IN]*	47	2F	/	79	4F	O	111	6F	o
16	10	*[DATA LINK ESCAPE]*	48	30	0	80	50	P	112	70	p
17	11	*[DEVICE CONTROL 1]*	49	31	1	81	51	Q	113	71	q
18	12	*[DEVICE CONTROL 2]*	50	32	2	82	52	R	114	72	r
19	13	*[DEVICE CONTROL 3]*	51	33	3	83	53	S	115	73	s
20	14	*[DEVICE CONTROL 4]*	52	34	4	84	54	T	116	74	t
21	15	*[NEGATIVE ACKNOWLEDGE]*	53	35	5	85	55	U	117	75	u
22	16	*[SYNCHRONOUS IDLE]*	54	36	6	86	56	V	118	76	v
23	17	*[ENG OF TRANS. BLOCK]*	55	37	7	87	57	W	119	77	w
24	18	*[CANCEL]*	56	38	8	88	58	X	120	78	x
25	19	*[END OF MEDIUM]*	57	39	9	89	59	Y	121	79	y
26	1A	*[SUBSTITUTE]*	58	3A	:	90	5A	Z	122	7A	z
27	1B	*[ESCAPE]*	59	3B	;	91	5B	[	123	7B	{
28	1C	*[FILE SEPARATOR]*	60	3C	<	92	5C	\	124	7C	|
29	1D	*[GROUP SEPARATOR]*	61	3D	=	93	5D	]	125	7D	}
30	1E	*[RECORD SEPARATOR]*	62	3E	>	94	5E	^	126	7E	~
31	1F	*[UNIT SEPARATOR]*	63	3F	?	95	5F	_	127	7F	*[DEL]*

Functions `chr(n)` and `ord(c)`

- Likewise, you can find character associated with a particular code using `chr` function, for example:

  `chr(65)` is `'A'`

- You can manipulate numbers in order to process characters

  `chr( ord('a') + 3)` is `chr(97)`, which is `'d'`

- Notice **digit *characters*** have codes too!

  `ord('6')` is `54`

Examples

- How can I find out what's 13 letters *after* '**e**'??
  - Easy answer: recite the alphabet from 'e' and count 13 places
  - Code answer: `chr( ord('e') + 13 )`, which is `'r'`
- How can I find out what's 19 letters *before* '**Z**'??
  - Code answer: `chr( ord('Z') - 19)`, which is `'G'`
- What's the ASCII code for the hashtag character??
  - Code answer: `ord('#')`, which is `35`

Harder Example…

- How can I do a "add" of 2 numeral characters to get another numerical character, like **'3'** and **'4'** and get **'7'**??

- First ask: how can I make '3' into 3? (***HINT***: We'll need a baseline…)
- That baseline is ord('0') --- how far away in the ASCII is '3' from '0'???
- ord('3') – **ord('0')** = 3
- So the "addition" is done like this:

ord('3') – ord('0') + ord('4') – ord('0') = 7

or, ord('3') + ord('4') – 2*ord('0') = 7

Then, to switch the answer from a number (7) to a character ('7'):

chr(ord('3') + ord('4') – 2*ord('0') + ord('0')) = '7'

So I Can Create a Function to do This!

```
def addChars(char1, char2):
    numAddASCII = ord(char1) +  ord(char2) – ord(‘0’)
    charNum = chr(numAddASCII)
    return charNum
```

Important Caveat!

Only works with 1 character numbers!

`str()`, `int()`, and `float()` Functions

- You can translate from one data type into another
- To change a variable from int or float into a string type, use the **str()** function
  - Example: **`a = 9.44`**, means that **`str(a) = '9.44'`**
- To change a variable from string, into an integer type use the **int()** function (or use the **float()** function if you need a float type)
  - Example1: **`s = '102'`**, means that **`int(s) = 102`**
  - Example2: **`s = '10.2'`**, means that **`float(s) = 102.0`**
- Why would we want to do this?

What's the Difference Between `return` and `print` ???

`print`

- Can go inside functions or outside of them
- Sends whatever's between the () to the "standard output"
  - **i.e. prints to your computer display, usually**

`return`

- Only belongs/used in functions
- Sends whatever's between the () back to whatever "*called*" the function
  - **i.e. no printing of any sort involved!**

What's the Difference Between `return` and `print` ???

`print`

Example:

```
def fun(a):
     b = 2*a + 1
     print(b)
     return(b)
…
x = fun(5)
# Call fun() with an argument of 5
y = x + 4
```

This will call fun() with 5, which prints the number 11 and also returns the number 11 to variable **x** in the main program, so that variable **y** can become the number 15.

`return`

Example:

```
def fun(a):
     b = 2*a + 1
     return(b)
…
x = fun(5)
# Call fun() with an argument of 5
y = x + 4
print(x)
```

This will call fun() with 5, which **ONLY** returns the number 11 to variable **x** in the main program, so that variable **y** can become the number 15. The main program also prints out the number 11.

Let's Go Over Some of the Review Questions!

YOUR TO-DOs

- Do **Homework4** (due **Wednesday 2/21**)
- No Lab this Week!
- Study for your **Midterm Exam!**

- Embrace randomness

</LECTURE>