Strings in Python 1
Midterm#1 Exam Review

CS 8: Introduction to Computer Science
Lecture #6

Ziad Matni
Dept. of Computer Science, UCSB

Administrative

- Turn in Homework #2 today
- Homework #3 is assigned and due next Thursday
- Lab #2 is due on Friday

- Your grades are now online!

  Access them through the class website
  and click on "**Class Grades, CMPSC 8, Spring 2017**"

- **Don't forget your TAs' and Instructor's office hours!! ☺**

MIDTERM IS COMING!

- Material: ***Everything*** we've done**, incl. up to Th. 4/20**
  - Homework, Labs, Lectures, Textbook
- **Tuesday, 4/25** in this classroom
- **Starts at 3:30pm **SHARP****
- **Pre-assigned seating**
- Duration: **1 hour long**
- Closed book: no calculators, no phones, no computers
- Only 1 sheet (single-sided) of written notes
  - Must be no bigger than 8.5" x 11"
  - **You have to turn it in with the exam**
- **You will write your answers on the exam sheet itself.**

Bring your UCSB IDs to the exam!!!

Study Session with a TA!!

- TA Sourav Medya (medya@cs.ucsb.edu) will lead a review session for anyone interested

- **Friday**, **April 21st** from **1:00 – 2:00 PM**
- In **PSYCH 1924**

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

Sample Question
Multiple Choice

What is the answer to this operation: 1+3j**2 ?

A. 1 + 9j

B. -9

C. -9 + 0j

D. -8

[✓ E. -8 + 0j]

Sample Question *Multiple Choice*

What is exactly printed by this code?

```
for z in range(3, 5, 1):
    print( z * z)
```

A. 3, 5, 1 on separate lines

[✓ B. 9, 16 on separate lines]

C. 9, 16, 25 on separate lines

D. 3, 5 on separate lines

E. None of the above

Sample Question
Short Answer

Write Python code that does the following: if the value of a variable, **v**, is less than 5, you will print out “UCSB” **v** times. Otherwise you will print out “Gaucho” once.

```
if v < 5:
   for j in range(v):
      print “UCSB”
else:
   print “Gaucho”
```

Starting Chapter 3

Strings

- Chapter 3’s problem context is **cryptography**, but mostly it is about **strings** and related ideas
- Strings are basically sequences of characters
- A string literal is enclosed in quotes (`' '` or ” ” in Python):

```
‘hello’ == “hello”    >>> True
```

Strings

- Actually objects of a Python class named `str`

```
    type('kitty')    >>> <class 'str'>
```

- Can assign names like any other type of object

```
    message = "Don't be late!"
    print(message)  >>> Don't be late!
```

- Lots of built-in functions work for string objects, and `class str` has useful operators and methods too

Operations on Strings

- Concatenation
  - Merging multiple strings into 1
  - Use the + operator
    - “say my” + “ ” + “name” = “say my name”

- Repetition
  - Easy way to multiply the contents of a string
  - Use the * operator
    - “ja ” * 3 = “ja ja ja ”

Indexing

- Every character in a string has an index associated with it

0	1	2	3	4	5	6	7	8
y	**o**		**m**	**a**	**m**	**a**	**’**	**s**

- In Python, indexing always starts at 0.
  - So the 1st character in the string is character #0
  - Indexing is called out with square brackets [*n*]

- If `name = “Jimbo Jones”`, then:

```
name[0] = “J”
name[4] = “o”
name[5] = “ ”
```
`name[15]` is undefined (error)

(Fun)ctions for Strings

- Length of string: len(*string*)
  - Example: len(“Gaucho”) = 6
- Slice a string into a smaller string: [*i*:*j*]
  - Where *i* = starting index, *j* = ending index
  - Example: “Gaucho”[2:4] = “uc”
- Combinations are possible!
  - Example:
    ((“o” + “Gaucho”[2:5] + “ ”) * 3) + “!”

More (Fun)ctions!

- Boolean operators **`in`** and **`not in`** to check if a sub-string is found inside a longer string

<u>Examples</u>:

- `“fun”` **`in`** `“functions”` = True
- `“fun”` **`in`** `“Functions”` = False
- `“Fan”` **`not in`** `“Functions”` = True

String Methods

See Table 3.2 in textbook

Assume: `name = ‘Bubba’`

- `name.center(9) = ‘  Bubba  ’` ← centers w/ spaces on each side
- `name.count(‘b’) = 2` ← counts how many times ‘b’ occurs
- `name.count(‘ubb’) = 1`
- `name.ljust(9) = ‘Bubba    ’` ← left justifies name in 9 spaces
- `name.rjust(9) = ‘    Bubba’` ← right justifies name in 9 spaces
- `name.upper() = ‘BUBBA’` ← all uppercase letters
- `name.lower() = ‘bubba’` ← all lowercase letters
- `name.index(‘bb’) = 2` ← Index of first occurrence of first letter
- `name.find(‘bb’) = 2` ← Index of first occurrence of first letter
- `name.find(‘z’) = -1` if not found, then returns -1
- `name.replace(‘bb’,‘dd’) = ‘Budda’`

Example

Assume string **s** =

“h	o	w		n	o	w		b	r	o	w	n		c	o	w		m	e	o	w	!”
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

What is:

- s.find(‘m’) = 18
- s.find(‘r’) = 9
- s.find(‘ow’) = 1
- s.find(‘s’) = -1
- s.replace(‘ meow’, ‘moo?’) = “how now brown cowmoo?!”

  ← *note: one space before **meow***

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

- Characters are stored as numbers in memory
  - There are standard codes for characters, e.g. ASCII codes, UTF-8, etc…

- For example, `'A'` has code `65` in ASCII
  - Use `ord` function to verify: `ord('A')  >>> 65`
  - Notice `'A'` is not same as `'a'`: `ord('a') >>> 97`

- Every character, **seen** (e.g. %, !, G, =, …) and **unseen** (e.g. CONTROL-X, newline…) has ASCII code

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

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

```
chr(65)   >>> 'A'
```

- Can manipulate numbers to process characters

```
chr( ord('a') + 3)  >>> 'd'
```

- Notice digit characters have codes too!

```
ord('6')  >>> 54
```

Examples

- How can I find out what's 13 letters after 'e'??
  - **chr(ord('e') + 13)**
- How can I "add" '3' and '4' and get '7'??
  - First ask: how can I make '3' into 3? ***HINT***: We need a baseline!
- 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:

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!

YOUR TO-DOs

- ❑ Finish reading **Chapter 3** for Thursday's class
- ❑ Finish **Homework3** (due **Thursday 4/27**)
- ❑ Finish **Lab2** (due **Friday 4/21**)

- ❑ **Study for Midterm #1!!!!**
  - ❑ Remember the study session: **Fri. 4/21 @ 1pm in PSYCH 1924**
  - ❑ Come see the prof. or the TAs during office hours
    if you have questions

- ❑ Run through an open meadow

</LECTURE>