

Strings in Python

CS 8: Introduction to Computer Science, Winter 2018
Lecture #7

Ziad Matni
Dept. of Computer Science, UCSB

Administrative

- Homework #3 due next Monday
- Lab #2 due end of today
 - If you're having trouble with submit, let me know
- Midterm is on **Wednesday, Feb 14th**

```

# Today's Date:02/06/18
# student functions go here

#factorial - returns n factorial
# assumes n is greater than or equal to 0
def factorial(n):
    factorial = 1
    if(n> 0):
        for i in range(1,n+1):
            factorial = factorial* i
        return factorial
    elif(n==0):
        return factorial
    else:
        print("you messed up")

# function to draw a rectangle
# parameters: name of a turtle, and the width and
# draws: a rectangle at the position of the turtle
#import turtle
#pat = turtle.Turtle("turtle")
def drawRectangle(myTurtle, length,width) :
    for i in range(2) :
        myTurtle.forward(length)
        myTurtle.right(90)
        myTurtle.forward(width)
        myTurtle.right(90)

```

```

for i in range(number):
    drawHouse(myTurtle,width)

```

IMPORTANT: Students: do not change any part of the code below

```

def main():

```

```

    # Test factorial function
    for i in range(1,11):
        print(factorial(i))

```

```

    # Test draw functions
    #drawRectangle()
    #drawHouse()
    #drawBlock()

```

```

main()

```

hello
你好

**Looking for Mandarin
bilinguals!**

If you are a native Mandarin speaker, please consider participating in our study. This study looks at perceptions of Mandarin speakers. You will be entered into a lottery where you can earn \$100 VISA Card.

Please contact
ahansia@umail.ucsb.edu
if interested.

Lecture Outline

- About the Midterm Exam
- Random Number Uses in Python
- Characters and Strings in Python

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.**



***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

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

Fill in the Blanks

The following code is supposed to print out these numbers on the same line and separated by spaces:

82 80 78 76 74 72 70

Complete the code below:

```
for num in range(82, 69, -2):  
    print(num, end=" ")
```

Sample Question

Coding

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")
```

But Wait! There's More!

- Sample exam questions are posted online for you to practice on
- I'll go over some of them on Monday

Random Values

- “Pseudo-random” values can be generated using special functions in most programming languages
- The `random` module
 - Simplest form is `random.random()`
 - Returns a floating point value between 0.0 and 1.0

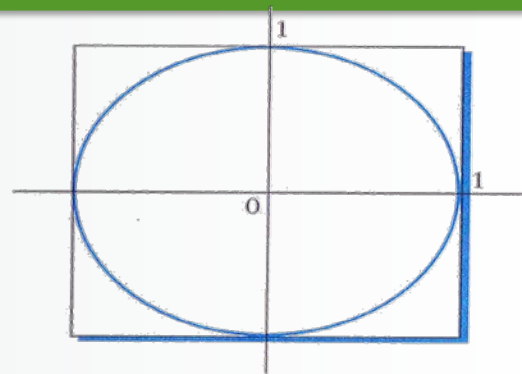
Random Values

- The `random` module has other functions too
- `random.randrange(n)`
 - Works like `range(n)`
 - Will generate a random number from 0 to n-1 every time
- `random.randint(low, high)`
 - A little more intuitive: will generate a random number from low to high (inclusive of both) every time
 - Best choice if you want to create a dice generator!
 - `random.randint(1, 6)`
- Try `help(random)` to learn more
- For more examples, see [Listing 2.5 in textbook](#)

CLASS DEMO: HOW TO USE random

Monte Carlo Simulation

- A popular statistical method using randomness to solve problems.
 - Used in many simulation – traffic flows, length of bank queues, etc...
- In the case of estimating pi – imagine throwing darts at a unit circle (i.e. $r = 1$) inscribed inside a square (i.e. whose side = $2r = 2$)
 - Circle area = $\pi r^2 = \pi$
 - Square area = $2 * 2 = 4$
 - So if n darts hit the square, how many darts (k) should land inside the circle by chance alone?
 - As it turns out, that's proportional to the area of the circle divided by the area of the square.
 - Answer: $k = n * \pi/4$. In other words, we can approximate $\pi_{est} = 4 * k/n$



See Listing 2.5 in textbook

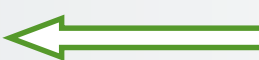
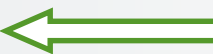
CLASS DEMO: HOW TO USE random

montePi(numDarts)

```
def montePi(numDarts):  
    # numDarts is the number of darts that we throw at the square  
    k = 0      # k is the nuber of darts that hit the circle inside the square  
  
    for i in range(numDarts):  
        x = random.random()      # x and y are random coordinates  
        y = random.random()      # representing the dart throw location  
        d = math.sqrt(x**2 + y**2) # d = distance between (x,y) and origin (0,0)  
        if d <= 1:               # if d <= 1, it means that the  
            k = k + 1            # hit is within the circle, so count those  
  
    pi = 4 * (k / numDarts)  
    return pi
```

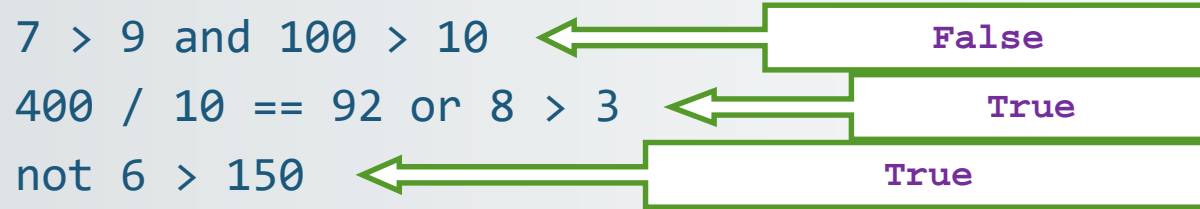
QUESTION: How close do we get to actual π using this method?
(see demo from class...)

Boolean Expressions

- Expressions that evaluate to `True` or `False`
- Relational operators: `<` `<=` `>` `>=` `==` `!=`
Example: `9 > 7` is `True`, while `(4.5 - 3) >= (3 - 1.3)` is `False`
- Watch out when using `==` or `!=` with *floating point numbers*
Example: `100/3 == 33.3333`  `False (why?)`
– Instead it's better to compare absolute difference to a small value
`abs(100/3 - 33.3333) < 0.0001`  `True (why?)`

Compound Boolean Expressions

- Logical operators: **and**, **or**, **not**
- Their operands are Boolean values:



- Special Python feature: `low <= value <= high`
- The special role that 0 and 1 play
 - See other behavior notes in Table 2.2 (p. 66)

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

(either `' '` or `" "` in Python):

```
>>> print('hello' == "hello" )  
True
```


Strings

- Strings are **objects** of a Python class named `str`
- This is not the case for other variables, like integers

```
type('kitty')    >>> <class 'str'>
type(13)         >>> <class <'int'>
type(13.3)       >>> <class <'float'>
```

- We can assign names to these variables like any other type of object

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

Operations on Strings

- Lots of built-in functions work for string objects, and `class str` has useful operators and methods too
- **Concatenation**
 - Merging multiple strings into 1
 - Use the `+` operator
 - `"say my" + " " + "name"` will become `"say my name"`
- **Repetition**
 - Easy way to multiply the contents of a string
 - Use the `*` operator
 - `"ja " * 3` is `"ja ja ja "` (why is there a space at the end?)

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 Ole")` is 10
- To slice a string into a smaller string, use **[i:j]**
 - Where i = starting index, j = ending index (NOT included)
 - Example: `"Gaucho"[2:4]` is "uc"
- Combinations are possible!
 - Example, what does this spell out?
`(("o" + "Gaucho"[2:5] + " ") * 3) + "!"`

More (Fun)ctions!

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

Examples:

- `"fun" in "functions"` = `True`
- `"fun" in "Functions"` = `False`
- `"Fan" not in "Functions"` = `True`

String Methods

Also see Table 3.2 in textbook

Try all of these out
as part of your
homework

Assume: name = 'Bubba'

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

Example

Assume string **s** = “how now brown cow meow”

“ 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 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 (not-found-in-Python) “add” of 2 numeral characters, 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:

$$\text{ord}('3') - \text{ord}('0') + \text{ord}('4') - \text{ord}('0') = 7$$

$$\text{or, } \underline{\text{ord}('3') + \text{ord}('4') - 2 * \text{ord}('0')} = 7$$

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

$$\text{chr}(\underline{\text{ord}('3') + \text{ord}('4') - 2 * \text{ord}('0')}) + \text{ord}('0') = '7'$$

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 (sections 1 and 2 only)** for next class
- ☐ Finish **Homework3** (due **Monday 2/12**)
- ☐ Finish **Lab2** (due **Wednesday 2/7**)
- ☐ Study for your **Midterm Exam (Wednesday 2/14)**
- ☐ Embrace randomness

</LECTURE>