CS 111: Program Design I

Lecture 12: Nested loops, Files Modules, Encodings, Supreme Ct

Robert H. Sloan & Richard Warner

University of Illinois at Chicago

Oct. 8, 2019 (First lecture after first Monday in October)

UIC

NESTED LOOPS: QUICK LOOK

Very common pattern: Loop inside a Loop

- Frequently useful pattern: *nested loops*
- Classic example is printing out a multiplication table, say 0 up to 10:

```
for i in range(11) # want 0 to 10
    for j in range(11)
        print(i * j)
```

Or print triangles

```
for i in range(1, n + 1):
    for j in range(i):
        print(char, end = ' ')
    print()
```

Labs and projects

- This week: the part of the web crawler that
  - finds 1 web address inside text of web page (lab)
  - finds *all* web addresses inside text of web page (Project 2)
- Then putting web crawler aside for a while
- Next big assignment: Data analyses of votes of Supreme Court Justices

FILES

Files

- Text
  - Collections of lines
    - Line = sequence of characters terminated by End Of Line (aka EOL, newline)
      - Can get End of Line explicitly with \n

- Binary: program (or zipped zip file)

Creating text files

- Any old editor *or Microsoft Word!*
  - Or download or …
- Observe

Open()

- Start working with file by using **open()** function.
  - Returns **file** object, most commonly used with two arguments (but can omit second, defaults to read)

- Syntax:

```
file_object = open(filename, mode)
```

Mode:

- **'r'** file will only be read
- **'w'** only writing (existing file with same name will be *erased*)
- **'a'** for appending; any data written to file automatically added to end.
- **'r+'** opens the file for both reading and writing.

File functions (after creating file with open)

file.write(*String*)

file.read(*num_characters*)

file.readline()
file.readlines()

file.close()

Creating a file

```
[In 1]: justices = 'Neil Gorsuch,
Clarence Thomas, Ruth Bader Ginsburg,
Stephen G. Breyer, John G. Roberts,
Samuel A. Alito, Sonia Sotomayor, Elena
Kagan, Brett Kavanaugh'
[In 2]: file = open("justices.txt", "w")
[In 3]: file.write(‘These are the
justices of the Supreme Court As of Oct 1
2019\n’)
[In 4]: file.write(justices)
[In 5]: file.close()
```

Reading a file

```
[In 6]: file = open("justices.txt", "r")
[In 7]: print(file.read())
[Out7]: These are the justices of the
Supreme Court as of Oct 1 2019
Neil Gorsuch, Clarence Thomas, Ruth
Bader Ginsburg, Stephen G. Breyer, John
G. Roberts, Samuel A. Alito, Sonia
Sotomayor, Elena Kagan, Brett Kavanaugh
```

PYTHON STANDARD LIBRARY & BEYOND: INTRO TO MODULES

Extending Python

- Every modern programming language has way to extend basic functions of language with new ones
- Python: **importing a module**
- module: Python file with new capabilities defined in it
- One you import module, it's as if you typed it in: you get all functions, objects, variables defined it immediately

Python Standard Library

- Python always comes with big set of modules
- List at https://docs.python.org/3/py-modindex.html
- Examples

csv	Read/write csv files
datetime	Basic date & time types
math	Math stuff (e.g., sin(), cos())
os	E.g., list files in your operating system
random	random number generation
urllib	Open URLs, parse URLs

Using Modules

- Use `import <module_name>` to make module's function's available
- Style: Put all import statements at top of file
- After `import module_name`, access its functions (and variables, etc.) through `module_name.function_name`

Module use continued

- If module_name is long, can abbreviate in `import` with `as`:
  - `import module_name as mn`
  - `mn.function_name`
  - There are several common modules where this is universal practice
  - E.g., `import matplotlib.pyplot as plt`
  - Observe: A few modules come in hierarchical organization with dots

If you prefer to save typing

- (I mostly do *not* do this)
- To access `function_name` without having to type `module_name` prefix, use:

```
from module_name import function_name
```

Dot notation remark

- Python makes two (2.5?) different uses of dot notation
  - methods, where as we've seen, we call method as
    - obj_name.method_name()
  - functions in modules
    - module_name.function_name
    - And a few hierarchically named modules, like matplotlib

Common but not standard

- matplotlib and pandas are *not* part of set of modules that must come with every Python 3
- matplotlib is very, very widely used, and pandas is widely used
- Both are among the many modules that come with the Anaconda distribution of Python 3
- For many that don't come automatically with Anaconda can tell Anaconda Navigator to get for you: Environments tab (E.g., scrapy)

TOWARDS PANDAS MODULE

A bit more on modules soon

- But we have plenty to be module users now

- Recall: ZyBooks ordering is tell you everything about one Python construct then more on to next; CS 111 Law ordering spiral
  - Why? We want you to be able to do real work with real datasets as soon as possible

pandas module: data science

- First module we will use heavily is pandas
  - Purpose: data science / data analytics
  - Very heavy majority of all data analytics work done in one or other of:
    1. R language
    2. Python using pandas module
- Next week's big project (will return to web crawler): data analytics on Supreme Court voting records

Why study pandas?

1. Data science fascinating and red-hot subject, and pandas is one of top two tools for data science
   - Can analyze *much* larger datasets much faster than in Excel; we'll only scratch surface
2. Opportunity to work with large, complex module, and learn our way around
3. Opportunity to work with a complex object (dataframe) with many methods and learn them

MORE ON ENCODINGS

Encodings again

- Recall that the smallest unit in a computer is the bit
- One bit can take on 2 possible values: 0 or 1
- Two bits can take on 4 possible values: 00, 01, 10, or 11
- Three bits can take on 8 possible values: 000, 001, 010, 011, 100, 101, 110, 111

How many distinct values can 4 bits take on?

A. 4
B. 8
C. 9
D. 13
E. 16

Ben Bitdiddle says

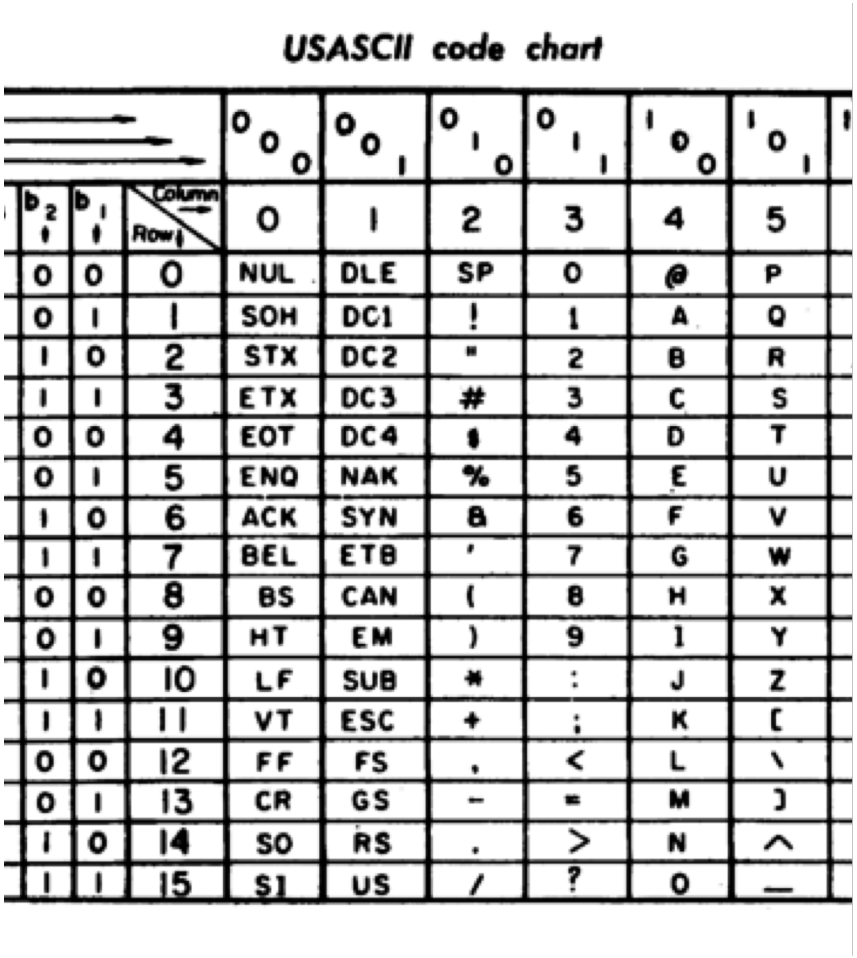
- n bits can take on 2 times as many values as n-1 bits = 2^n values

How many distinct values can 1 byte take on

- (Recall that a byte = 8 bits)

A. 2
B. 8
C. 64
D. 128
E. 256

Encoding characters in bytes: 1960s



USASCII code chart

b_2	b_1	Row \ Column	0 $0\,0\,0$	1 $0\,0\,1$	2 $0\,1\,0$	3 $0\,1\,1$	4 $1\,0\,0$	5 $1\,0\,1$
0	0	0	NUL	DLE	SP	0	@	P
0	1	1	SOH	DC1	!	1	A	Q
1	0	2	STX	DC2	"	2	B	R
1	1	3	ETX	DC3	#	3	C	S
0	0	4	EOT	DC4	$	4	D	T
0	1	5	ENQ	NAK	%	5	E	U
1	0	6	ACK	SYN	&	6	F	V
1	1	7	BEL	ETB	'	7	G	W
0	0	8	BS	CAN	(	8	H	X
0	1	9	HT	EM	)	9	I	Y
1	0	10	LF	SUB	*	:	J	Z
1	1	11	VT	ESC	+	;	K	[
0	0	12	FF	FS	,	<	L	\
0	1	13	CR	GS	-	=	M	]
1	0	14	SO	RS	.	>	N	^
1	1	15	SI	US	/	?	O	_

- ASCII: Use 1 byte to encode 95 *printing* characters
- The ones on every computer keyboard to this day
- Pretty much *all encodings agree with ASCII on those 95 characters*
- ASCII also has some nonprinting characters like newline and tab

Communicating common non-printing characters in Python

- \n is used to denote newline in a string literal
- \t is used to denote tab in a string literal
- And so double backslash is used to denote a backslash in a string literal.
- What is len("\\")?
  A. 0
  B. 1
  C. 2

But

- What about Viginère, Beyoncé Knowles, and Renée Zellweger?

- А что насчет Арабского?

Encoding more characters

- Unicode: over 128,000 characters covering 135 modern and historical scripts, and symbols
- 2 bytes not enough for all of it
  - 2^{16} is "only" 65,536
- Various character encodings for all or a subset
- Python supports Unicode