

Miscellaneous Topics



FILE I/O
THE ORD AND CHR FUNCTIONS
MARCH 11

File I/O



- More often than not programs read from files, rather than from input typed at the keyboard.
- Often one program reads what another program outputs.
- More and more, programs are reading data produced by other hardware, e.g., sensors, telescopes, microarrays, etc.
- In these instances very little, if any, input is provided at the keyboard.

File objects



- Simplest Python statement for opening a file:

```
f = open("war.txt")
```

- Assuming that there is a file called “`war.txt`” in the same directory as your Python program, this statement *opens* the file for reading.
- Subsequently, the file can be accessed via the variable `f`.
- Since `f` is a variable, it has a type. Try `type(f)`.

File objects



- The variable `f` is often called a *file object*.
- If the file is missing from the directory, an error message is issued.

```
>>> g = open("hello.txt")
Traceback (most recent call last):
File "<string>", line 1, in <fragment>
IOError: [Errno 2] No such file or directory: 'hello.txt'
```

- Once a file object is successfully connected to a file residing on your machine, we can use the file object to read from the file in a variety of ways.

Reading from a file



- `s = f.read()`

Reads everything from the file into the string s

- `s = f.readline()`

Reads the next line from the files into s

- `for line in f:`

```
    print line.split()
```

Allows us to read and process the file line by line

Let us solve these problems on “War and Peace”



1. Build a dictionary of words extracted from the text that we might be able to use later, maybe in a spellchecker.
2. Compute the number of sentences in the text.
3. Compute the frequencies of letters in the text.

Two useful built-in Python functions that can help in solving Problem 3 are `ord` and `chr`.

Two useful functions



- `ord(ch)`
if `ch` is a single character string, this function returns the ASCII code for `ch`
- `chr(i)`
returns a string of one character whose ASCII code is the integer `i`

What is ASCII?

It stands for the *American Standard Code for Information Interchange*. It assigns a number in the range 0..255 to every character that can be entered at the keyboard.

More on ASCII



- The numbers 0..31 are reserved for unprintable characters, e.g., the tab character ("\\t"), the end of line character ("\\n"), etc.
- 32 is the ASCII value of the space character (" ")
- 33..47 is used for some punctuation characters
- 48..57 is used for digits “0” through “9”
- 65..90 is used for upper case letters
- 97..122 is used for lower case letters

ASCII Table



Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	Ø	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	Ø	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	:	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

Some examples of `chr` and `ord` in action



```
>>> ord("a")
97
>>> chr(97)
'a'
>>> ord(" ")
32
>>> ord("o")
48
>>> chr(48)
'o'
>>> chr(49)
'1'
>>> ord("A")
65
>>> ord("B")
66
```

How are these functions useful?



- Because of the fact that all the upper case letters occur consecutively in the ASCII table, the expression `ord(ch) - ord("A")` has value 0 for `ch = "A"`, value 1 for `ch = "B"`, has value 2 for `ch = "C"`, etc.
- Similarly, `ord(ch)-ord("a")` has value 0 for `ch = "a"` , has value 1 for `ch = "b"`, has value 2 for `ch = "c"`, etc.

A program to count letter frequencies



```
f = open("war.txt")
L = [0]*26
s = f.read()
for ch in s:
    if ch.isupper():
        L[ord(ch)-ord("A")] = L[ord(ch)-ord("A")] + 1
    elif ch.islower():
        L[ord(ch)-ord("a")] = L[ord(ch)-ord("a")] + 1
print L
```

Notice how `ord(ch)-ord("A")` and `ord(ch)-ord("a")` are used to index into the list `L`.

Another example



- The `ord` and `chr` functions can be used to perform Caeser's Cipher (Problem 3, HW 7).
- Try this: `chr(ord("a") + 4)`
- What does this expression evaluate to?