61A Lecture 10

Wednesday, September 21

Strings are an Abstraction

Representing data:

```
'200'      '1.2e-5'      'False'      '(1, 2)'
```

Representing language:

```
"""O! methinks how slow
This old moon wanes; she lingers my desires ,
Like to a step dame, or a dowager
Long withering out a young man's revenue."""
```

Representing programs:

```
'curry = lambda f: lambda x: lambda y: f(x, y)'
```


Representing Strings: the ASCII Standard

American Standard Code for Information Interchange

"Bell" "Line feed"

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

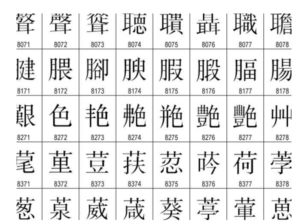
3 bits (rows) · 4 bits (columns)

- Layout was chosen to support sorting by character code
- Rows indexed 2-5 are a useful 6-bit (64 element) subset
- Control characters were designed for transmission

Representing Strings: the Unicode Standard

Bonus Material

- 109,000 characters
- 93 scripts (organized)
- Enumeration of character properties, such as case
- Supports bidirectional display order
- 32 bits per character number
- A canonical name for every character



http://ian-albert.com/unicode_chart/unichart-chinese.jpg

U+0058 LATIN CAPITAL LETTER X

U+263a WHITE SMILING FACE

U+2639 WHITE FROWNING FACE

'☺' '☹'

Demo

Representing Strings: UTF-8 Encoding

Bonus Material

UTF: (UCS (Universal Character Set) Transformation Format)

Unicode: Correspondence between characters and 32 bit numbers

UTF-8: Correspondence between numbers and bytes

A byte is 8 bits, and can encode any integer 0-255

bytes	integers
00000000	0
00000001	1
00000010	2
00000011	3

Variable-length encoding: integers vary in the number of bytes required to encode them!

In Python: string length in characters, bytes length in bytes

Demo

Strings are Sequences

```
>>> city = 'Berkeley'
>>> len(city)
8
>>> city[3]
'k'
```

An element of a string is itself a string!

Length. A sequence has a finite length.

Element selection. A sequence has an element corresponding to any non-negative integer index less than its length, starting at 0 for the first element.

```
>>> 'Berkeley' + ', CA'
'Berkeley, CA'
>>> 'Shabu ' * 2
'Shabu Shabu '
```

String arithmetic is like tuple arithmetic

String Membership Differs from Other Sequences

The "in" and "not in" operators match substrings

```
>>> 'here' in "Where's Waldo?"
True
```

Why? Working with strings, we care about words, not characters

The "count" method also matches substrings

```
>>> 'Mississippi'.count('i')
4
>>> 'Mississippi'.count('issi')
1
```

the number of non-overlapping occurrences of a substring

String Literals Have Three Forms

```
>>> 'I am string!'
'I am string!'
>>> "I've got an apostrophe"
"I've got an apostrophe"
>>> '您好'
'您好'
```

Single- and double-quoted strings are equivalent

```
>>> """The Zen of Python
claims, Readability counts.
Read more: import this."""
'The Zen of Python\nclaims, "Readability counts."\nRead more: import this.'
```

A backslash "escapes" the following character

"Line feed" character represents a new line

String Coercion

Any object can be "coerced" into a string.

Coercion doesn't change an object; it produces a corresponding object of a different type.

```
>>> digits
(1, 8, 2, 8)
>>> 2 in digits
True
>>> str(2) + ' is an element of ' + str(digits)
'2 is an element of (1, 8, 2, 8)'
```

The constructor for a string can take any object as its argument

How is string coercion implemented? October 10

Methods on Strings

```
>>> '1234'.isnumeric()
True
>>> 'rOBERT dE nIRO'.swapcase()
'Robert De Niro'
>>> 'snakeyes'.upper().endswith('YES')
True
```

Demo

Sequences as Conventional Interfaces

Consider two problems:

- Sum the even members of the first n Fibonacci numbers.
- List the letters in the acronym for a name, which includes the first letter of each capitalized word.

enumerate naturals:	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11.
map fib:	0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55.
filter iseven:	0, 2, 8, 34, .
accumulate sum:	44

Sequences as Conventional Interfaces

Consider two problems:

- Sum the even members of the first n Fibonacci numbers.
- List the letters in the acronym for a name, which includes the first letter of each capitalized word.

enumerate words:	'University', 'of', 'California', 'Berkeley'
filter iscap:	'University', 'California', 'Berkeley'
map first:	'U', 'C', 'B'
accumulate tuple:	('U', 'C', 'B')

Mapping a Function over a Sequence

Apply a function to each element of the sequence

```
>>> alternates = (-1, 2, -3, 4, -5)

>>> tuple(map(abs, alternates))
(1, 2, 3, 4, 5)
```

The returned value of map is an iterable map object

A constructor for the built-in map type

The returned value of filter is an iterable filter object

Demo

Accumulation and Iterable Values

Iterable objects give access to some elements in order.

Many built-in functions take iterable objects as argument.

tuple	Return a tuple containing the elements
sum	Return the sum of the elements
min	Return the minimum of the elements
max	Return the maximum of the elements

For statements also operate on iterable values.

Demo

Generator Expressions

One large expression that evaluates to an iterable object

(<map exp> for <name> in <iter exp> if <filter exp>)

- Evaluates to an iterable object.
- <iter exp> is evaluated when the generator expression is evaluated.
- Remaining expressions are evaluated when elements are accessed

(<map exp> for <name> in <iter exp>)

Precise evaluation rule introduced in Chapter 4.

Reducing a Sequence

Reduce is a higher-order generalization of max, min, & sum.

```
>>> from operator import mul

>>> from functools import reduce

>>> reduce(mul, (1, 2, 3, 4, 5))
120
```

Similar to accumulate from Homework 2

Demo