Objectives for class 7

--- Chapter 4 ---

4.1 To solve mathematics problems by using the functions in the math module (§4.2)

4.2 To represent and process strings and characters (§4.3).

4.3 To encode characters using ASCII and Unicode (§4.3.1). 4.4 To use the ord function to obtain ar numerical code for a character and the chrfunction to convert a numerical code to a character (§4.3.2).

4.5 To represent special characters using the escape sequence (§4.3.3).

4.6 To test substrings using the in and not in operators (§4.3.8).

4.7 To compare strings (§4.3.9). 4.8 To use string functions min, max, and len (§4.3.10).

Solve math problems using Python Built-in Functions

```
>>> max(2, 3, 4) # Returns a maximum number
4
>>> min(2, 3, 4) # Returns a minimu number
2
>>> round(4.51) # Rounds to its nearest integer
5
>>> round(4.4) # Rounds to its nearest integer
4
>>> abs(-3) # Returns the absolute value
3
>>> pow(2, 3) # Same as 2 ** 3
8
```

Solve math problems using the math Functions

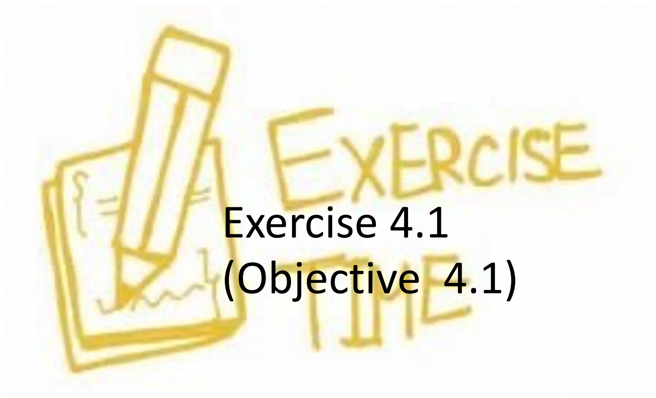
Function	**Description**	**Example**
fabs(x)	Returns the absolute value of the argument.	fabs(-2) is 2
ceil(x)	Rounds x up to its nearest integer and returns this integer.	ceil(2.1) is 3 ceil(-2.1) is -2
floor(x)	Rounds x down to its nearest integer and returns this integer.	floor(2.1) is 2 floor(-2.1) is -3
exp(x)	Returns the exponential function of x (e ** x).	exp(1) is 2.71828
log(x)	Returns the natural logarithm of x.	log(2.71828) is 1.0
log(x, base)	Returns the logarithm of x for the specified base.	log10(10, 10) is 1
sqrt(x)	Returns the square root of x.	sqrt(4.0) is 2
sin(x)	Returns the sine of x. x represents an angle in radians.	sin(3.14159 / 2) is 1 sin(3.14159) is 0
asin(x)	Returns the angle in radians for the inverse of sine.	asin(1.0) is 1.57 asin(0.5) is 0.523599
cos(x)	Returns the cosine of x. x represents an angle in radians.	cos(3.14159 / 2) is 0 cos(3.14159) is -1

acos(x)	Returns the angle in radians for the inverse of cosine.	acos(1.0) is 0 acos(0.5) is 1.0472
tan(x)	Returns the tangent of x. x represents an angle in radians.	tan(3.14159 / 4) is 1 tan(0.0) is 0
fmod(x, y)	Returns the remainder of x/y as double.	fmod(2.4, 1.3) is 1.1
degrees(x)	Converts angle x from radians to degrees	degrees(1.57) is 90
radians(x)	Converts angle x from degrees to radians	radians(90) is 1.57

Import math library for math functions

```
import math # import Math module to use the math functions
# Test algebraic functions
print("exp(1.0) =", math.exp(1))
print("log(2.718) =", math.log(math.e))
print("log10(10, 10) =", math.log(10, 10))
print("sqrt(4.0) =", math.sqrt(4.0))

# Test trigonometric functions
print("sin(PI / 2) =", math.sin(math.pi / 2))
print("cos(PI / 2) =", math.cos(math.pi / 2))
print("tan(PI / 2) =", math.tan(math.pi / 2))
print("degrees(1.57) =", math.degrees(1.57))
print("radians(90) =", math.radians(90))
```



Exercise 4.1
(Objective 4.1)

String is a sequence of characters

- No character type in Python
- A **single-character string** represents a character.
- *String* literals enclosed in matching *single quotes* (') or *double quotes* (").

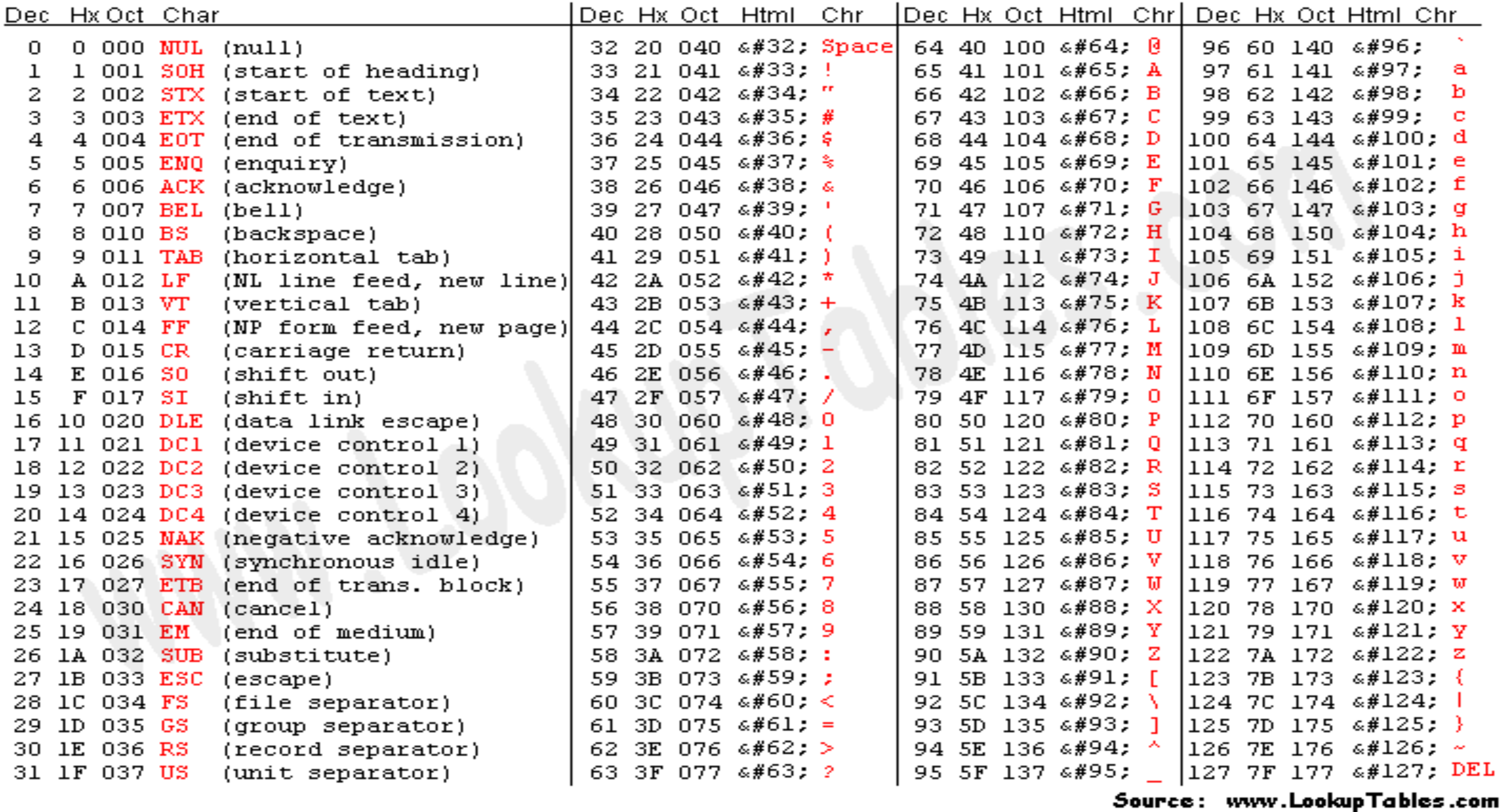
```
letter = 'A' # Same as letter = "A"
numChar = '4' # Same as numChar = "4"
message = "Good morning"
# Same as message = 'Good morning'
```

Unicode/ASCII Assigns a Code to a Character

- A specification
- List every character and assign each character a unique code.
- Rules translating characters into bytes are called **encoding**.

Python	**Encoded as**
`'4'`	`0000000000110100`
`4`	`0000000000000100`

ASCII Table and Description



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

Source: www.LookupTables.com

Unicode (16 bit) vs. ASCII (8 bit)

- Unicode represents **65,536** distinct characters.
- ASCII represents **256** distinct characters.
- ASCII is a small subset of Unicode
- Unicode represents international characters
- Python encode characters using Unicode
  - E.g., `\u6B22`

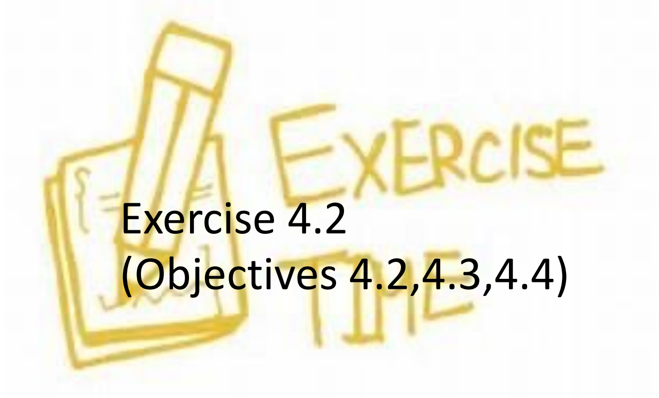
Function **ord**: Obtain Unicode of a Character

```
>>> ch = 'a'
>>> ord(ch)
97
>>> ord(‘b’)
98
>>> ord(‘a’)+1
98
>>> ord(‘A’)
65
```

```
>>> ord(‘1’)
49
>>> ord(‘ ’)
32
>>> ord(‘,’)
44
```

Function **chr** : Obtain a Character by its Unicode

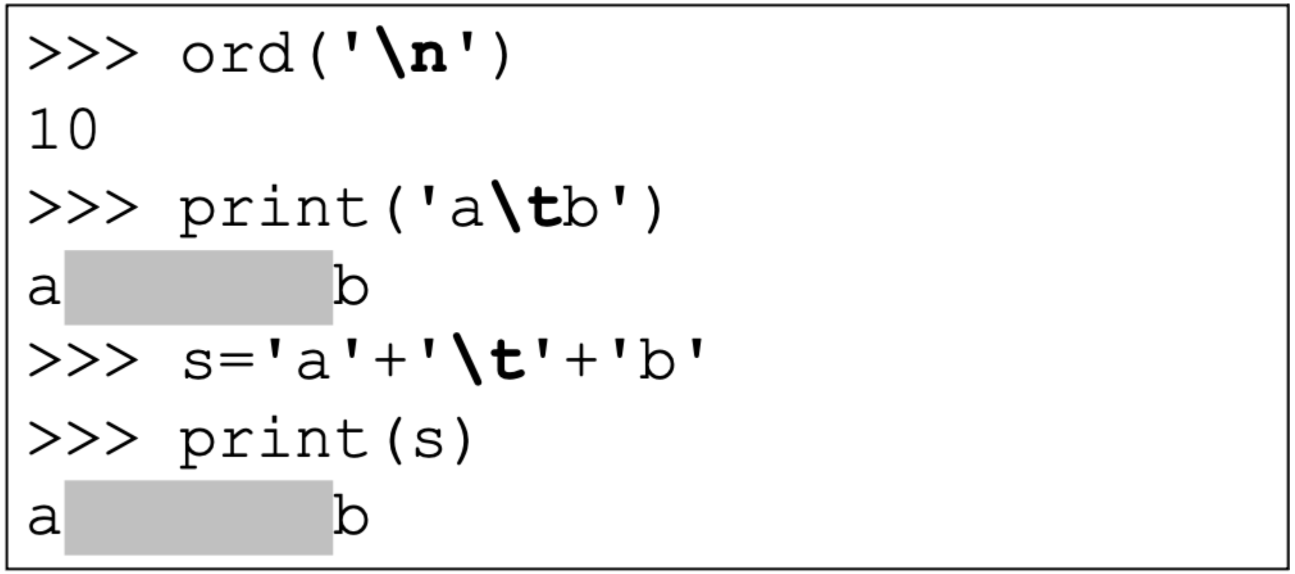
```
>>> chr(98)
'b'
>>> chr(99)
'c'
>>> chr(10)
'\n'
>>> chr(4)
'\x04’
>>> chr(ord('a')+1)
'b'
```



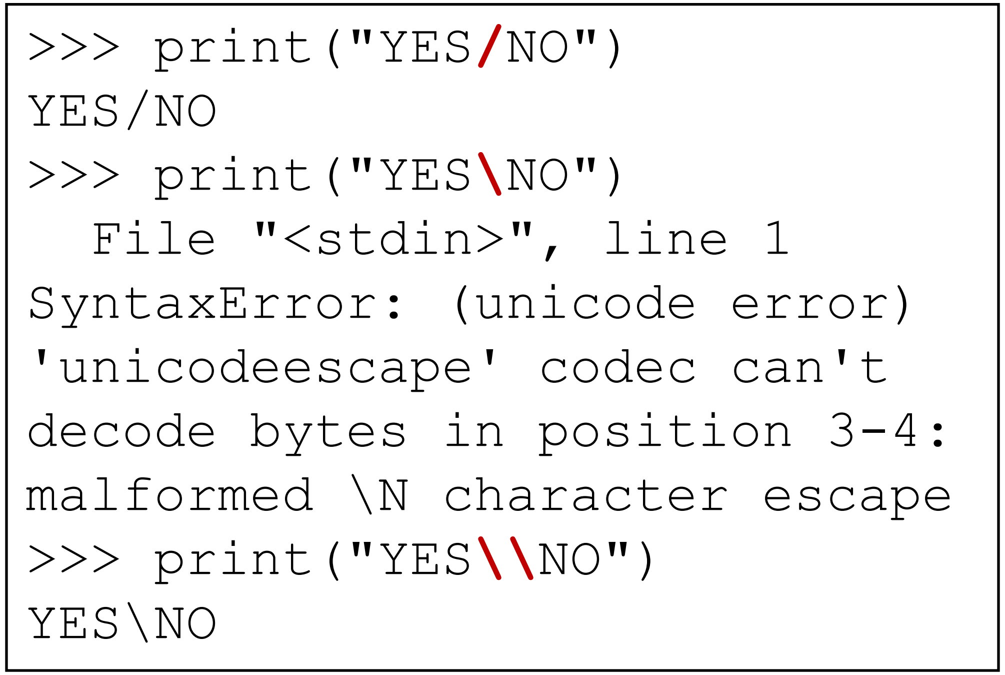
Exercise 4.2
(Objectives 4.2,4.3,4.4)

Represent special characters using Escape Sequences

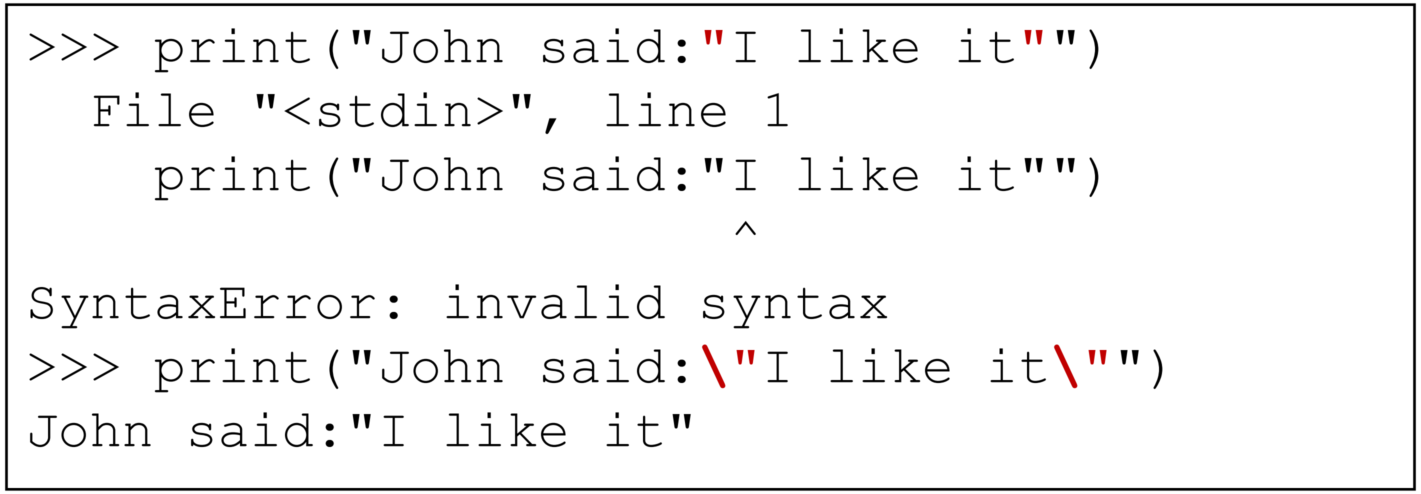
Description	*Escape Sequence*	*Unicode*
Backspace	`\b`	`\u0008`
Tab	`\t`	`\u0009`
Linefeed	`\n`	`\u000A`
Carriage return	`\r`	`\u000D`
Backslash	`\\`	`\u005C`
Single Quote	`\'`	`\u0027`
Double Quote	`\"`	`\u0022`



```
>>> ord('\n')
10
>>> print('a\tb')
a        b
>>> s='a'+'\t'+'b'
>>> print(s)
a        b
```



```
>>> print("YES/NO")
YES/NO
>>> print("YES\NO")
  File "<stdin>", line 1
SyntaxError: (unicode error)
'unicodeescape' codec can't
decode bytes in position 3-4:
malformed \N character escape
>>> print("YES\\NO")
YES\NO
```



```
>>> print("John said:"I like it"")
  File "<stdin>", line 1
    print("John said:"I like it"")
                       ^
SyntaxError: invalid syntax
>>> print("John said:\"I like it\"")
John said:"I like it"
```

in and **not in** Operators: a String in or not in Another String?

```
>>> s1="Welcome"
>>> "come" in s1
True
>>> "come" not in s1
False
```

Practice

- What will be the output if user enters “iPython”?

```
s = input("Enter a string: ")
if "Python" in s:
    print("Python", "is in", s)
else:
    print("Python", "is not in", s)
```

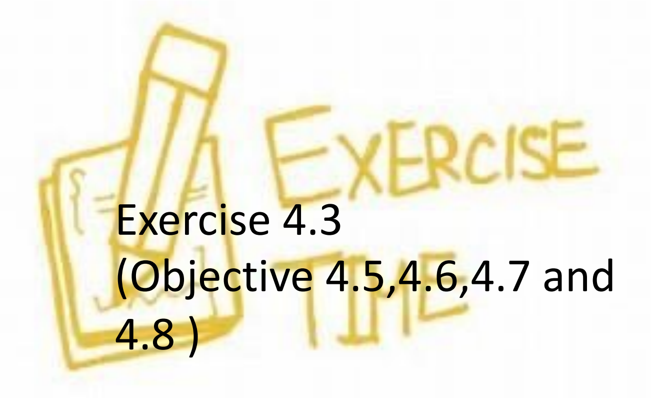
How to compare two strings?

- Characters’ numeric codes(Unicode/ASCII) are compared from left to right.

```
>>> "green" == "glow"
False
>>> "green" != "glow"
True
>>> "green" > "glow"
True
>>> "green" >= "glow"
True
```

```
>>> "green" < "glow"
False
>>> "green" <= "glow"
False
>>> "ab" <= "abc"
True
```

Built-in Functions for Strings



Exercise 4.3
(Objective 4.5,4.6,4.7 and 4.8)

```
>>> s = "Welcome"
>>> len(s)  # return the length of a string
7
>>> max(s) # return the largest character
'o'
>>> min(s) # return the smallest character
'W'
```