SWCON104
Web & Python Programming

Conditional Statement

Department of Software Convergence

경희대학교
KYUNG HEE UNIVERSITY

Today

- Review the type bool
- Boolean operators: and, or, not
- Relational operators: >, <, >=, <=, ==, !=
- Comparing strings (ASCII)
- if statement

[Textbook]
Practical Programming
(An Introduction to Computer Science Using Python),
by Paul Gries, Jennifer Campbell, Jason Montojo,
The Pragmatic Bookshelf, 2017

Practice

- Practice_07_ConditionalStatement.ipynb

Making choices

- A Boolean type, bool can have the value either true or false.
- Boolean operators: and, or, not
  - not is a unary operator: the operator is applied to just one value
  - and, or are binary operators: the operator is applied to two values.

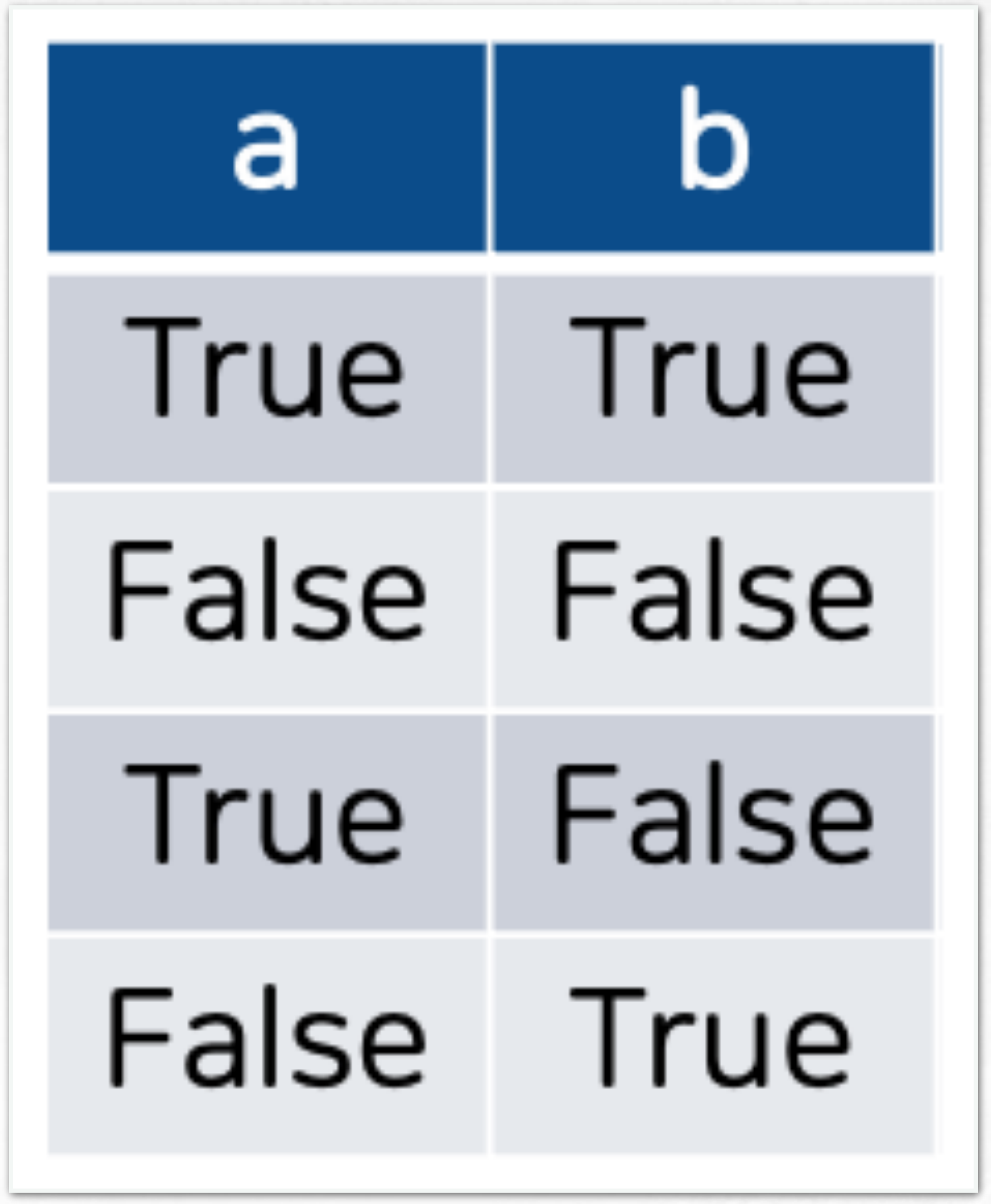
```
>>> not True
False
>>> not False
True
```

```
>>> True and True
True
>>> False and False
False
>>> True and False
False
>>> False and True
False
```

```
>>> True or True
True
>>> False or False
False
>>> True or False
True
>>> False or True
True
```

Truth table

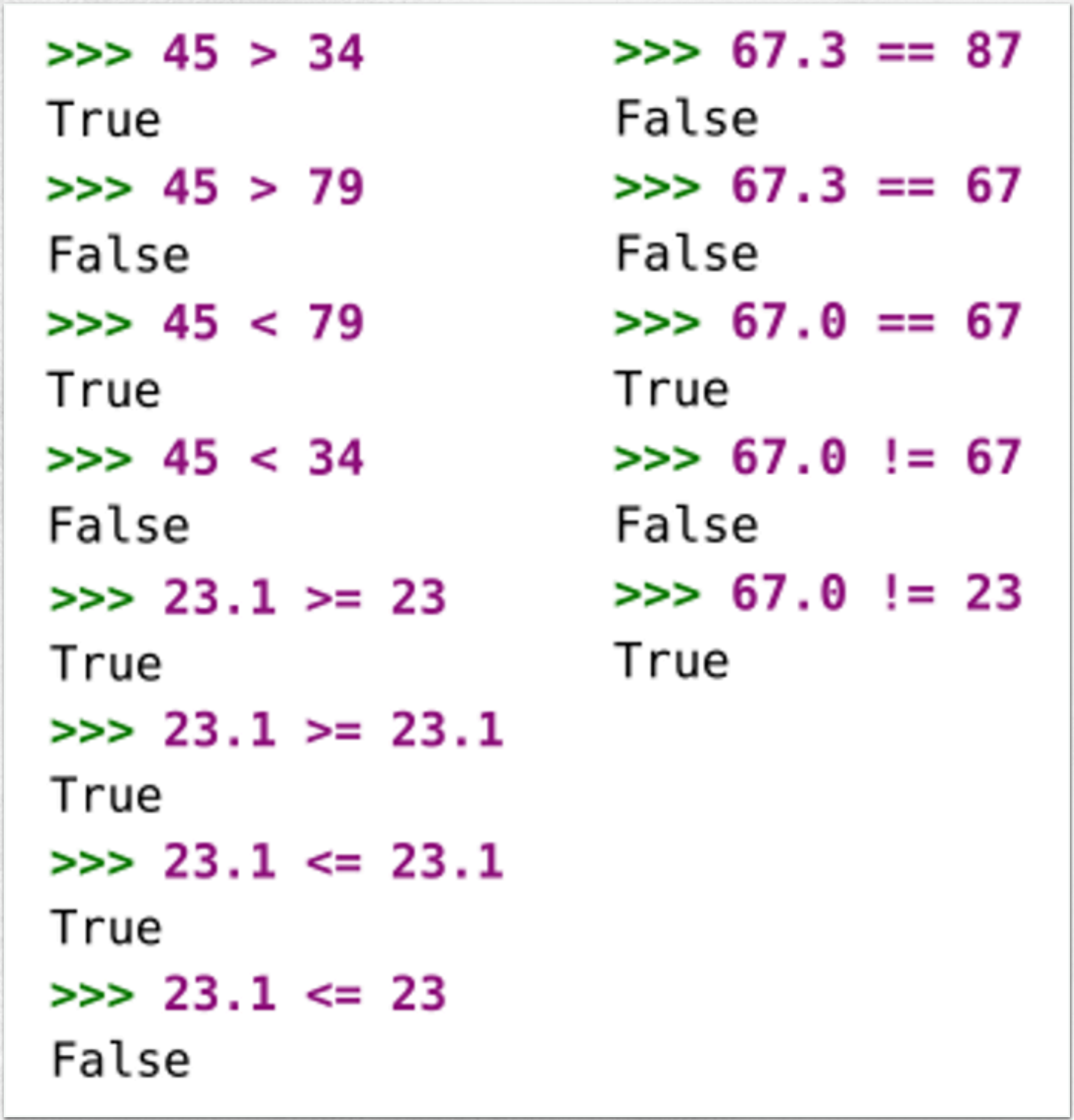
- When a and b are Boolean type variables,



a	b
True	True
False	False
True	False
False	True

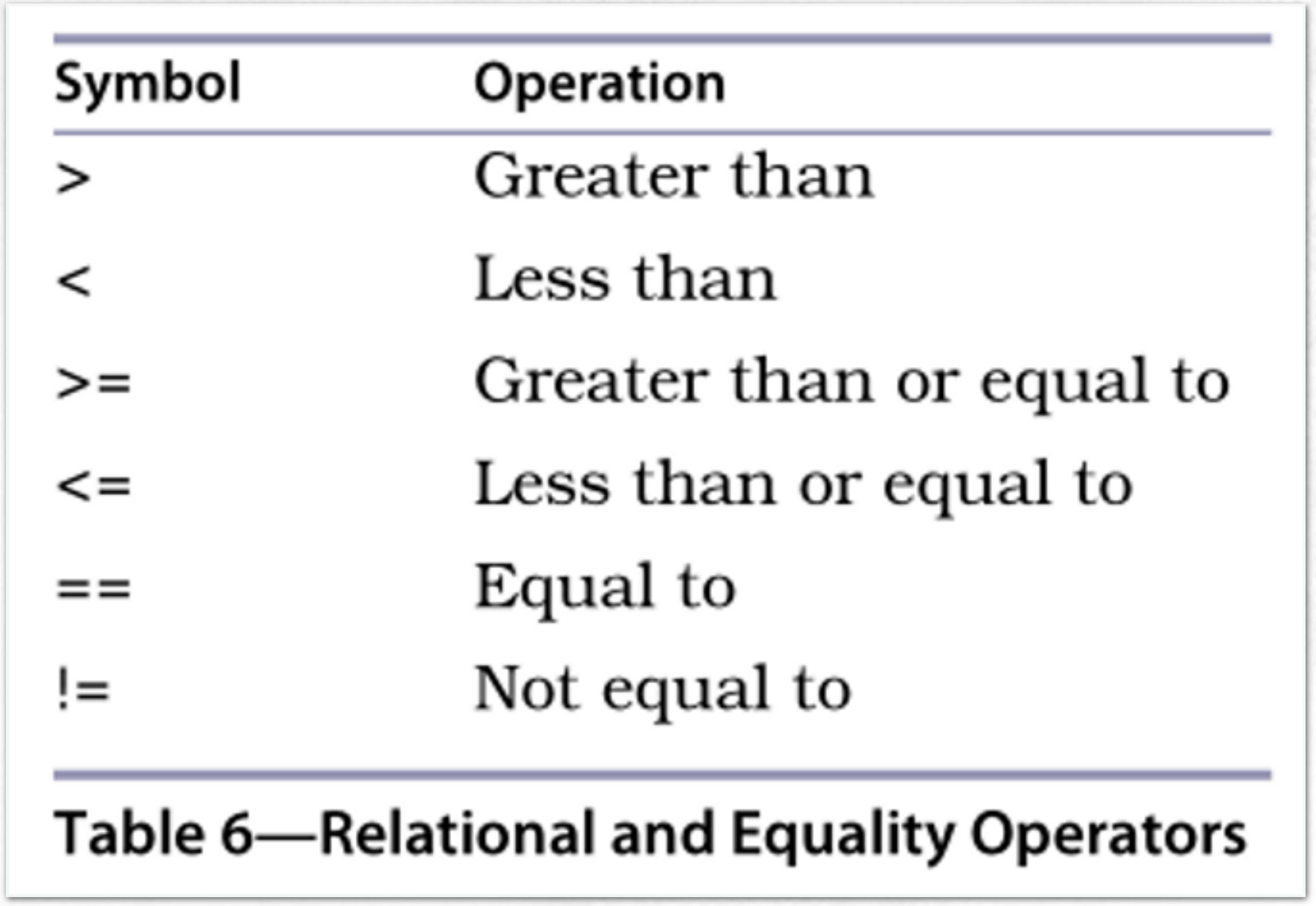
- Inclusive or (OR) vs. Exclusive or (XOR)
  - Inclusive or: a or b (False if and only if both are False)
  - Exclusive or: Do you want to meet on Monday or Tuesday?
  - a XOR b is represented as `(a and not b) or (not a and b)`

Relational operators



```
>>> 45 > 34
True
>>> 45 > 79
False
>>> 45 < 79
True
>>> 45 < 34
False
>>> 23.1 >= 23
True
>>> 23.1 >= 23.1
True
>>> 23.1 <= 23.1
True
>>> 23.1 <= 23
False
```

```
>>> 67.3 == 87
False
>>> 67.3 == 67
False
>>> 67.0 == 67
True
>>> 67.0 != 67
False
>>> 67.0 != 23
True
```

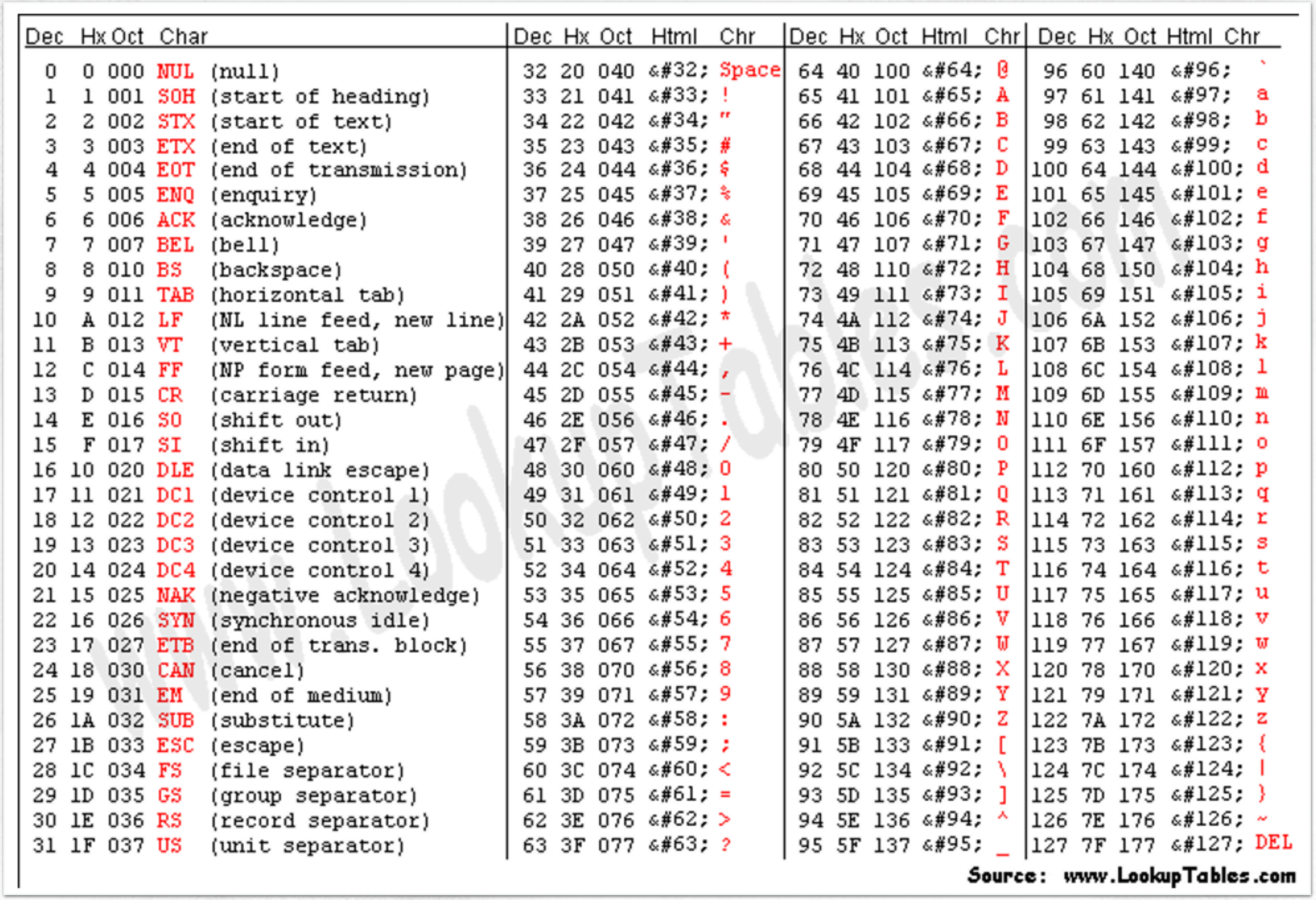


Symbol	Operation
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to

Table 6—Relational and Equality Operators

Comparing strings

- ASCII: American Standard Code for Information Interchange

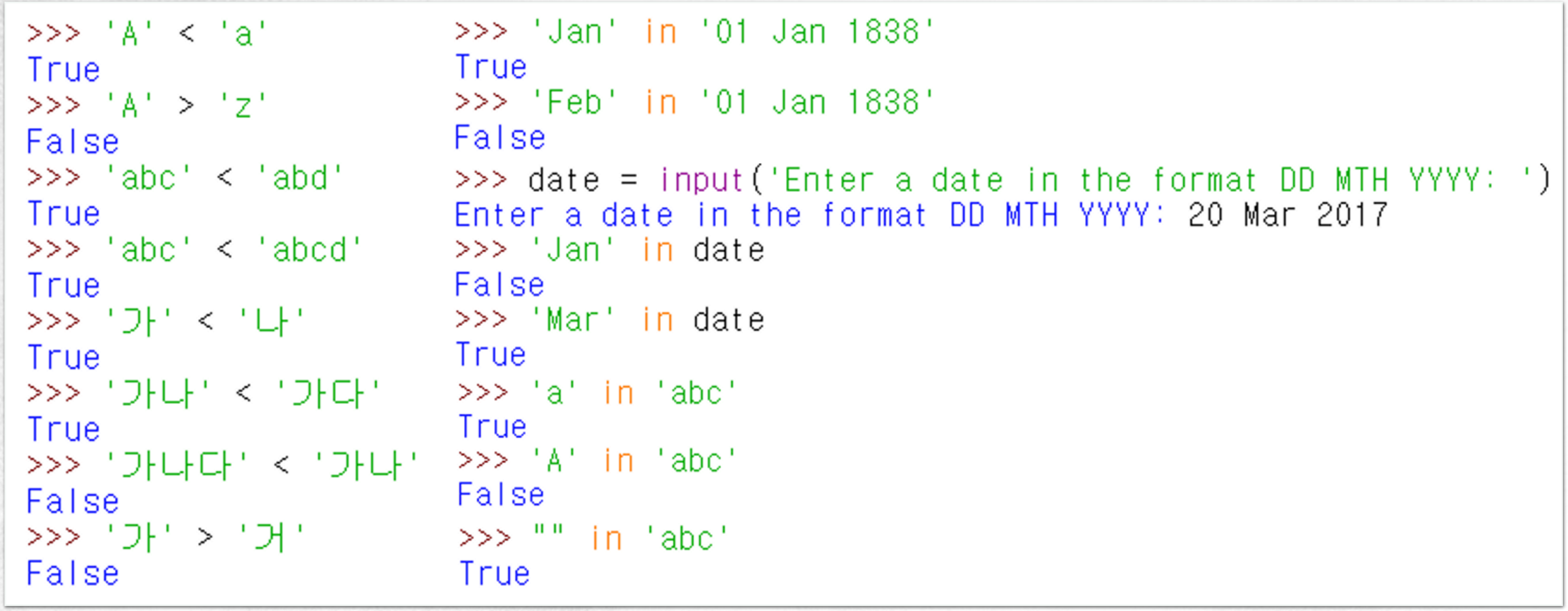


Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	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	7	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	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

Comparing strings

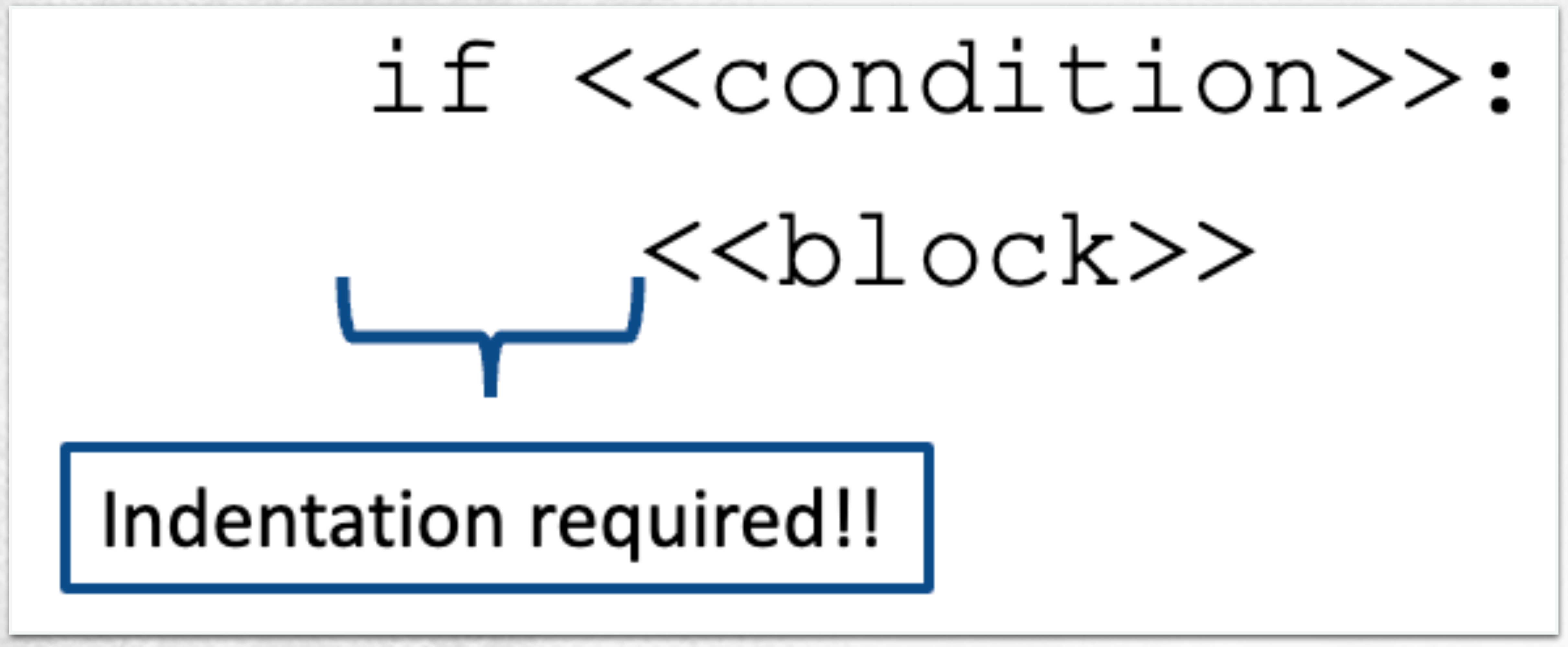
- Lexicographically
- Checks whether one string appears inside another one
- Case sensitive
- Empty string is always a substring of every string



```
>>> 'A' < 'a'
True
>>> 'A' > 'z'
False
>>> 'abc' < 'abd'
True
>>> 'abc' < 'abcd'
True
>>> '가' < '나'
True
>>> '가나' < '가다'
True
>>> '가나다' < '가나'
False
>>> '가' > '거'
False
```

```
>>> 'Jan' in '01 Jan 1838'
True
>>> 'Feb' in '01 Jan 1838'
False
>>> date = input('Enter a date in the format DD MTH YYYY: ')
Enter a date in the format DD MTH YYYY: 20 Mar 2017
>>> 'Jan' in date
False
>>> 'Mar' in date
True
>>> 'a' in 'abc'
True
>>> 'A' in 'abc'
False
>>> "" in 'abc'
True
```

if statement

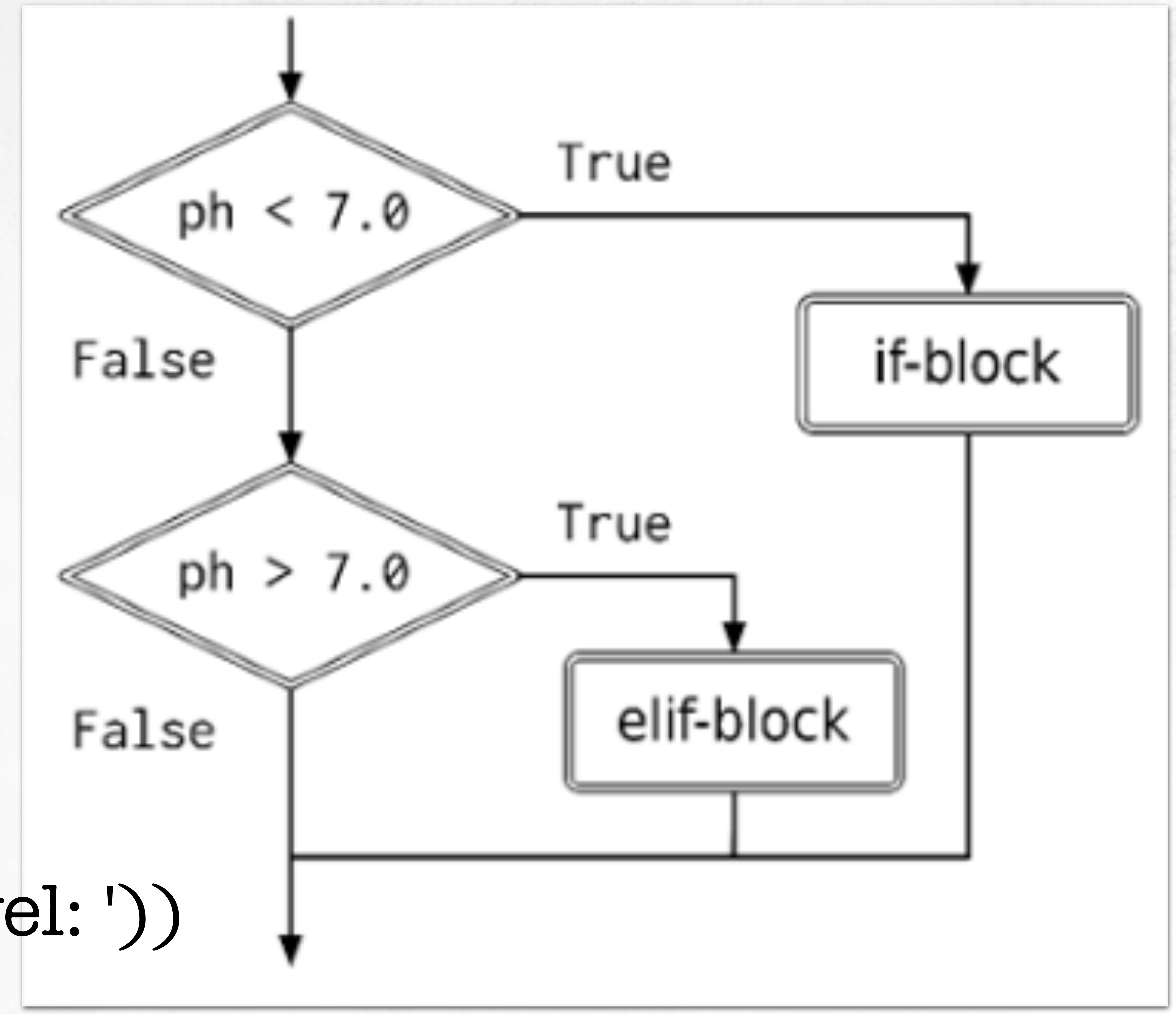


```
if <<condition>>:
    <<block>>
```

- Condition
  - Usually a Boolean expression
  - Has be an expression that can be interpreted as `True` or `False`
- Block
  - If the condition is true, the statements in the block are executed.
  - Otherwise, they are not executed.

if statement example

- A table of solution categories based on pH level



```
ph = float(input('Enter the pH level: '))

if ph < 7.0:
    print(ph, "is acidic.")
elif ph > 7.0:
    print("Be careful with that!")
```

if/elif statement

- If the two conditions are related, use **if/elif** instead of two **ifs**.

```
ph = float(input('Enter the pH
level: '))


if ph < 7.0:

    ph = 8.0


if ph > 7.0:

    print(ph, "is acidic.")
```

```
ph = float(input('Enter the pH
level: '))


if ph < 7.0:

    ph = 8.0

elif ph > 7.0:

    print(ph, "is acidic.")
```

Multiple elif

```
compound = input('Enter the compound: ‘)


if compound == "H2O":
    print("Water")
elif compound == "NH3":
    print("Ammonia")
elif compound == "CH4":
    print("Methane")
```

```
>>> Enter the compound: CH4
Methane
>>>
>>> Enter the compound: H2SO4
>>>
```

Multiple elif

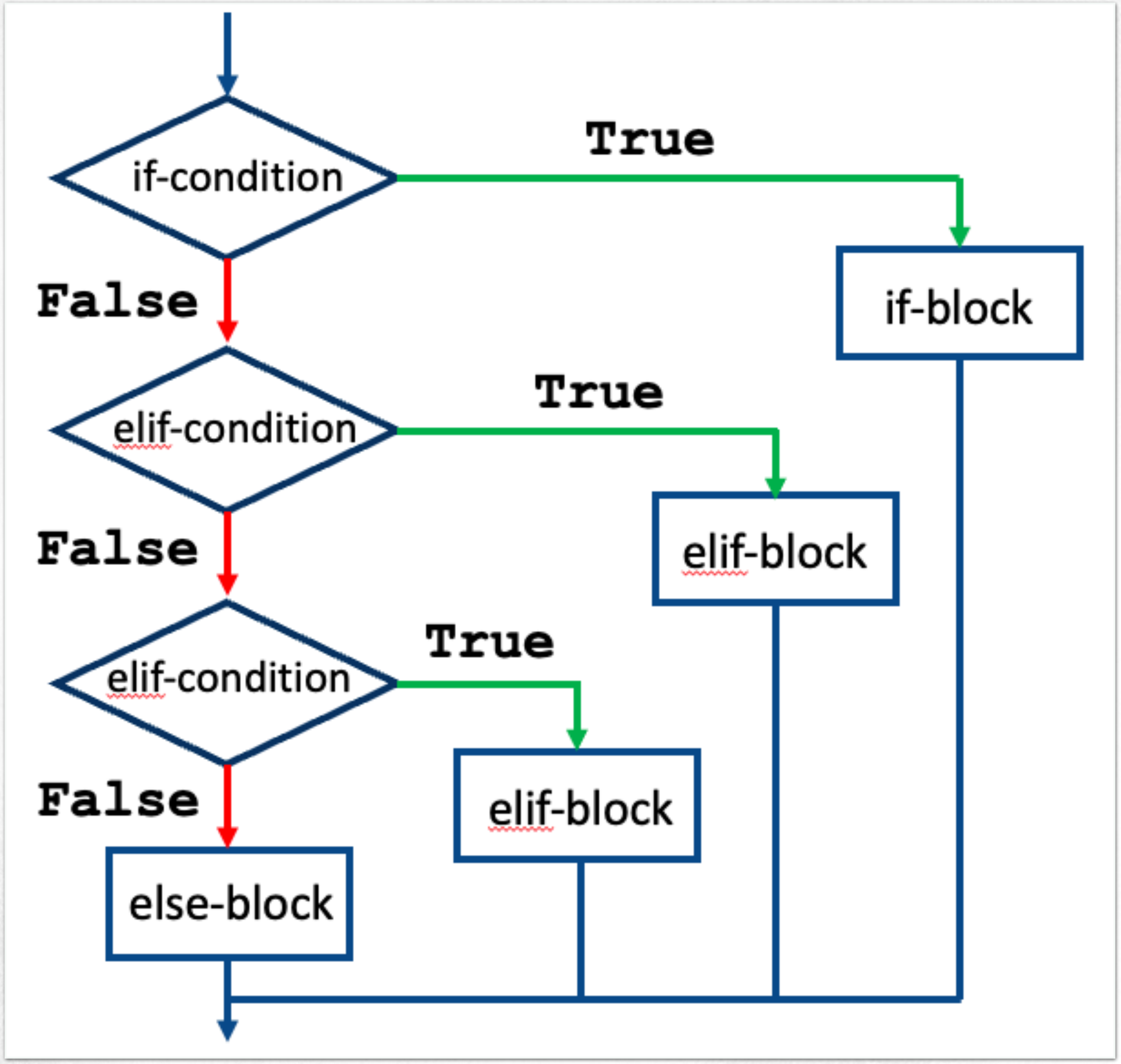
```
compound = input('Enter the compound: ')

if compound == "H2O":
    print("Water")
elif compound == "NH3":
    print("Ammonia")
elif compound == "CH4":
    print("Methane")
else:
    print("Unknown compound")
```

```
>>> Enter the compound: CH4
Methane
>>>
>>> Enter the compound: H2SO4
Unknown compound
>>>
```

Typical if statement and flow chart

```
if <<if-condition>>:
    <<if_block>>
elif <<elif-condition>>:
    <<elif_block>>
elif <<elif-condition>>:
    <<elif_block>>
else:
    <<else_block>>
```



Nested if statements

```
ph = float(input(‘Enter the pH level: ‘))


if 0<= ph <=14:
    if ph < 7.0:
        print(ph, "is acidic.")
    elif ph > 7.0:
        print(ph, "is basic.")
    else:
        print(ph, "is neutral.")
else:
    print("pH value has to be a number between 0 and 14.")
```

Use of Boolean variable

```
if age < 45:
    if bmi < 22.0:
        risk = 'low'
    else:
        risk = 'medium'
else:
    if bmi < 22.0:
        risk = 'medium'
    else:
        risk = 'high'
```

```
young = age < 45
slim = bmi < 22.0
if young:
    if slim:
        risk = 'low'
    else:
        risk = 'medium'
else:
    if slim:
        risk = 'medium'
    else:
        risk = 'high'
```

Use of Boolean variable

```
young = age < 45
slim = bmi < 22.0
if young and slim:
    risk = 'low'
elif young and not slim:
    risk = 'medium'
elif not young and slim:
    risk = 'medium'
elif not young and not slim:
    risk = 'high'
```

```
young = age < 45
slim = bmi < 22.0
if young:
    if slim:
        risk = 'low'
    else:
        risk = 'medium'
else:
    if slim:
        risk = 'medium'
    else:
        risk = 'high'
```

Summary

- if statements control the flow of execution. As with function definitions, the bodies of if statements are indented, as are the bodies of elif and else clauses.

Thank you

경희대학교
KYUNG HEE UNIVERSITY