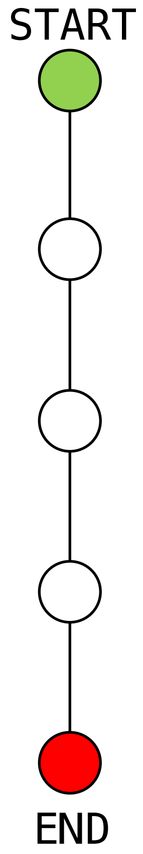
Control Flow: Conditionals

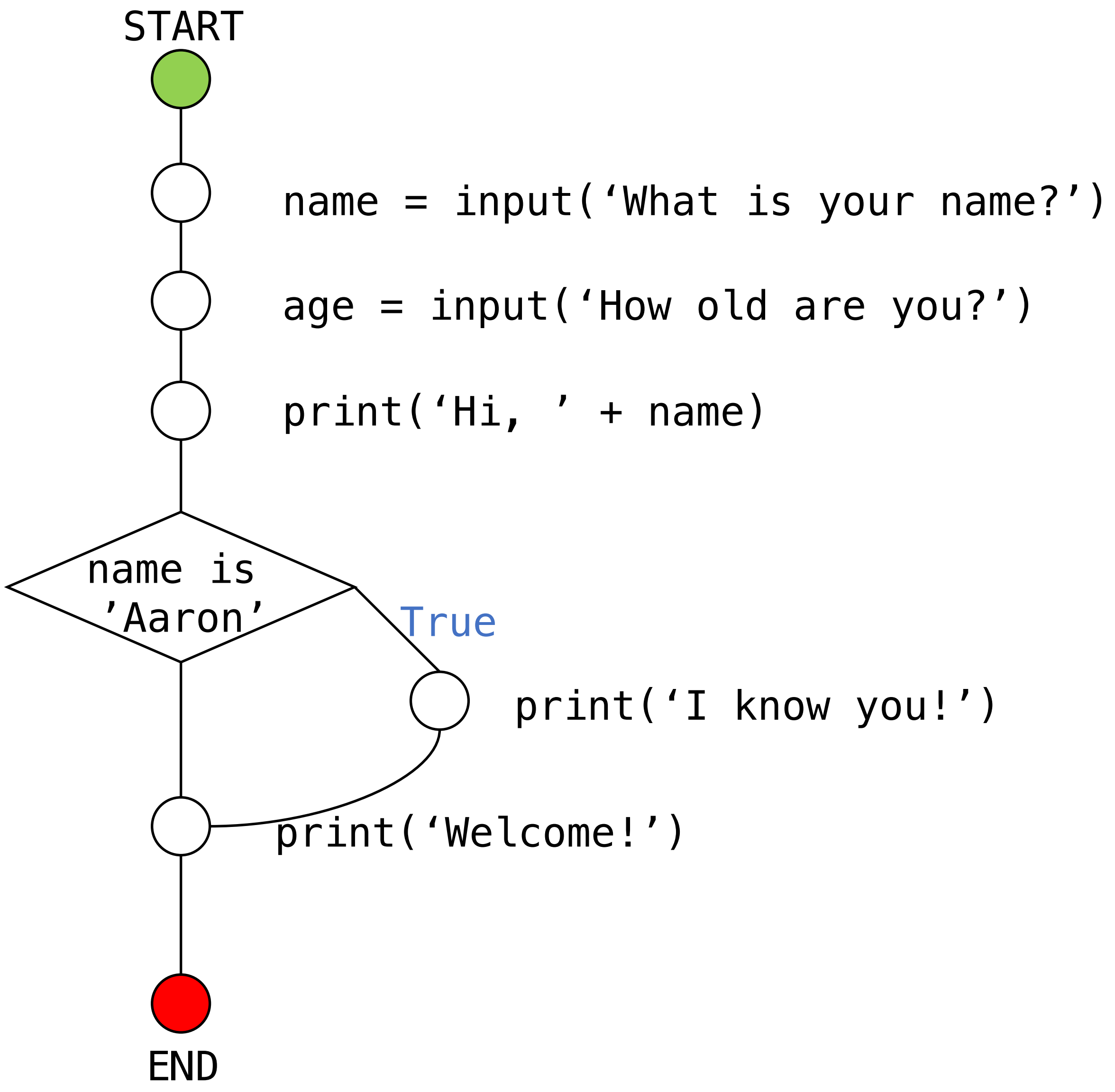
Contents

Sequential Program Flow

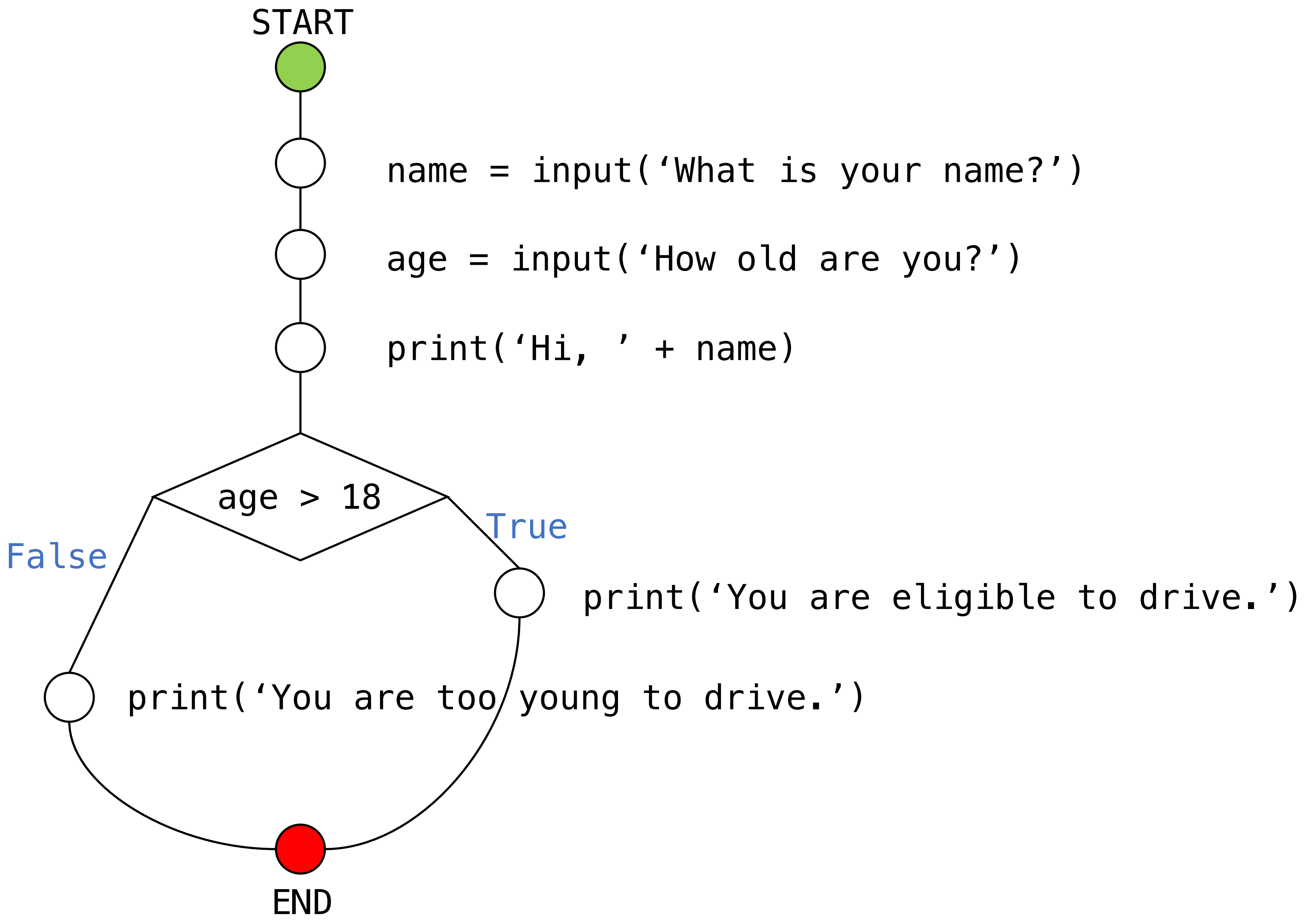


```
name = input(‘What is your name?’)

age = input(‘How old are you?’)

print(‘Hi, ’ + name)
```

Conditional Flow



Conditional Flow



```
name = input('What is your name?')
age = input('How old are you?')
print('Hi, ' + name)
print('You are eligible to drive.')
print('You are too young to drive.')
```

Boolean Data Type

String data type examples:

`‘John’, ‘pink dolphin’, ‘One upon a time…’, etc`

Integer data type examples:

`-49, 0, 5, 1800943, etc`

Float data type examples:

`56.6, 100.06, 0.398, etc`

The boolean data type has only two values, i.e.:

`True, False`

Boolean Data Type

```
privilege_member = True
greeting = ‘Welcome’

if privilege_member:
    greeting = ‘Welcome, privilege member’

print(greeting)
```

Boolean Data Type

```
mammal = True

if mammal:
    text = ‘Mammals are warm-blooded.’
else:
    text = ‘Not a mammal.’

print(text)
```

Boolean Expressions

Boolean expressions evaluate to the boolean value `True` or `False`

```
age < 18
```

Comparison operator

```
age > 18 and eyesight_passed
```

Logical operator

Comparison Operators

Operator	Description	Example	Result
`==`	Equal to	`8 == 9`	`False`
`!=`	Not equal to	`8 != 9`	`True`
`>`	Greater than	`8 > 9`	`False`
`<`	Less than	`8 < 9`	`True`
`>=`	Greater than or equal to	`8 >= 9`	`False`
`<=`	Less than or equal to	`8 <= 9`	`True`

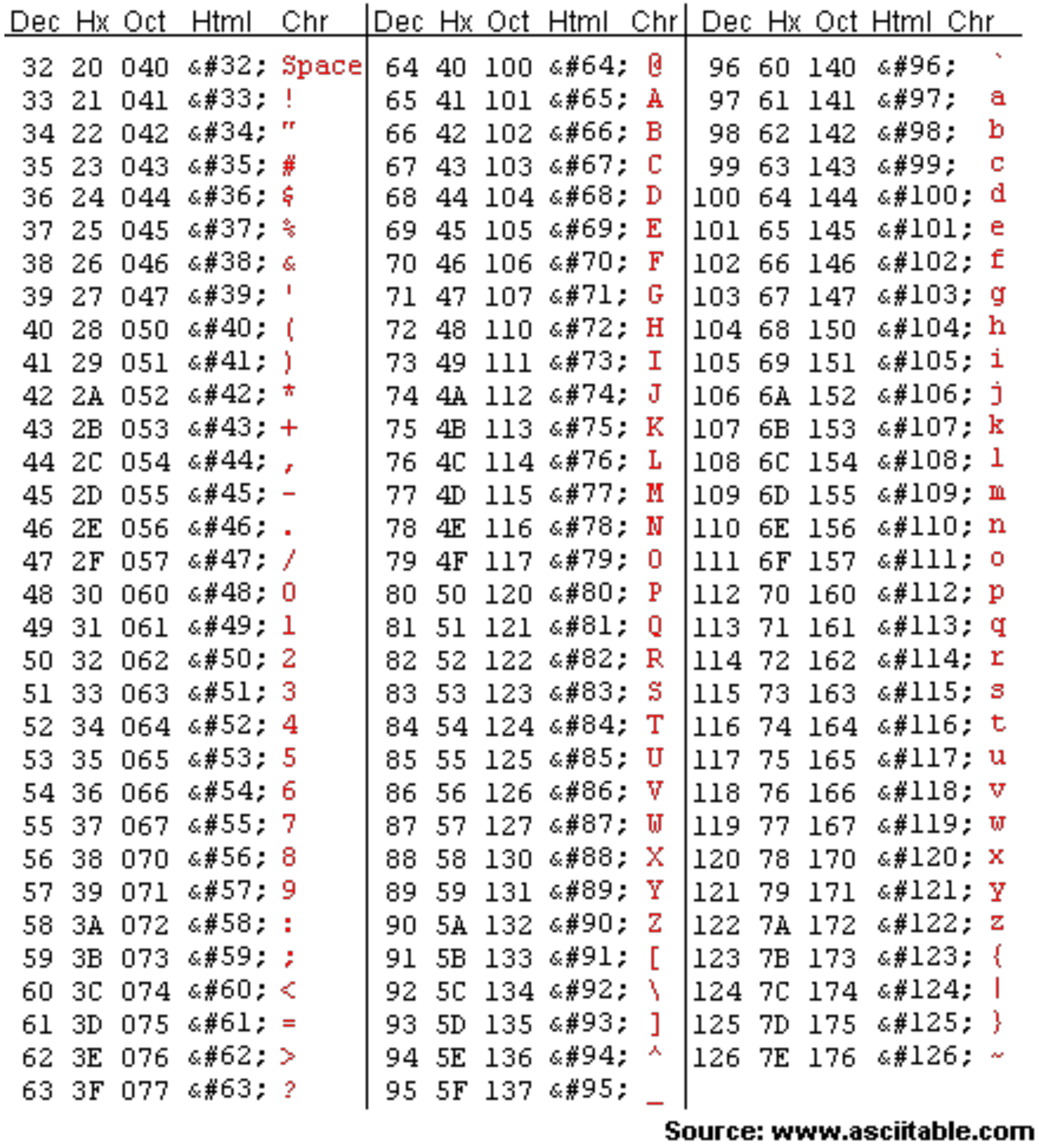
Comparing Strings

Python compares strings lexicographically using the ASCII value of each characters.

Operator	**Example**	**Result**
`==`	`‘pear’ == ‘peace’`	`False`
`!=`	`‘pear’ != ‘peace’`	`True`
`>`	`‘pear’ > ‘peace’`	`True`
`<`	`‘pear’ < ‘peace’`	`False`
`>=`	`‘pear’ >= ‘peace’`	`True`
`<=`	`‘pear’ <= ‘peace’`	`False`

Comparing Strings

A character is a piece of data. In the ASCII standard, each character is represented with a numeric value. The lexicographical order of ASCII characters is as follows (next slide):



Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
63	3F	077	&#63;	?	95	5F	137	&#95;	_					

Source: www.asciitable.com

Logical Operators

Logical operators take boolean expression as an operand. The operand must be an expression that evaluates to a boolean value.

Three logical operators:

- `not`
- `and`
- `or`

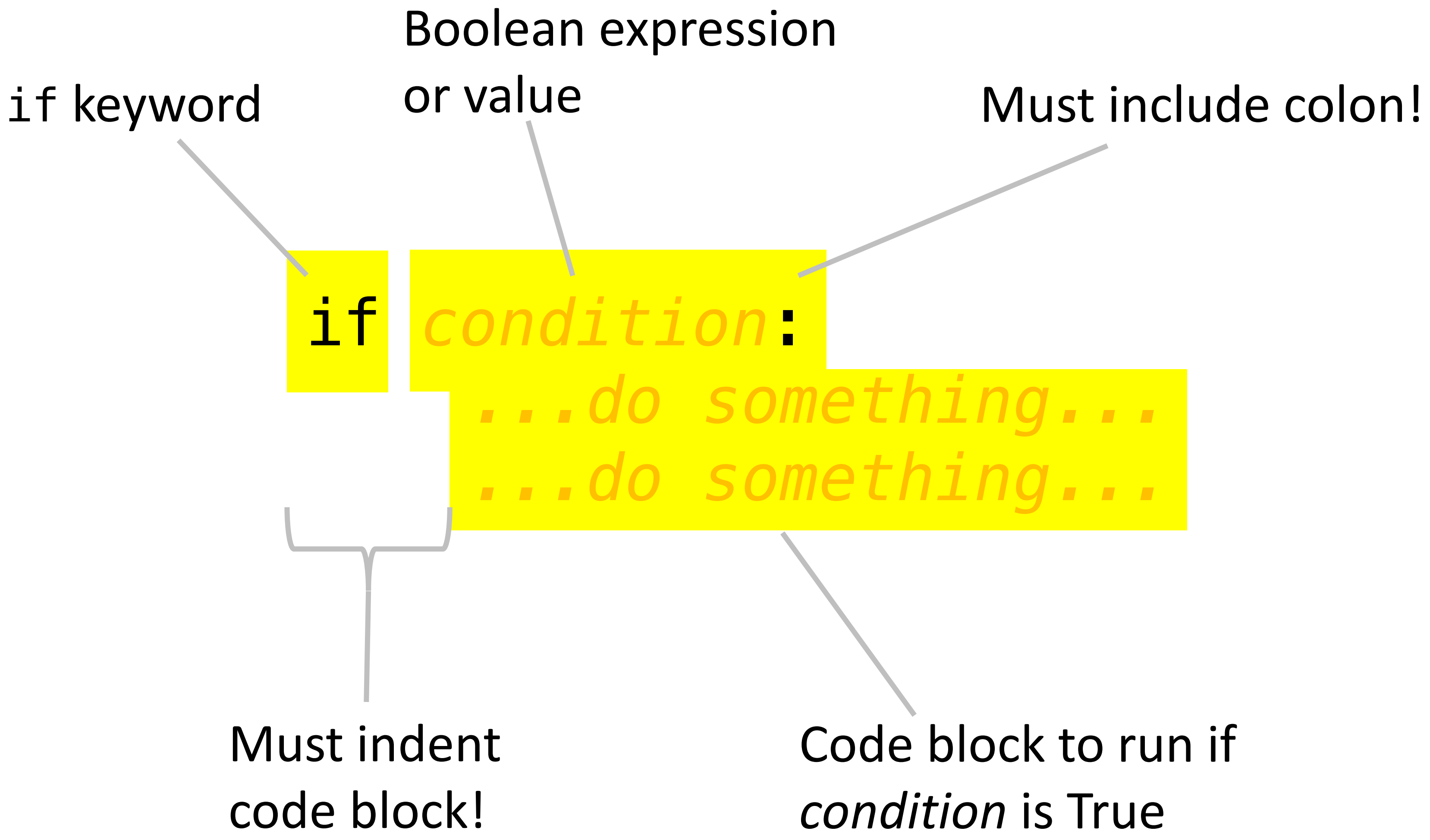
Examples:

```
not privilege_member
age > 10 and age < 20
name == ‘Max’ or name == ‘Maximillian’
```

Truth Table

`x`	`y`	`x and y`	`x or y`	`not x`	`not y`
`True`	`True`	`True`	`True`	`False`	`False`
`True`	`False`	`False`	`True`	`False`	`True`
`False`	`True`	`False`	`True`	`True`	`False`
`False`	`False`	`False`	`False`	`True`	`True`

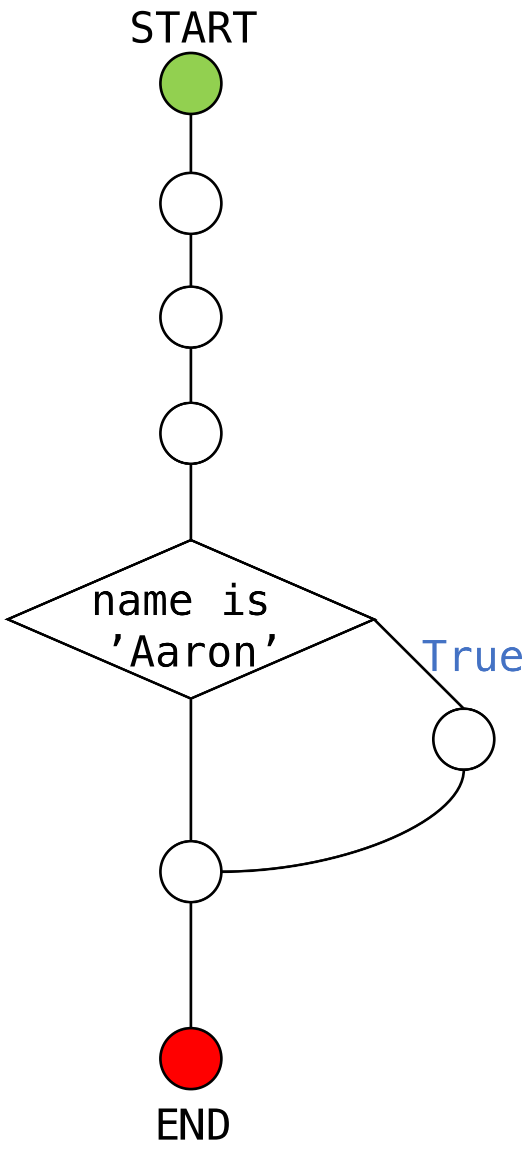
if Statement



```
if condition:
    ...do something...
    ...do something...
```

if keyword
Boolean expression or value
Must include colon!
Must indent code block!
Code block to run if *condition* is True

if Statement



```
name = input(‘What is your name?’)

age = input(‘How old are you?’)

print(‘Hi, ’ + name)

if name == ‘Aaron’:

    print(‘I know you!’)


print(‘Welcome!’)
```

`if` Statement

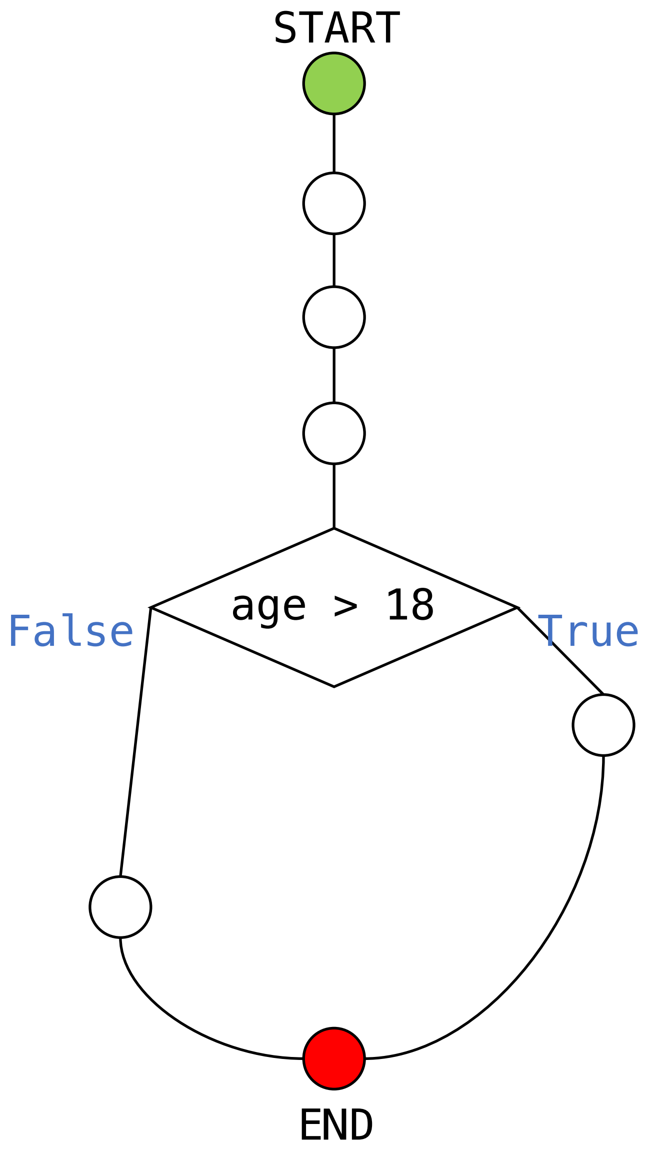
Code block to run if *condition* is True

```
if condition:
    ...do something...
else:
    ...do something else...
```

Must indent code block!

Code block to run if *condition* is False

if Statement



```
name = input('What is your name?')

age = int(input('How old are you?'))

print('Hi, ' + name)


if age > 18:

    print('You are eligible to drive.')

else:

    print('You are too young to drive.')
```

if Statement

```
if condition 1:
    ...do something...
elif condition 2:
    ...do something...
elif condition 3:
    ...do something...
else:
    ...do something else...
```

if Statement

```
age = int(input(‘Enter age: ’))

if age < 11:
    print(’Child’)
elif age < 19:
    print(’Teen’)
else:
    print(‘Adult’)
```

Evaluations of Numbers & Strings as Boolean

- An empty string value is evaluated to False, True otherwise.
- Zero value is valuated to False, True otherwise.

Evaluations of Numbers & Strings as Boolean

```
text = ’’

if text:
    print(‘”’ + text + ’” evaluated to True’)
else:
    print(‘”’ + text + ’” evaluated to False’)
```

Evaluations of Numbers & Strings as Boolean

```
age = -1

if age:
    print(str(age) + ’ evaluated to True’)
else:
    print(str(age) + ’ evaluated to False’)
```

Idiomatic Python

The characteristic of Python is readability. Idioms in programming languages lend to readability.

Instead of:

```
a = True
if a:
    x = 1
else:
    x = 0
```

Idiom:

```
a = True
x = 1 if a else 0
```

Idiomatic Python

Instead of:

```
if a == True:
    # do something
if b == False:
    # do something
```

Idiom:

```
if a:
    # do something
if not b:
    # do something
```

Idiomatic Python

Instead of:

```
if a <= b and b <= c:
    # do something
```

Idiom:

```
if a <= b <= c:
    # do something
```