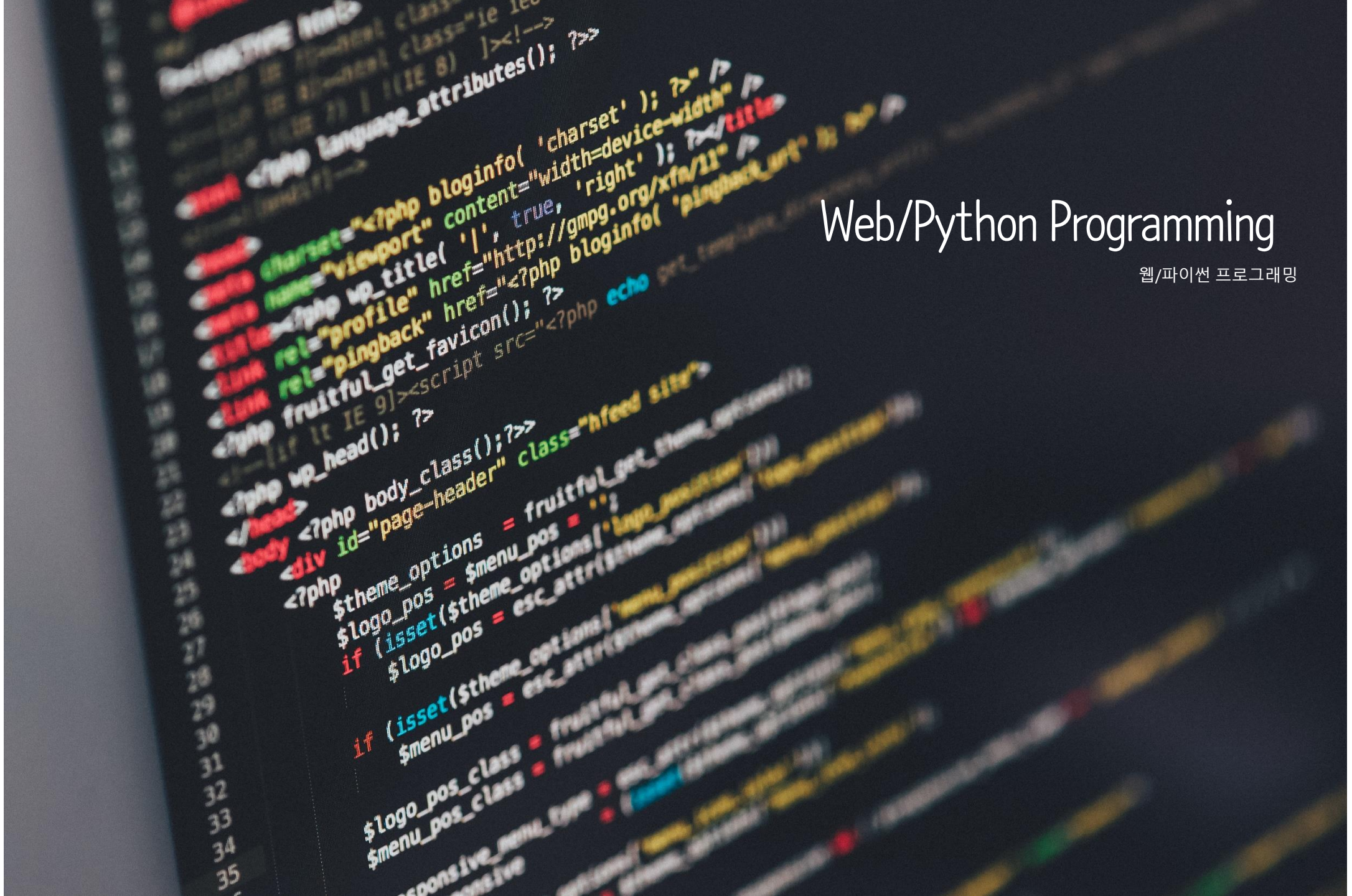

Web/Python Programming

웹/파이썬 프로그래밍

Today

- Review the type `str`
- Type `bool`: A Boolean type
- Boolean operators: and, or, not
- Relational operators: >, <, >=, <=, ==, !=
- Truth table
- Combining comparisons (precedence)
- Short-circuit evaluation
- Comparing strings
- Covers Chapter 5 in your textbook

String

- In Python, text is represented as a string.
- String is a type. (`str`)
- String is a sequence of characters.
- Characters include letters, digits, and symbols.
- Characters include Latin alphabet, 한글, chemical symbols, musical symbols, and much more.

How to define a string?

- Single quotes
- Double quotes

- The opening and closing quotes must match.

```
>>> 'Aristotle'
'Aristotle'
>>> "Issac Newton"
'Issac Newton'
>>> 'Charles Darwin"
SyntaxError: EOL while scanning string literal
>>>
```

Empty string

- ‘’
- “”
- It contains no character.
- It’s not a blank. It’s an empty string.

Cf. How long can a string be?

- Limited only by computer memory.

Operations on strings

- Python built-in functions for string
  - len( ) : returns the length of a string
  - + : concatenates two strings
  - * : repeats and concatenates strings
  - int( ) : converts a string of numbers to integer type
  - float( ) : converts a string of numbers to floating-point type

print( )

```
>>> a = 'one'
>>> a
'one'
>>> print(a)
one
>>> b = 'one₩ntwo₩nthree'
>>> b
'one₩ntwo₩nthree'
>>> print(b)
one
two
three
>>> c = 'one₩ttwo₩nthree₩tfour'
>>> c
'one₩ttwo₩nthree₩tfour'
>>> print(c)
one     two
three   four
```

input( )

```
>>> species = input()
Homo sapiens
>>> species
'Homo sapiens'
>>> population = input()
6973738433
>>> population
'6973738433'
>>> type(population)
<class 'str'>
```

```
>>> population = input()
6973738433
>>> population
'6973738433'
>>> population = int(population)
>>> population
6973738433
>>> population = population + 1
>>> population
6973738434
```

```
>>> species = input("Please enter a species: ")
Please enter a species: Python curtus
>>> print(species)
Python curtus
```

```
>>> population = int(input())
6973738433
>>> population = population + 1
6973738434
```

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

How to use Booleans?

```
def is_positive(x):
    """
    (number) -> bool
    Return True iff x is positive.

    >>> is_positive(3)
    True

    >>> is_positive(-4.6)
    False

    >>> is_positive(0.0)
    False
    """
    return x > 0
```

```
 RESTART: C:/Users/jiyoung/AppData/Local/Programs
/Python/Python35/Scripts/test04.py
>>> is_positive(3)
True
>>> is_positive(4.6)
True
>>> is_positive(-4.6)
False
>>> is_positive(0)
False
```

a	b	a != b (a XOR b)
True	True	False
False	False	False
True	False	True
False	True	True

Combining comparisons

```
>>> x = 2
>>> y = 5
>>> z = 7
>>> x < y and  y < z
True
>>> (x < y) and (y < z)
True
>>> (1 < x) and (x <= 5)
True
>>> (1 < z) and (z <= 5)
False
```

```
>>> x = 3
>>> 1 < x <= 5
True
>>> 3 < 5 != True
True
>>> 3 < 5 != False
True
>>> (3 < 5) and (5 != True)
True
>>> (3 < 5) and (5 != False)
True
```

Numbers and Strings with Booleans

- Numbers
  - 0 and 0.0 are treated as False
  - All other numbers are True

```
>>> not 0
True
>>> not 1
False
>>> not -34.2
False
```

- Strings
  - Empty string is treated as False
  - All other strings are True

```
>>> not ''
True
>>> not ""
True
>>> not 'bad'
False
```

- `None` is treated as False

Short-circuit Evaluation

- When Python evaluates an expression containing `or`, if the first operand is `True`, Python doesn't evaluate the second operand.

- When Python evaluates an expression containing `and`, if the first operand is `False`, Python doesn't evaluate the second operand.

```
>>> 1/0
Traceback (most recent call last):
  File "<pyshell#34>", line 1, in <module>
    1/0
ZeroDivisionError: division by zero
>>> (2<3) or (1/0)
True
```

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		Space	64	40	100	@	@	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	0	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	&#127;	DEL

Source: www.LookupTables.com

Comparing Strings

- Lexicographically

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

- Checks whether one string appears inside another one:

```
>>> '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
```

#case sensitive!!

#empty string is always a substring of every string