오늘의 강의 목표

- String에 대한 이해
- String indexing과 slicing에 대한 이해
- String operator들에 대한 이해
- String formatting에 대한 이해
- Character coding에 대한 이해
- String method들에 대한 이해

Python String Type

- Single quote (') 혹은 double quote (") 로 표현

```
>>> s1 = 'Hello World'
>>> s2 = "Goodbye World"
>>> print(s1 + ". " + s2 + ".")
Hello World. Goodbye World.
```

- '와 " 차이는 없으나...

```
>>> s3 = 'I'm happy'
SyntaxError: invalid syntax
>>> s4 = "I'm happy"
>>>
```

- 여러 행에 걸친 경우 (''' 혹은 """ 이용)

```
>>> s5 = """Light of the moon
Moves west, flowers' shadows
Creep eastward."""
```

String Indexing

- String은 character의 연속
- 각 character는 index 값을 가짐
  - 좌측부터 시작할 경우 0부터 시작
  - 우측부터 시작할 경우 -1부터 시작

index:	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
	0	1	2	3	4	5	6	7	8	9	10
string:	H	e	l	l	o		W	o	r	l	d

String Indexing

index:	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
	0	1	2	3	4	5	6	7	8	9	10
string:	H	e	l	l	o		W	o	r	l	d

```
>>> s = "Hello World"
>>> print(s[0])
H
>>> print(s[1])
e
>>> print(s[-1])
d
>>> print(s[-2])
l
```

String Slicing

index:	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
	0	1	2	3	4	5	6	7	8	9	10
string:	H	e	l	l	o		W	o	r	l	d

```
>>> s = "Hello World"
>>> print(s[0:2])
He
>>> print(s[3:5])
lo
>>> print(s[-10:-7])
ell
```

```
>>> print(s[:]) # 전체
Hello World
>>> print(s[:3]) # 시작생략
Hel
>>> print(s[6:]) # 끝생략
World
```

More on string slicing at
http://www.pythoncentral.io/cutting-and-slicing-strings-in-python/

String Length

index:	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
	0	1	2	3	4	5	6	7	8	9	10
string:	H	e	l	l	o		W	o	r	l	d

```
>>> a = "Hello World"
>>> len(a)    # String의 Length를 리턴하는 Function
11
```

Practice

- 아래와 같이 동작하는 프로그램을 작성하세요

```
Enter a string: Hello
Hello
ello
llo
lo
o
H
He
Hel
Hell
Hello
```

String Operators

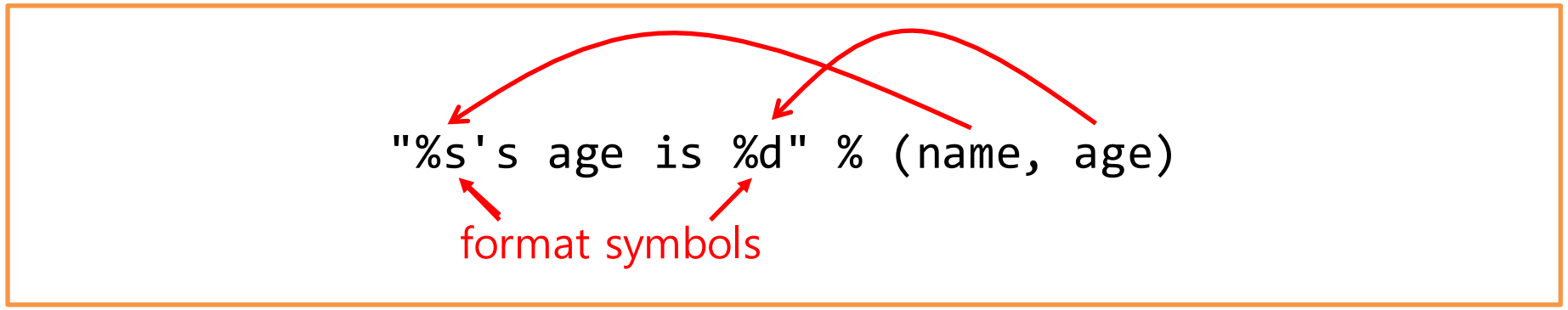
Operator	Description
x in s	s가 x를 포함하면 True, 아니면 False
s not in s	s가 x를 포함하면 False, 아니면 True
s + t	이어 붙이기
s * n or n * s	s를 n번 반복
s[i]	i번째 character
s[i:j]	i번째에서 j번째까지의 slice
s[i:j:k]	i번째에서 j번째까지의 slice (단, step size가 k)

String Operators (+, *, in, not in)

```
>>> a = "Hello"
>>> b = "World"
>>> c = a + " " + b
>>> print(c)
Hello World
>>> c = a * 3 + b * 3
>>> print(c)
HelloHelloHelloWorldWorldWorld
```

```
>>> print("H" in a)
True
>>> print("H" not in a)
False
```

String Formatting




```
>>> name = "Kim"
>>> age = 38
>>> print("%s's age is %d" % (name, age))
Kim's age is 38
```

- 아래는 + operator를 이용하는 기존 방법

```
>>> name = "Kim"
>>> age = 38
>>> print(name + "'s age is " + str(age))
Kim's age is 38
```

String Format Symbols

Format Symbol	Description
%c	Character (integer 혹은 single character string)
%s	String
%i or %d	10진수
%o	8진수
%x	16진수 (소문자 a-f)
%X	16진수 (대문자 A-F)
%e	지수 표현 (소문자 e)
%E	지수 표현 (대문자 E)
%f or %F	실수
%g	%f와 %e 중에 짧은 쪽
%G	%F와 %E 중에 짧은 쪽

String Formatting Examples

```
while True:
    var = input("Enter a decimal int (or 'q' to quit): ")
    if var == 'q':
        break
    var = int(var)
    print("Dec %d = Hex %x = Oct %o" % (var, var, var))
```

Format Symbols

변수 var의 값은 결국 메모리에 이진수로 저장됨. %d, %x, %o는 이진수로 저장된 변수 var의 값을 어떤 형태(10진수, 16진수, 8진수)로 표현할 것인지를 결정.

String Formatting Examples

```
>>> pi = 3.141592
>>> print("pi = %f" % (pi))
pi = 3.141592
>>> print("pi = %e" % (pi))
pi = 3.141592e+00
>>> print("pi = %d" % (pi))
pi = 3
```

%d를 사용했기 때문에 소수점 이하 잘림

More on String Formatting

- 이 외에도 자리 수 지정, zero-padding, 부호 표시 등 더 상세한 formatting 가능

```
>>> a = 39
>>> print("%5d" % (a))     # 5자리
   39
>>> print("%05d" % (a))    # Zero-padding
00039
>>> print("%+05d" % (a))   # 부호 출력
+0039
```

More on string formatting at:
https://docs.python.org/3/library/stdtypes.html#printf-style-string-formatting

컴퓨터는 어떻게 문자를 저장할까?

- 컴퓨터는 이진수밖에 저장하지 못함
- 모든 문자에 대응하는 숫자를 할당 (코드 값)
- 초기에는 0에서 127까지의 숫자만 사용 (ASCII)
  - 한 글자 저장에 1 byte (0 ~ 255)면 충분
- 더 다양한 문자 지원을 위해 Unicode로 확장
  - 한 글자 저장에 2 or 3 bytes 까지도 필요

결론 : 컴퓨터에서 문자는 숫자임

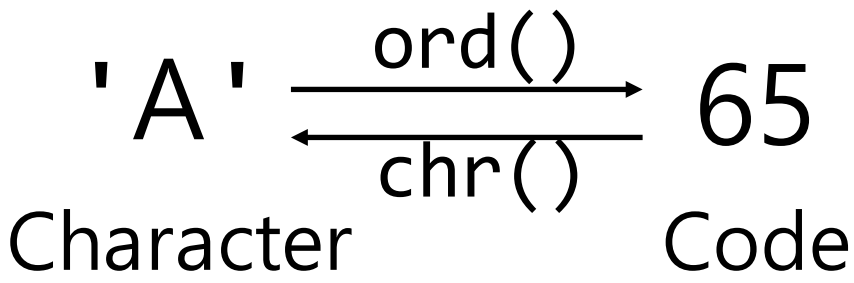
More on character coding at
http://www.joelonsoftware.com/articles/Unicode.html

ASCII Code Table

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

ASCII Code Conversion Functions




```
>>> ord("A")
65
>>> ord('A')
65
>>> chr(65)
'A'
```

ASCII Code Example

- String formatting을 이용한 code to character 변환

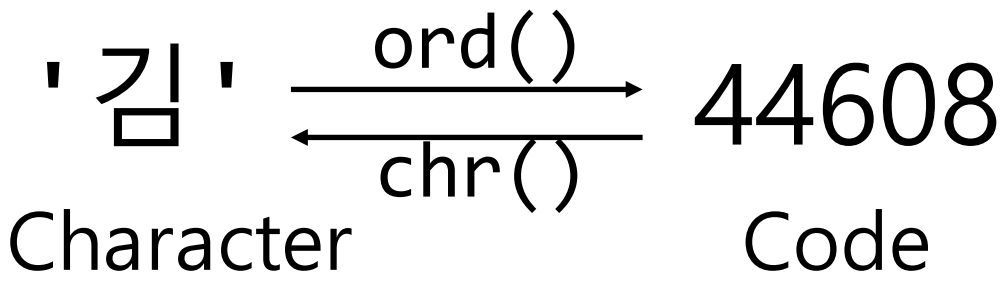
```
>>> for i in range(32, 127):
      print("%c's ascii code is %d" % (i, i))

 's ascii code is 32
!'s ascii code is 33
"'s ascii code is 34
…
~'s ascii code is 126
```

Unicode Table

기본적으로 ASCII Table의 확장임.

Visit http://unicode-table.com

Unicode Conversion Functions




```
>>> ord('김')
44608
>>> chr(44608)
'김'
```

Unicode Example

- String formatting을 이용한 code to character 변환

```
>>> for i in range(44608, 44708):
      print("%c's unicode is %d" % (i, i))



김's unicode is 44608
깁's unicode is 44609
깂's unicode is 44610
…
꺟's unicode is 44707
```

String Methods

- str.capitalize()
- str.casefold()
- str.center()
- str.count()
- str.encode()
- str.endswith()
- str.expandtabs()
- str.find()
- str.format()
- str.format_map()
- str.index()
- str.isalnum()
- str.isalpha()
- str.isdecimal()
- str.isdigit()
- str.isidentifier()
- str.islower()
- str.isnumeric()
- str.isprintable()
- str.isspace()
- str.istitle()
- str.isupper()
- str.join()
- str.ljust()
- str.lower()
- str.lstrip()
- str.maketrans()
- str.partition()
- str.replace()
- str.rfind()
- str.rindex()
- str.rjust()
- str.rpartition()
- str.rsplit()
- str.rstrip()
- str.split()
- str.splitlines()
- str.startswith()
- str.strip()
- str.swapcase()
- str.title()
- str.translate()
- str.upper()
- str.zfill()

More at
https://docs.python.org/3/library/stdtypes.html#string-methods

String Methods Examples

- str.capitalize() : 첫 글자는 대문자로, 나머지는 소문자로 변환

```
>>> a = "you are RIGHT."
>>> a.capitalize()
'You are right.'
```

- str.count(…) : 특정 substring 카운트

```
>>> a = "To be, or not to be, that is the question"
>>> a.count("be")
2
>>> a.count("o")
5
>>> a.count("th")
2
```

String Methods Examples

- str.find() : substring의 첫 번째 index 리턴. 없으면 -1 리턴

```
>>> a = "I am a boy"
>>> a.find("am")
2
>>> a.find("boy")
7
>>> a.find("You")
-1
```

- str.replace(…) : 특정 substring을 변경

```
>>> a = "I am a boy"
>>> a = a.replace("I", "You")
>>> a = a.replace("am", "are")
>>> print(a)
You are a boy
```

String Methods Examples

- str.isalpha(…): 모두 알파벳 문자인가?

```
>>> a = "abcdefg"
>>> a.isalpha()
True
>>> a = "abc505"
>>> a.isalpha()
False
```

- str.isalnum(…): 모두 알파벳이나 숫자인가?

```
>>> a = "abc505"
>>> a.isalnum()
True
>>> a = "abc 505"
>>> a.isalnum()
False
```

String Methods Examples

- str.islower(…) : 모두 소문자인가?
- str.isupper(…) : 모두 대문자인가?
- str.isdecimal(…) : 모두 (10진수) 숫자인가?
- str.isspace(…) : 모두 공백인가?

String Methods Examples

- str.split() : Split string into substrings

```
>>> a = "I am a boy"
>>> a.split()
['I', 'am', 'a', 'boy']
>>> a = "10 20 30 40 50"
>>> a.split()
['10', '20', '30', '40', '50']
>>> a = "10,20,30,40,50"
>>> a.split(',')    # Comma(,)를 seperator로 사용
['10', '20', '30', '40', '50']
```

Raw String

- " 앞에 r 혹은 R을 붙여서 만듬
- Escape character를 문자 그대로 해석

```
>>> a = "note: \n is a newline character"
>>> b = r"note: \n is a newline character"
>>> print(a)
note:
 is a newline character
>>> print(b)
note: \n is a newline character
```

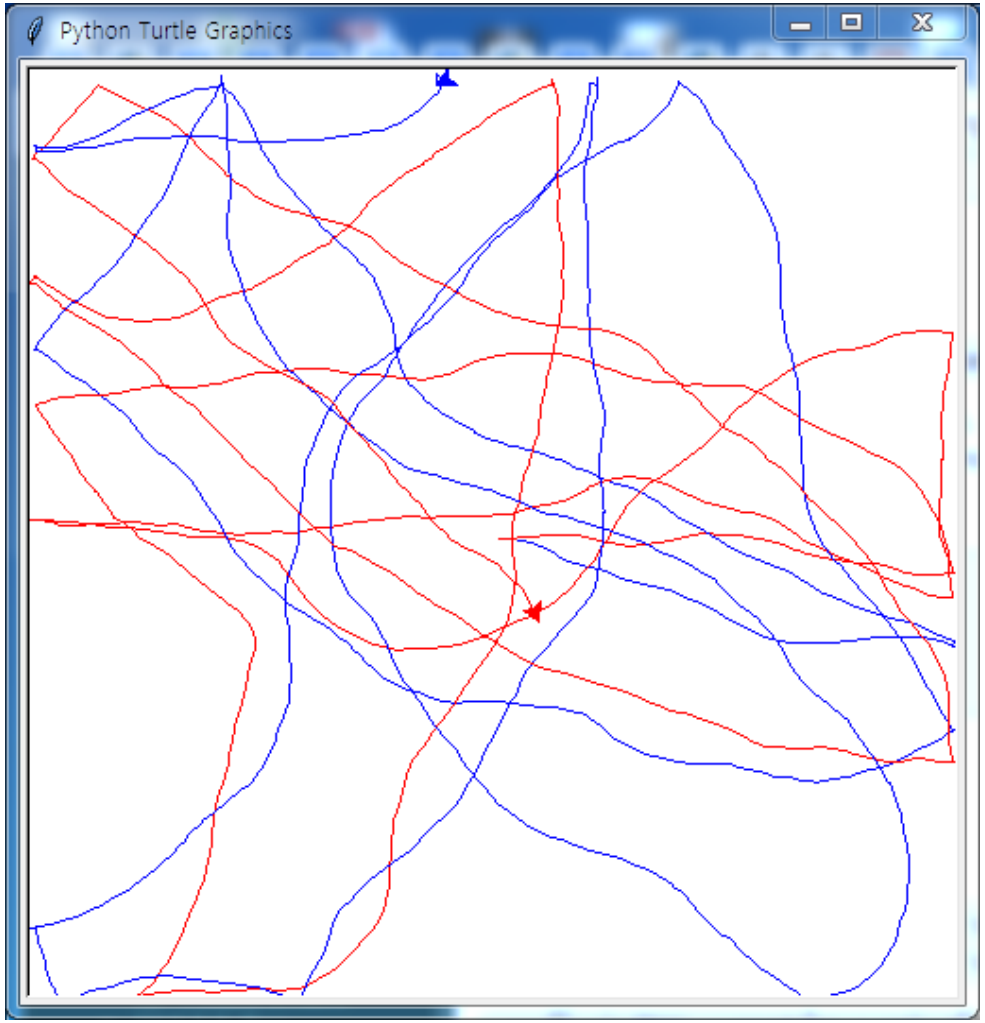
Practice

- 패스워드 검증
  - 최소 길이 10
  - Login id 포함하면 안됨
  - 대문자 1개, 소문자 1개, 숫자 1개 포함해야 함

```
Enter login id: kookmin
Enter password: haha1004
Too short
Enter password: haha1004kookmin
Contains login id
Enter password: haha1004bingo
Needs uppercase and lowercase and numeric characters
Enter password: Haha1004Bingo
Your password is perfect!
```

HW

- 두 개의 Turtle 생성 (Red, Blue 컬러)
- 두 개의 Turtle이 무작위로 움직이도록 함



Questions