DS2500 – Intermediate Programming with Data

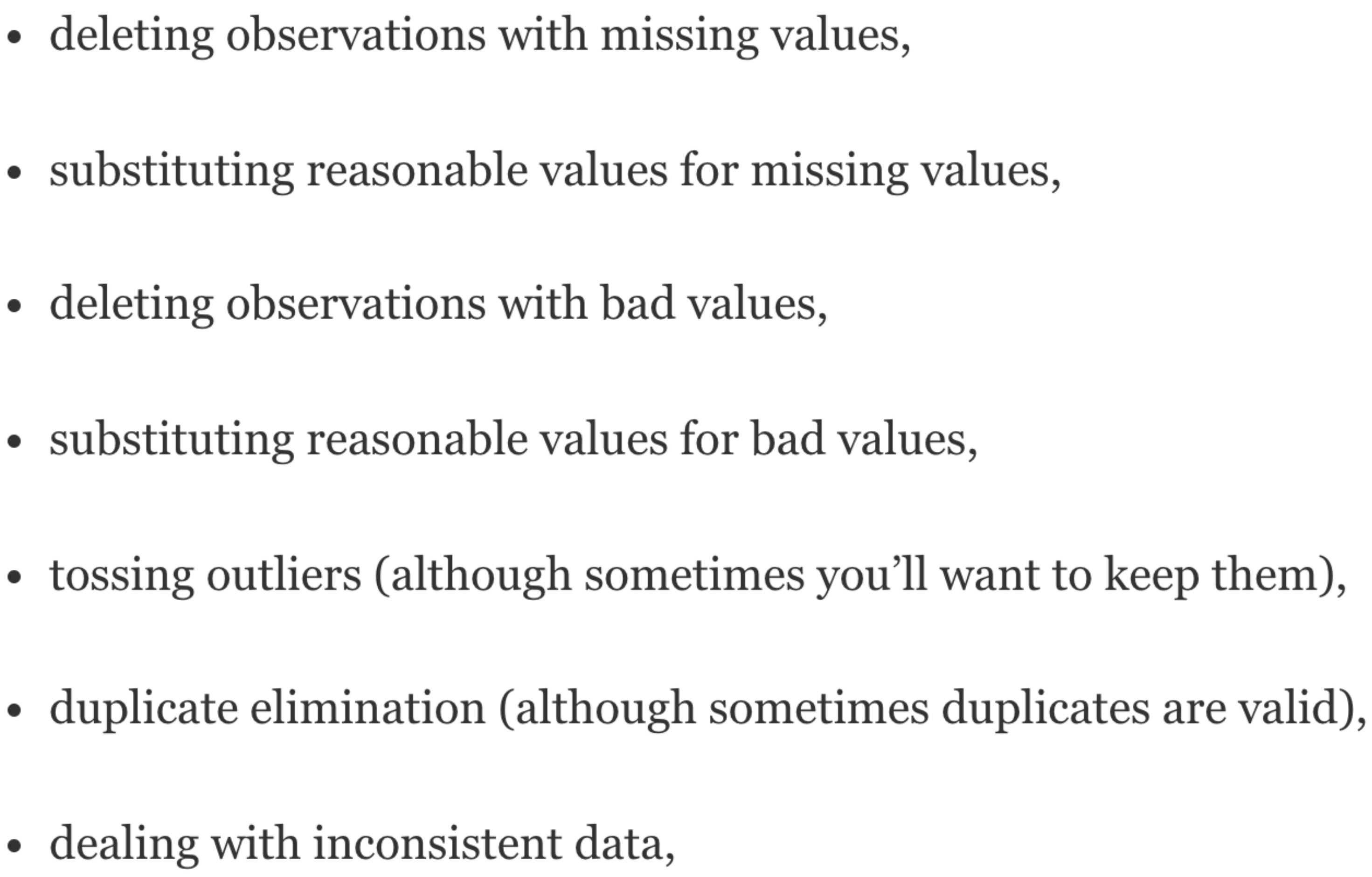
08. Strings and Regular Expressions

John Rachlin, PhD
Northeastern University

Northeastern University

Data Munging (Data Wrangling)

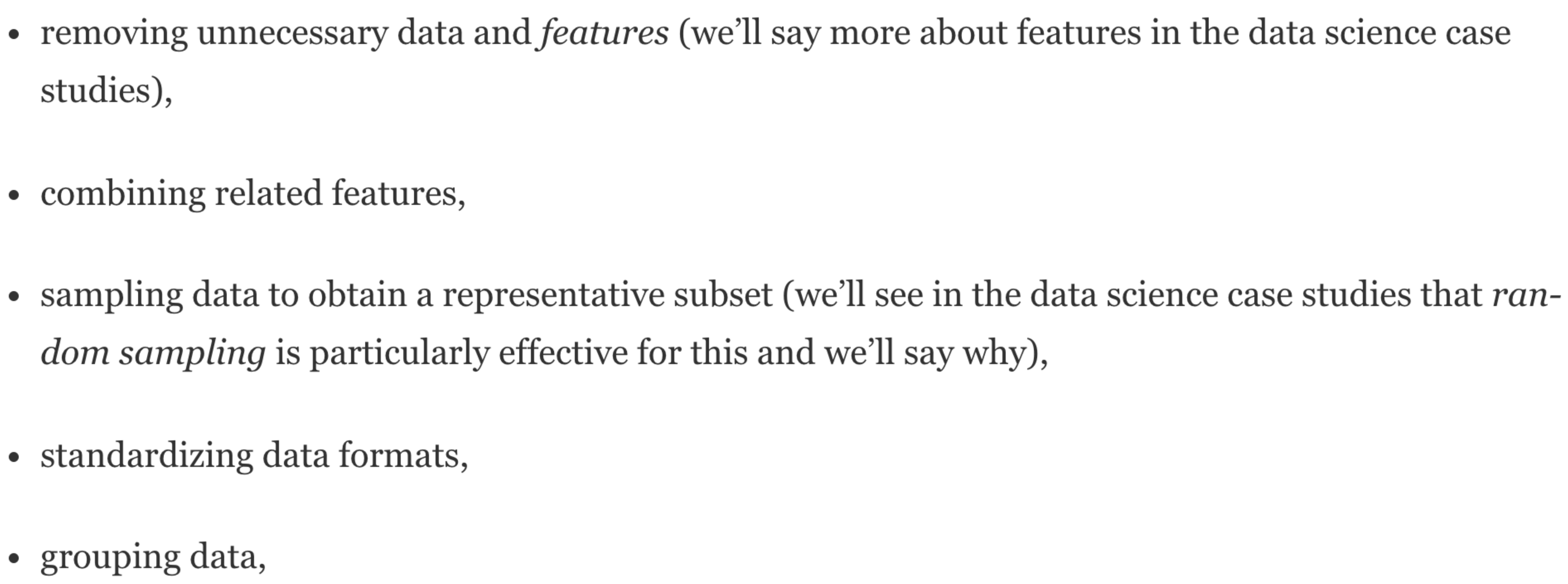
a) Data Cleaning



- deleting observations with missing values,
- substituting reasonable values for missing values,
- deleting observations with bad values,
- substituting reasonable values for bad values,
- tossing outliers (although sometimes you’ll want to keep them),
- duplicate elimination (although sometimes duplicates are valid),
- dealing with inconsistent data,

Data Munging (Data Wrangling)

b) Transformations

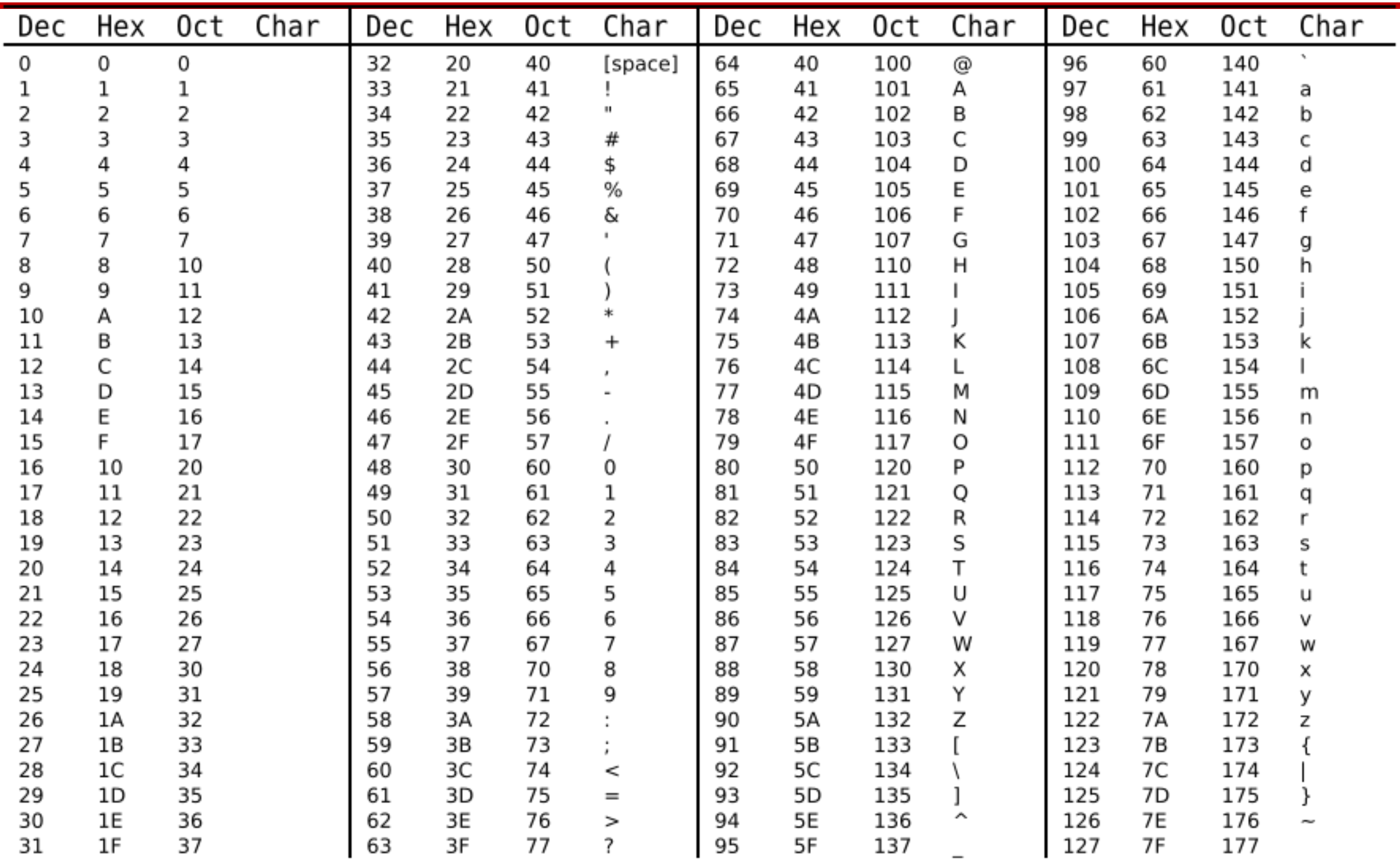


- removing unnecessary data and *features* (we'll say more about features in the data science case studies),
- combining related features,
- sampling data to obtain a representative subset (we'll see in the data science case studies that *random sampling* is particularly effective for this and we'll say why),
- standardizing data formats,
- grouping data,

Applications

Anagrams
Automated grading of written homework
Automated teaching systems
Categorizing articles
Chatbots
Compilers and interpreters
Creative writing
Cryptography
Document classification
Document similarity
Document summarization
Electronic book readers
Fraud detection
Grammar checkers
Inter-language translation
Legal document preparation
Monitoring social media posts
Natural language understanding
Opinion analysis
Page-composition software
Palindromes
Parts-of-speech tagging
Project Gutenberg free books
Reading books, articles, documentation and absorbing knowledge
Search engines
Sentiment analysis
Spam classification
Speech-to-text engines
Spell checkers
Steganography
Text editors
Text-to-speech engines
Web scraping
Who authored Shakespeare's works?
Word clouds
Word games
Writing medical diagnoses from x-rays, scans, blood tests
and many more…

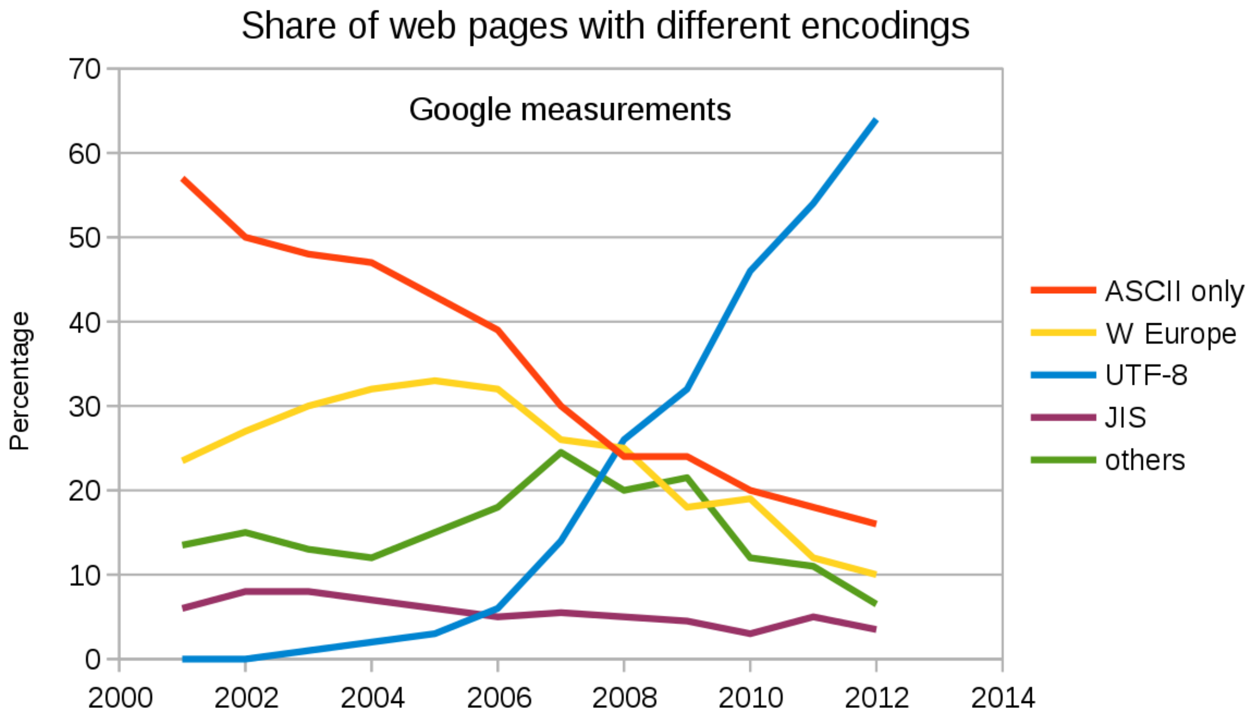
ASCII Table (American Standard Code for Information Interchange)



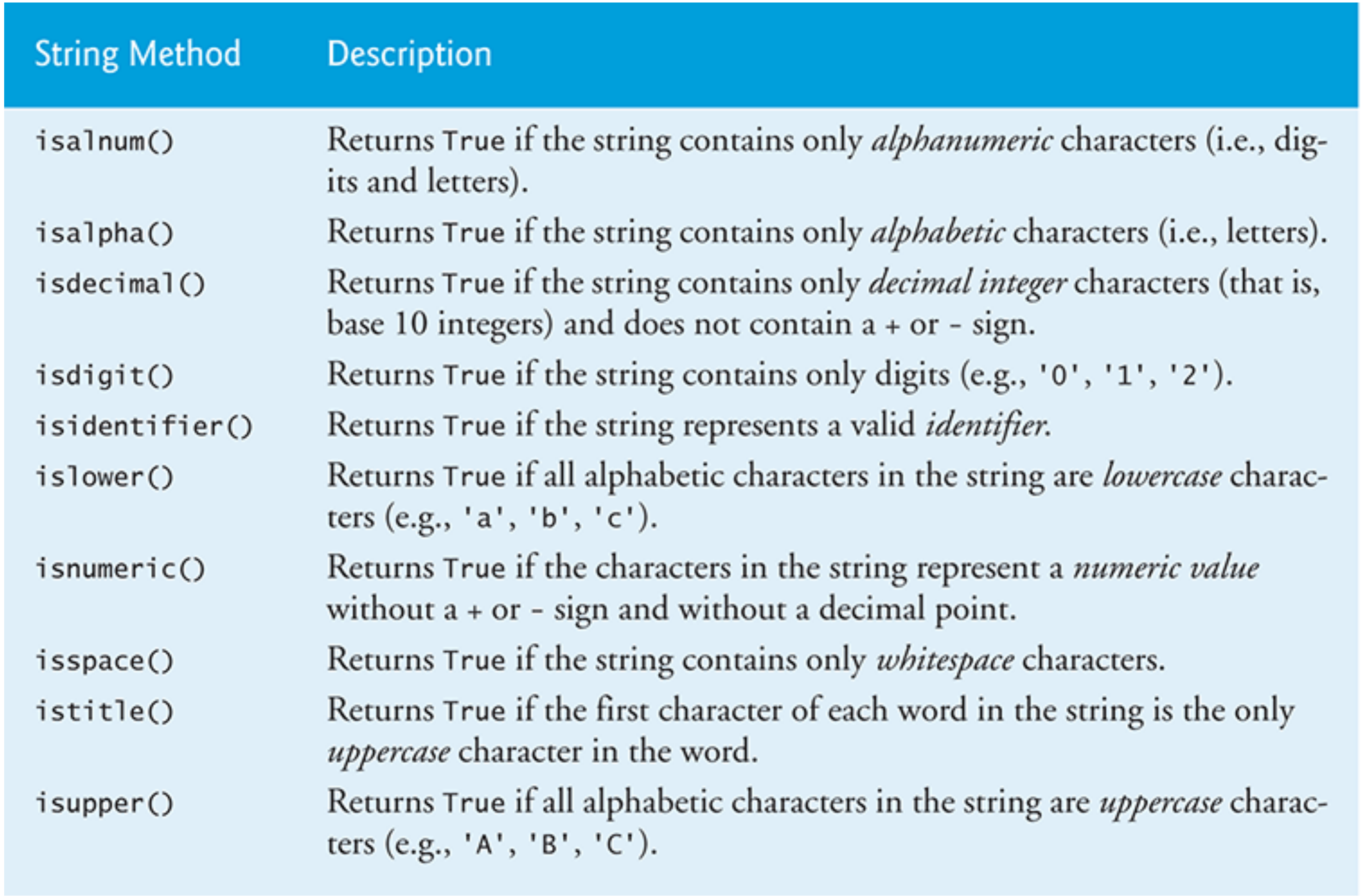
Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Character Encoding

Name	Size	Characters
ASCII	7 bits	128
Extended ASCII	8 bits	256
UTF-8 (Unicode)	8-32 bits	1,112,064



Character testing methods (validation)



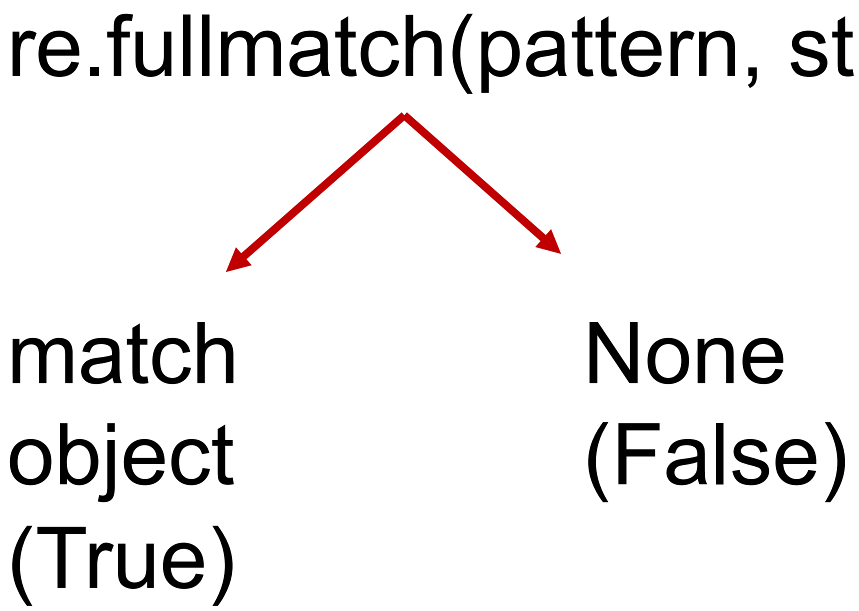
String Method	Description
`isalnum()`	Returns `True` if the string contains only *alphanumeric* characters (i.e., digits and letters).
`isalpha()`	Returns `True` if the string contains only *alphabetic* characters (i.e., letters).
`isdecimal()`	Returns `True` if the string contains only *decimal integer* characters (that is, base 10 integers) and does not contain a + or - sign.
`isdigit()`	Returns `True` if the string contains only digits (e.g., `'0'`, `'1'`, `'2'`).
`isidentifier()`	Returns `True` if the string represents a valid *identifier*.
`islower()`	Returns `True` if all alphabetic characters in the string are *lowercase* characters (e.g., `'a'`, `'b'`, `'c'`).
`isnumeric()`	Returns `True` if the characters in the string represent a *numeric value* without a + or - sign and without a decimal point.
`isspace()`	Returns `True` if the string contains only *whitespace* characters.
`istitle()`	Returns `True` if the first character of each word in the string is the only *uppercase* character in the word.
`isupper()`	Returns `True` if all alphabetic characters in the string are *uppercase* characters (e.g., `'A'`, `'B'`, `'C'`).

Regex Methods

Name	Return	Description
fullmatch	**match** object or **None**	Match regex pattern to a string that should *entirely* match the pattern.
search	**match** object or **None**	First occurrence of a substring that matches a regular expression. Use group() to return the substring.
findall	**List** of matching strings	Finds every matching substring in a string
finditer	**Iterable** of **match** objects	Works like **findall** but returns a lazy iterable of match objects.

Regular expressions

import re

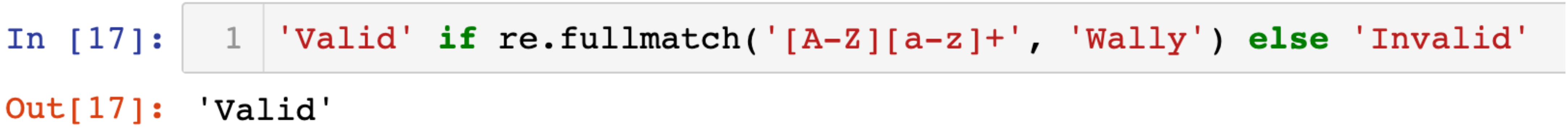


re.fullmatch(pattern, string)

match
object
(True)

None
(False)

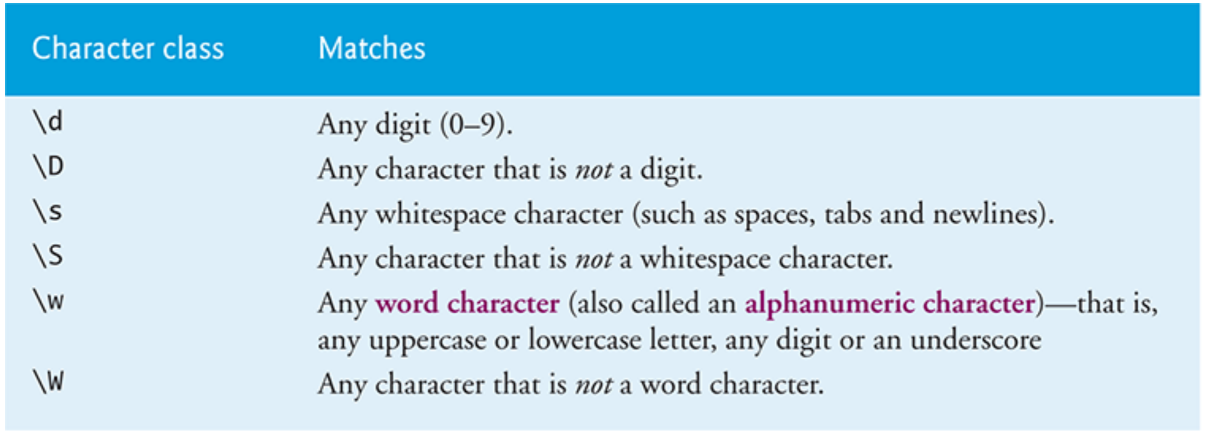
A conditional expression:



```
In [17]: 'Valid' if re.fullmatch('[A-Z][a-z]+', 'Wally') else 'Invalid'

Out[17]: 'Valid'
```

Regular expressions



Character class	Matches
\d	Any digit (0–9).
\D	Any character that is *not* a digit.
\s	Any whitespace character (such as spaces, tabs and newlines).
\S	Any character that is *not* a whitespace character.
\w	Any **word character** (also called an **alphanumeric character**)—that is, any uppercase or lowercase letter, any digit or an underscore
\W	Any character that is *not* a word character.

*	0 or more
+	1 or more
?	0 or 1
{*n*}	n occurrences
{*n*,*m*}	between n and m occurrences
^	not (when inside brackets) or it can mean ‘starting with’
$	‘ending with’
|	or
[]	custom list of characters matching 1 character, e.g. [a-z], [aeiou], [0-5]
()	grouping