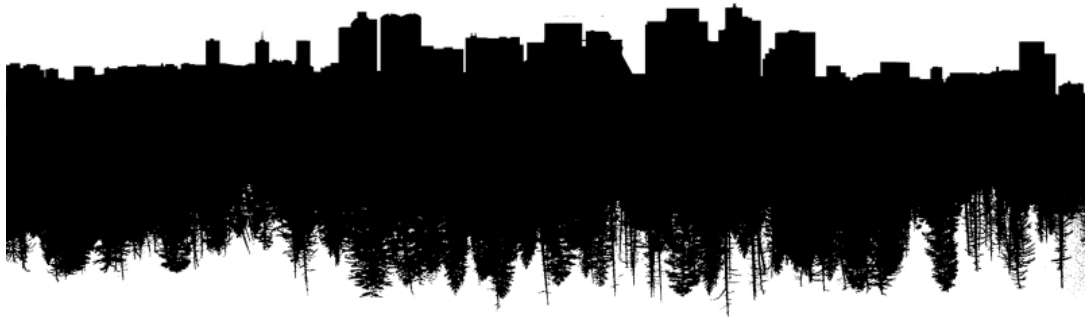
UNDERSTANDING DATA 2

URBAN STUDIES AND PLANNING
DIGITAL URBAN
MONDAY 24TH JANUARY 9:00-12:00

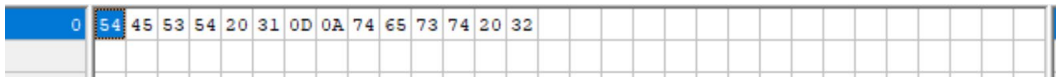
Anssi Joutsiniemi
D.Sc(Tech), Architect
Professor of Practice
Aalto University

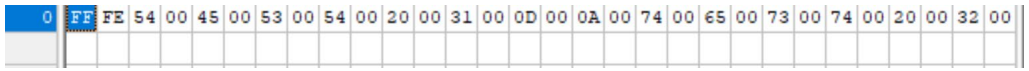


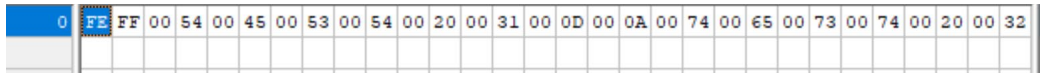
STRINGS & VALUES

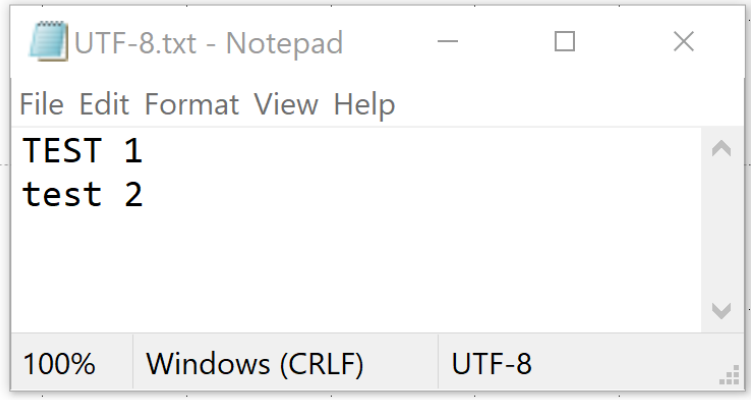
Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	*[NULL]*	32	20	*[SPACE]*	64	40	@	96	60	`
1	1	*[START OF HEADING]*	33	21	!	65	41	A	97	61	a
2	2	*[START OF TEXT]*	34	22	"	66	42	B	98	62	b
3	3	*[END OF TEXT]*	35	23	#	67	43	C	99	63	c
4	4	*[END OF TRANSMISSION]*	36	24	$	68	44	D	100	64	d
5	5	*[ENQUIRY]*	37	25	%	69	45	E	101	65	e
6	6	*[ACKNOWLEDGE]*	38	26	&	70	46	F	102	66	f
7	7	*[BELL]*	39	27	'	71	47	G	103	67	g
8	8	*[BACKSPACE]*	40	28	(	72	48	H	104	68	h
9	9	*[HORIZONTAL TAB]*	41	29	)	73	49	I	105	69	i
10	A	*[LINE FEED]*	42	2A	*	74	4A	J	106	6A	j
11	B	*[VERTICAL TAB]*	43	2B	+	75	4B	K	107	6B	k
12	C	*[FORM FEED]*	44	2C	,	76	4C	L	108	6C	l
13	D	*[CARRIAGE RETURN]*	45	2D	-	77	4D	M	109	6D	m
14	E	*[SHIFT OUT]*	46	2E	.	78	4E	N	110	6E	n
15	F	*[SHIFT IN]*	47	2F	/	79	4F	O	111	6F	o
16	10	*[DATA LINK ESCAPE]*	48	30	0	80	50	P	112	70	p
17	11	*[DEVICE CONTROL 1]*	49	31	1	81	51	Q	113	71	q
18	12	*[DEVICE CONTROL 2]*	50	32	2	82	52	R	114	72	r
19	13	*[DEVICE CONTROL 3]*	51	33	3	83	53	S	115	73	s
20	14	*[DEVICE CONTROL 4]*	52	34	4	84	54	T	116	74	t
21	15	*[NEGATIVE ACKNOWLEDGE]*	53	35	5	85	55	U	117	75	u
22	16	*[SYNCHRONOUS IDLE]*	54	36	6	86	56	V	118	76	v
23	17	*[ENG OF TRANS. BLOCK]*	55	37	7	87	57	W	119	77	w
24	18	*[CANCEL]*	56	38	8	88	58	X	120	78	x
25	19	*[END OF MEDIUM]*	57	39	9	89	59	Y	121	79	y
26	1A	*[SUBSTITUTE]*	58	3A	:	90	5A	Z	122	7A	z
27	1B	*[ESCAPE]*	59	3B	;	91	5B	[	123	7B	{
28	1C	*[FILE SEPARATOR]*	60	3C	<	92	5C	\	124	7C	|
29	1D	*[GROUP SEPARATOR]*	61	3D	=	93	5D	]	125	7D	}
30	1E	*[RECORD SEPARATOR]*	62	3E	>	94	5E	^	126	7E	~
31	1F	*[UNIT SEPARATOR]*	63	3F	?	95	5F	_	127	7F	*[DEL]*

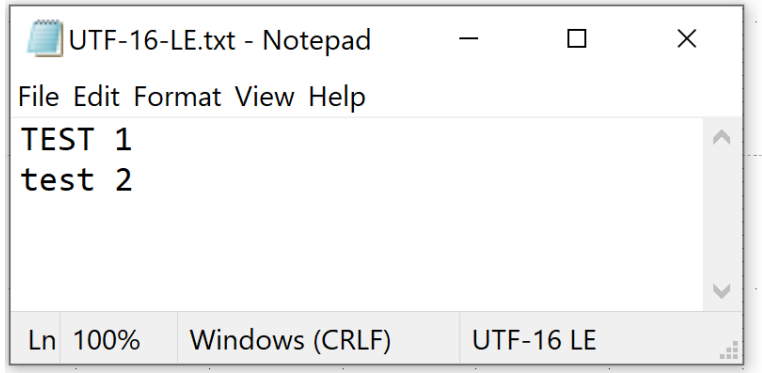
EXAMPLES

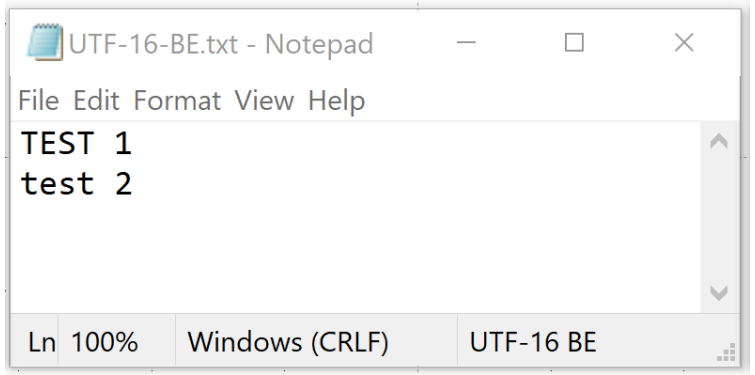

0 54 45 53 54 20 31 0D 0A 74 65 73 74 20 32


0 FF FE 54 00 45 00 53 00 54 00 20 00 31 00 0D 00 0A 00 74 00 65 00 73 00 74 00 20 00 32 00


0 FE FF 00 54 00 45 00 53 00 54 00 20 00 31 00 0D 00 0A 00 74 00 65 00 73 00 74 00 20 00 32


UTF-8.txt - Notepad
File Edit Format View Help
TEST 1
test 2
100%
Windows (CRLF)
UTF-8


UTF-16-LE.txt - Notepad
File Edit Format View Help
TEST 1
test 2
Ln 100%
Windows (CRLF)
UTF-16 LE


UTF-16-BE.txt - Notepad
File Edit Format View Help
TEST 1
test 2
Ln 100%
Windows (CRLF)
UTF-16 BE

DIFFERENCES IN OPERATING SYSTEMS

Coding new line i.e. pressing <ENTER>

Mac OS & Apple II family:	`0D`	(carriage return)
Linux/Unix:	`0A`	(line feed)
Windows:	`0D 0A`	(carriage return + line feed)

Memory storage for data: `90 AB 12 CD`

Little Endian (IBM): DWORD: `CD 12 AB 90` WORD `AB 90` + `CD 12`

(i.e. least significant byte to the most significant byte)

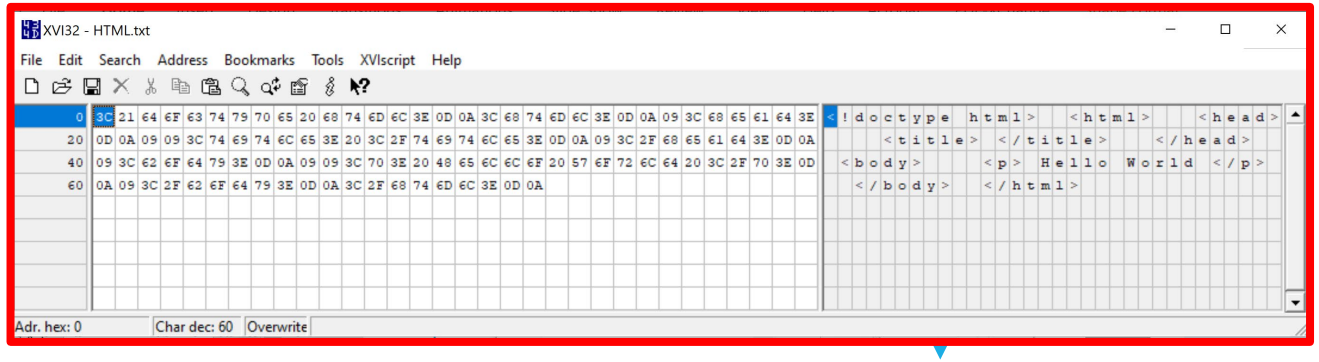
Big Endian (Sun): DWORD: `90 AB 12 CD` WORD `90 AB` + `12 CD`

(i.e. most significant byte to the least significant byte)

Tutorial: https://www.youtube.com/watch?v=T1C9Kj_78ek

TIFF

Tagged Image File Format




```
<!doctype html>
<html>
        <head>
                <title> </title>
        </head>
        <body>
                <p> Hello World </p>
        </body>
</html>
```

SAMPLE FILE:

0	49	49	2A	00	08	00	00	00	19	00	00	01	03	00	01	00	00	00	E8	03	00	00	01	01	03	00	01	00	00	00	33	02
20	00	00	02	01	03	00	04	00	00	00	3A	01	00	00	03	01	03	00	01	00	00	00	05	00	00	00	06	01	03	00	01	00
40	00	00	02	00	00	00	0D	01	02	00	68	00	00	00	42	01	00	00	0E	01	02	00	12	00	00	00	AA	01	00	00	11	01
60	04	00	05	00	00	00	BC	01	00	00	12	01	03	00	01	00	00	00	01	00	00	00	15	01	03	00	01	00	00	00	04	00
80	00	00	16	01	03	00	01	00	00	00	80	00	00	00	17	01	04	00	05	00	00	00	D0	01	00	00	1A	01	05	00	01	00
A0	00	00	E4	01	00	00	1B	01	05	00	01	00	00	00	EC	01	00	00	1C	01	03	00	01	00	00	00	01	00	00	00	1D	01
C0	02	00	0A	00	00	00	F4	01	00	00	28	01	03	00	01	00	00	00	02	00	00	00	31	01	02	00	0D	00	00	00	FE	01
E0	00	00	32	01	02	00	14	00	00	00	0C	02	00	00	3D	01	03	00	01	00	00	00	02	00	00	00	4A	01	04	00	01	00
100	00	00	C8	04	00	00	52	01	03	00	01	00	00	00	01	00	00	00	53	01	03	00	04	00	00	00	20	02	00	00	69	87
120	04	00	01	00	00	00	6A	05	00	00	73	87	07	00	A0	02	00	00	28	02	00	00	00	00	00	00	08	00	08	00	08	00
140	08	00	5C	5C	68	6F	6D	65	2E	6F	72	67	2E	61	61	6C	74	6F	2E	66	69	5C	6A	6F	75	74	73	69	61	31	5C	64

TIFF general: https://en.wikipedia.org/wiki/TIFF

TIFF 6.0 Specification: https://www.itu.int/itudoc/itu-t/com16/tiff-fx/docs/tiff6.pdf

WELL-KNOWN TEXT REPRESENTATION OF GEOMETRY (WKT)

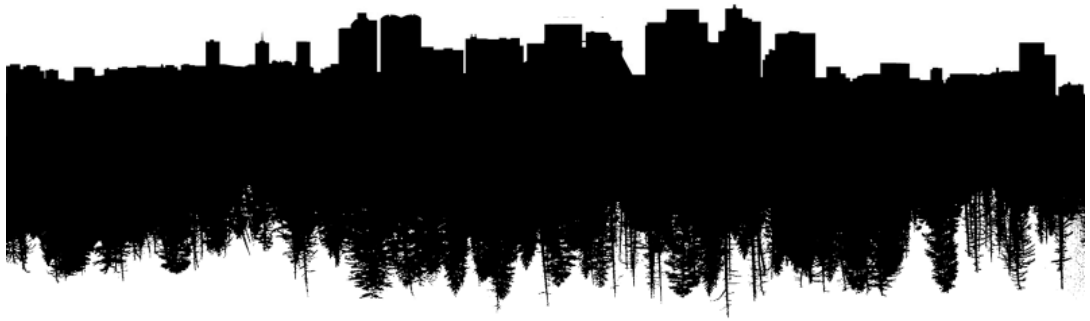
Revisiting text – binary conversion in Geopandas

- WKT is text markup language for vector geometry. WKB is the binary version.

- Is part of geopandas data frame (WKT / WKB)

```
def convert(wkt_text):
    p = shapely.wkt.loads(wkt_text)
    from shapely import wkb
    return (wkb.dumps(p, hex=True, srid=32636))
```

https://en.wikipedia.org/wiki/Well-known_text_representation_of_geometry



PROGRAMMING STYLES

OBJECT ORIENTED, PROCEDURAL & FUNCTIONAL

Object-Oriented Programming (OOP)

- Program flow is encapsulated within *Classes*
- Objects are instances of these classes
- *Methods* are encapsulated functions within classes

Procedural Programming

- Programs are sequences of instructions to be executed.
- Contains sets of instructions called *Procedures*, analogous to Functions.

Functional Programming

- *Function* is reusable set of instructions.
- Takes usually one or more input and returns output.

attributes:

- speed
- gear

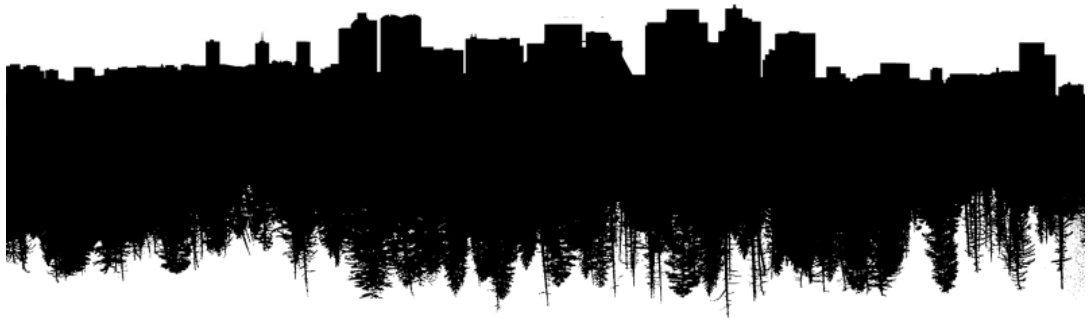
behaviours:

- speed up
- apply brake
- change gear

```
class bicycle:
    ''' properties'''
    # Class variables.
    gear = 1
    speed = 0

    def __init__(self, gear, speed):
        self.gear = gear
        self.speed = speed

    def speedUp(self, increase):
        self.speed += increase

    def changeGear(self,newGear):
        self.gear = newGear

    def applyBrake(self, decrease):
        self.speed -= decrease
```

```
print('{} is {} years old'.format(n, a) )
```

https://scoutapm.com/blog/functional-vs-procedural-vs-oop



SEQUENTIAL THINKING

PROGRAM FLOW CONTROL

There are ONLY TWO BASIC STRUCTURES in sequantial programming:

Branching structures

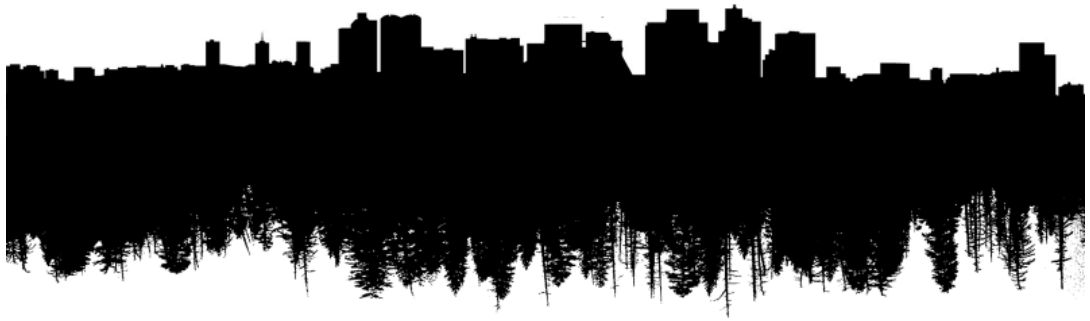
- IF/ELIF/ELSE (in other languages f.ex. SWITCH-CASE statements etc.)

Looping structures

- FOR and WHILE structures (also range(), enumerate() etc. methods)

```
for x in lst:
    print(x)
```

https://www.youtube.com/watch?v=_a95RaIZyf0
https://jovian.ai/aakashns/python-branching-and-loops



FUNCTIONS & OPERATORS

https://www.pythoncheatsheet.org/
https://cheatography.com/davechild/cheat-sheets/python/

OPERATORS

Arithmetic operators

+	Addition	2 + 2 = 4
-	Subtraction	5 - 2 = 3
*	Multiplication	3 * 3 = 9
/	Division	22 / 8 = 2.75
**	Exponent	2 ** 3 = 8
%	Modulus/Remainder	22 % 8 = 6
//	Integer division	22 // 8 = 2

Comparison

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

Note also: Assignment operators, Boolean operators & Augmented Assignment Operators

STRING FUNCTIONS

Variable type conversion

str(), int(), float()

String methods

upper(), lower()

join() and split()

strip(), rstrip(), and lstrip()

format(<var>, <var>)

more… => Regular Expressions

DATA STRUCTURES

List (i.e. Array):

```
animal = ['cat', 'bat', 'rat', 'elephant']
animal[0]
```

Note: All Strings are lists!

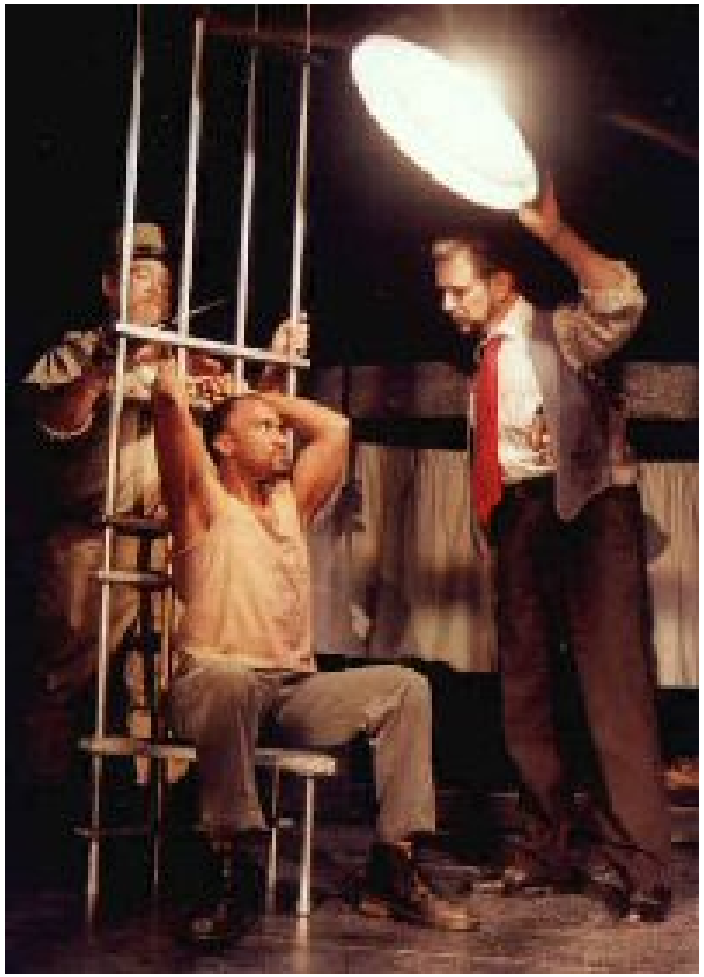
Dictionaries:

```
<dictionary> = {<key : <value>, <key : <value> ... }
spam = {'color': 'black', 'age': 78}
```

Dictionary methods:

```
.keys()
.values()
.items()
.get(<key>, <default>)
```

QUESTIONS ?



Thank you!