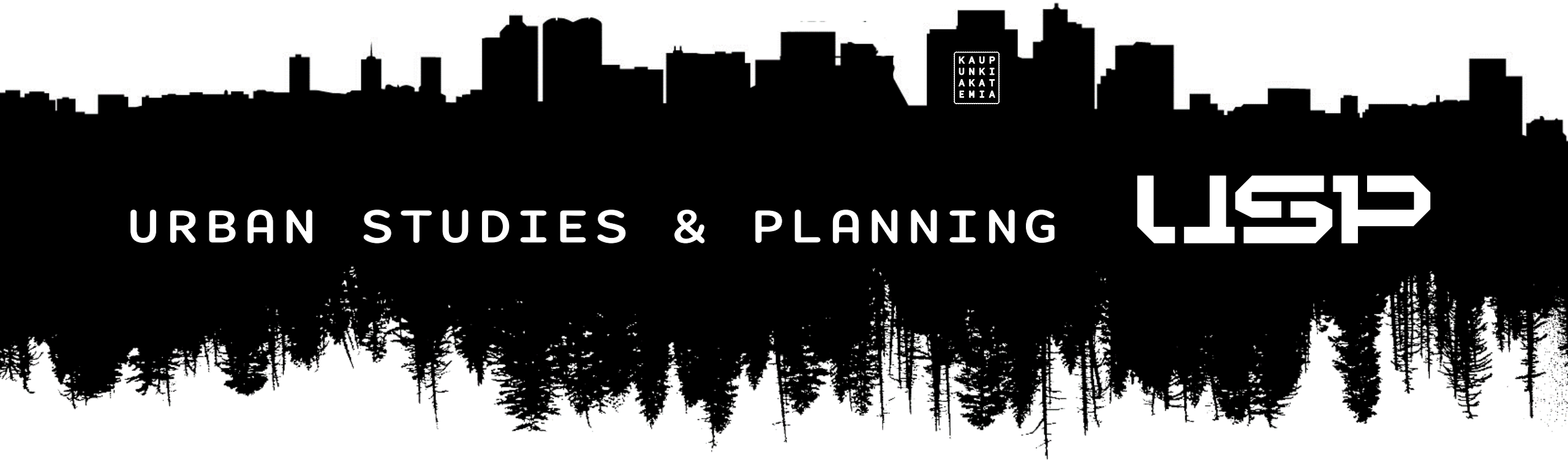
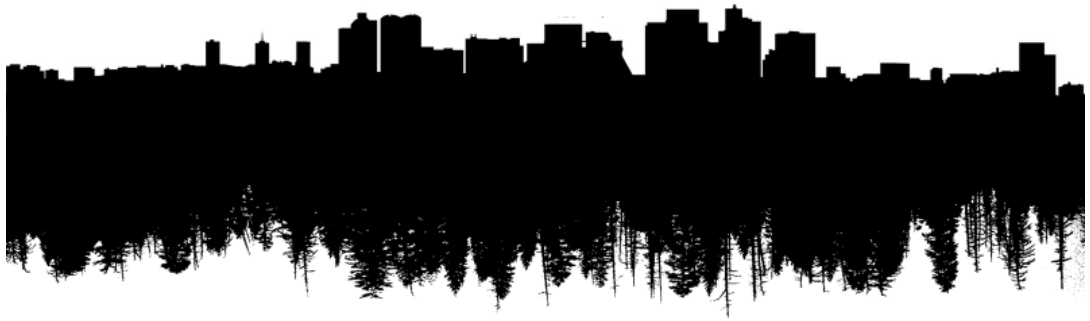


USP-E0361
Complex Adaptive Systems

Lecture 2

26.4.2023 Anssi Joutsiniemi

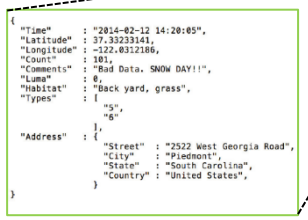


GENERALIZED DATA FLOW

Data feed

Coding manipulation

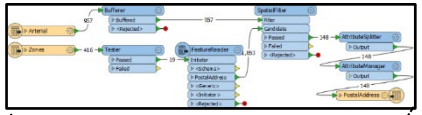
Representation

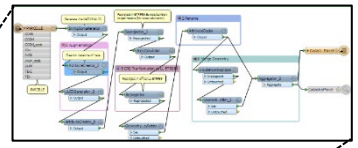


DATA

DATA

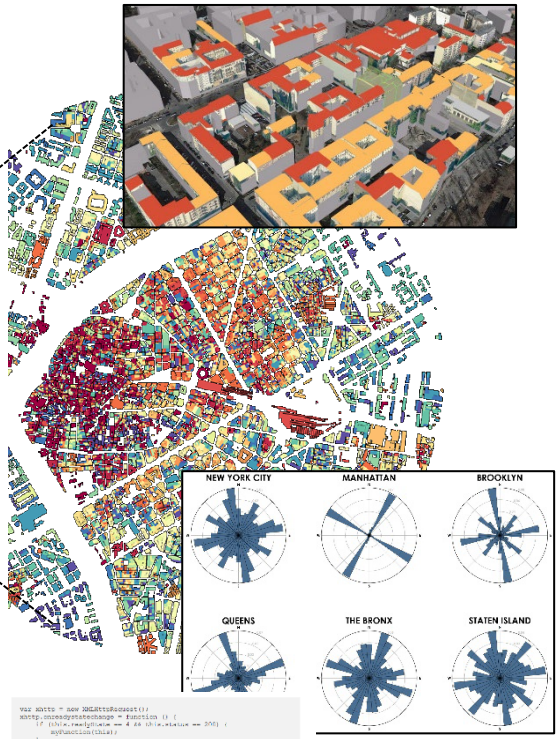
DATA





Collecting → Processing → Presenting

Iterative loop in GAMA



MESSY COMPUTAITION TERMINOLOGY

open vs. propritary

packed vs. unpacked

Unicode vs. ASCII vs. binary

numbers vs. text

HTML vs. XML vs. JSON

code vs. comments

RGB vs. CMYK

Windows vs. Mac vs. Linux

BASIC TERMS

FILE

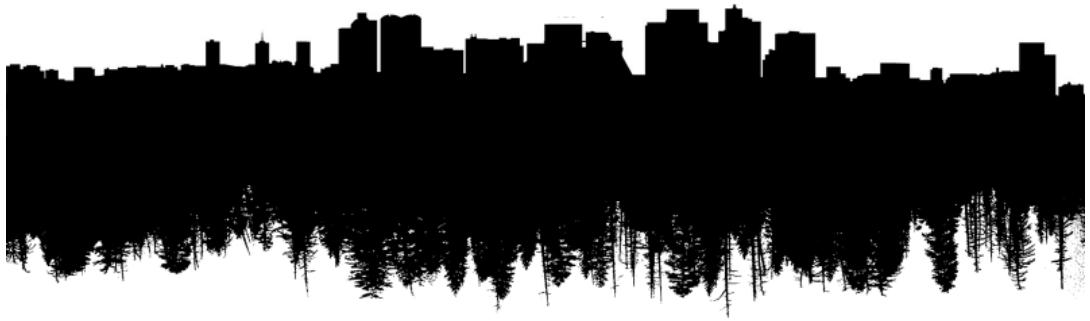
=> Data storage (structured information according to specification)

DATA

=> Processed entities (input values, processing variables & attributes, output items)

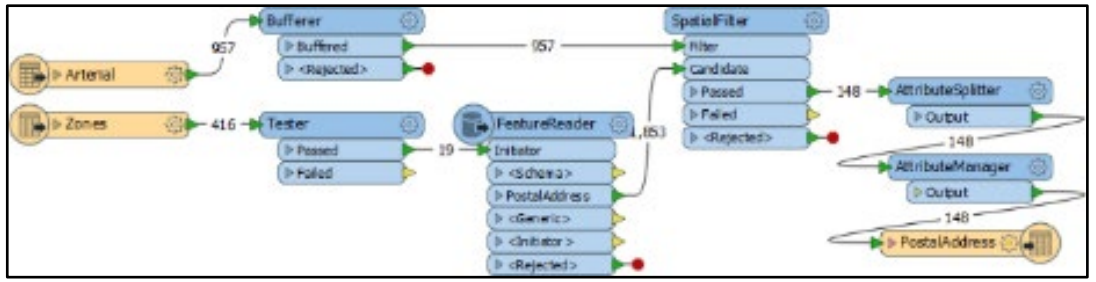
CODE

=> Set of instructions (using explicit syntax)



RECAP OF PRINCIPLES

CODE ENTITIES



Variables

- <gen.> Able to vary.
- <math.> A symbolic name associated with an entity whose associated value may be changed
- <comp.> A small section of **memory** in which a program can store intermediate results and from which it can read them.

Code

- A sequence of instructions

Comment

-Human-readable instruction

//Species representing the ground agents

Modularity

"Creating reusable and/or hierarchical packages of instructions"

- *Function* is reusable set of instructions.

  doMyThing(attribute)

- *Methods* are just functions encapsulated within classes

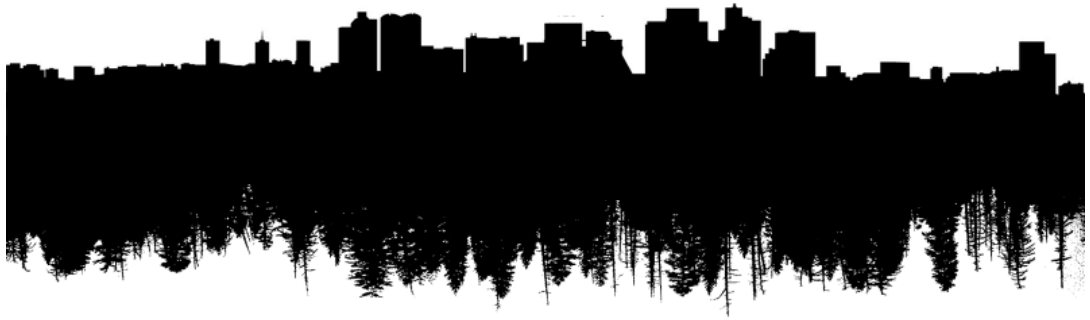
  class.doMyThing(attribute)

Attributes (=*data*)

- *Have type and structure*

Return values (=*data*)

- *Have type and structure*



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]

DATA TYPES

Numbers

▪ Bit & Nybble	1bit & 4 bits	(max. 2 & 16)	
▪ Byte	8-bits	(max. 256)	Byte
▪ Word	2 bytes, 16 bits	(max. 65 536)	Small Integer (signed/unsigned)
▪ Double word	4 bytes, 32 bits	(max. 4 294 967 296)	Integer (signed/unsigned)
▪ Quad word	8 bytes, 64 bits	(max.18 446 744 073 709 551 616)	Floating point values

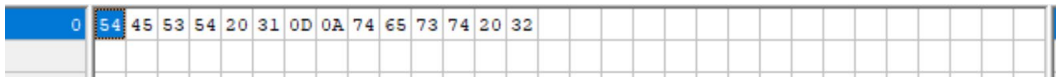
Text

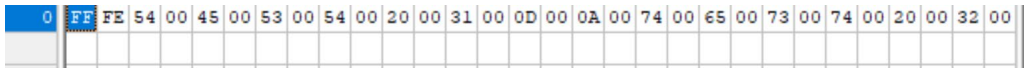
▪ ASCII/ANSI	1 byte	(max. 256)	Character
▪ UNICODE	2 bytes	(max. 65 536)	Unicode character

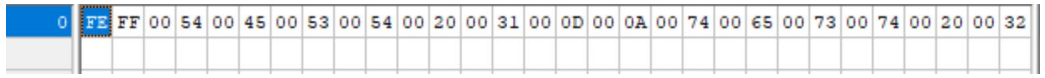
Date & Time (YYYY-MM-DD hh:mm:ss)

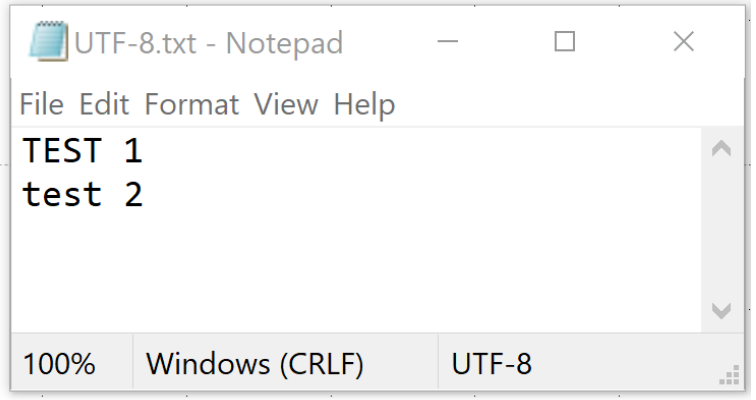
▪ Small datatime	4 bytes	1900-01-01 through 2079-06-06	1 minute accuracy
▪ Datetime	8 bytes	1753-01-01 through 9999-12-31	0.00333 second accuracy

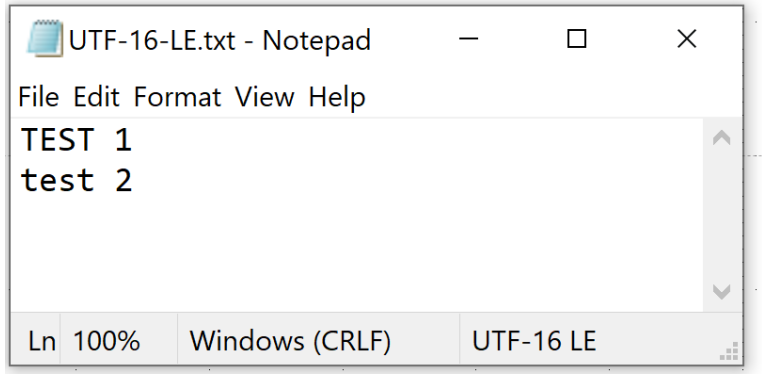
DATA SEEN & DATA STORED

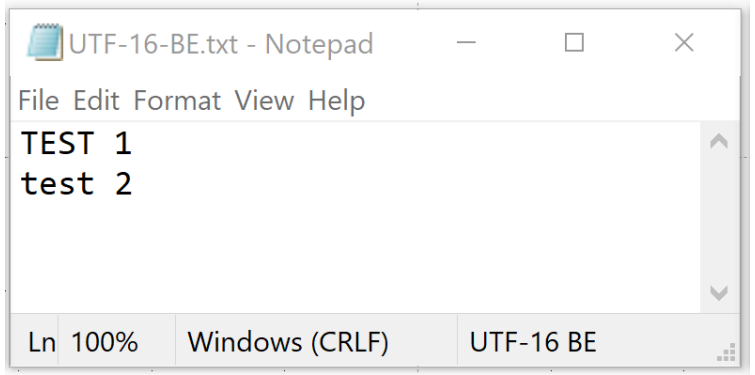




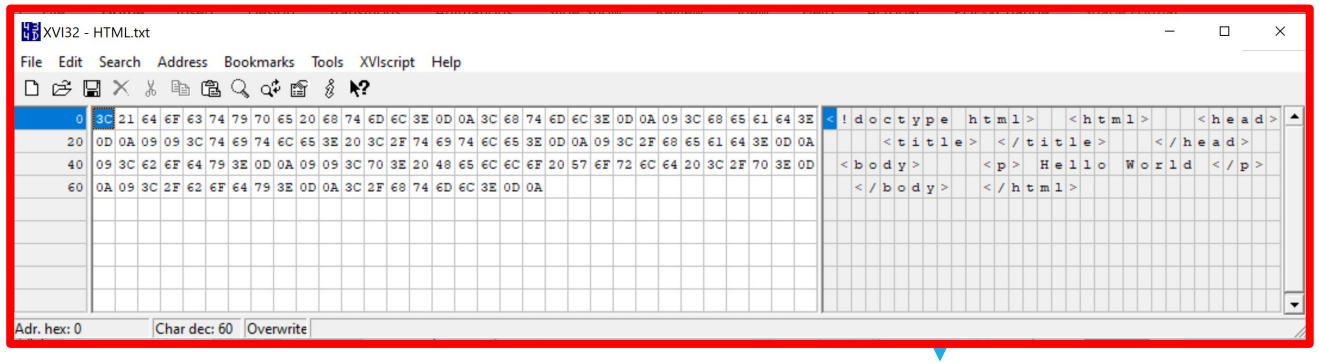








FILES & DATA

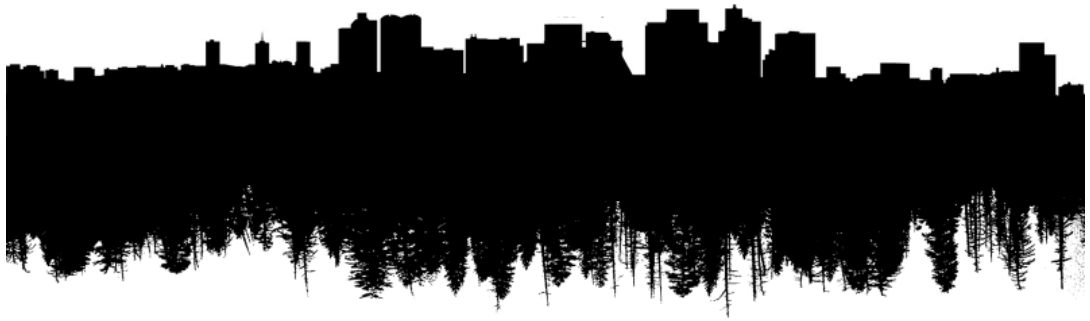



```
<!doctype html>
<html>
        <head>
                <title> </title>
        </head>
        <body>
                <p> Hello World </p>
        </body>
</html>
```

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



SEQUENTIAL THINKING

PROGRAM FLOW CONTROL

There are ONLY TWO BASIC STRUCTURES in sequential programming:

Branching structures

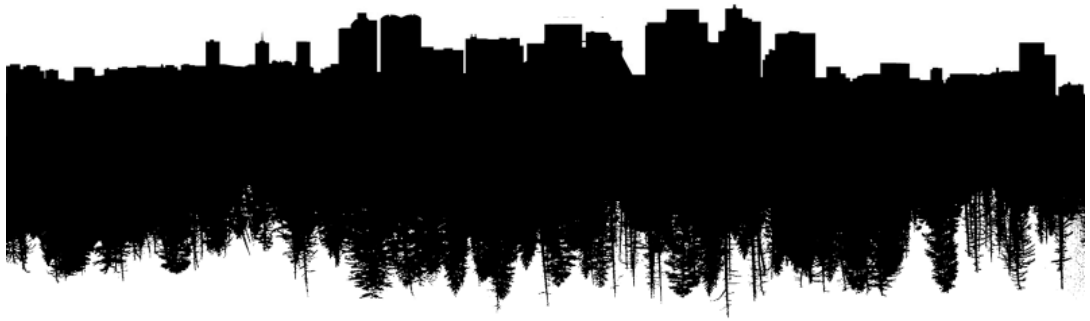
- IF/ELIF/ELSE (in other languages f.ex. SWITCH-CASE statements etc.)

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://github.com/gama-platform/gama/wiki/BasicProgrammingConceptsInGAML



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 (COLLECTION OF INDIVIDUAL DATA ITEMS)

List (i.e. Array):

```
animal = ['cat', 'bat', 'rat', 'elephant']
animal[1]
```

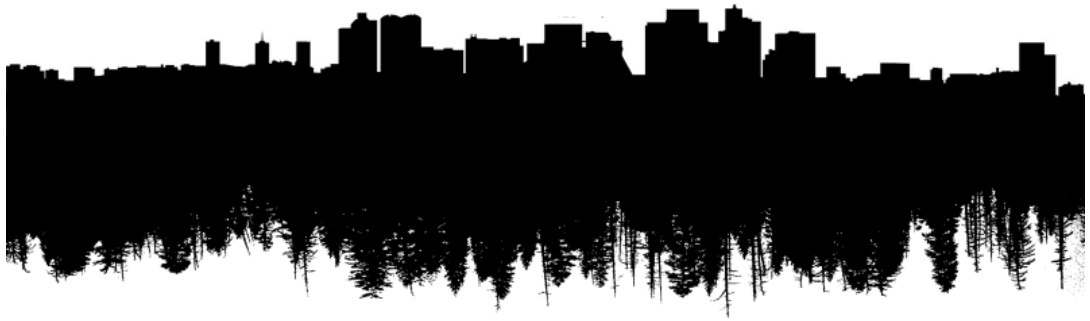
(*Advanced note: All Strings are lists!)*

Dictionaries:

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

Dictionary methods:

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

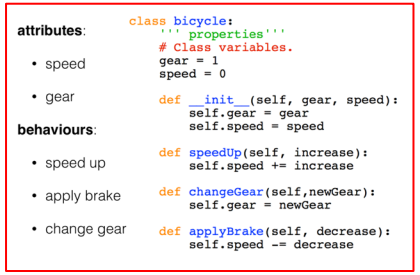


PROGRAMMING STYLES

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



Objects in GAMA:
https://www.youtube.com/watch?v=2GrR3VGIJ8g

Procedural Programming

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

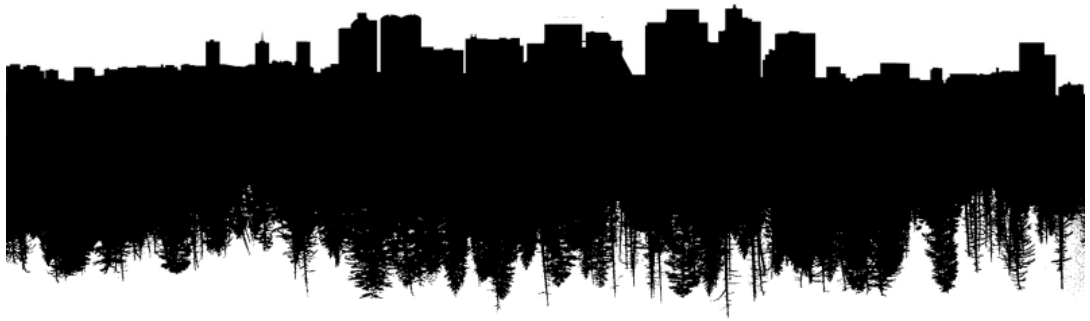
Functional Programming

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

```
In C:
printf("Character is %c \n", ch);

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

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



ART OF DEBUGGING

WHAT IS DEBUGGING

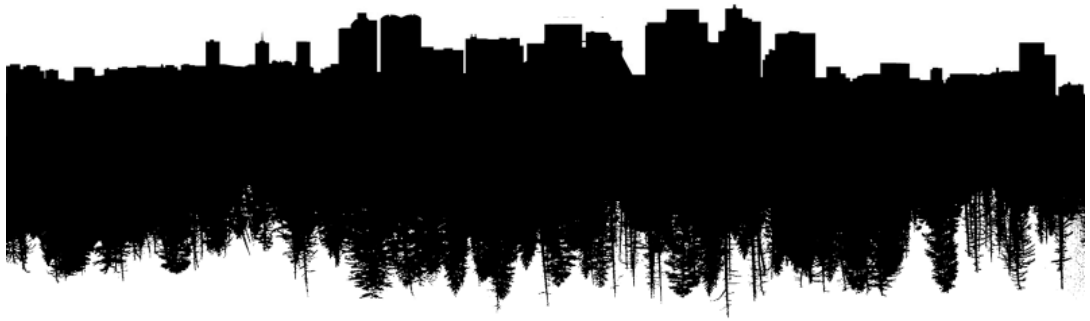
"Debugging occurs as a consequence of successful testing. That is, when a test case uncovers an error, debugging is the process that results in the removal of the error. Although debugging can and should be an orderly process, it is still very much an art."

https://www.1000sourcecodes.com/2012/05/software-engineering-art-of-debugging.html

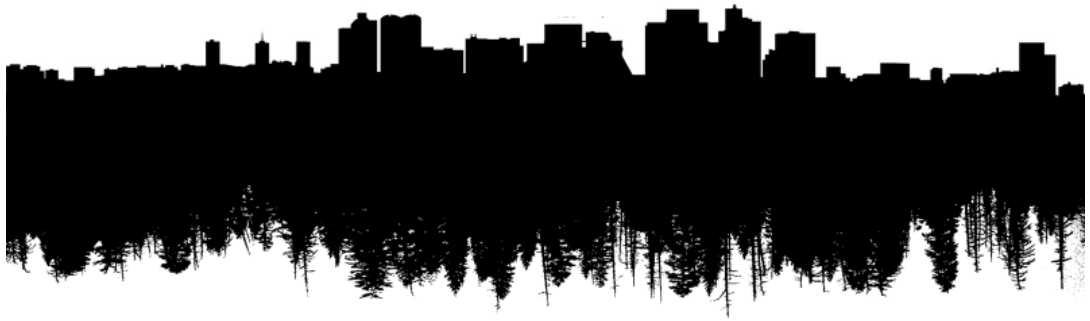
"Fixing a buggy program is a process of confirming, one by one, that the many things you *believe* to be true about the code actually are true. When you find that one of your assumptions is not true, you have found a clue to the location (if not the exact nature) of a bug."

Matloff, Norman & Salzman, Peter Jay (2008). *The Art of Debugging*. No Starch Press.

1. Start small
2. Use a top-down approach (modularity, hierarchy)
3. Pay attention to variable names and use plenty of comments
4. Spot exceptions arbitrary breaks (ex. Infinite loops)
5. Issue an interrupts to check the data validity



ADVANCED TOPICS



DATA BASICS CHEAT SHEET

NUMBER SYSTEMS

Decimal	(10-base)	[Values: 0,1,2,3,4,5,6,7,8,9]
Binary	(2-base)	[Values: 0,1]
Octal	(8-base)	[Values: 0,1,2,3,4,5,6,7]
Hexadecimal	(16-base)	[Values: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F]

https://www.youtube.com/watch?v=aW3qCcH6Dao

https://www.youtube.com/watch?v=GPnLy6YO-0M

CODING TEXT

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]

ASCII - American Standard Code for Information Interchange 7-bit

ANSI - American National Standards Institute 8-bit

Unicode (see: https://en.wikipedia.org/wiki/List_of_Unicode_characters)

Hex-to-ASCII https://www.rapidtables.com/convert/number/hex-to-ascii.html

ASCII-to-Hex https://www.rapidtables.com/convert/number/ascii-to-hex.html

CODING COLOUR

Color spaces are typically of DWORD length i.e. 4 bytes (32 bits) long.

Threfore there is 1 byte (256 values) per color component.

Additive colors (RGB):

https://www.youtube.com/watch?v=LCs8mK1rzc0

Substractive colors (CMYK):

https://www.youtube.com/watch?v=r8ejTUNwgTo

Colors in WWW: https://en.wikipedia.org/wiki/Web_colors

http://htmlcolorcodes.com/

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

DATA COMPRESSION I.E. PACKING

UNPACKED COMPRESSED

The process of reducing the size of a data file.

Compression can be either lossy or lossless.

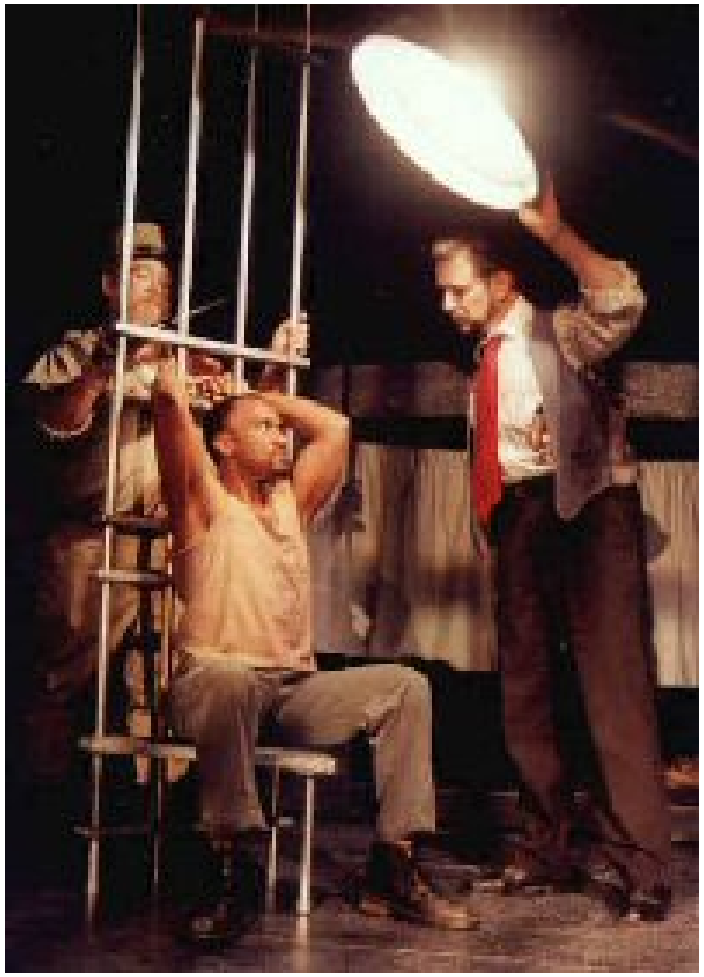
No information is lost in lossless compression. Lossy compression reduces bits by removing unnecessary or less important information.

- The Lempel–Ziv (LZ) compression methods are among the most popular algorithms for lossless storage.
- DEFLATE is a variation on LZ optimized for decompression speed and compression ratio, but compression can be slow. DEFLATE is used in PKZIP, Gzip, and PNG.
- LZW (Lempel–Ziv–Welch) is used in GIF images.
- Look for z-ending filenames: .klmz, .svgz etc.

Becoming more and more popular due to openness requirements. (*vrt. .doc vs .docx*)

MS-format specifications: https://msdn.microsoft.com/en-us/library/office/cc313105(v=office.12).aspx

QUESTIONS ?



Thank you!