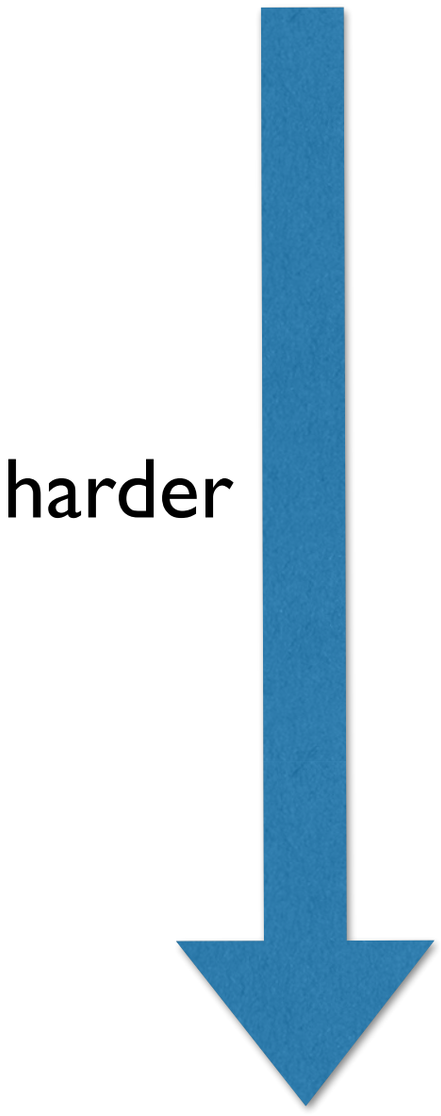
Data Types

David Grellscheid

UNIVERSITETET I BERGEN

From Python, I want to...



use output from other progs

run other programs

call other functions

Use output from other progs

data files in **standard** formats

ideally plain text (txt, csv, json, xml, ...)

or self-documented binary formats (netCDF,...)

watch out for encoding issues!

Use output from other progs

Core Python: `with open(...):`

Standard libraries: csv, json, xml, ...

External libs: xarray, ...

Run other programs

https://docs.python.org/3/library/subprocess.html

```
import subprocess
result = subprocess.run(
    ["ls", "-l"],
    capture_output=True,
    text=True
)
print(result.stdout)
```

Parallel tasks: multiprocessing, queue
https://docs.python.org/3/library/concurrency.html

Call other functions

Foreign function interface (FFI)

Python-C: ctypes, cython

Python-Fortran: f2py (in numpy)

Python-Java: jython, py4j, ...

Python-R: rpy2 / R-Python: reticulate

https://rpy2.github.io/doc/v3.5.x/html/introduction.html

Call external functions

Data types need to match!

integers

floating-point numbers

string format / encoding

array layout

Binary representation

All data is stored in bytes

1 byte = 8 bits

e.g. 0100 0001 is one byte

Binary / hexadecimal

"0000"	0	"1000"	8
"0001"	1	"1001"	9
"0010"	2	"1010"	a
"0011"	3	"1011"	b
"0100"	4	"1100"	c
"0101"	5	"1101"	d
"0110"	6	"1110"	e
"0111"	7	"1111"	f

4 binary digits make 1 hex digit. Easier to write

Binary representation

1 byte = 8 bits = 2 hex digits.
Can have 256 possible values

binary	hex			
"0000 0000"	"00"			
"0000 0001"	"01"			
"0000 0010"	"02"			
...	...			
"0100 0001"	"41"			
...	...			
"1110 0110"	"e6"			
...	...			
"1111 1101"	"fd"			
"1111 1110"	"fe"			
"1111 1111"	"ff"			

Binary representation

Many possible interpretations of those bit patterns.
Here: unsigned or signed integers

binary	hex	uint8	int8	
"0000 0000"	"00"	0	0	
"0000 0001"	"01"	1	1	
"0000 0010"	"02"	2	2	
...	...	...	...	
"0100 0001"	"41"	65	65	
...	...	...	...	
"1110 0110"	"e6"	230	-26	
...	...	...	...	
"1111 1101"	"fd"	253	-3	
"1111 1110"	"fe"	254	-2	
"1111 1111"	"ff"	255	-1	

Binary representation

Many possible interpretations of those bit patterns.
Here: text

binary	hex	ascii	iso8859-1	iso8859-2
"0000 0000"	"00"	*NUL*	*NUL*	*NUL*
"0000 0001"	"01"	*SOH*	*SOH*	*SOH*
"0000 0010"	"02"	*STX*	*STX*	*STX*
...	...	...		
"0100 0001"	"41"	A	A	A
...	...	...		
"1110 0110"	"e6"	✘	æ	ć
...	...	...		
"1111 1101"	"fd"	✘	ý	ý
"1111 1110"	"fe"	✘	þ	ţ
"1111 1111"	"ff"	✘	ÿ	˙

Larger representations

int32 - 4 bytes

int64 - 8 bytes

big5 (chinese chars) - 2 bytes

a4-48 : 人

utf-8: flexible byte length

Unicode representation

Each glyph has one number from 0x0 to 0x10ffff

glyph	unicode point	name	utf-8
A	U+0041	*Latin Capital letter A*	"41"
æ	U+00e6	*Latin Small letter Æ*	"c3 a6"
ć	U+0107	*Latin Small letter C with acute*	"c4 87"
ن	U+0646	*Arabic Letter Noon*	"d9 86"
و	U+0648	*Arabic Letter Waw*	"d9 88"
ر	U+0631	*Arabic Letter Reh*	"d8 b1"
凯	U+51ef	*CJK Unified Ideograph-51EF*	"e5 87 af"
𐏈	U+103c8	*Old Persian Sign Auramazdaa*	"f0 90 8f 88"
🤓	U+1f913	*Nerd Face*	"f0 9f a4 93"
🇿🇦	-	-	-
🇿	U+1F1FF	*Regional Indicator Symbol Letter Z*	"f0 9f 87 bf"
🇦	U+1F1E6	*Regional Indicator Symbol Letter A*	"f0 9f 87 a6"