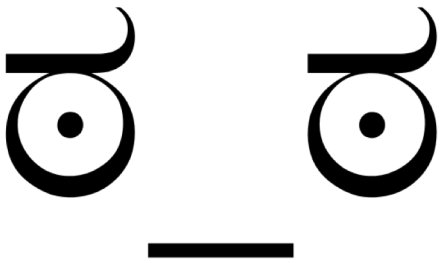
The str/bytes nightmare before python2 EOL

Kir Chou

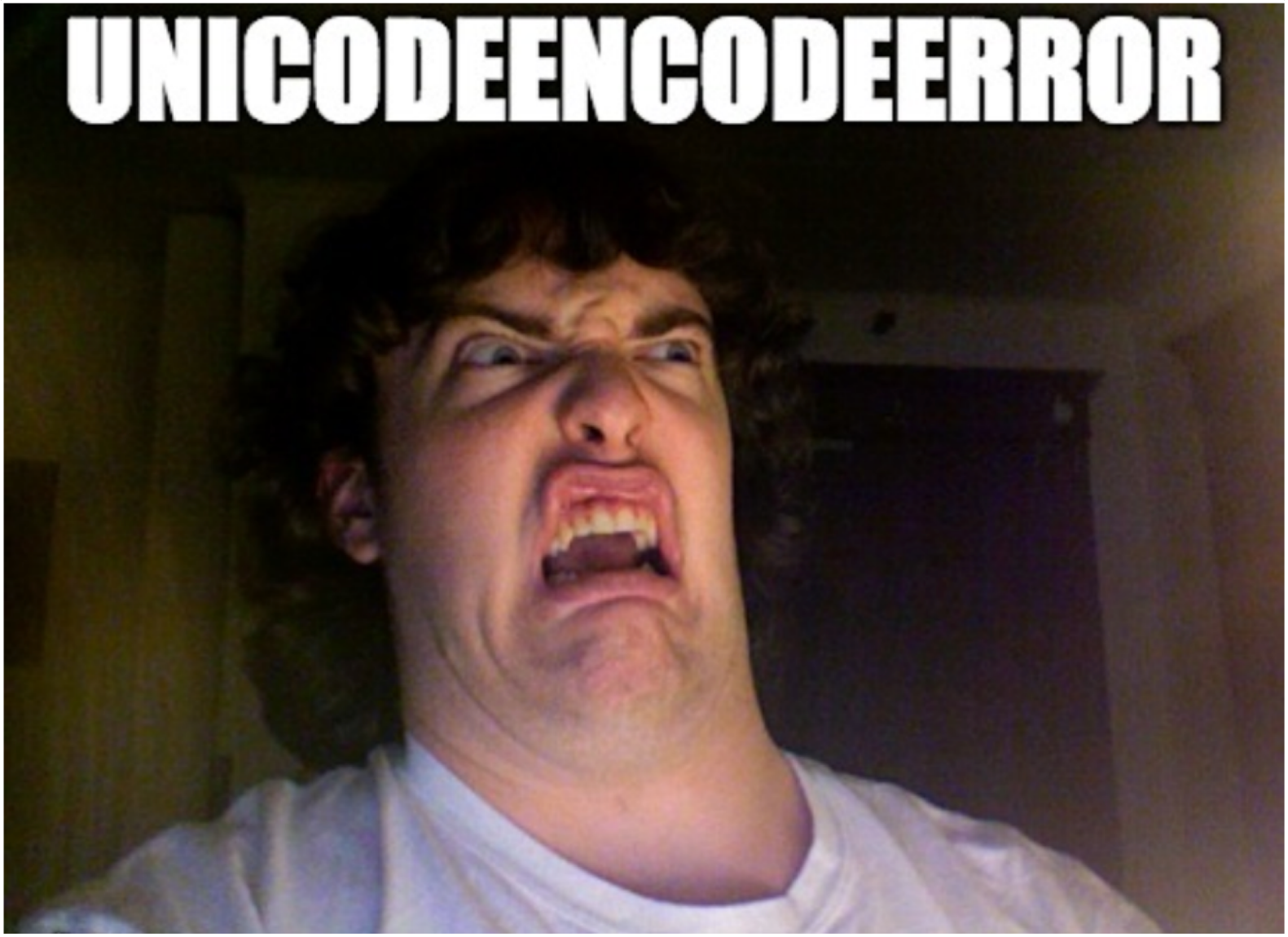
A9 (Amazon Product Search)

🚨 Confusing Terms 🚨

- **Str:** str() of python
- **Bytes:** bytes() of python
- **Text:** unicode() in Python2 or str() in Python3
- **String:** Text ∪ Bytes

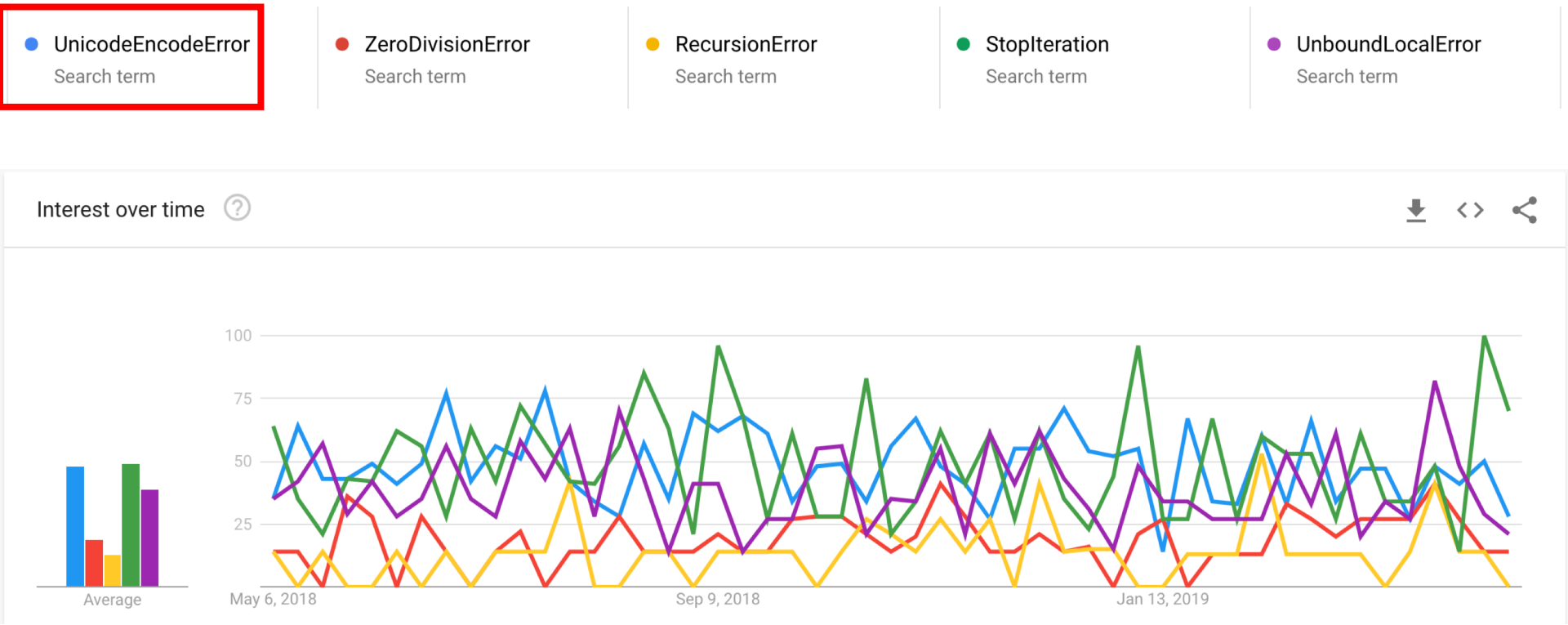


You just need to know that those terms are different in this talk.


UNICODEENCODEERROR

Google Trend

The frequency of **encoding** ~= **iterator** or **variable scope.**

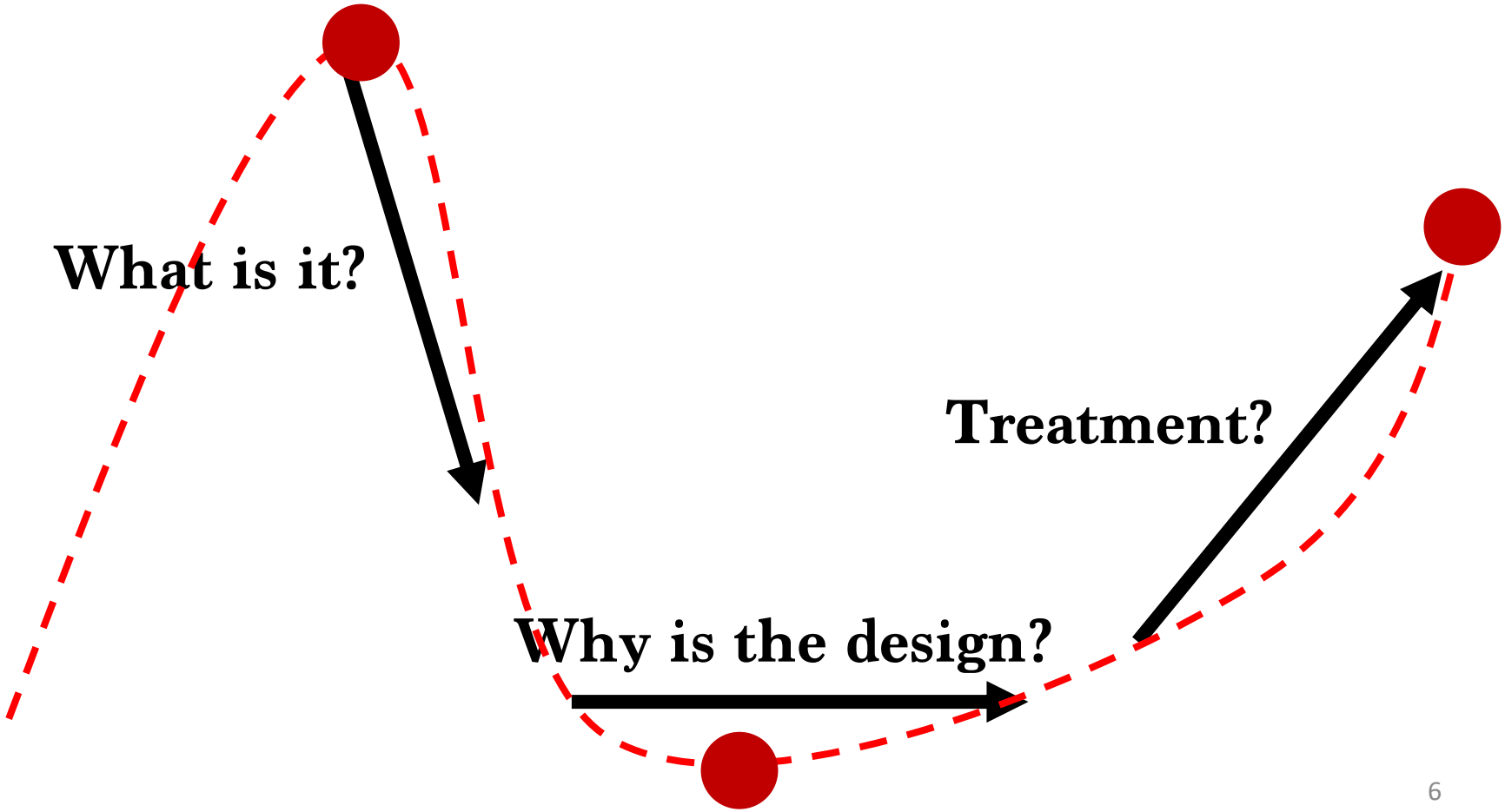


REF

Objective

D-K effect curve

For any level pythonist, nothing new in this talk



Outline

1. ~~(Covered) Talk Objective~~
2. Motivation
3. Python String 101 – What is string?
4. Python String 101 – Why is this design?
5. 5 Treatments

Motivation

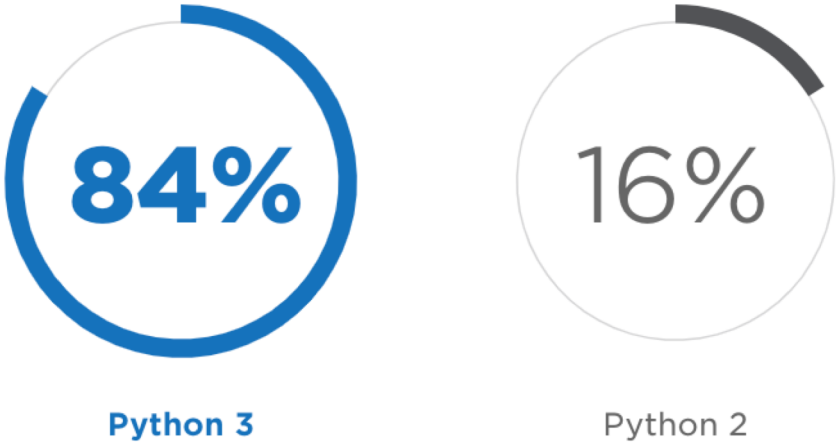
Python2 End-Of-Life

https://pythonclock.org/

Optimistic Numbers

∵ The number of python2 user ↓

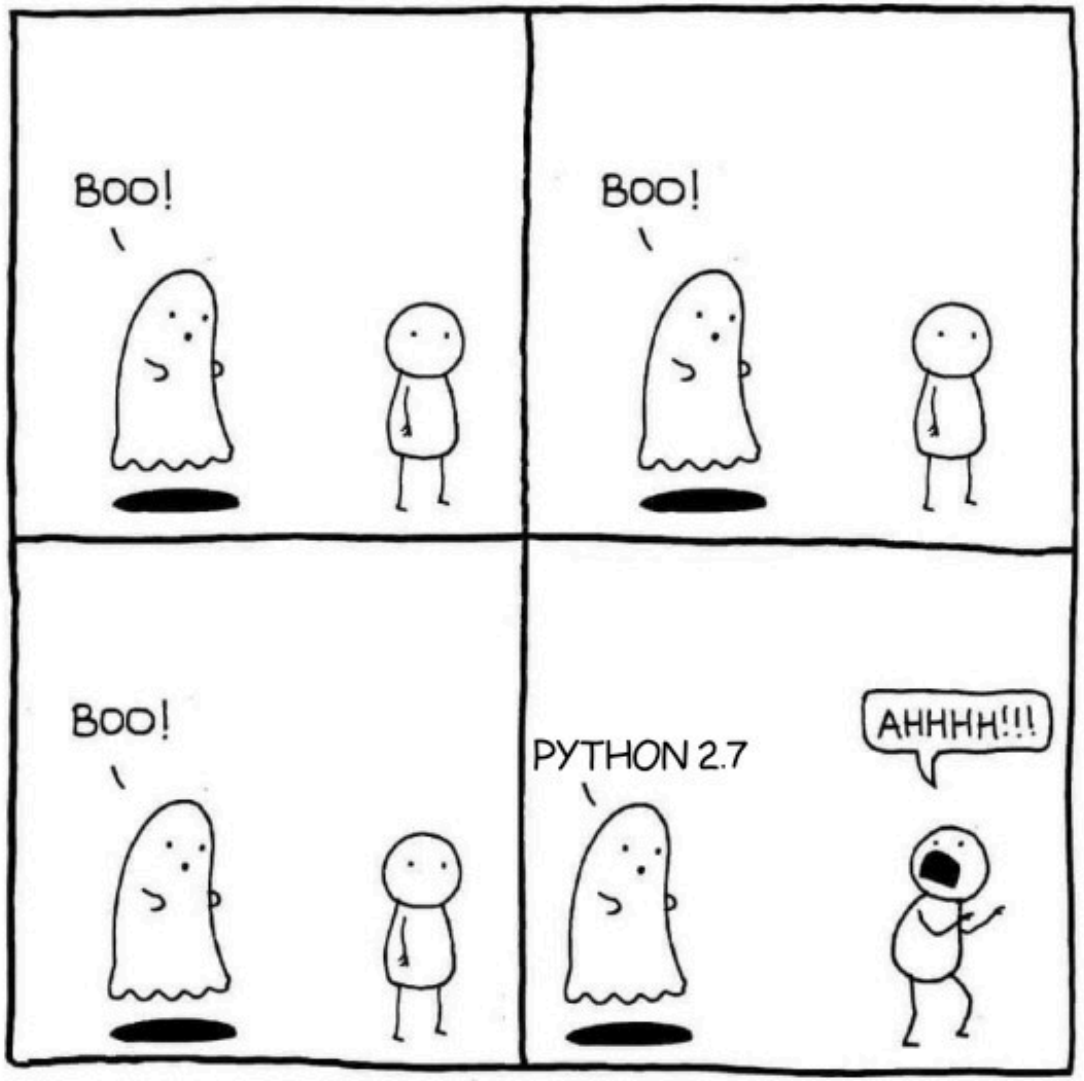
∴ The number of new package will be written in python3

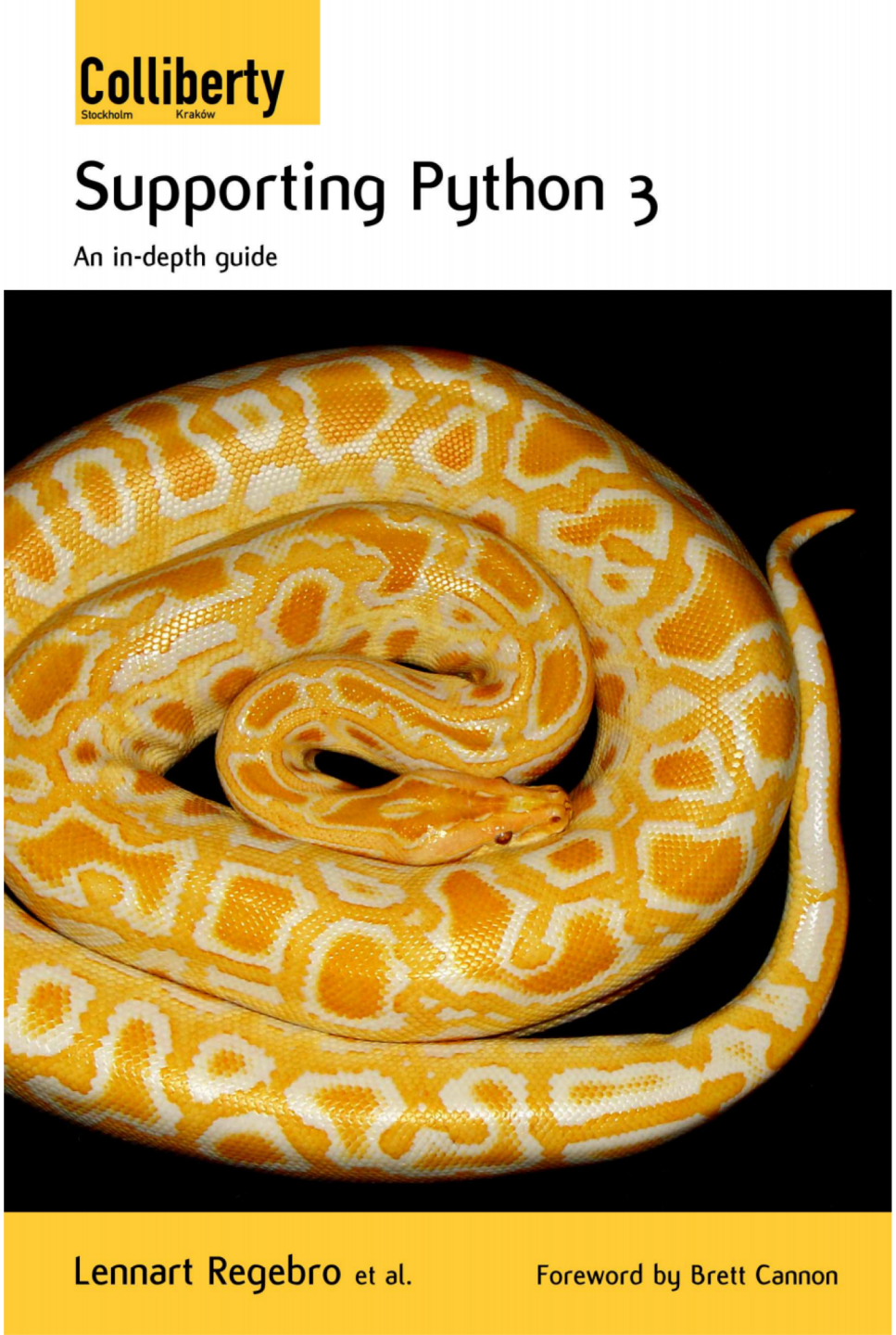


By Jetbrains 2018 survey

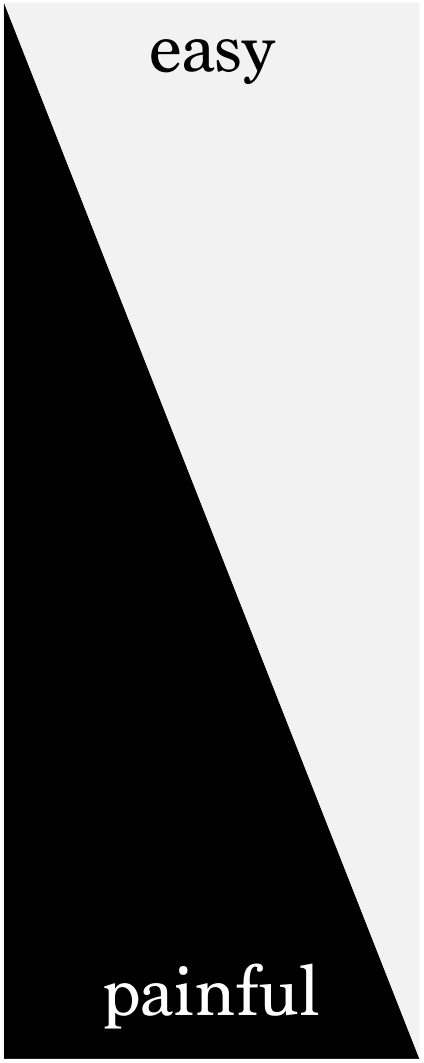
Pessimistic Facts

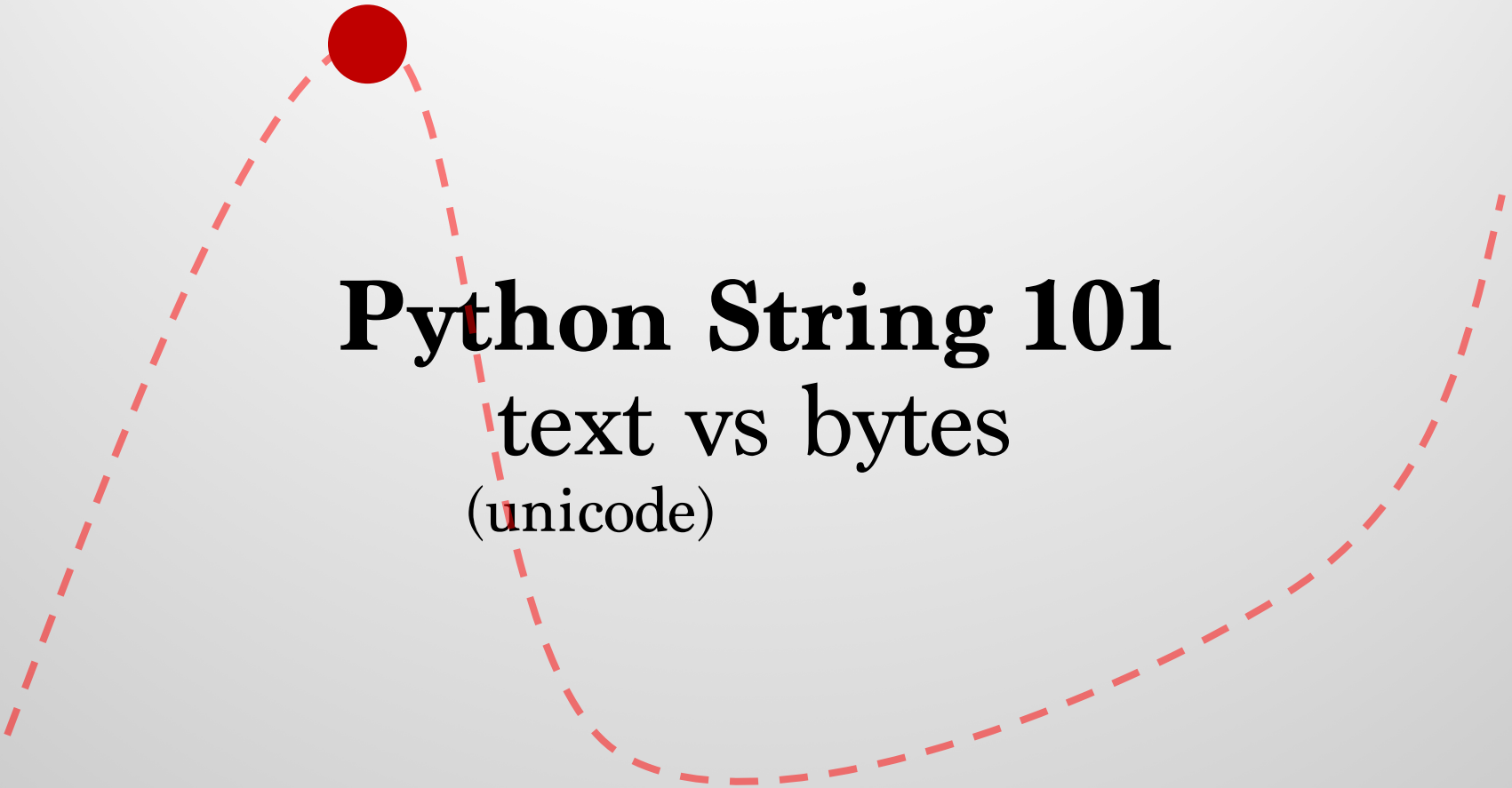
legacy dependencies




Colliberty
Stockholm Kraków
Supporting Python 3
An in-depth guide
Lennart Regebro et al.
Foreword by Brett Cannon

Migration – Pain Points



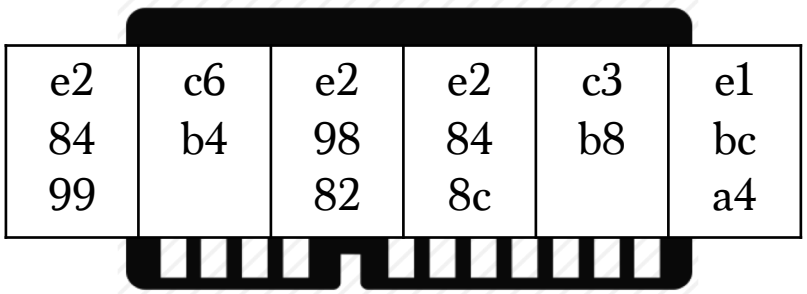


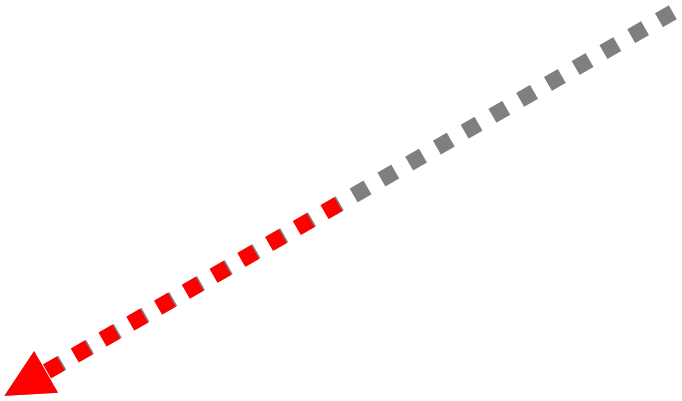
Python String 101

text vs bytes

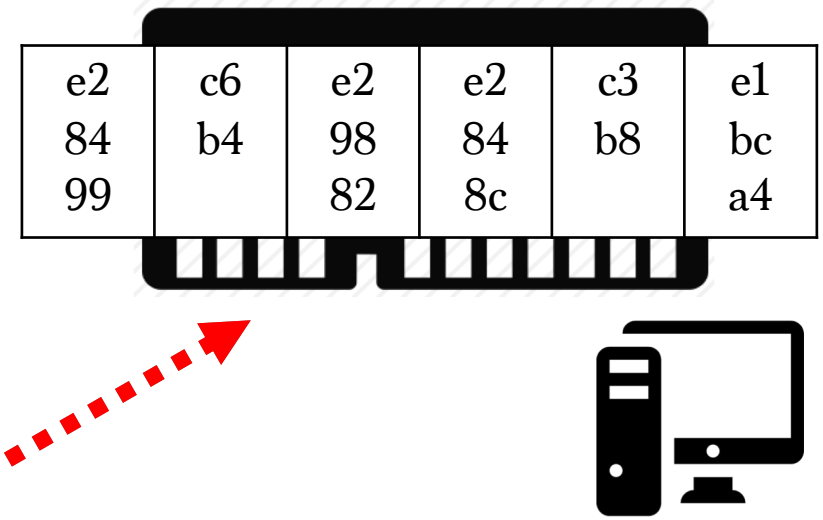
(unicode)

Text: How to present info in memory?





Bytes: How to store info into memory?



Python3

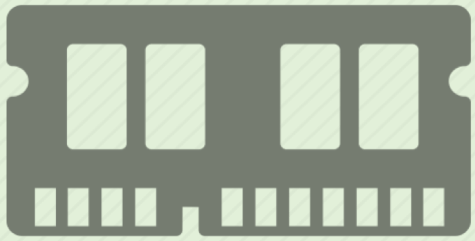
Default: text

```
>>> 'python'
'python'

>>> '𝔓𝔶  𝔊𝔬𝔥'
'𝔓𝔶  𝔊𝔬𝔥'
```

Python3

b' ' : ascii encode (0~127)
text.encode(**encoding**)

```
>>> b'python'
b'python'
>>> 'python'.encode(encoding='ascii')
b'python '
```



```
>>> b'ℙy𝔍  ℌø𝔂'
```

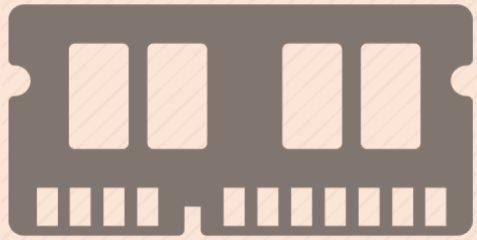
SyntaxError: bytes can only contain ASCII literal characters.

```
>>> 'ℙy𝔍  ℌø𝔂'.encode(encoding='utf-8')
b'\xe2\x84\x99\xc6\xb4\xe2\x98\x82\xe2\x84\x8c\xc3\xb8\xe1\xbc\xa4'
```

Python2

Default: auto-encoded bytes

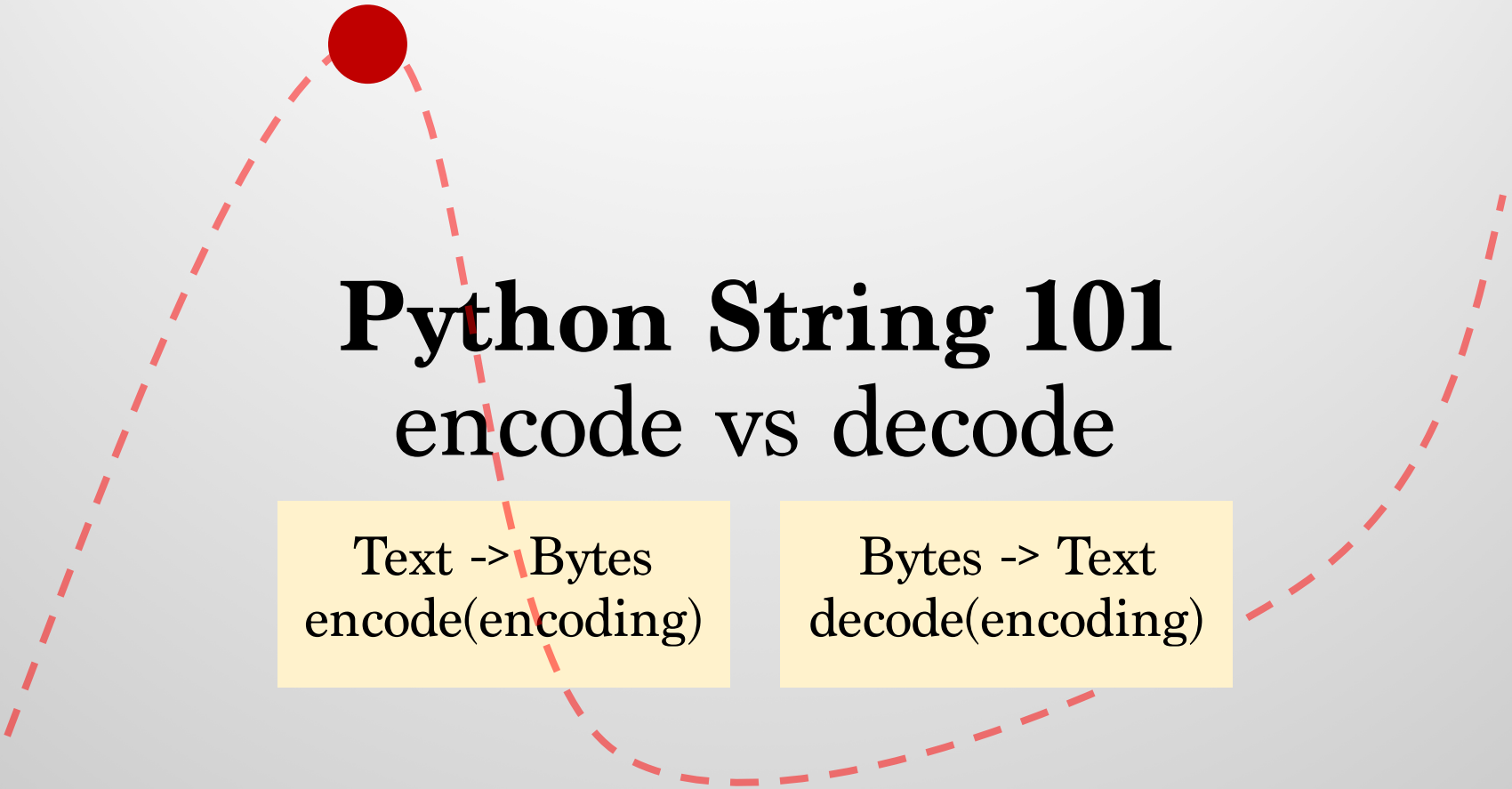
```
>>> 'python'
'python'

>>> '𝕻𝔂 𝔥𝔬̈𝔫'
'\xe2\x84\x99\xc6\xb4\xe2\x98\x82\xe2\x84\x8c\xc3\xb8\xe1\xbc\xa4'
```



Python2

u' ': text

```
>>> u'ℙy𝔱𝔥ǒ𝔫'
u'\u2119\u01b4\u2602\u210c\xf8\u1f24'
```



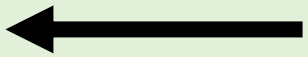
Python String 101
encode vs decode

Text -> Bytes
encode(encoding)

Bytes -> Text
decode(encoding)

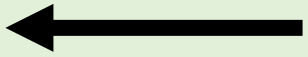
Python3

```
>>> dir(str())
[..., 'encode', ...]
```



Text -> Bytes
encode(encoding)

```
>>> dir(bytes())
[..., 'decode', ...]
```



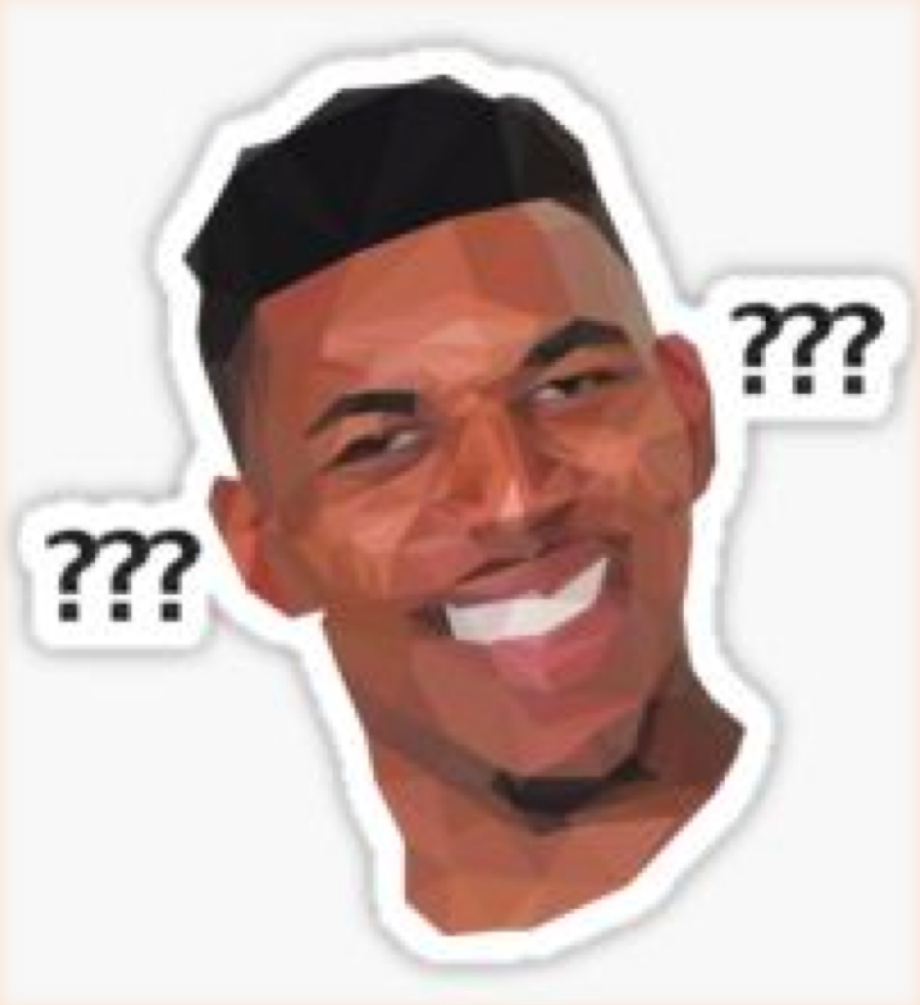
Bytes -> Text
decode(encoding)

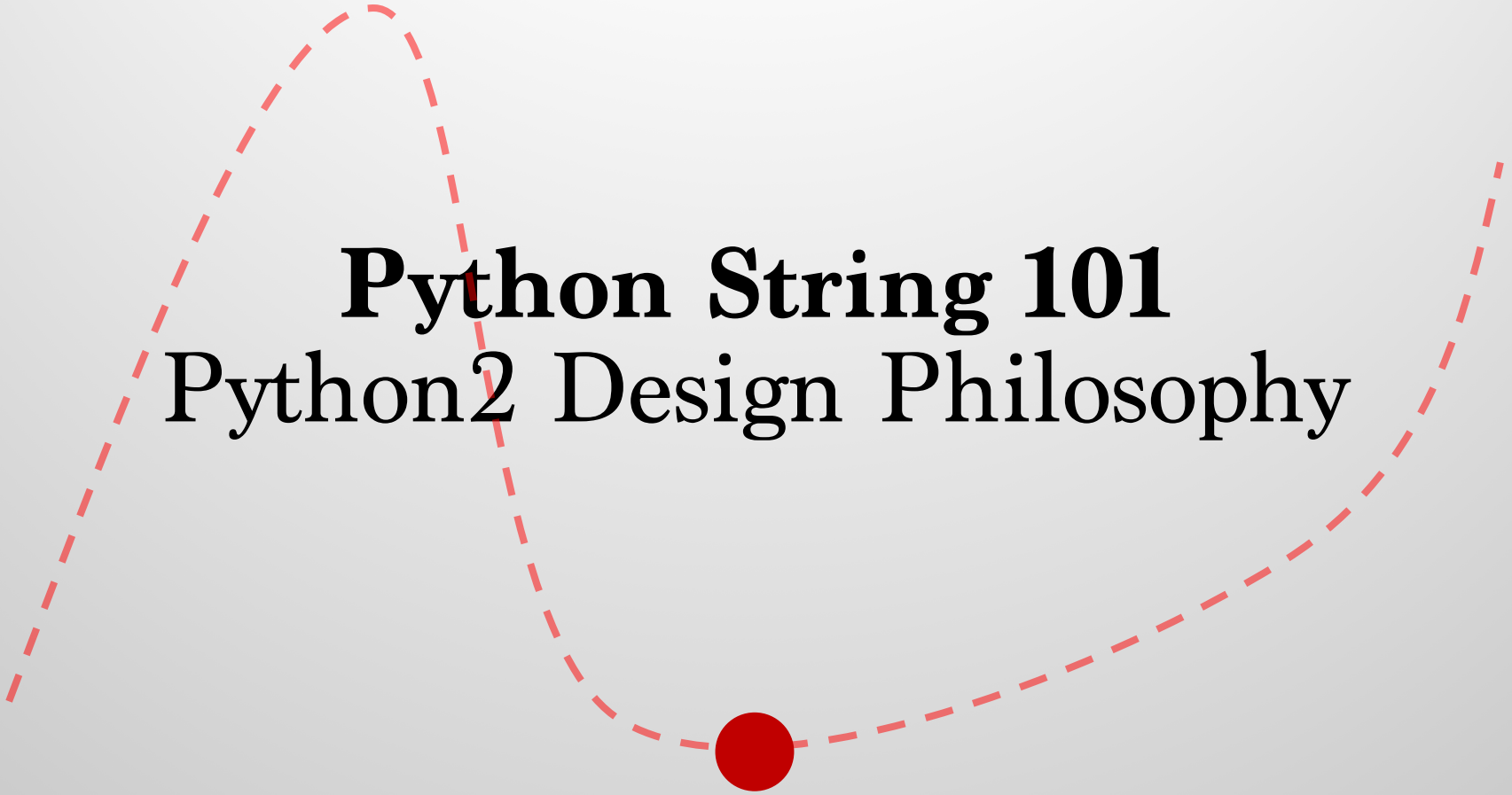
Python2

```
>>> dir(str())
[..., 'decode', 'encode', ...]

>>> dir(bytes())
[..., 'decode', 'encode', ...]
```

Python2

```
>>> dir(str())
[..., 'decode', 'encode', ...]

>>> dir(bytes())
[..., 'decode', 'encode', ...]
```





Python String 101

Python2 Design Philosophy

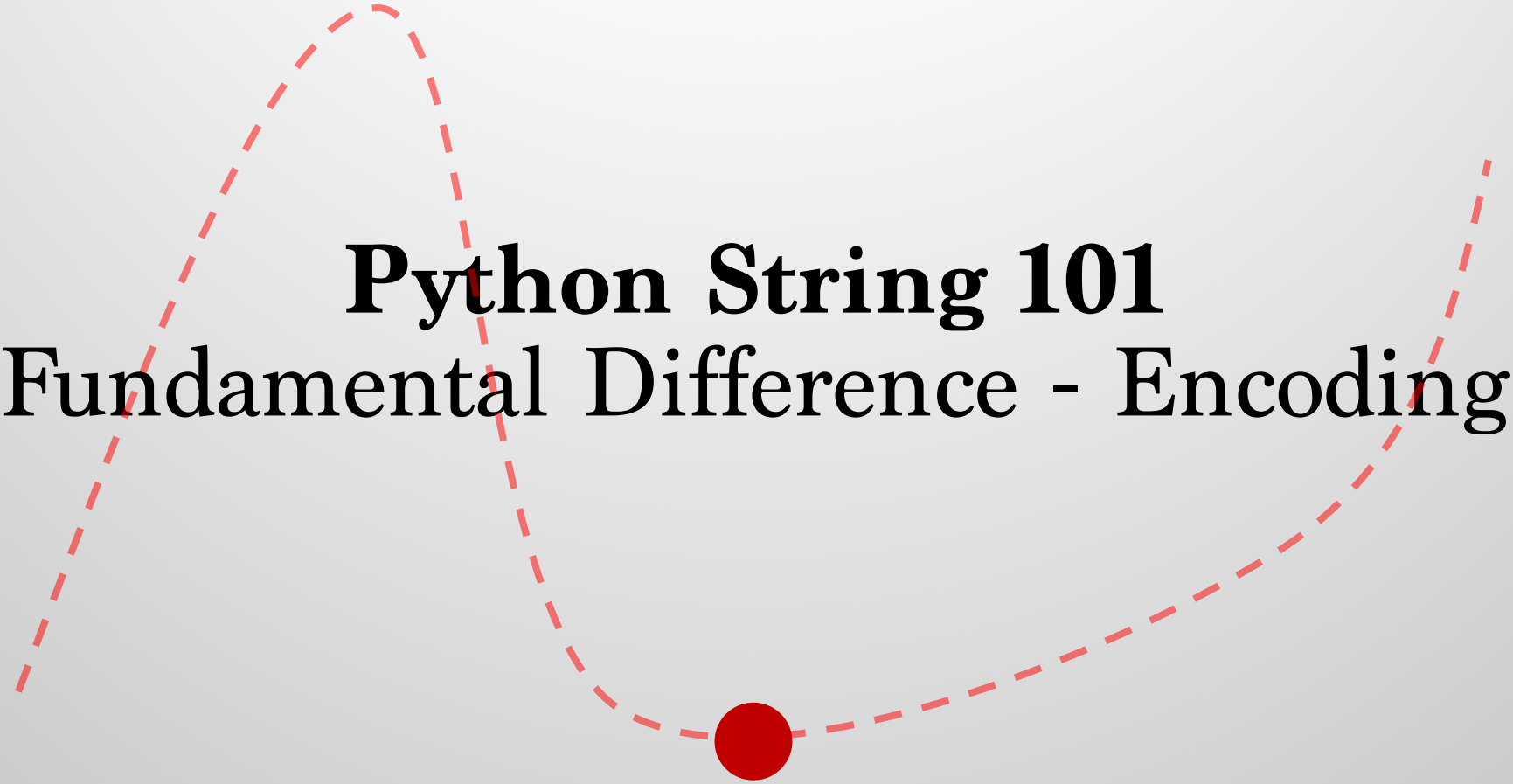
ASCII TABLE

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]

Python2 cares ascii encoding at first, BUT...

- Most of encoding supports one way: **utf-8, latin-1**
- But there are few exceptions: **base64, rot13**

```
>>> 'python'.encode('rot13')
'clguba'

>>> 'python'.decode('rot13')
u'clguba'
```



Python String 101
Fundamental Difference - Encoding

ASCII encoding behave similar,
BUT...

```
>>> b'python' == u'python'
False
```

3

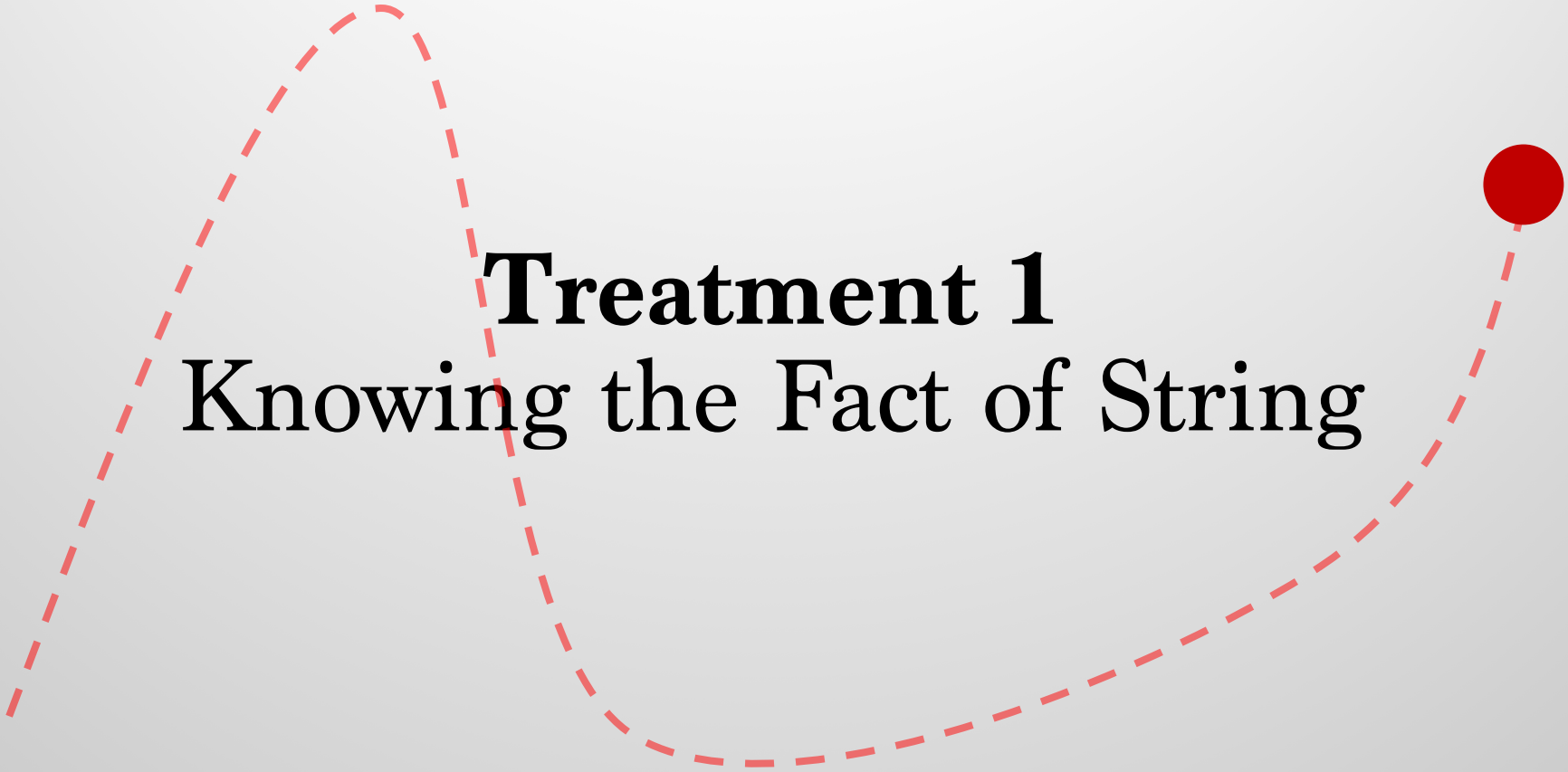
```
>>> b'python' == u'python'
True
```

2

You must be told, or you have to guess

By Ned Batchelder

	48	69	e2	84	99	c6	b4	e2	98	82	e2	84	8c	c3	b8	e1	bc	a4
utf-8	H	i	ℙ			ƴ		☂			ℌ			ø		ἤ		
iso8859-1	H	i	â	„	™	Æ	´	â	˜	‚	â	„	Œ	Ã	¸	á	¼	¤
utf-16-le	楈		蓢		욙		□		芘		蓢		쎌		□		꒼	
utf-16-be	䡩		□		駆		듢		颂		□		賃		룡		벤	
shift-jis	H	i	邃		巵		ｴ	笘		や		ъ		ﾃ	ｸ	眈		､



Treatment 1
Knowing the Fact of String

The Fact of String

By Ned Batchelder

- **Python3: every string are unicode**
  - **Encoding needs to be handled manually**
  - One-way encode/decode behind str/bytes *(Good)*

- **Python2: every string are auto encoded to bytes**
  - Ascii is consistently handled
  - Two-way encode/decode behind str/bytes *(Broken after python is broadly used in many different human languages.)*

Treatment 2
Explicit Declaration
(From community approach to supporting python3)

Consistent IO

- Standard I/O (bytes)
  - Python2: sys.stdin / sys.stdout
  - Python3: sys.buffer.stdin / sys.buffer.stdout

- File IO

```
import io  # consistent api in both versions
io.open('path/to/file', 'wt')  # text, bytes
```

Bytes or Text (with Encoding)

- u" or b"?
  - No more raw string "
- Which encoding is used for the text?
  - No more guess, always provide encoding: latin-1, utf-8...

2

```
def my_encrypt(text, encoding='utf-8'):
    ...
```

3

Encoding of your Operating System

2

```
>>> # -*- coding: utf-8 -*-
sys.getdefaultencoding()
```

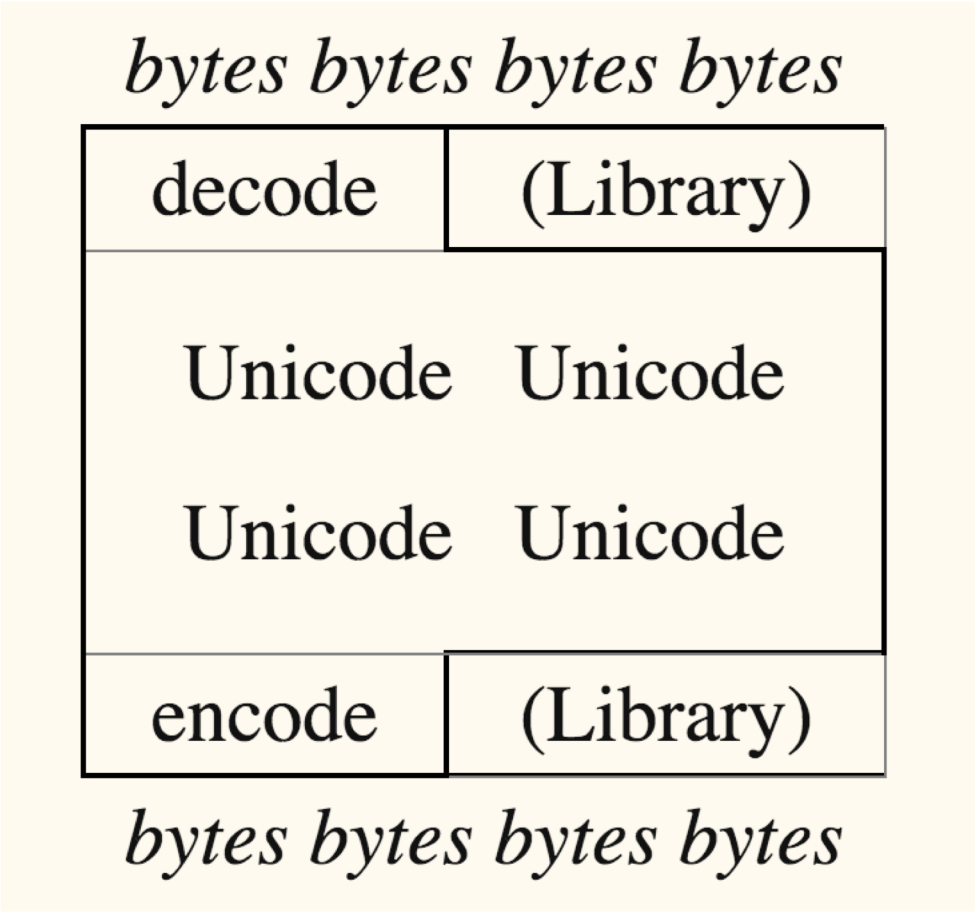
3

Treatments 3
Unicode Sandwich
(From community approach to supporting python3)

Unicode Sandwich

Decode ASAP, Encode ALAP

Don't care about encoding!



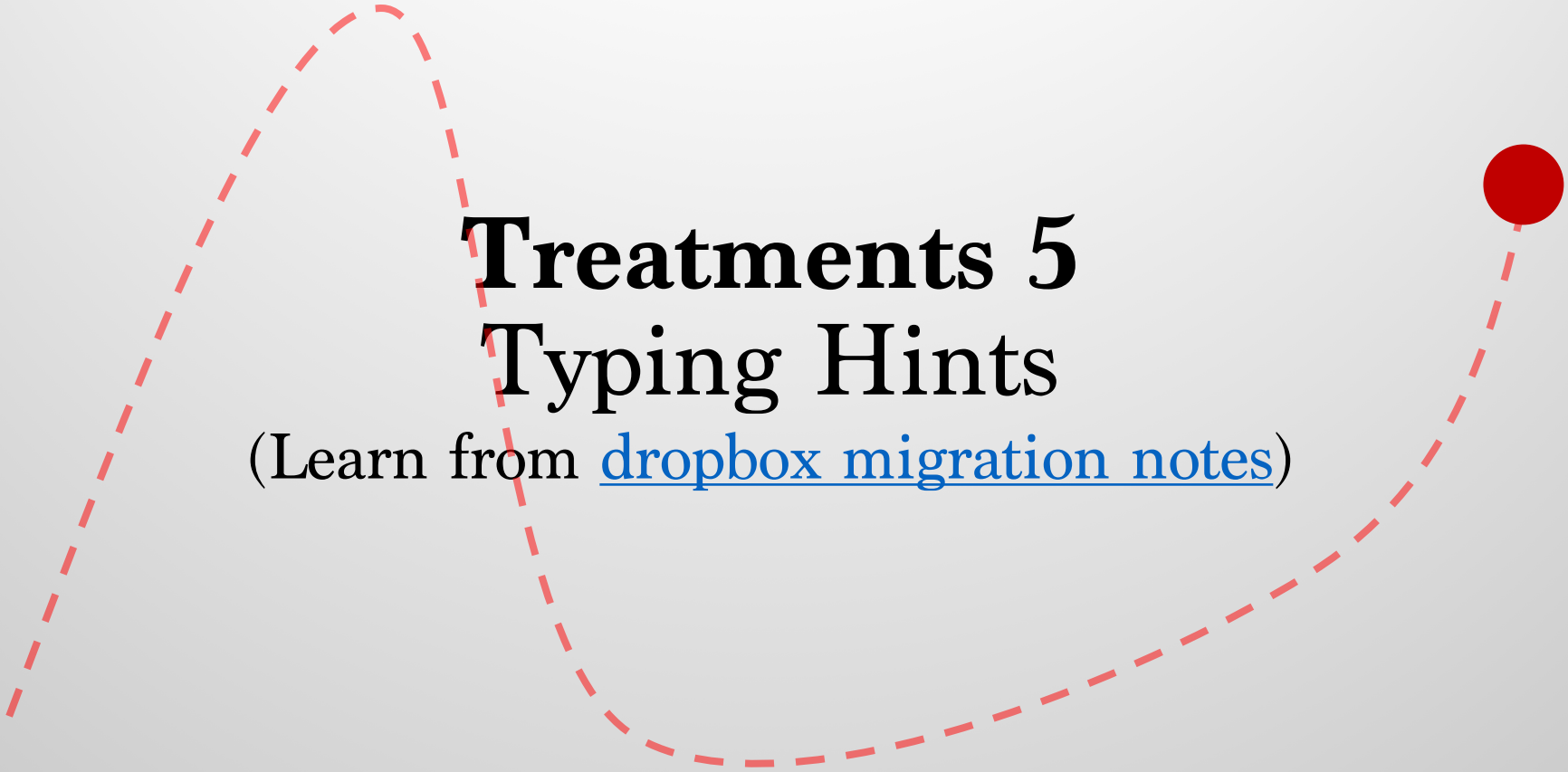
Treatments 4 (minor)
Python2/3 compatibility
(From community approach to supporting python3)

Add unicode_literals

after porting to python3

```
>>> from __future__ import unicode_literals
```

2



Treatments 5
Typing Hints

(Learn from dropbox migration notes)

Typing Hints [mypy] [pyre]

2

```
def encrypt(data, key):
    return cipher
```

3

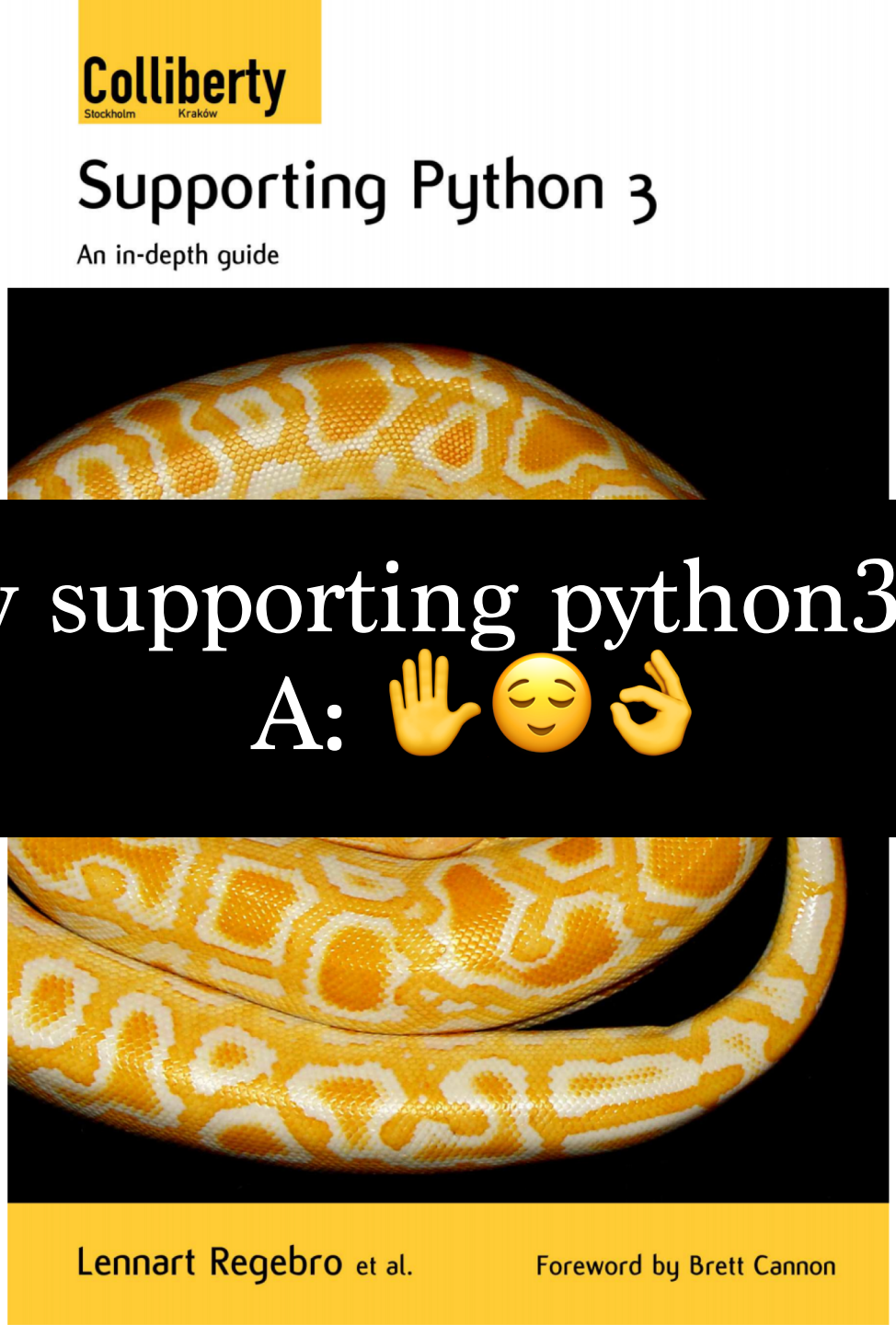
```
def encrypt(data: bytes, key: bytes) -> bytes:
    return cipher
```

3

```
def encrypt(data, key):
    # (bytes, bytes) -> bytes
    return cipher
```

2

Conclusion



Q: Why supporting python3 is hard?
A: 🖐😌👌

Take-Home Messages

Write text/bytes explicitly with typing hints

Always provide **encoding** for bytes

Apply **Unicode Sandwich** if possible

Copy encode/decode from StackOverflow

Python String is no longer a nightmare! 🎉

Homework

(Verify your understanding)

1. Search “UnicodeEncodeError” in StackOverflow
2. Randomly pick one solved question
3. Read the content
4. Explain to yourself what happened and why is it

Q&A

- String
- SupportingPython3

Reference: Docs

- Python2 unicode
- Python3 unicode
- pyporting
- python-future.org/compatible_idioms.pdf
- 2018 Jetbrains Survey
- Dropbox Migration Notes

Reference: Talks

- Ned Batchelder: Pragmatic Unicode, or, How do I stop the pain?
- Guido van Rossum: BDFL Python 3 retrospective
- Brett Cannon - How to make your code Python 2/3 compatible - PyCon 2015
- Writing Python 2/3 compatible code by Edward Schofield
- Brandon Rhodes - Oh, Come On Who Needs Bytearrays