Python Fundamentals

Strings and Collections

Austin Bingham
@austin_bingham
austin@sixty-north.com

Presenter

Robert Smallshire
@robsmallshire
rob@sixty-north.com

pluralsight
hardcore developer training

Python Collections

Python Collections


Python
Collections
str

Python Collections

str

bytes

Python Collections

`str`

`bytes`

`list`

Python Collections

- `str`
- `bytes`
- `list`
- `dict`

Python Collections

str

bytes

list

dict

for-loops

python™

str

immutable sequences of Unicode codepoints

String Literals

```
‘This is a string’
```

String Literals

```
'This is a string'
```

```
"This is also a string"
```

Moment of Zen

Practicality beats purity

Beautiful text strings
Rendered in literal form
Simple elegance

Strings with Newlines

Strings with Newlines

1. **Multiline** strings

Strings with Newlines

1. **Multiline** strings

2. **Escape** sequences

Windows

‘\r\n’

or

Linux

OS X

‘\n’

?

Universal Newlines

‘\n’

Universal Newlines

‘\n’

PEP278

www.python.org/dev/peps/pep-0278/

Escape Sequences

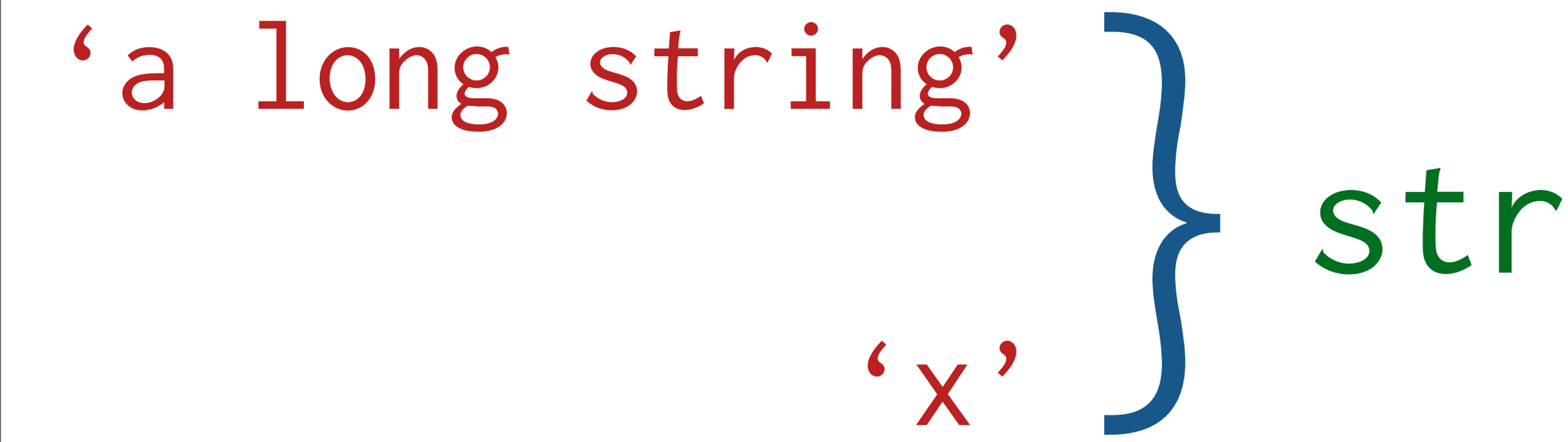
Sequence	Meaning
`\newline`	Backslash and newline ignored
`\\`	Backslash (\)
`\'`	Single quote (')
`\a`	ASCII Bell (BEL)
`\b`	ASCII Backspace (BS)
`\f`	ASCII Formfeed (FF)
`\n`	ASCII Linefeed (LF)
`\r`	ASCII Carriage Return (CR)
`\t`	ASCII Horizontal Tab (TAB)
`\v`	ASCII Vertical Tab (VT)
`\ooo`	Character with octal value ooo
`\xhh`	Character with hex value hh
Only recognized in string literals	
`\N{name}`	Character named name in the Unicode database
`\uxxxx`	Character with 16-bit hex value xxxx
`\Uxxxxxxxx`	Character with 32-bit hex value xxxxxxxx

Escape Sequences

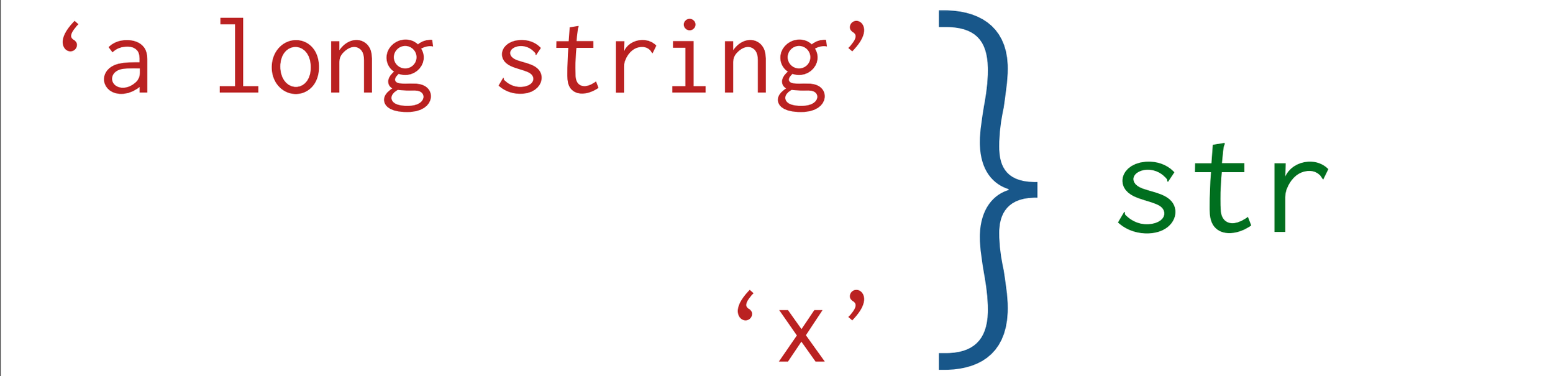
Sequence	Meaning
`\newline`	Backslash and newline ignored
`\\`	Backslash (\)
`\'`	Single quote (')
`\a`	ASCII Bell (BEL)
`\b`	ASCII Backspace (BS)
`\f`	ASCII Formfeed (FF)
`\n`	ASCII Linefeed (L
`\r`	
	Character with hex value hh
Only recognized in string literals	
`\N{name}`	Character named name in the Unicode database
`\uxxxx`	Character with 16-bit hex value xxxx
`\Uxxxxxxxx`	Character with 32-bit hex value xxxxxxxx

docs.python.org/3/reference/lexical_analysis.html#strings

No Separate Character Type

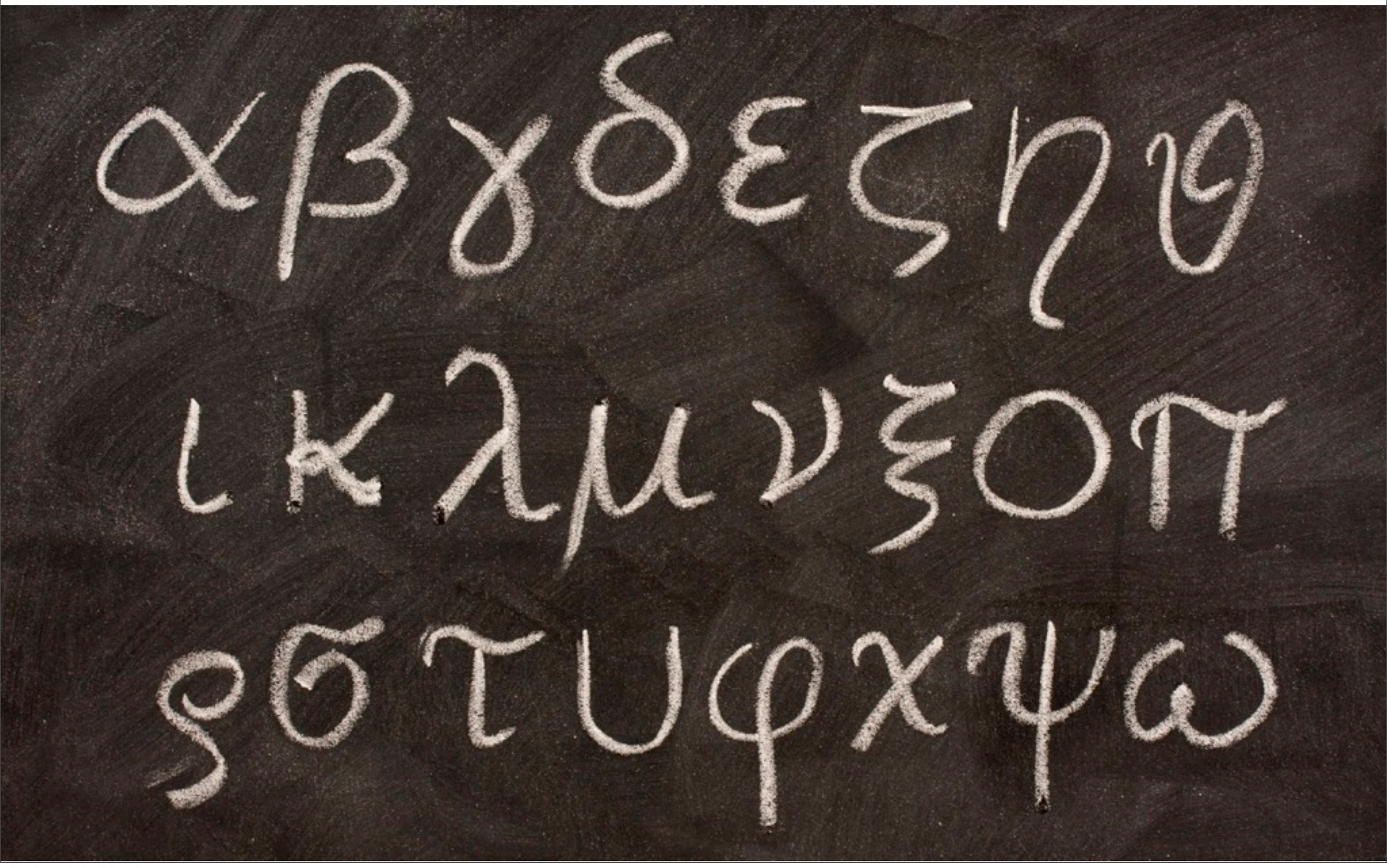


No Separate Character Type



"characters" are simply one-element strings

Python Strings Are Unicode



Python Strings Are Unicode

bytes

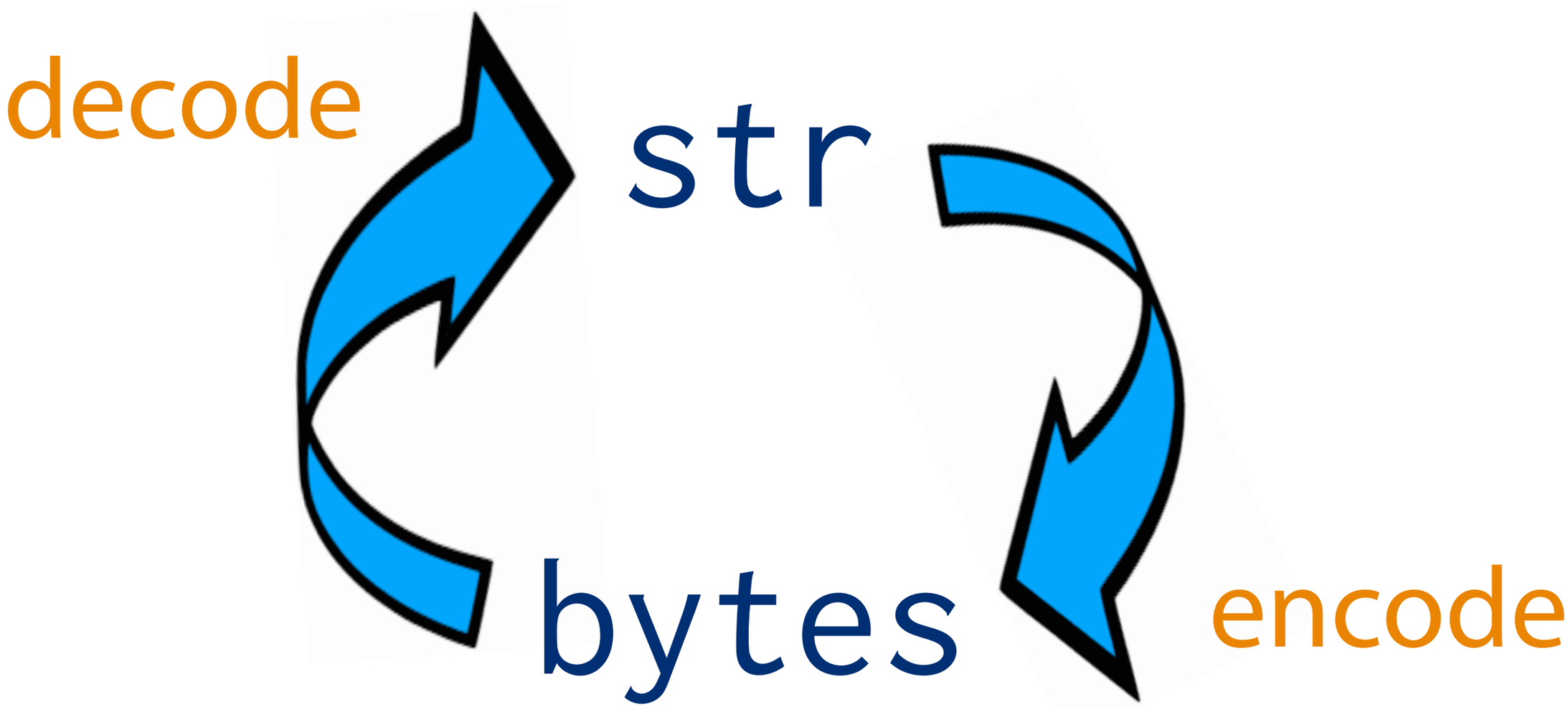
immutable sequences of bytes

Bytes Literals

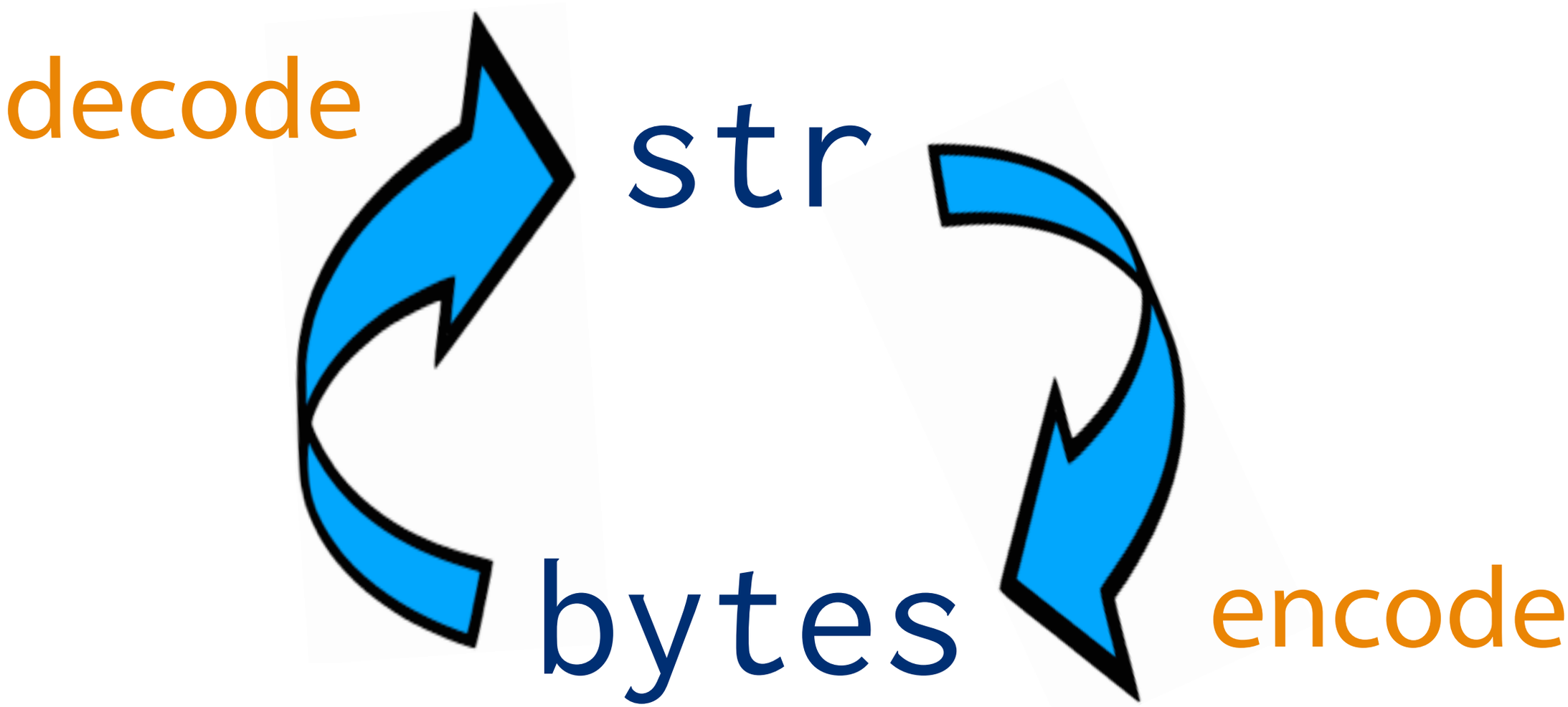
```
b‘data’
```

```
b“data”
```

Converting Between Strings and Bytes



Converting Between Strings and Bytes



Encodings

docs.python.org/3/library/codecs.html#standard-encodings

list

mutable sequences of objects

List Literals

```
[a, b, c, d]
```

dict

mutable mappings of keys to values

Dict Literals

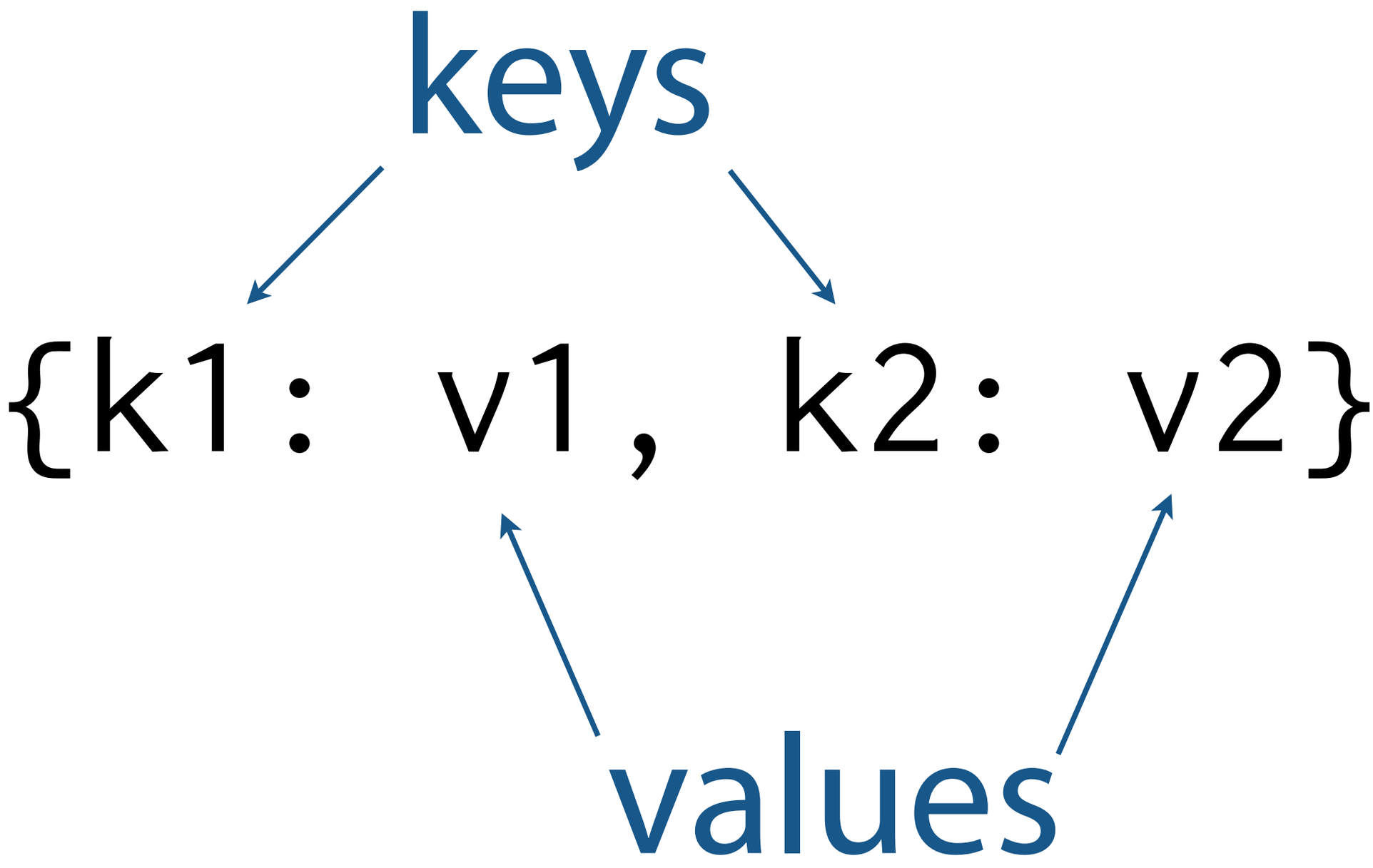
```
{k1: v1, k2: v2}
```

Dict Literals

keys

```
{k1: v1, k2: v2}
```

Dict Literals



for-loop

visit each item in a sequence

For-Loop Syntax

```
for ITEM in SEQUENCE:
    ...body...
```

Exit the REPL

Ctrl-Z

Ctrl-D

Putting it all together


Putting it all together
• urllib.urlopen()


Putting it all together
• urllib.urlopen()
• with-statement


Putting it all together
• urllib.urlopen()
• with-statement
• list


Putting it all together
• urllib.urlopen()
• with-statement
• list
• for-loop

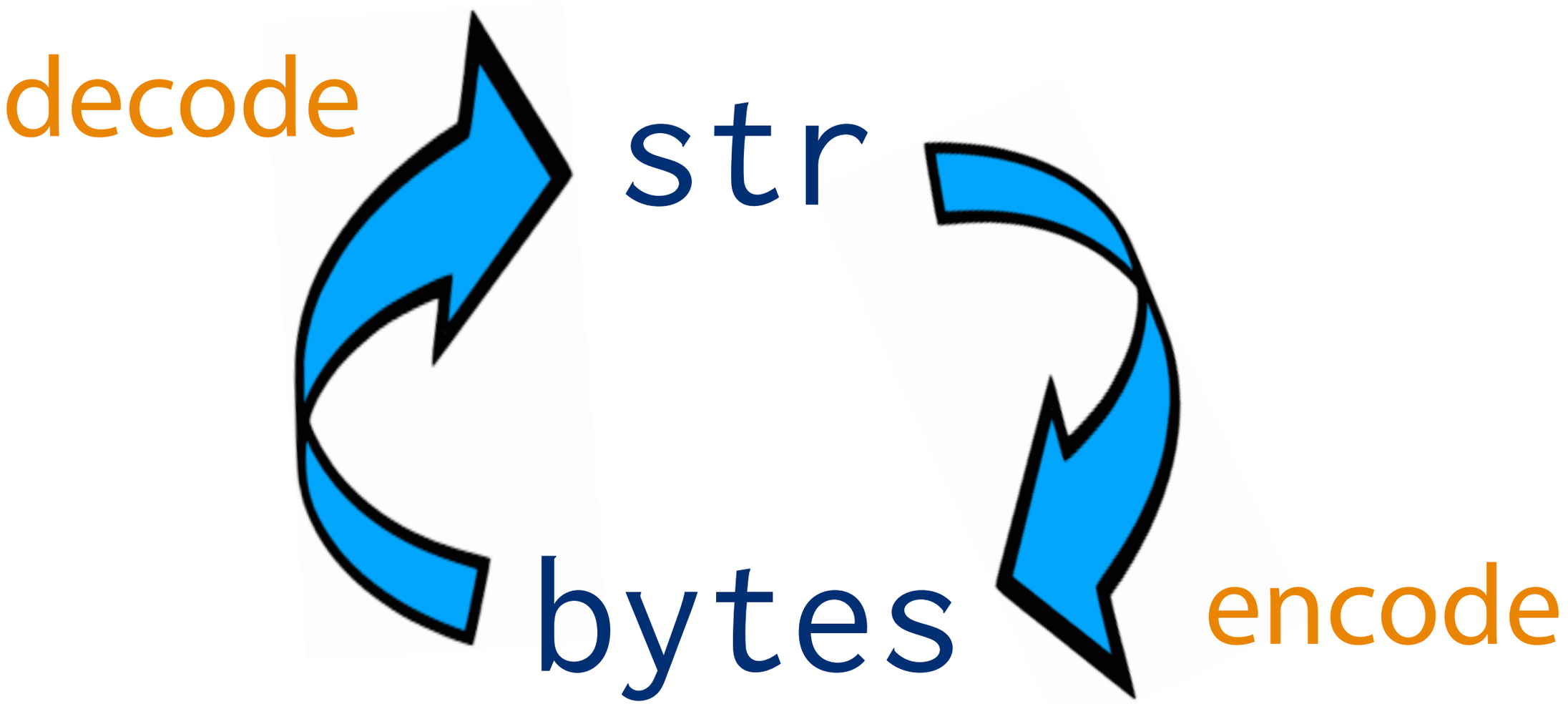

Putting it all together
• urllib.urlopen()
• with-statement
• list
• for-loop
• bytes.split()

Recall: Bytes Literals

```
b‘data’
```

```
b“data”
```

Recall: Converting Between Strings and Bytes




Putting it all together
• urllib.urlopen()
• with-statement
• list
• for-loop
• bytes.split()
• bytes.decode()


Putting it all together
• urllib.urlopen()
• with-statement
• list
• for-loop
• bytes.split()
• bytes.decode()
• str.split()

Summary: Strings and Bytes

Summary: Strings and Bytes

- Single- and multi-line string quoting

Summary: Strings and Bytes

- Single- and multi-line string quoting
- Adjacent string literal concatenation

Summary: Strings and Bytes

- Single- and multi-line string quoting
- Adjacent string literal concatenation
- Universal newlines

Summary: Strings and Bytes

- Single- and multi-line string quoting
- Adjacent string literal concatenation
- Universal newlines
- Escape sequences for control characters

Summary: Strings and Bytes

- Single- and multi-line string quoting
- Adjacent string literal concatenation
- Universal newlines
- Escape sequences for control characters
- Raw strings suppress the escaping mechanism

Summary: Strings and Bytes

- Single- and multi-line string quoting
- Adjacent string literal concatenation
- Universal newlines
- Escape sequences for control characters
- Raw strings suppress the escaping mechanism
- Convert other types with the `str()` constructor

Summary: Strings and Bytes

- Single- and multi-line string quoting
- Adjacent string literal concatenation
- Universal newlines
- Escape sequences for control characters
- Raw strings suppress the escaping mechanism
- Convert other types with the `str()` constructor
- Zero-based square-bracket indexing of strings

Summary: Strings and Bytes

- Single- and multi-line string quoting
- Adjacent string literal concatenation
- Universal newlines
- Escape sequences for control characters
- Raw strings suppress the escaping mechanism
- Convert other types with the `str()` constructor
- Zero-based square-bracket indexing of strings
- Rich variety of string methods

Summary: Strings and Bytes

- Single- and multi-line string quoting
- Adjacent string literal concatenation
- Universal newlines
- Escape sequences for control characters
- Raw strings suppress the escaping mechanism
- Convert other types with the `str()` constructor
- Zero-based square-bracket indexing of strings
- Rich variety of string methods
- Python 3 source encoding is UTF-8I

Summary: Strings and Bytes

- Single- and multi-line string quoting
- Adjacent string literal concatenation
- Universal newlines
- Escape sequences for control characters
- Raw strings suppress the escaping mechanism
- Convert other types with the `str()` constructor
- Zero-based square-bracket indexing of strings
- Rich variety of string methods
- Python 3 source encoding is UTF-8I
- `bytes` is a sequence of bytes, `str` is a sequence of Unicode codepoints

Summary: Strings and Bytes

- Single- and multi-line string quoting
- Adjacent string literal concatenation
- Universal newlines
- Escape sequences for control characters
- Raw strings suppress the escaping mechanism
- Convert other types with the `str()` constructor
- Zero-based square-bracket indexing of strings
- Rich variety of string methods
- Python 3 source encoding is UTF-8I
- `bytes` is a sequence of bytes, `str` is a sequence of Unicode codepoints
- `bytes` literals prefixed with a lowercase `b`

Summary: Strings and Bytes

- Single- and multi-line string quoting
- Adjacent string literal concatenation
- Universal newlines
- Escape sequences for control characters
- Raw strings suppress the escaping mechanism
- Convert other types with the `str()` constructor
- Zero-based square-bracket indexing of strings
- Rich variety of string methods
- Python 3 source encoding is UTF-8I
- `bytes` is a sequence of bytes, `str` is a sequence of Unicode codepoints
- `bytes` literals prefixed with a lowercase `b`
- Convert `str` to `bytes` with `encode()`, `bytes` to `str` with `decode()`

Summary: Lists

```
[a, b, c, d]
```

Summary: Lists

- Lists are mutable, heterogeneous sequences of objects

```
[a, b, c, d]
```

Summary: Lists

- Lists are mutable, heterogeneous sequences of objects
- List literals delimited by square brackets, items separated by commas

```
[a, b, c, d]
```

Summary: Lists

- Lists are mutable, heterogeneous sequences of objects
- List literals delimited by square brackets, items separated by commas
- Zero-based, square-bracket indexing to retrieve objects

```
[a, b, c, d]
```

Summary: Lists

- Lists are mutable, heterogeneous sequences of objects
- List literals delimited by square brackets, items separated by commas
- Zero-based, square-bracket indexing to retrieve objects
- Square-bracket assignment to replace objects

```
[a, b, c, d]
```

Summary: Lists

- Lists are mutable, heterogeneous sequences of objects
- List literals delimited by square brackets, items separated by commas
- Zero-based, square-bracket indexing to retrieve objects
- Square-bracket assignment to replace objects
- Grow lists with `append()`

```
[a, b, c, d]
```

Summary: Lists

- Lists are mutable, heterogeneous sequences of objects
- List literals delimited by square brackets, items separated by commas
- Zero-based, square-bracket indexing to retrieve objects
- Square-bracket assignment to replace objects
- Grow lists with `append()`
- Construct from other sequences using `list()` constructor

```
[a, b, c, d]
```

Summary: Dictionaries

```
{k1: v1, k2: v2}
```

Summary: Dictionaries

- Dictionaries associate keys with values

```
{k1: v1, k2: v2}
```

Summary: Dictionaries

- Dictionaries associate keys with values
- Literal dicts delimited by curly braces

```
{k1: v1, k2: v2}
```

Summary: Dictionaries

- Dictionaries associate keys with values
- Literal dicts delimited by curly braces
- Literal key-value pairs separated by commas, with a colon between each key and value

```
{k1: v1, k2: v2}
```

Summary: For-loops

```
for ITEM in SEQUENCE:
    ...body...
```

Summary: For-loops

- Take items one-by-one from an iterable object, binding a name to the current item

```
for ITEM in SEQUENCE:
    ...body...
```

Summary: For-loops

- Take items one-by-one from an iterable object, binding a name to the current item
- Correspond to *for-each* loops in other languages

```
for ITEM in SEQUENCE:
    ...body...
```

- Tak… curr… …e
- Cor…

Next time in Python Fundamentals

- functions
- modules
- execution model
- command-line arguments