[220 / 319] Iterators and comprehensions

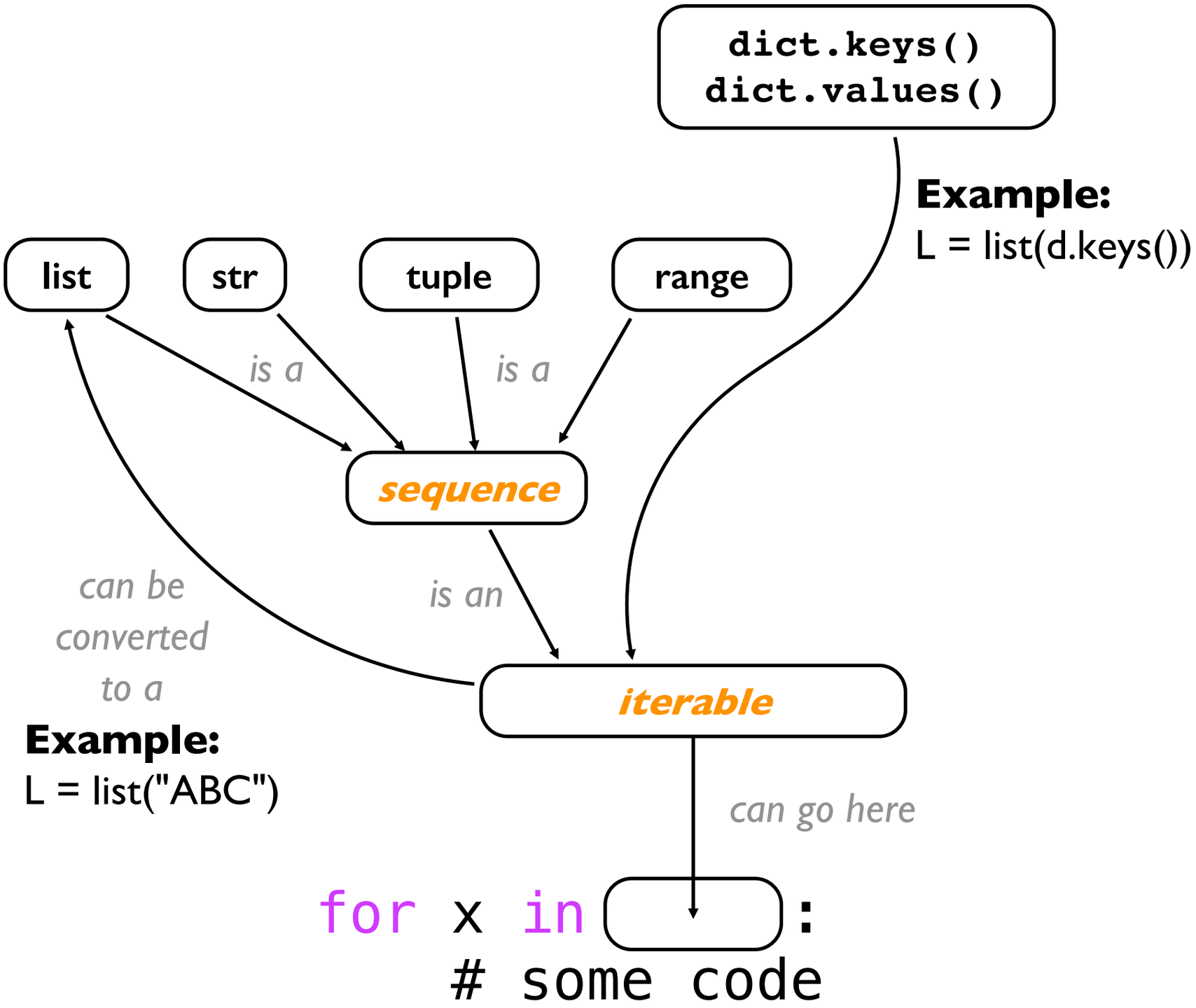
Meena Syamkumar
Andy Kuemmel

Iterators and comprehensions

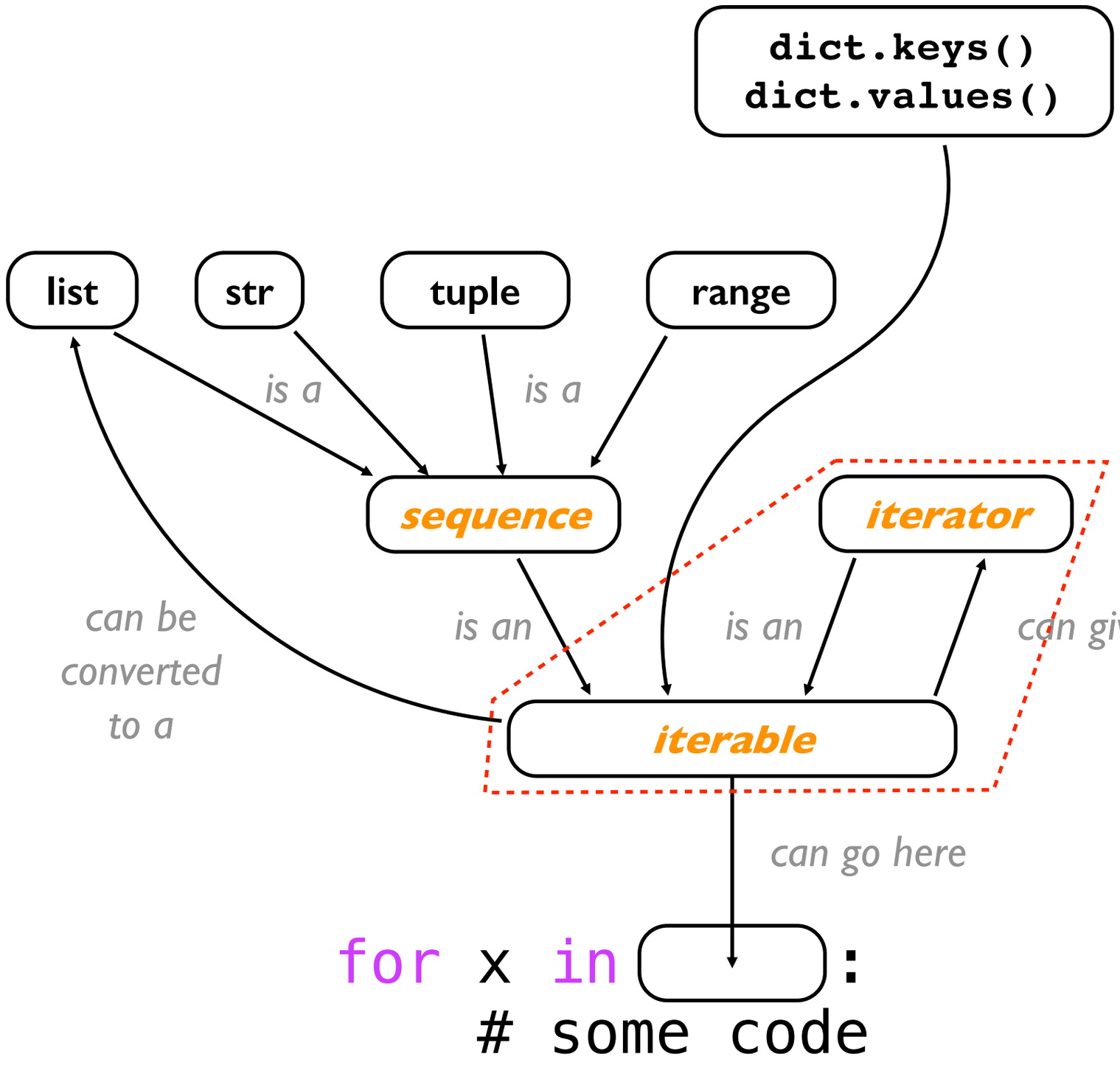
Outline

- review problems
  - recursion
  - function object
  - sorted / .sort() + lambda
- the scary vocabulary of iteration
  - notebook examples
- comprehensions
  - list comprehensions
  - dict comprehensions
  - tuple unpacking
  - notebook examples
- the open function

The Vocabulary of Iteration



The Vocabulary of Iteration



Example:

```
it = iter("ABC")
first = next(it)
```

let's differentiate these better...

is `x` iterable?

if this works, then yes:

`iter(x)` **returns an iterator over x**

is `y` an iterator?

if this works, then yes:

`next(y)` **returns next value from y**

is `x` iterable?

if this works, then yes:

`y = iter(x)` **returns an iterator over x**

is `y` an iterator?

if this works, then yes:

`next(y)` **returns next value from y**

Notebook examples: Can you classify x, y, and z?

```
x = [1, 2, 3]
y = enumerate(['A', 'B', 'C'])
z = 3
```

Things to try:

```
iter(x)
next(x)
```

etc.

Iterators

Outline

- review problems
  - recursion
  - function object
  - sorted / .sort() + lambda
- the scary vocabulary of iteration
  - notebook examples
- comprehensions
  - list comprehensions
  - dict comprehensions
  - tuple unpacking
  - notebook examples
- the open function

List and dict comprehensions – basic syntax

Enable you to generate new lists and dictionaries

```
new_list = [expression for val in iterable if
conditional_expression]
```

```
new_list = [expression if conditional_expression else
alternate_expression for val in iterable ]
```

```
{key: val for val in iterable if condition}
```

```
dict([expression for val in iterable if condition])
```

Iterators and comprehensions

Outline

- review problems
  - recursion
  - function object
  - sorted / .sort() + lambda
- the scary vocabulary of iteration
  - notebook examples
- comprehensions
  - list comprehensions
  - dict comprehensions
  - tuple unpacking
  - notebook examples
- the open function

Reading Files

```
path = “file.txt”
f = open(path)
```

open(…) function is built in

Reading Files

```
path = “file.txt”
f = open(path)
```

it takes a string argument,
which contains path to a file

file.txt

```
This is a test!
3
2
1
Go!
```

c:\users\meena\my-doc.txt

/var/log/events.log

../data/input.csv

Reading Files

```
path = “file.txt”
f = open(path)
```

it returns a file object

file.txt

```
This is a test!
3
2
1
Go!
```

file objects are iterators!

Reading Files

```
path = “file.txt”
f = open(path)

for line in f:
    print(line)
```

file.txt

```
This is a test!
3
2
1
Go!
```

Output

This is a test!

3

2

1

Go!