algorithms and data structures

Algorithms and Data Structures programming in Python revisited sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serialization

defining data structures, for example: a set using the Pickle module an application to network programming

> MCS 275 Lecture 36 Programming Tools and File Management Jan Verschelde, 14 April 2008

・ロト・西ト・山田・山田・山下

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serialization

defining data structures, for example: a set using the Pickle module

algorithms and data structures

Algorithms and Data Structures programming in Python revisited sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serialization

defining data structures, for example: a set using the Pickle module an application to network programming

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, ists

Persistent Data

storing information between executions using DBM files

Object Serializati

defining data structures, for example: a set using the Pickle module

programming in Python revisited

Niklaus Wirth: programs = algorithms + data structures

Three basic control structures in any algorithm:

- 1. sequence of statements
- 2. conditional statement: if else
- 3. iteration: while and for loop

For every control structure,

we have a matching data structure:

	control structures	data structures
1	sequence	tuple
2	if else	dictionary
3	while / for	list

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

equences, dictionaries, ists

Persistent Data

storing information between executions using DBM files

Object Serialization

defining data structures, for example: a set using the Pickle module

programming in Python revisited

Niklaus Wirth: programs = algorithms + data structures

Three basic control structures in any algorithm:

- 1. sequence of statements
- 2. conditional statement: if else
- 3. iteration: while and for loop
- For every control structure,
- we have a matching data structure:

	control structures	data structures
1	sequence	tuple
2	if else	dictionary
3	while / for	list

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

equences, dictionaries, ists

Persistent Data

storing information between executions using DBM files

Object Serializatio

defining data structures, for example: a set using the Pickle module

an application to network programming

・ロト・西ト・山田・山田・山下

programming in Python revisited

Niklaus Wirth: programs = algorithms + data structures

Three basic control structures in any algorithm:

- 1. sequence of statements
- 2. conditional statement: if else
- 3. iteration: while and for loop

For every control structure,

we have a matching data structure:

	control structures	data structures
1	sequence	tuple
2	if else	dictionary
3	while / for	list

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, ists

Persistent Data

storing information between executions using DBM files

Object Serializatio

defining data structures, for example: a set using the Pickle module

an application to network programming

programming in Python revisited

Niklaus Wirth: programs = algorithms + data structures

Three basic control structures in any algorithm:

- 1. sequence of statements
- 2. conditional statement: if else
- 3. iteration: while and for loop

For every control structure, we have a matching data structure:

	control structures	data structures
1	sequence	tuple
2	if else	dictionary
3	while / for	list

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, ists

Persistent Data

storing information between executions using DBM files

Object Serializati

defining data structures, for example: a set using the Pickle module

programming in Python revisited

Niklaus Wirth: programs = algorithms + data structures

Three basic control structures in any algorithm:

- 1. sequence of statements
- 2. conditional statement: if else
- 3. iteration: while and for loop

For every control structure, we have a matching data structure:

	control structures	data structures
1	sequence	tuple
2	if else	dictionary
3	while / for	list

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, ists

Persistent Data

storing information between executions using DBM files

Object Serializatio

defining data structures, for example: a set using the Pickle module

an application to network programming

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □

programming in Python revisited

Niklaus Wirth: programs = algorithms + data structures

Three basic control structures in any algorithm:

- 1. sequence of statements
- 2. conditional statement: if else
- 3. iteration: while and for loop

For every control structure,

we have a matching data structure:

	control structures	data structures
1	sequence	tuple
2	if else	dictionary
3	while / for	list

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● ● ● ● ●

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, ists

Persistent Data

storing information between executions using DBM files

Object Serializatio

defining data structures, for example: a set using the Pickle module

programming in Python revisited

Niklaus Wirth: programs = algorithms + data structures

Three basic control structures in any algorithm:

- 1. sequence of statements
- 2. conditional statement: if else
- 3. iteration: while and for loop

For every control structure,

we have a matching data structure:

	control structures	data structures
1	sequence	tuple
2	if else	dictionary
3	while / for	list

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, ists

Persistent Data

storing information between executions using DBM files

Object Serializatio

defining data structures, for example: a set using the Pickle module

an application to network programming

programming in Python revisited

Niklaus Wirth: programs = algorithms + data structures

Three basic control structures in any algorithm:

- 1. sequence of statements
- 2. conditional statement: if else
- 3. iteration: while and for loop

For every control structure,

we have a matching data structure:

	control structures	data structures
1	sequence	tuple
2	if else	dictionary
3	while / for	list

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● ● ● ● ●

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, ists

Persistent Data

storing information between executions using DBM files

Object Serializati

defining data structures, for example: a set using the Pickle module

programming in Python revisited

Niklaus Wirth: programs = algorithms + data structures

Three basic control structures in any algorithm:

- 1. sequence of statements
- 2. conditional statement: if else
- 3. iteration: while and for loop

For every control structure,

we have a matching data structure:

	control structures	data structures
1	sequence	tuple
2	if else	dictionary
3	while / for	list

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, ists

Persistent Data

storing information between executions using DBM files

Object Serializatio

defining data structures, for example: a set using the Pickle module

algorithms and data structures

Algorithms and Data Structures

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serialization

defining data structures, for example: a set using the Pickle module an application to network programming

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

rogramming in Python evisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serializatio

defining data structures, for example: a set using the Pickle module

tuples as sequences manipulated by functions

All data are sequences of bits, or bit tuples.

Swapping values:

```
>>> a = 1
>>> b = 2
>>> (b,a) = (a,b)
>>> b
1
>>> a
2
```

Functions take sequences of arguments on input and return sequences on output.



14 April 2008

Algorithms and Data Structures

rogramming in Python evisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serializatio

defining data structures, for example: a set using the Pickle module

an application to network programming

tuples as sequences manipulated by functions

All data are sequences of bits, or bit tuples.

Swapping values:

```
>>> a = 1
>>> b = 2
>>> (b,a) = (a,b)
>>> b
1
>>> a
2
```

Functions take sequences of arguments on input and return sequences on output.



14 April 2008

Algorithms and Data Structures

evisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serializatio

defining data structures, for example: a set using the Pickle module

an application to network programming

tuples as sequences manipulated by functions

All data are sequences of bits, or bit tuples.

Swapping values:

Functions take sequences of arguments on input and return sequences on output.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

rogramming in Python evisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serializatio

defining data structures, for example: a set using the Pickle module

an application to network programming

tuples as sequences manipulated by functions

All data are sequences of bits, or bit tuples.

Swapping values:

Functions take sequences of arguments on input and return sequences on output.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

rogramming in Python evisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serialization

defining data structures, for example: a set using the Pickle module

Storing Conditions

dictionaries and if else statements

We can represent an if tt else statement

```
>>> import time
>>> hour = time.localtime()[3]
>>> if hour < 12:
... print 'good morning'
... else:
... print 'good afternoon'
...
good afternoon</pre>
```

```
via a dictionary:
>>> d = { True:'good morning',
... False : 'good afternoon'}
>>> d[hour<12]
'good afternoon'
```

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

rogramming in Python evisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serializatio

defining data structures, for example: a set using the Pickle module

Storing Conditions

dictionaries and if else statements

We can represent an if tt else statement

```
>>> import time
>>> hour = time.localtime()[3]
>>> if hour < 12:
... print 'good morning'
... else:
... print 'good afternoon'
...
good afternoon
via a dictionary:</pre>
```

>>> d = { True:'good morning', ... False : 'good afternoon'} >>> d[hour<12] 'good afternoon'

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python evisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serializatio

defining data structures, for example: a set using the Pickle module

Loops and Lists storing the results of a for loop

Printing all lower case characters:

```
>>> for i in range(ord('a'),ord('z')):
... print chr(i)
```

A list of all lower case characters:

```
>>> L = range(ord('a'),ord('z'))
>>> map(chr,L)
```

map() returns a list of the results of applying a function to a sequence of arguments.

The while statement combines for with if else: conditional iteration.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

rogramming in Python evisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serializatior

defining data structures, for example: a set using the Pickle module an application to network

an application to network programming

storing the results of a for loop

Printing all lower case characters:

>>> for i in range(ord('a'),ord('z')):
... print chr(i)

A list of all lower case characters:

```
>>> L = range(ord('a'),ord('z'))
>>> map(chr,L)
```

map() returns a list of the results of applying a function to a sequence of arguments.

The while statement combines for with if else: conditional iteration.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python evisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serialization

defining data structures, for example: a set using the Pickle module

an application to network programming

storing the results of a for loop

Printing all lower case characters:

```
>>> for i in range(ord('a'),ord('z')):
... print chr(i)
```

A list of all lower case characters:

```
>>> L = range(ord('a'),ord('z'))
>>> map(chr,L)
```

map() returns a list of the results of applying a function to a sequence of arguments.

The while statement combines for with if else: conditional iteration.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

rogramming in Python evisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serialization

defining data structures, for example: a set using the Pickle module an application to network

an application to network programming

・ロト・西ト・山田・山田・山下

storing the results of a for loop

Printing all lower case characters:

```
>>> for i in range(ord('a'),ord('z')):
... print chr(i)
```

A list of all lower case characters:

```
>>> L = range(ord('a'),ord('z'))
>>> map(chr,L)
```

map() returns a list of the results of applying a function to a sequence of arguments.

The while statement combines for with if else: conditional iteration.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

rogramming in Python evisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serializatio

defining data structures, for example: a set using the Pickle module

an application to network programming

storing the results of a for loop

Printing all lower case characters:

```
>>> for i in range(ord('a'),ord('z')):
... print chr(i)
```

A list of all lower case characters:

```
>>> L = range(ord('a'),ord('z'))
>>> map(chr,L)
```

map() returns a list of the results of applying a function to a sequence of arguments.

The while statement combines for with if else: conditional iteration.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python evisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serialization

defining data structures, for example: a set using the Pickle module

an application to network programming

・ロト・西ト・山田・山田・山下

storing the results of a for loop

Printing all lower case characters:

```
>>> for i in range(ord('a'),ord('z')):
... print chr(i)
```

A list of all lower case characters:

```
>>> L = range(ord('a'),ord('z'))
>>> map(chr,L)
```

map() returns a list of the results of applying a function to a sequence of arguments.

The while statement combines for with if else: conditional iteration.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

rogramming in Python evisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serializatio

defining data structures, for example: a set using the Pickle module

◆□▶ ◆□▶ ◆三▶ ◆三▶ ・三 のへぐ

defining lists in a short way

Instead of map(), filter(), etc... (eventually with lambda functions), *list comprehensions* provide a shorter way to create lists:

To sample integer points on the parabola $y = x^2$:

>>> [(x,x**2) for x in range(0,3)] [(0, 0), (1, 1), (2, 4)]

Generating three random numbers:

>>> from random import uniform
>>> L = [uniform(0,1) for i in range(0,3)]
>>> ['%.3f' % x for x in L]
['0.843', '0.308', '0.272']

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

rogramming in Python evisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serialization

defining data structures, for example: a set using the Pickle module an application to network

programming

defining lists in a short way

Instead of map(), filter(), etc... (eventually with lambda functions), *list comprehensions* provide a shorter way to create lists:

To sample integer points on the parabola $y = x^2$:

>>> [(x,x**2) for x in range(0,3)]
[(0, 0), (1, 1), (2, 4)]

Generating three random numbers:

>>> from random import uniform
>>> L = [uniform(0,1) for i in range(0,3)]
>>> ['%.3f' % x for x in L]
['0.843', '0.308', '0.272']

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

rogramming in Python wisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serialization

defining data structures, for example: a set using the Pickle module an application to network

defining lists in a short way

Instead of map(), filter(), etc... (eventually with lambda functions), *list comprehensions* provide a shorter way to create lists:

To sample integer points on the parabola $y = x^2$:

>>> [(x,x**2) for x in range(0,3)]
[(0, 0), (1, 1), (2, 4)]

Generating three random numbers:

>>> from random import uniform
>>> L = [uniform(0,1) for i in range(0,3)]
>>> ['%.3f' % x for x in L]
['0.843', '0.308', '0.272']

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

rogramming in Python evisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serialization

defining data structures, for example: a set using the Pickle module an application to network

defining lists in a short way

Instead of map(), filter(), etc... (eventually with lambda functions), *list comprehensions* provide a shorter way to create lists:

To sample integer points on the parabola $y = x^2$:

>>> [(x,x**2) for x in range(0,3)]
[(0, 0), (1, 1), (2, 4)]

Generating three random numbers:

>>> from random import uniform
>>> L = [uniform(0,1) for i in range(0,3)]
>>> ['%.3f' % x for x in L]
['0.843', '0.308', '0.272']

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

rogramming in Python evisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serialization

defining data structures, for example: a set using the Pickle module an application to network

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 - のへで

algorithms and data structures

Algorithms and Data Structures programming in Python revisited sequences, dictionaries, lists

Persistent Data storing information between executions using DBM files

Object Serialization

defining data structures, for example: a set using the Pickle module an application to network programming

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

lists

Persistent Data

storing information between executions

using DBM files

Object Serializati

defining data structures, for example: a set using the Pickle module

an application to network programming

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □

storing information between executions

Data that is *persistent* outlives programs.

Objects constructed by a script are lost as soon as the script ends.

Two extremes to make data persistent:

- 1. files: store string representations,
- 2. MySQL: store data in tables in a database.

Intermediate solution: DBM files.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

lists

storing information between executions

using DBM files

Object Serializatio

defining data structures, for example: a set using the Pickle module

an application to network programming

・ロト・西ト・山田・山田・山下

storing information between executions

Data that is *persistent* outlives programs.

Objects constructed by a script are lost as soon as the script ends.

Two extremes to make data persistent:

- 1. files: store string representations,
- 2. MySQL: store data in tables in a database.

Intermediate solution: DBM files.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited sequences dictionaries

lists

Persistent Data

storing information between executions

using DBM files

Object Serializatio

defining data structures, for example: a set using the Pickle module

an application to network programming

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □

storing information between executions

Data that is *persistent* outlives programs.

Objects constructed by a script are lost as soon as the script ends.

Two extremes to make data persistent:

- 1. files: store string representations,
- 2. MySQL: store data in tables in a database.

Intermediate solution: DBM files.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions

using DBM files

Object Serializatio

defining data structures, for example: a set using the Pickle module

an application to network programming

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □

storing information between executions

Data that is *persistent* outlives programs.

Objects constructed by a script are lost as soon as the script ends.

Two extremes to make data persistent:

- 1. files: store string representations,
- 2. MySQL: store data in tables in a database.

Intermediate solution: DBM files.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

Persistent Data

storing information between executions

using DBM files

Object Serializatio

defining data structures, for example: a set using the Pickle module

an application to network programming

storing information between executions

Data that is *persistent* outlives programs.

Objects constructed by a script are lost as soon as the script ends.

Two extremes to make data persistent:

- 1. files: store string representations,
- 2. MySQL: store data in tables in a database.

Intermediate solution: DBM files.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

Persistent Data

storing information between executions

using DBM files

Object Serializatio

defining data structures, for example: a set using the Pickle module

an application to network programming

algorithms and data structures

Algorithms and Data Structures

programming in Python revisited sequences, dictionaries, lists

Persistent Data storing information between executions using DBM files

Object Serialization

defining data structures, for example: a set using the Pickle module an application to network programming

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python evisited

sequences, dictionaries, ists

Persistent Data

storing information between executions

using DBM files

Object Serialization

defining data structures, for example: a set using the Pickle module

Using DBM Files

DBM files are standard in the Python library.

\$ python

>>> import anydbm

>>> libdb = anydbm.open('library','c')

opened a new dbm with read-write access (flag = ' c')

MCS 275 L-36

14 April 2008

using DBM files

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● ● ● ● ●
Using DBM Files

DBM files are standard in the Python library.

\$ python

>>> import anydbm

>>> libdb = anydbm.open('library','c')

opened a new dbm with read-write access (flag = ' c ')

- >>> libdb['0'] = str({'author':'Rashi Gupta',
- ... 'title':'Making Use of Python'})

keys and values must be of type string

```
>>> libdb.keys()
['0']
>>> libdb.values()
["{'title': 'Making Use of Python', 'au
$ ls
```

ightarrow library is a file in current directory.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions

using DBM files

Object Serialization

```
defining data structures, for
example: a set
using the Pickle module
an application to network
```

```
'Rashi
```

▲□▶▲□▶▲□▶▲□▶ □ ● ●

Using DBM Files

DBM files are standard in the Python library.

```
$ python
```

- >>> import anydbm
- >>> libdb = anydbm.open('library','c')

opened a new dbm with read-write access (flag = ' c ')

```
>>> libdb['0'] = str({'author':'Rashi Gupta',
```

... 'title':'Making Use of Python'})

keys and values must be of type string

```
>>> libdb.keys()
```

```
[′0′]
```

```
>>> libdb.values()
```

["{'title': 'Making Use of Python', 'author': 'Rashi \$ ls

 \rightarrow library is a file in current directory.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions

```
using DBM files
```

Object Serialization

```
defining data structures, for
example: a set
using the Pickle module
an application to network
```

```
◆□ > ◆□ > ◆豆 > ◆豆 >  ̄豆 → ���
```

Using DBM Files

DBM files are standard in the Python library.

```
$ python
```

- >>> import anydbm
- >>> libdb = anydbm.open('library','c')

opened a new dbm with read-write access (flag = ' c ')

```
>>> libdb['0'] = str({'author':'Rashi Gupta',
```

... 'title':'Making Use of Python'})

keys and values must be of type string

```
>>> libdb.keys()
['0']
>>> libdb.values()
["{'title': 'Making Use of Python', 'author': 'Rashi of
$ ls
```

 \rightarrow library is a file in current directory.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions

```
using DBM files
```

```
Object
Serialization
```

```
defining data structures, for
example: a set
using the Pickle module
an application to network
```

▲□▶▲□▶▲□▶▲□▶ □ ● ●

and selecting books using the key

```
>>> import anydbm
>>> mylib = anydbm.open('library','c')
>>> mylib.keys()
['0']
>>> mylib['1'] = str({'author':'S. Ceri e
... 'title':'The Art and Craft of Computi
>>> mylib.values()
["{'title': 'Making Use of Python', 'author':
    "{'title': 'The Art and Craft of Computing',
```

Selecting the author of book with key 1:

```
>>> V = mylib.values()
>>> d = V[int(mylib.keys()[1])]
>>> eval(d)['author']
'S. Ceri et al.'
```

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions

```
using DBM files
```

Object Serialization

defining data structures, for example: a set using the Pickle module an application to network programming

▲□▶▲□▶▲□▶▲□▶ □ ● ●

and selecting books using the key

```
>>> import anydbm
>>> mylib = anydbm.open('library','c')
>>> mylib.keys()
['0']
>>> mylib['1'] = str({'author':'S. Ceri et al
... 'title':'The Art and Craft of Computing'}
>>> mylib.values()
["{'title': 'Making Use of Python', 'author': 'Ra an application be
regularized by
object
>>> mylib.values()
["{'title': 'The Art and Craft of Computing', 'author': 'Ra an application be
regularized by
object
>>> mylib.values()
```

Selecting the author of book with key 1:

```
>>> V = mylib.values()
>>> d = V[int(mylib.keys()[1])]
>>> eval(d)['author']
'S. Ceri et al.'
```

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □

MCS 275 L-36

and selecting books using the key

```
>>> import anydbm
>>> mylib = anydbm.open('library','c')
>>> mylib.keys()
['0']
>>> mylib['1'] = str({'author':'S. Ceri et al
... 'title':'The Art and Craft of Computing'}
>>> mylib.values()
["{'title': 'Making Use of Python', 'author': 'Rashic.Gupta
"{'title': 'The Art and Craft of Computing', 'author': 'S.
```

Selecting the author of book with key 1:

```
>>> V = mylib.values()
>>> d = V[int(mylib.keys()[1])]
>>> eval(d)['author']
'S. Ceri et al.'
```

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □

MCS 275 L-36

and selecting books using the key

```
>>> import anydbm
>>> mylib = anydbm.open('library','c')
>>> mylib.keys()
['0']
>>> mylib['1'] = str({'author':'S. Ceri et al
... 'title':'The Art and Craft of Computing'}
>>> mylib.values()
["{'title': 'Making Use of Python', 'author': 'Rashic«Gupta
"{'title': 'The Art and Craft of Computing', 'author': 'S.
```

Selecting the author of book with key 1:

```
>>> V = mylib.values()
>>> d = V[int(mylib.keys()[1])]
>>> eval(d)['author']
'S. Ceri et al.'
```

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □

MCS 275 L-36

and selecting books using the key

```
>>> import anydbm
>>> mylib = anydbm.open('library','c')
>>> mylib.keys()
['0']
>>> mylib['1'] = str({'author':'S. Ceri et al
... 'title':'The Art and Craft of Computing'}
>>> mylib.values()
["{'title': 'Making Use of Python', 'author': 'Rashic«Gupta
"{'title': 'The Art and Craft of Computing', 'author': 'S.
```

Selecting the author of book with key 1:

```
>>> V = mylib.values()
>>> d = V[int(mylib.keys()[1])]
>>> eval(d)['author']
'S. Ceri et al.'
```

▲□▶▲□▶▲□▶▲□▶ = のへで

MCS 275 L-36

an overview

MCS 275 L-36

14 April 2008

files

description	sequences
load module anydbm	Persiste
create or open dbm	storing info executions
file with name n	using DBM
assign value for key	
load value for key	defining da example: a
number of entries stored	an applicati
see if entry for key	F - 3
remove entry for key	
close dbm file	
	description load module anydbm create or open dbm file with name n assign value for key load value for key number of entries stored see if entry for key remove entry for key close dbm file

- values are dictionaries, stored as strings

an overview

MCS 275 L-36

14 April 2008

BS

<pre>import anydbm f = anydbm.open('n','c') f['key'] = 'value' value = f['key'] count = len(f) found = f.has_key('key') del f['key'] f.close()</pre> load module anydbm create or open dbm create or open dbm file with name n objet stratage datage create or open dbm create or open dbm file with name n objet stratage datage create or open dbm file with name n objet stratage create or open dbm file with name n objet stratage create or open dbm file with name n objet stratage datage create or open dbm file with name n objet stratage create or open dbm file with name n objet stratage create or open dbm file with name n objet stratage create or open dbm file with name n objet stratage create or open dbm file with name n objet stratage create or open dbm file with name n objet stratage create or open dbm file with name n objet stratage create or open dbm file stratage fi	Python code	description	sequences, d lists
<pre>f = anydbm.open('n','c') f = anydbm.open('n','c') f = anydbm.open('n','c') f = 'value' value = f['key'] count = len(f) found = f.has_key('key') del f['key'] f.close()</pre> create or open dbm file with name n assign value for key load value for key remove entry for key close dbm file	import anydbm	load module anydbm	Persistent
file with name nusing DBM filef['key'] = 'value'assign value for keyvalue = f['key']load value for keycount = len(f)number of entries storedfound = f.has_key('key')see if entry for keydel f['key']remove entry for keyf.close()close dbm file	<pre>f = anydbm.open('n','c')</pre>	create or open dbm	storing inform executions
<pre>f['key'] = 'value' value = f['key'] count = len(f) found = f.has_key('key') del f['key'] f.close()</pre> assign value for key load value for key number of entries stored see if entry for key close dbm file		file with name n	using DBM file
<pre>value = f['key'] count = len(f) found = f.has_key('key') del f['key'] f.close()</pre> load value for key number of entries stored see if entry for key close dbm file	f['key'] = 'value'	assign value for key	
count = len(f)number of entries stored an application programmingfound = f.has_key('key')see if entry for keydel f['key']remove entry for keyf.close()close dbm file	value = f['key']	load value for key	defining data example: a se
found = f.has_key('key')see if entry for keydel f['key']remove entry for keyf.close()close dbm file	count = len(f)	number of entries stored	an application programming
del f['key']remove entry for keyf.close()close dbm file	found = f.has_key('key')	see if entry for key	
f.close() close dbm file	del f['key']	remove entry for key	
	f.close()	close dbm file	

н

- values are dictionaries, stored as strings

an overview

MCS 275 L-36

14 April 2008

using DBM files

У

an application to network

Python code	description
import anydbm	load module anydbm
<pre>f = anydbm.open('n','c')</pre>	create or open dbm
	file with name n
f['key'] = 'value'	assign value for key
<pre>value = f['key']</pre>	load value for key
count = len(f)	number of entries sto
<pre>found = f.has_key('key')</pre>	see if entry for key
del f['key']	remove entry for key
f.close()	close dbm file

Т

- values are dictionaries, stored as strings

dia a substitution

an overview

MCS 275 L-36

Python code	description	sequences, dic lists
import anydbm	load module anydbm	Persistent
<pre>f = anydbm.open('n','c')</pre>	create or open dbm	storing informa executions
	file with name n	using DBM file
f['key'] = 'value'	assign value for key	
<pre>value = f['key']</pre>	load value for key	defining data s example: a set
count = len(f)	number of entries stored	an application to programming
<pre>found = f.has_key('key')</pre>	see if entry for key	F - 0 0
del f['key']	remove entry for key	
f.close()	close dbm file	

- values are dictionaries, stored as strings

import anydbm

Python code

an overview

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited sequences, dictionaries,

storing information between executions

using DBM files

Object Serialization

defining data structures, for example: a set using the Pickle module an application to network programming

Typical use:

- every record in database has unique key
- values are dictionaries, stored as strings

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □

description load module anydbm

f = anydbm.open('n','c')
f = anydbm.open('n','c')
f = 'value'
value = f['key']
count = len(f)
found = f.has_key('key')
del f['key']
f close()
create or open dbm
file with name n
assign value for key
load value for key
see if entry for key
remove entry for key
close dbm file

an overview

MCS 275 L-36

Python code	description	sequences, dict lists
import anydbm	load module anydbm	Persistent [
<pre>f = anydbm.open('n','c')</pre>	create or open dbm	storing informat executions
	file with name n	using DBM files
f['key'] = 'value'	assign value for key	
value = f['key']	load value for key	defining data st example: a set
count = len(f)	number of entries stored	an application to
<pre>found = f.has_key('key')</pre>	see if entry for key	programming
del f['key']	remove entry for key	
f.close()	close dbm file	

- values are dictionaries, stored as strings

an overview

MCS 275 L-36

14 April 2008

iles

Python code	description	sequences,
import anydbm	load module anydbm	Persisten
<pre>f = anydbm.open('n','c')</pre>	create or open dbm	storing inform executions
	file with name n	using DBM f
f['key'] = 'value'	assign value for key	
value = f['key']	load value for key	defining data example: a s
count = len(f)	number of entries stored	an applicatio
<pre>found = f.has_key('key')</pre>	see if entry for key	1.0.
del f['key']	remove entry for key	
f.close()	close dbm file	

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● ● ● ● ●

.

- values are dictionaries, stored as strings

an overview

MCS 275 L-36

14 April 2008

Python code	description	sequences, diction
import anydbm	load module anydbm	Persistent Da
<pre>f = anydbm.open('n','c')</pre>	create or open dbm	storing information executions
	file with name n	using DBM files
f['key'] = 'value'	assign value for key	
value = f['key']	load value for key	defining data struc example: a set
count = len(f)	number of entries stored	an application to n
<pre>found = f.has_key('key')</pre>	see if entry for key	F
del f['key']	remove entry for key	
f.close()	close dbm file	

Typical use:

- every record in database has unique key
- values are dictionaries, stored as strings

an overview

MCS 275 L-36

14 April 2008

Python code	description	sequences, diction
import anydbm	load module anydbm	Persistent Da
<pre>f = anydbm.open('n','c')</pre>	create or open dbm	storing information executions
	file with name n	using DBM files
f['key'] = 'value'	assign value for key	
<pre>value = f['key']</pre>	load value for key	defining data struc example: a set
count = len(f)	number of entries stored	an application to n
<pre>found = f.has_key('key')</pre>	see if entry for key	programming
del f['key']	remove entry for key	
f.close()	close dbm file	

Typical use:

- every record in database has unique key
- values are dictionaries, stored as strings

algorithms and data structures

Algorithms and Data Structures

programming in Python revisited sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serialization defining data structures, for example: a set using the Pickle module an application to network programming

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, ists

Persistent Data

storing information between executions using DBM files

Object

Serialization

defining data structures, for example: a set

using the Pickle module

an application to network programming

▲□▶▲□▶▲□▶▲□▶ □ ● ●

object oriented design of data structures

Builtin data types offer

- storage: object data attributes,
- methods: object functional attributes.

Example: define a class Set. A set is a list without duplicates.

```
>>> from class_set import
>>> E = Set()
>>> E
{}
>>> A = Set(2,5,2)
>>> A
{2,5}
```

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object

defining data structures, for

example: a set

using the Pickle module an application to network

```
・ロト・西ト・山田・山田・山下
```

object oriented design of data structures

Builtin data types offer

- storage: object data attributes,
- methods: object functional attributes.

```
Example: define a class Set.
A set is a list without duplicates.
```

```
>>> from class_set import
>>> E = Set()
>>> E
{}
>>> A = Set(2,5,2)
>>> A
{2,5}
```

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object

Serialization

defining data structures, for example: a set

using the Pickle module an application to network

・ロト・西ト・山田・山田・山下

object oriented design of data structures

Builtin data types offer

- storage: object data attributes,
- methods: object functional attributes.

Example: define a class Set. A set is a list without duplicates.

```
>>> from class_set import
>>> E = Set()
>>> E
{}
>>> A = Set(2,5,2)
>>> A
{2,5}
```

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object

Serialization

defining data structures, for example: a set

using the Pickle module an application to network

programming

・ロト・西ト・山田・山田・山下

object oriented design of data structures

Builtin data types offer

- storage: object data attributes,
- methods: object functional attributes.

Example: define a class Set. A set is a list without duplicates.

```
>>> from class_set import *
>>> E = Set()
>>> E
{}
>>> A = Set(2,5,2)
>>> A
{2,5}
```

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, ists

Persistent Data

storing information between executions using DBM files

Object

Serialization

defining data structures, for example: a set

using the Pickle module an application to network

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □

object oriented design of data structures

Builtin data types offer

- storage: object data attributes,
- methods: object functional attributes.

Example: define a class Set. A set is a list without duplicates.

```
>>> from class_set import *
>>> E = Set()
>>> E
{}
>>> A = Set(2,5,2)
>>> A
{2,5}
```

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object

▲ロ ▶ ▲周 ▶ ▲ 国 ▶ ▲ 国 ▶ ● ● ● ● ●

Serialization

defining data structures, for example: a set

using the Pickle module an application to network

programming

Incremental Design of a Class

first constructor and string representation

```
class Set:
```

. . .

A set is a list without duplicates.

def __init__(self,*elements):

II II II

Turns a sequence of arguments in elements into a set.

II II II

def ___str__(self):

II II II

Returns a string representation of a set as { elements }.

11 11 11

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object

Serialization

defining data structures, for example: a set

using the Pickle module

an application to network programming

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □

Incremental Design of a Class

first constructor and string representation

```
class Set:
```

. . .

A set is a list without duplicates.

def __init__(self,*elements):

. . .

Turns a sequence of arguments in elements into a set.

. . .

def ___str__(self):

II II II

Returns a string representation of a set as { elements }.

11 11 11

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object

Serialization

defining data structures, for example: a set

using the Pickle module

an application to network programming

Incremental Design of a Class

first constructor and string representation

```
class Set:
```

. . .

A set is a list without duplicates.

def __init__(self,*elements):

. . .

Turns a sequence of arguments in elements into a set.

. . .

def __str__(self):

. . .

Returns a string representation of a set as { elements }. MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object

Serialization

defining data structures, for example: a set

using the Pickle module

an application to network programming

◆□▶ ◆□▶ ◆三▶ ◆三▶ ・三 のへで

Code for the Constructor

in the function ____init____

```
def __init__(self,*elements):
    """
    Turns a sequence of arguments in
    elements into a set.
    """
    self.s = []
    for e in elements:
        if not e in self.s:
            self.s.append(e)
```

```
def __repr__(self):
```

п. п. п

The string representation defines the set representation. """

return str(self)

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited sequences, dictionaries,

Persistent Data

storing information between executions using DBM files

```
Object
```

```
Serialization
```

```
defining data structures, for 
example: a set
```

```
using the Pickle module
an application to network
```

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □

Code for the Constructor

```
in the function ___init___
```

```
def __init__(self,*elements):
    """
    Turns a sequence of arguments in
    elements into a set.
    """
    self.s = []
    for e in elements:
        if not e in self.s:
            self.s.append(e)
```

def __repr__(self):

11 11 11

The string representation defines the set representation.

```
return str(self)
```

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited sequences, dictionaries,

Persistent Data

storing information between executions using DBM files

```
Object
```

```
Serialization
```

```
defining data structures, for 
example: a set
```

```
using the Pickle module
an application to network
```

```
programming
```

◆□ > ◆□ > ◆豆 > ◆豆 > ̄豆 → ���

Code for the Constructor

```
in the function ____init____
```

```
def init (self,*elements):
   .....
   Turns a sequence of arguments in
   elements into a set.
   . . . .
   self.s = []
   for e in elements:
      if not e in self.s:
         self.s.append(e)
def repr (self):
   .....
   The string representation defines
   the set representation.
   .....
   return str(self)
```

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited sequences, dictionaries,

Persistent Data

storing information between executions using DBM files

Object

Serialization

defining data structures, for example: a set

using the Pickle module an application to network

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □

in the function ___str___

```
def str (self):
   . . .
   Returns a string representation
   of a set as { elements }.
   . . . .
   n = len(self.s) - 1
```

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited sequences dictionaries

Persistent Data

storing information between executions using DBM files

Object

Serialization

defining data structures, for example: a set

using the Pickle module an application to network

programming

◆□▶ ◆□▶ ◆三▶ ◆三▶ ● □ ● ●

in the function ___str___

```
def str (self):
   . . .
   Returns a string representation
   of a set as { elements }.
   . . . .
   n = len(self.s) - 1
   if n < 0:
      r = '{}'
```

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

lists

Persistent Data

storing information between executions using DBM files

Object

Serialization

```
defining data structures, for 
example: a set
```

```
using the Pickle module
```

an application to network programming

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 - のへで

in the function ___str___

```
def str (self):
   . . .
   Returns a string representation
   of a set as { elements }.
   . . . .
   n = len(self.s) - 1
   if n < 0:
      r = '\{\}'
   else:
      r = '{'
      for i in range(0,n):
          r = r + str(self.s[i]) + ','
```

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object

Serialization

defining data structures, for example: a set

using the Pickle module

an application to network programming

in the function ___str___

```
def str (self):
   .....
   Returns a string representation
   of a set as { elements }.
   . . . .
   n = len(self.s) - 1
   if n < 0:
      r = '\{\}'
   else:
      r = '{'
      for i in range(0,n):
         r = r + str(self.s[i]) + ','
      r = r + str(self.s[n]) + ' 
   return r
```

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited sequences, dictionaries

Persistent Data

storing information between executions using DBM files

Object

Serialization

defining data structures, for example: a set

using the Pickle module an application to network

programming

algorithms and data structures

Algorithms and Data Structures

programming in Python revisited sequences, dictionaries, lists

ersistent Data storing information between execution using DBM files

Object Serialization

defining data structures, for example: a set using the Pickle module an application to network programming

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, ists

Persistent Data

storing information between executions using DBM files

Object Serializatio

defining data structures, for example: a set

using the Pickle module

an application to network programming

▲□▶▲□▶▲□▶▲□▶ □ ● ●

Pickled Objects

mapping data structures to serial strings

Main limitation of DBM files:

data stored under a key must be a string.

The str and eval works for dictionaries, but not for class instances. Recreating objects from standard string representations is in general not possible.

Serialization is the conversation of objects to strings. Arbitrarily data structures in memory are mapped to a serial string form.

The pickle module is standard in Python. Its C implementation cPickle is more efficient.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serializatior

defining data structures, for example: a set

using the Pickle module

an application to network programming

▲□▶▲□▶▲□▶▲□▶ □ ● ●

Pickled Objects

mapping data structures to serial strings

Main limitation of DBM files:

data stored under a key must be a string.

The str and eval works for dictionaries, but not for class instances. Recreating objects from standard string representations is in general not possible.

Serialization is the conversation of objects to strings. Arbitrarily data structures in memory are mapped to a serial string form.

The pickle module is standard in Python. Its C implementation cPickle is more efficient.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, ists

Persistent Data

storing information between executions using DBM files

Object Serializatior

lefining data structures, for example: a set

using the Pickle module

an application to network programming

・ロト・西ト・山田・山田・山下
Pickled Objects

mapping data structures to serial strings

Main limitation of DBM files:

data stored under a key must be a string.

The str and eval works for dictionaries, but not for class instances. Recreating objects from standard string representations is in general not possible.

Serialization is the conversation of objects to strings. Arbitrarily data structures in memory are mapped to a serial string form.

The pickle module is standard in Python. Its C implementation cPickle is more efficient.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, ists

Persistent Data

storing information between executions using DBM files

Object Serializatior

lefining data structures, for example: a set

using the Pickle module

an application to network programming

・ロト・西ト・山田・山田・山下

Pickled Objects

mapping data structures to serial strings

Main limitation of DBM files:

data stored under a key must be a string.

The str and eval works for dictionaries, but not for class instances. Recreating objects from standard string representations is in general not possible.

Serialization is the conversation of objects to strings. Arbitrarily data structures in memory are mapped to a serial string form.

The pickle module is standard in Python. Its C implementation cPickle is more efficient.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, ists

Persistent Data

storing information between executions using DBM files

Object Serializatior

lefining data structures, for example: a set

using the Pickle module

Using cPickle

\$ python >>> from class_set import * >>> A = Set('a',3,Set(3,'five')) >>> import cPickle >>> setdb = open('oursets','w') >>> cPickle.dump(A,setdb)

oursets is a file. A new Python session:

```
$ python
>>> from class_set import *
>>> import cPickle
>>> f = open('oursets','r')
>>> S = cPickle.load(f)
>>> S
{a,3,{3,five}}
```

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, ists

Persistent Data

storing information between executions using DBM files

Object Serializatio

defining data structures, for example: a set

using the Pickle module

an application to network programming

・ロト・西ト・山田・山田・山下

Using cPickle

\$ python >>> from class_set import * >>> A = Set('a',3,Set(3,'five')) >>> import cPickle >>> setdb = open('oursets','w') >>> cPickle.dump(A,setdb)

oursets is a file. A new Python session:

```
$ python
>>> from class_set import *
>>> import cPickle
>>> f = open('oursets','r')
>>> S = cPickle.load(f)
>>> S
{a,3,{3,five}}
```

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serializatio

defining data structures, for example: a set

using the Pickle module

an application to network programming

Using cPickle to store sets

\$ python >>> from class_set import * >>> A = Set('a',3,Set(3,'five')) >>> import cPickle >>> setdb = open('oursets','w') >>> cPickle.dump(A,setdb)

oursets is a file. A new Python session:

```
$ python
>>> from class_set import *
>>> import cPickle
>>> f = open('oursets','r')
>>> S = cPickle.load(f)
>>> S
{a,3,{3,five}}
```

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, ists

Persistent Data

storing information between executions using DBM files

Object Serializatio

defining data structures, for example: a set

using the Pickle module

an application to network programming

Using cPickle to store sets

```
$ python
>>> from class_set import *
>>> A = Set('a',3,Set(3,'five'))
>>> import cPickle
>>> setdb = open('oursets','w')
>>> cPickle.dump(A,setdb)
```

oursets is a file. A new Python session:

```
$ python
>>> from class_set import *
>>> import cPickle
>>> f = open('oursets','r')
>>> S = cPickle.load(f)
>>> S
{a,3,{3,five}}
```

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serializatio

defining data structures, for example: a set

using the Pickle module

an application to network programming

Using cPickle to store sets

```
$ python
>>> from class_set import *
>>> A = Set('a',3,Set(3,'five'))
>>> import cPickle
>>> setdb = open('oursets','w')
>>> cPickle.dump(A,setdb)
```

oursets is a file. A new Python session:

```
$ python
>>> from class_set import *
>>> import cPickle
>>> f = open('oursets','r')
>>> S = cPickle.load(f)
>>> S
{a,3,{3,five}}
```

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serializatio

defining data structures, for example: a set

using the Pickle module

an application to network programming

the File oursets

(iclass set Set p1 (dp2 S's' (lp3 S'a' aI3 a(iclass_set Set p4 (dp5 S's' (lp6 ТЗ aS'five' p7 asbasb.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serialization

defining data structures, for example: a set

using the Pickle module

an application to network programming

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 のへで

Syntax for general Use

two methods: dump and load

General syntax to pickle and unpickle:

```
1. pickling
  dumping an object to a file:
    import cPickle
    < file > = open( < name > , 'w' )
    cPickle.dump( < object > , < file > )
```

2. unpickling

loading an object from file:

< file > = open(< name > , 'r')

< object > = cPickle.load(< name >)

Not all objects can be pickled, e.g.: files. An exception PickleError is provided.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serialization

defining data structures, for example: a set

using the Pickle module

an application to network programming

・ロト・西ト・山田・山田・山下

Syntax for general Use

two methods: dump and load

General syntax to pickle and unpickle:

- 1. pickling
 dumping an object to a file:
 import cPickle
 < file > = open(< name > , 'w')
 cPickle.dump(< object > , < file >)
- 2. unpickling loading an object from file:
 - < file > = open(< name > , 'r')
 - < object > = cPickle.load(< name >)

Not all objects can be pickled, e.g.: files. An exception PickleError is provided.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serialization

lefining data structures, for example: a set

using the Pickle module

Syntax for general Use

two methods: dump and load

General syntax to pickle and unpickle:

- 1. pickling
 dumping an object to a file:
 import cPickle
 < file > = open(< name > , 'w')
 cPickle.dump(< object > , < file >)
- 2. unpickling loading an object from file:

Not all objects can be pickled, e.g.: files. An exception PickleError is provided.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serialization

lefining data structures, for example: a set

using the Pickle module

algorithms and data structures

Algorithms and Data Structures

programming in Python revisited sequences, dictionaries, lists

ersistent Data storing information between execut using DBM files

Object Serialization

defining data structures, for example: a set using the Pickle module an application to network programming

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, ists

Persistent Data

storing information between executions using DBM files

Object Sorializat

defining data structures, for example: a set

using the Pickle module

an application of pickling to network programming

Suppose we want to send sets from server to client or from client to server.

Using pickling as follows:

- 1. sender dumps object to local file
- 2. sender reads file into one string
- 3. sender sends the string
- 4. receiver writes strings to file
- 5. receiver loads object from file

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object

Serialization

defining data structures, for example: a set

using the Pickle module

an application to network programming

an application of pickling to network programming

Suppose we want to send sets from server to client or from client to server.

Using pickling as follows:

- 1. sender dumps object to local file
- 2. sender reads file into one string
- 3. sender sends the string
- 4. receiver writes strings to file
- 5. receiver loads object from file

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object

Serialization

defining data structures, for example: a set

using the Pickle module

an application to network programming

an application of pickling to network programming

Suppose we want to send sets from server to client or from client to server.

Using pickling as follows:

- 1. sender dumps object to local file
- 2. sender reads file into one string
- 3. sender sends the string
- 4. receiver writes strings to file
- 5. receiver loads object from file

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object

Serialization

defining data structures, for example: a set

using the Pickle module

an application to network programming

an application of pickling to network programming

Suppose we want to send sets from server to client or from client to server.

Using pickling as follows:

- 1. sender dumps object to local file
- 2. sender reads file into one string
- 3. sender sends the string
- 4. receiver writes strings to file
- 5. receiver loads object from file

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object

Serialization

defining data structures, for example: a set

using the Pickle module

an application to network programming

an application of pickling to network programming

Suppose we want to send sets from server to client or from client to server.

Using pickling as follows:

- 1. sender dumps object to local file
- 2. sender reads file into one string
- 3. sender sends the string
- 4. receiver writes strings to file

5. receiver loads object from file

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object

Serialization

defining data structures, for example: a set

using the Pickle module

an application to network programming

an application of pickling to network programming

Suppose we want to send sets from server to client or from client to server.

Using pickling as follows:

- 1. sender dumps object to local file
- 2. sender reads file into one string
- 3. sender sends the string
- 4. receiver writes strings to file
- 5. receiver loads object from file

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object

Serialization

defining data structures, for example: a set

using the Pickle module

an application to network programming

Writing to Strings

using the module StringIO

Instead of working with a temporary file, we can directly write to strings.

```
>>> from class_set import *
>>> import cPickle
>>> f = open('oursets','r')
>>> A = cPickle.load(f)
>>> A
{a,3,{3,five}}
```

>>> import StringIO
>>> output = StringIO.StringIO()
>>> cPickle.dump(A,output)
>>> output.getvalue()
"(iclass_set\nSet\np1\n(dp2\nS's'\n(lp3\nS'a'\nal3\na

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Sorializat

defining data structures, for

using the Pickle module

an application to network programming

Writing to Strings

using the module StringIO

Instead of working with a temporary file, we can directly write to strings.

```
>>> from class_set import *
>>> import cPickle
>>> f = open('oursets','r')
>>> A = cPickle.load(f)
>>> A
{a,3,{3,five}}
```

```
>>> import StringIO
```

>>> output = StringIO.StringIO()

```
>>> cPickle.dump(A,output)
```

>>> output.getvalue()

"(iclass_set\nSet\np1\n(dp2\nS's'\n(lp3\nS'a'<mark>\naI3\</mark>na

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited

sequences, dictionaries, lists

Persistent Data

storing information between executions using DBM files

Object Serializati

defining data structures, for example: a set

using the Pickle module

an application to network programming

Writing to Strings

using the module StringIO

Instead of working with a temporary file, we can directly write to strings.

```
>>> from class_set import *
>>> import cPickle
>>> f = open('oursets','r')
>>> A = cPickle.load(f)
>>> A
{a,3,{3,five}}
```

>>> import StringIO
>>> output = StringIO.StringIO()
>>> cPickle.dump(A,output)
>>> output.getvalue()
"(iclass_set\nSet\npl\n(dp2\nS's'\n(lp3\nS'a'\naI3\na

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited sequences, dictionaries

Persistent Data

storing information between executions using DBM files

Object Serializati

defining data structures, for

using the Pickle module

Summary + Assignments

We restarted *Making Use of Python...* Assignments:

- Use list comprehensions to generate points (x, y) uniformly distributed on the circle: x² + y² = 1. (For some angle t: x = cos(t), y = sin(t).)
- Extend the Class set with a method member e.g.:
 A.member(3) returning True or False accordingly.
- 3. Augment the Class set with the method add, to add a sequence of elements to a set. The number of elements varies, e.g.: S.add(2), S.add(2,3), etc. are all valid uses of add.
- 4. Define the union and intersect operations on sets.
- 5. Give code for client and server to interchange a set.

MCS 275 L-36

14 April 2008

Algorithms and Data Structures

programming in Python revisited sequences, dictionaries,

Persistent Data

storing information between executions using DBM files

Object Serialization

defining data structures, for example: a set