

Guessing
Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous
Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre-
hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary +
Assignments

- 1 **Guessing Secrets**
functions returning functions
oracles and trapdoor functions
- 2 **Anonymous Functions**
lambda forms
map(), reduce(), filter(), eval(), and apply()
- 3 **List Comprehensions**
algorithms and data structures
sequences, dictionaries, lists
- 4 **Summary + Assignments**

MCS 260 Lecture 15
Introduction to Computer Science
Jan Vershelde, 15 February 2010

Guessing Secrets

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

A little game: try to guess a number.

Typical *repeat until*:

```
generate secret
```

```
repeat
```

```
    ask for a guess
```

```
until guess equals secret
```

This game is typical for password verification.

Guessing Secrets

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

A little game: try to guess a number.

Typical *repeat until*:

```
generate secret
```

```
repeat
```

```
    ask for a guess
```

```
until guess equals secret
```

This game is typical for password verification.

Guessing Secrets

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

A little game: try to guess a number.

Typical *repeat until*:

```
generate secret
```

```
repeat
```

```
    ask for a guess
```

```
until guess equals secret
```

This game is typical for password verification.

lambda forms

list comprehensions

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

1 **Guessing Secrets**
functions returning functions
oracles and trapdoor functions

2 **Anonymous Functions**
lambda forms
map(), reduce(), filter(), eval(), and apply()

3 **List Comprehensions**
algorithms and data structures
sequences, dictionaries, lists

4 **Summary + Assignments**

Functions returning Functions

implementing oracles

Guessing Secrets

functions returning
functions

oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

Instead of storing the secret explicitly, we use *an oracle*.

For a given input, the oracle will return True if the input matches the secret and return False otherwise.

Our number guessing game with an oracle:

```
oracle = generate_secret()  
repeat  
    guess = input('give number : '  
until oracle(guess)
```

The function `oracle()` is a function computed by the function `generate_secret()`.

Functions returning Functions

implementing oracles

Guessing Secrets

functions returning
functions

oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

Instead of storing the secret explicitly, we use *an oracle*.

For a given input, the oracle will return True if the input matches the secret and return False otherwise.

Our number guessing game with an oracle:

```
oracle = generate_secret()  
repeat  
    guess = input('give number : ')  
until oracle(guess)
```

The function `oracle()` is a function computed by the function `generate_secret()`.

lambda forms

list comprehensions

Guessing Secrets

functions returning functions

oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures

sequences, dictionaries, lists

Summary + Assignments

1 Guessing Secrets

functions returning functions
oracles and trapdoor functions

2 Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

3 List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

4 Summary + Assignments

Oracles and Trapdoor Functions

password security

Guessing Secrets

functions returning functions

oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

Guarding of passwords on Unix:

- the password is encrypted,
- only the encrypted password is saved on file.

Password verification consists in

- 1 calling the encryption algorithm on user input,
- 2 checking if the result of the encryption equals the encrypted password stored on file.

The encryption algorithm acts as an oracle.

The oracle is typically a *trapdoor* function:

- 1 efficient to compute output for any input,
- 2 very hard to compute the inverse of an output.

Oracles and Trapdoor Functions

password security

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

Guarding of passwords on Unix:

- the password is encrypted,
- only the encrypted password is saved on file.

Password verification consists in

- 1 calling the encryption algorithm on user input,
- 2 checking if the result of the encryption equals the encrypted password stored on file.

The encryption algorithm acts as an oracle.

The oracle is typically a *trapdoor* function:

- 1 efficient to compute output for any input,
- 2 very hard to compute the inverse of an output.

Oracles and Trapdoor Functions

password security

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

Guarding of passwords on Unix:

- the password is encrypted,
- only the encrypted password is saved on file.

Password verification consists in

- 1 calling the encryption algorithm on user input,
- 2 checking if the result of the encryption equals the encrypted password stored on file.

The encryption algorithm acts as an oracle.

The oracle is typically a *trapdoor* function:

- 1 efficient to compute output for any input,
- 2 very hard to compute the inverse of an output.

lambda forms

list comprehensions

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

1 Guessing Secrets
functions returning functions
oracles and trapdoor functions

2 Anonymous Functions
lambda forms
map(), reduce(), filter(), eval(), and apply()

3 List Comprehensions
algorithms and data structures
sequences, dictionaries, lists

4 Summary + Assignments

Guessing
Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous
Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre-
hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary +
Assignments

Lambda Forms

or anonymous functions

Lambda forms are functions without name, syntax:

```
lambda < arguments > : < expression >
```

Often we want to create functions rapidly,
or to give shorter, more meaningful names.
For example, to simulate the rolling of a die:

```
>>> import random  
>>> die = lambda : random.randint(1,6)  
>>> die()  
2
```

The function `die()` has no arguments, but just as with any other function, lambda forms can have default values, keyword and optional arguments.

Lambda Forms

or anonymous functions

Lambda forms are functions without name, syntax:

```
lambda < arguments > : < expression >
```

Often we want to create functions rapidly,
or to give shorter, more meaningful names.
For example, to simulate the rolling of a die:

```
>>> import random  
>>> die = lambda : random.randint(1,6)  
>>> die()  
2
```

The function `die()` has no arguments, but just as with any other function, lambda forms can have default values, keyword and optional arguments.

Functions as Objects

computing with functions

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

Use default arguments to extend `die()`:

```
>>> import random
>>> die = lambda a=1,b=6: random.randint(a,b)
>>> die()
2
>>> die(0,100)
34
>>> die(a=-100)
-29
```

Functions are also objects:

```
>>> type(die)
<type 'function'>
>>> die
<function <lambda> at 0x40247294>
```

Functions as Objects

computing with functions

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

Use default arguments to extend `die()`:

```
>>> import random
>>> die = lambda a=1,b=6: random.randint(a,b)
>>> die()
2
>>> die(0,100)
34
>>> die(a=-100)
-29
```

Functions are also objects:

```
>>> type(die)
<type 'function'>
>>> die
<function <lambda> at 0x40247294>
```

Functions returning Functions

with lambda forms

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

Recall the guessing of a secret. The secret itself is less important than its function: we want an oracle to separate those who know the secret from those who don't.

Our application is to return a function

```
def make_oracle():  
    "returns an oracle as a lambda form"  
    n = input('Give secret number : ')  
    f = lambda x: x == n  
    return f
```

In the main program:

```
oracle = make_oracle()
```

Functions returning Functions

with lambda forms

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

Recall the guessing of a secret. The secret itself is less important than its function: we want an oracle to separate those who know the secret from those who don't.

Our application is to return a function

```
def make_oracle():  
    "returns an oracle as a lambda form"  
    n = input('Give secret number : ')  
    f = lambda x: x == n  
    return f
```

In the main program:

```
oracle = make_oracle()
```

Functions returning Functions

with lambda forms

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

Recall the guessing of a secret. The secret itself is less important than its function: we want an oracle to separate those who know the secret from those who don't.

Our application is to return a function

```
def make_oracle():  
    "returns an oracle as a lambda form"  
    n = input('Give secret number : ')  
    f = lambda x: x == n  
    return f
```

In the main program:

```
oracle = make_oracle()
```

The Guessing Game

with a lambda form

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

```
# L-15 MCS 260 Mon 15 Feb 2010 : an oracle
#
# With lambda forms we can make functions
# that return functions.
#
def make_oracle():
    "returns an oracle as a lambda form"
    n = input('Give secret number : ')
    f = lambda x: x == n
    return f

oracle = make_oracle()
while True:
    g = input('Guess the secret : ')
    if oracle(g): break
    print 'wrong, try again'
print 'found the secret'
```

The Guessing Game

with a lambda form

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

```
# L-15 MCS 260 Mon 15 Feb 2010 : an oracle
#
# With lambda forms we can make functions
# that return functions.
#
def make_oracle():
    "returns an oracle as a lambda form"
    n = input('Give secret number : ')
    f = lambda x: x == n
    return f

oracle = make_oracle()
while True:
    g = input('Guess the secret : ')
    if oracle(g): break
    print 'wrong, try again'
print 'found the secret'
```

lambda forms

list comprehensions

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
`map()`, `reduce()`,
`filter()`, `eval()`, and
`apply()`

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

- 1 Guessing Secrets
functions returning functions
oracles and trapdoor functions
- 2 Anonymous Functions
lambda forms
`map()`, `reduce()`, `filter()`, `eval()`, and `apply()`
- 3 List Comprehensions
algorithms and data structures
sequences, dictionaries, lists
- 4 Summary + Assignments

The map() Function

mapping functions to lists

map() performs the same function on a sequence, syntax:

```
map ( < function > , < sequence > )
```

where the function is often anonymous.

Enumerate all letters of the alphabet:

```
>>> ord('a')
97
>>> chr(97)
'a'
>>> map(chr, range(97, 97+26))
['a', 'b', 'c', ..., 'y', 'z']
```

observe the use of range

The map() Function

mapping functions to lists

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

map() performs the same function on a sequence, syntax:

```
map ( < function > , < sequence > )
```

where the function is often anonymous.

Enumerate all letters of the alphabet:

```
>>> ord('a')  
97  
>>> chr(97)  
'a'  
>>> map(chr, range(97, 97+26))  
['a', 'b', 'c', ..., 'y', 'z']
```

observe the use of range

The map() Function

mapping functions to lists

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

map() performs the same function on a sequence, syntax:

```
map ( < function > , < sequence > )
```

where the function is often anonymous.

Enumerate all letters of the alphabet:

```
>>> ord('a')
97
>>> chr(97)
'a'
>>> map(chr, range(97, 97+26))
['a', 'b', 'c', ..., 'y', 'z']
```

observe the use of range

The map() Function

mapping functions to lists

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

map() performs the same function on a sequence, syntax:

```
map ( < function > , < sequence > )
```

where the function is often anonymous.

Enumerate all letters of the alphabet:

```
>>> ord('a')  
97  
>>> chr(97)  
'a'  
>>> map(chr, range(97, 97+26))  
['a', 'b', 'c', .. , 'y', 'z']
```

observe the use of range

Combining Lists with map()

list processing

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

Adding corresponding elements of lists:

```
>>> a = range(0,3)
>>> b = range(3,6)
>>> map(lambda x,y: x+y, a,b)
[3, 5, 7]
```

Creating tuples, mapping None to lists (or use zip()):

```
>>> m = map(None, a, b)
>>> m
[(0, 3), (1, 4), (2, 5)]
>>> zip(a,b)
```

Let us add the elements in the tuples of m:

```
>>> map(lambda x: x[0]+x[1], m)
[3, 5, 7]
```

Combining Lists with map()

list processing

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

Adding corresponding elements of lists:

```
>>> a = range(0,3)
>>> b = range(3,6)
>>> map(lambda x,y: x+y, a,b)
[3, 5, 7]
```

Creating tuples, mapping None to lists (or use zip()):

```
>>> m = map(None, a, b)
>>> m
[(0, 3), (1, 4), (2, 5)]
>>> zip(a, b)
```

Let us add the elements in the tuples of m:

```
>>> map(lambda x: x[0]+x[1], m)
[3, 5, 7]
```

Combining Lists with map()

list processing

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

Adding corresponding elements of lists:

```
>>> a = range(0,3)
>>> b = range(3,6)
>>> map(lambda x,y: x+y, a,b)
[3, 5, 7]
```

Creating tuples, mapping None to lists (or use zip()):

```
>>> m = map(None, a, b)
>>> m
[(0, 3), (1, 4), (2, 5)]
>>> zip(a, b)
```

Let us add the elements in the tuples of m:

```
>>> map(lambda x: x[0]+x[1], m)
[3, 5, 7]
```

The reduce() Function

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

The function `reduce()` returns one single value from a list, for example to compute the sum:

```
>>> r = range(0,10)
>>> reduce(lambda x,y: x+y , r)
45
```

The function given as argument to `reduce()` must

- take two elements on input,
- return one single element.

`reduce()` repeatedly replaces the first two elements of the list by the result of the function, applied to those first two elements, until only one element in the list is left

The reduce() Function

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

The function `reduce()` returns one single value from a list, for example to compute the sum:

```
>>> r = range(0,10)
>>> reduce(lambda x,y: x+y , r)
45
```

The function given as argument to `reduce()` must

- take two elements on input,
- return one single element.

`reduce()` repeatedly replaces the first two elements of the list by the result of the function, applied to those first two elements, until only one element in the list is left

The filter() Function

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

Filters a sequence, subject to a criterion, syntax:

```
filter ( < criterion > , < sequence > )
```

where `criterion` is a function returning a boolean, and the `sequence` is typically a list.

The list on return contains all elements of the input list for which the criterion is True.

Sieve methods to compute primes:

```
>>> s = range(2,100)
>>> s = filter(lambda x: x%2 != 0,s)
>>> s = filter(lambda x: x%3 != 0,s)
>>> s = filter(lambda x: x%5 != 0,s)
>>> s = filter(lambda x: x%7 != 0,s)
```

first element of the list `s` is always a prime

The filter() Function

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

Filters a sequence, subject to a criterion, syntax:

```
filter ( < criterion > , < sequence > )
```

where `criterion` is a function returning a boolean, and the `sequence` is typically a list.

The list on return contains all elements of the input list for which the criterion is True.

Sieve methods to compute primes:

```
>>> s = range(2,100)
>>> s = filter(lambda x: x%2 != 0,s)
>>> s = filter(lambda x: x%3 != 0,s)
>>> s = filter(lambda x: x%5 != 0,s)
>>> s = filter(lambda x: x%7 != 0,s)
```

first element of the list `s` is always a prime

The filter() Function

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

Filters a sequence, subject to a criterion, syntax:

```
filter ( < criterion > , < sequence > )
```

where `criterion` is a function returning a boolean, and the `sequence` is typically a list.

The list on return contains all elements of the input list for which the criterion is True.

Sieve methods to compute primes:

```
>>> s = range(2,100)
>>> s = filter(lambda x: x%2 != 0,s)
>>> s = filter(lambda x: x%3 != 0,s)
>>> s = filter(lambda x: x%5 != 0,s)
>>> s = filter(lambda x: x%7 != 0,s)
```

first element of the list `s` is always a prime

The filter() Function

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

Filters a sequence, subject to a criterion, syntax:

```
filter ( < criterion > , < sequence > )
```

where `criterion` is a function returning a boolean,
and the `sequence` is typically a list.

The list on return contains all elements of the input list for
which the criterion is True.

Sieve methods to compute primes:

```
>>> s = range(2,100)
>>> s = filter(lambda x: x%2 != 0,s)
>>> s = filter(lambda x: x%3 != 0,s)
>>> s = filter(lambda x: x%5 != 0,s)
>>> s = filter(lambda x: x%7 != 0,s)
```

first element of the list `s` is always a prime

The filter() Function

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

Filters a sequence, subject to a criterion, syntax:

```
filter ( < criterion > , < sequence > )
```

where `criterion` is a function returning a boolean,
and the `sequence` is typically a list.

The list on return contains all elements of the input list for
which the criterion is True.

Sieve methods to compute primes:

```
>>> s = range(2,100)
>>> s = filter(lambda x: x%2 != 0,s)
>>> s = filter(lambda x: x%3 != 0,s)
>>> s = filter(lambda x: x%5 != 0,s)
>>> s = filter(lambda x: x%7 != 0,s)
```

first element of the list `s` is always a prime

Evaluation of Expressions

with eval()

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

The `eval()` executes an expression string.

```
def make_fun():
    s = raw_input('give an expression : ')
    x = raw_input('give the variable name : ')
    f = lambda x : eval(s)
    return f

g = make_fun()
v = input('give a value : ')
y = g(v)
print 'the expression evaluated at %f' % v
print 'gives %f ' % y
```

Evaluation of Expressions

with `eval()`

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
`map()`, `reduce()`,
`filter()`, `eval()`, and
`apply()`

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

The `eval()` executes an expression string.

```
def make_fun():
    s = raw_input('give an expression : ')
    x = raw_input('give the variable name : ')
    f = lambda x : eval(s)
    return f

g = make_fun()
v = input('give a value : ')
y = g(v)
print 'the expression evaluated at %f' % v
print 'gives %f ' % y
```

Illustration of evalshow.py

delayed evaluation

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

Running `evalshow.py` at the command prompt \$:

```
$ python evalshow.py  
give an expression : 2*x**6 - x + 9.9  
give the variable name : x  
give a value : -0.4512  
the expression evaluated at -0.451200  
gives 10.368075
```

With `eval()` we delay the evaluation of the expression entered by the user, till a value for the variable is provided.

Illustration of evalshow.py

delayed evaluation

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

Running `evalshow.py` at the command prompt \$:

```
$ python evalshow.py  
give an expression : 2*x**6 - x + 9.9  
give the variable name : x  
give a value : -0.4512  
the expression evaluated at -0.451200  
gives 10.368075
```

With `eval()` we delay the evaluation of the expression entered by the user, till a value for the variable is provided.

The apply() Function

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

Syntax:

```
apply ( < function name > , < arguments > )
```

Although $y = \text{apply}(f, x)$ equals $y = f(x)$,
the `apply` is useful when not only the arguments,
but also the function is only known at run time.

Typical example: a simple calculator.

```
$ python calculator.py  
give first operand : 3  
give second operand : 4  
operator ? (+, -, *) *  
3 * 4 = 12
```

The apply() Function

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

Syntax:

```
apply ( < function name > , < arguments > )
```

Although $y = \text{apply}(f, x)$ equals $y = f(x)$,
the `apply` is useful when not only the arguments,
but also the function is only known at run time.

Typical example: a simple calculator.

```
$ python calculator.py  
give first operand : 3  
give second operand : 4  
operator ? (+, -, *) *  
3 * 4 = 12
```

The Program calculator.py

apply() avoids if else elif

Guessing Secrets

functions returning
functions

oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures

sequences,
dictionaries, lists

Summary + Assignments

```
# L-15 MCS 260 Mon 15 Feb 2010 : apply
#
# apply() in a simple # calculator
#
from operator import add, sub, mul

ops = { '+':add, '-':sub, '*': mul }

a = input('give first operand : ')
b = input('give second operand : ')
op = raw_input('operator ? (+,-,*) ')

c = apply(ops[op],(a,b))

print '%d %s %d = %d' % (a,op,b,c)
```

The Program calculator.py

apply() avoids if else elif

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

```
# L-15 MCS 260 Mon 15 Feb 2010 : apply
#
# apply() in a simple # calculator
#
from operator import add, sub, mul

ops = { '+':add, '-':sub, '*':mul }

a = input('give first operand : ')
b = input('give second operand : ')
op = raw_input('operator ? (+,-,*) ')

c = apply(ops[op],(a,b))

print '%d %s %d = %d' % (a,op,b,c)
```

The Program calculator.py

apply() avoids if else elif

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

```
# L-15 MCS 260 Mon 15 Feb 2010 : apply
#
# apply() in a simple # calculator
#
from operator import add, sub, mul

ops = { '+':add, '-':sub, '*':mul }

a = input('give first operand : ')
b = input('give second operand : ')
op = raw_input('operator ? (+,-,* ) ')

c = apply(ops[op],(a,b))

print '%d %s %d = %d' % (a,op,b,c)
```

The Program calculator.py

apply() avoids if else elif

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

```
# L-15 MCS 260 Mon 15 Feb 2010 : apply
#
# apply() in a simple # calculator
#
from operator import add, sub, mul

ops = { '+':add, '-':sub, '*':mul }

a = input('give first operand : ')
b = input('give second operand : ')
op = raw_input('operator ? (+,-,* ) ')

c = apply(ops[op],(a,b))

print '%d %s %d = %d' % (a,op,b,c)
```

The Program calculator.py

apply() avoids if else elif

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

```
# L-15 MCS 260 Mon 15 Feb 2010 : apply
#
# apply() in a simple # calculator
#
from operator import add, sub, mul

ops = { '+':add, '-':sub, '*':mul }

a = input('give first operand : ')
b = input('give second operand : ')
op = raw_input('operator ? (+,-,* ) ')

c = apply(ops[op],(a,b))

print '%d %s %d = %d' % (a,op,b,c)
```

Monte Carlo without Loops

a functional implementation

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

Recall the Monte Carlo method to estimate π :

- 1 generate n points P in $[0, 1] \times [0, 1]$
- 2 $m := \{ (x, y) \in P : x^2 + y^2 \leq 1 \}$
- 3 the estimate is then $4 \times m/n$

Main ingredients in *a functional implementation*:

- 1 `u = lambda i: random.uniform(0,1)`
- 2 `map u on range(0,n) twice for x and y`
- 3 `map(None, x, y)` returns list of tuples
- 4 `t = lambda (x,y): x**2 + y**2 <= 1`
- 5 `f = filter(t,z)`

without loops!

Monte Carlo without Loops

a functional implementation

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

Recall the Monte Carlo method to estimate π :

- 1 generate n points P in $[0, 1] \times [0, 1]$
- 2 $m := \{ (x, y) \in P : x^2 + y^2 \leq 1 \}$
- 3 the estimate is then $4 \times m/n$

Main ingredients in *a functional implementation*:

- 1 `u = lambda i: random.uniform(0, 1)`
- 2 `map u on range(0, n) twice for x and y`
- 3 `map(None, x, y)` returns list of tuples
- 4 `t = lambda (x, y): x**2 + y**2 <= 1`
- 5 `f = filter(t, z)`

without loops!

Monte Carlo without Loops

a functional implementation

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

Recall the Monte Carlo method to estimate π :

- 1 generate n points P in $[0, 1] \times [0, 1]$
- 2 $m := \{ (x, y) \in P : x^2 + y^2 \leq 1 \}$
- 3 the estimate is then $4 \times m/n$

Main ingredients in *a functional implementation*:

- 1 `u = lambda i: random.uniform(0, 1)`
- 2 `map u on range(0, n) twice for x and y`
- 3 `map(None, x, y)` returns list of tuples
- 4 `t = lambda (x, y): x**2 + y**2 <= 1`
- 5 `f = filter(t, z)`

without loops!

Monte Carlo without Loops

a functional implementation

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

Recall the Monte Carlo method to estimate π :

- 1 generate n points P in $[0, 1] \times [0, 1]$
- 2 $m := \{ (x, y) \in P : x^2 + y^2 \leq 1 \}$
- 3 the estimate is then $4 \times m/n$

Main ingredients in *a functional implementation*:

- 1 `u = lambda i: random.uniform(0, 1)`
- 2 `map u on range(0, n) twice for x and y`
- 3 `map(None, x, y) returns list of tuples`
- 4 `t = lambda (x, y): x**2 + y**2 <= 1`
- 5 `f = filter(t, z)`

without loops!

Monte Carlo without Loops

a functional implementation

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

Recall the Monte Carlo method to estimate π :

- 1 generate n points P in $[0, 1] \times [0, 1]$
- 2 $m := \{ (x, y) \in P : x^2 + y^2 \leq 1 \}$
- 3 the estimate is then $4 \times m/n$

Main ingredients in *a functional implementation*:

- 1 `u = lambda i: random.uniform(0,1)`
- 2 `map u on range(0,n) twice for x and y`
- 3 `map(None, x, y)` returns list of tuples
- 4 `t = lambda (x,y): x**2 + y**2 <= 1`
- 5 `f = filter(t,z)`

without loops!

Monte Carlo without Loops

a functional implementation

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

Recall the Monte Carlo method to estimate π :

- 1 generate n points P in $[0, 1] \times [0, 1]$
- 2 $m := \{ (x, y) \in P : x^2 + y^2 \leq 1 \}$
- 3 the estimate is then $4 \times m/n$

Main ingredients in *a functional implementation*:

- 1 `u = lambda i: random.uniform(0,1)`
- 2 `map u on range(0,n) twice for x and y`
- 3 `map(None, x, y)` returns list of tuples
- 4 `t = lambda (x,y): x**2 + y**2 <= 1`
- 5 `f = filter(t,z)`

without loops!

Guessing
Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous
Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre-
hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary +
Assignments

Estimating π

Monte Carlo without loops

```
# L-15 MCS 260 Mon 15 Feb 2010 : mc4pi2
#
# Illustration of lambda forms in
# a Monte Carlo method for Pi.
#
import random
n = input('Give number of samples : ')
r = range(0,n)
u = lambda i: random.uniform(0,1)
x = map(u,r)
y = map(u,r)
z = map(None,x,y)
t = lambda (x,y): x**2 + y**2 <= 1
f = filter(t,z)
p = 4.0*len(f)/n
print 'estimate for Pi : %f' % p
```

Guessing
Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous
Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre-
hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary +
AssignmentsEstimating π
Monte Carlo without loops

```
# L-15 MCS 260 Mon 15 Feb 2010 : mc4pi2
#
# Illustration of lambda forms in
# a Monte Carlo method for Pi.
#
import random
n = input('Give number of samples : ')
r = range(0,n)
u = lambda i: random.uniform(0,1)
x = map(u,r)
y = map(u,r)
z = map(None,x,y)
t = lambda (x,y): x**2 + y**2 <= 1
f = filter(t,z)
p = 4.0*len(f)/n
print 'estimate for Pi : %f' % p
```

Guessing
Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous
Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre-
hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary +
AssignmentsEstimating π
Monte Carlo without loops

```
# L-15 MCS 260 Mon 15 Feb 2010 : mc4pi2
#
# Illustration of lambda forms in
# a Monte Carlo method for Pi.
#
import random
n = input('Give number of samples : ')
r = range(0,n)
u = lambda i: random.uniform(0,1)
x = map(u,r)
y = map(u,r)
z = map(None,x,y)
t = lambda (x,y): x**2 + y**2 <= 1
f = filter(t,z)
p = 4.0*len(f)/n
print 'estimate for Pi : %f' % p
```

lambda forms list comprehensions

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

1 **Guessing Secrets**
functions returning functions
oracles and trapdoor functions

2 **Anonymous Functions**
lambda forms
map(), reduce(), filter(), eval(), and apply()

3 **List Comprehensions**
algorithms and data structures
sequences, dictionaries, lists

4 **Summary + Assignments**

Algorithms and Data Structures

a summary of Python in the small

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

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

Algorithms and Data Structures

a summary of Python in the small

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

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

Algorithms and Data Structures

a summary of Python in the small

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

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

Algorithms and Data Structures

a summary of Python in the small

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

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

Algorithms and Data Structures

a summary of Python in the small

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

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

Algorithms and Data Structures

a summary of Python in the small

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

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

Algorithms and Data Structures

a summary of Python in the small

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

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

Algorithms and Data Structures

a summary of Python in the small

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

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

Algorithms and Data Structures

a summary of Python in the small

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

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

lambda forms

list comprehensions

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

- 1 Guessing Secrets
functions returning functions
oracles and trapdoor functions
- 2 Anonymous Functions
lambda forms
map(), reduce(), filter(), eval(), and apply()
- 3 List Comprehensions
algorithms and data structures
sequences, dictionaries, lists
- 4 Summary + Assignments

Guessing
Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous
Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre-
hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary +
Assignments

Programs are Data Transformations

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.

Guessing
Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous
Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre-
hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary +
Assignments

Programs are Data Transformations

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.

Guessing
Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous
Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre-
hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary +
Assignments

Programs are Data Transformations

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.

Guessing
Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous
Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre-
hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary +
Assignments

Programs are Data Transformations

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.

Storing Conditions

dictionaries and if else statements

We can represent an if else statement

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

via a dictionary:

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

Guessing
Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous
Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre-
hensions

algorithms and data
structures

sequences,
dictionaries, lists

Summary +
Assignments

Storing Conditions

dictionaries and if else statements

We can represent an if else statement

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

via a dictionary:

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

Guessing
Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous
Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre-
hensions

algorithms and data
structures

sequences,
dictionaries, lists

Summary +
Assignments

Guessing
Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous
Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre-
hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary +
Assignments

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.

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.

Guessing
Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous
Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre-
hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary +
Assignments

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.

Guessing
Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous
Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre-
hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary +
Assignments

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.

Guessing
Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous
Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre-
hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary +
Assignments

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.

Guessing
Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous
Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre-
hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary +
Assignments

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.

List Comprehensions

defining lists in a short way

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

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']
```

List Comprehensions

defining lists in a short way

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

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']
```

List Comprehensions

defining lists in a short way

Guessing Secrets

functions returning functions
oracles and trapdoor functions

Anonymous Functions

lambda forms
map(), reduce(), filter(), eval(), and apply()

List Comprehensions

algorithms and data structures
sequences, dictionaries, lists

Summary + Assignments

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']
```

List Comprehensions

defining lists in a short way

Guessing Secrets

functions returning
functions
oracles and trapdoor
functions

Anonymous Functions

lambda forms
map(), reduce(),
filter(), eval(), and
apply()

List Compre- hensions

algorithms and data
structures
sequences,
dictionaries, lists

Summary + Assignments

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']
```

Summary + Assignments

Background reading for this lecture:

- pages 181-182 in *Python Programming in Context*,
- pages 256-258 in *Computer Science, an overview*.

Assignments:

- 1 Generate the list $[(1,1),(1,2),(1,3),(1,4), \dots, (1,n)]$, for any given n . Use this list then to create all fractions $1.0/k$, for k from 1 to n . Finally, use `round()` to round all fractions to two decimal places.

- 2 Approximate the exponential function as $\sum_{k=0}^n \frac{x^k}{k!}$.

Write a Python program using `map()` and `reduce()`, to evaluate this approximation for given x and n .

- 3 Use list comprehensions to generate points (x, y) uniformly distributed on the circle: $x^2 + y^2 = 1$. (For some angle t : $x = \cos(t)$, $y = \sin(t)$.)