Outline

1. Simulation
   Monte Carlo methods
   random numbers

2. Repeat Until
   binary expansion
   break statement

3. Arrays and nested for Loops
   arrays represent matrices
   searching a two dimensional array

4. Summary + Assignments

MCS 260 Lecture 12
Introduction to Computer Science
Jan Verschelde, 22 September 2008
Simulation
Monte Carlo methods

- In a mathematical model with uncertainties, events occur with assigned probabilities.

- Simulation consists in the repeated drawing of samples according to a probability distribution. We count the number of successful samples.

- The Law of Large Numbers states that the arithmetic average of the observed successes converges to the expected value or mean of the experiment, as the number of experiments increases.

- Monte Carlo methods are listed among the Top Ten Algorithms of the 20th century.
Simulation

Monte Carlo methods

- In a mathematical model with uncertainties, events occur with assigned probabilities.

- Simulation consists in the repeated drawing of samples according to a probability distribution. We count the number of successful samples.

- The Law of Large Numbers states that the arithmetic average of the observed successes converges to the expected value or mean of the experiment, as the number of experiments increases.

- Monte Carlo methods are listed among the Top Ten Algorithms of the 20th century.
Simulation
Monte Carlo methods

- In a mathematical model with uncertainties, events occur with assigned probabilities.

- Simulation consists in the repeated drawing of samples according to a probability distribution. We count the number of successful samples.

- The Law of Large Numbers states that the arithmetic average of the observed successes converges to the expected value or mean of the experiment, as the number of experiments increases.

- Monte Carlo methods are listed among the Top Ten Algorithms of the 20th century.
Simulation
Monte Carlo methods

- In a mathematical model with uncertainties, events occur with assigned probabilities.

- Simulation consists in the repeated drawing of samples according to a probability distribution. We count the number of successful samples.

- The Law of Large Numbers states that the arithmetic average of the observed successes converges to the expected value or mean of the experiment, as the number of experiments increases.

- Monte Carlo methods are listed among the Top Ten Algorithms of the 20th century.
Running Simulations
repeat until: break

1. **Simulation**
   - Monte Carlo methods
   - random numbers

2. **Repeat Until**
   - binary expansion
   - break statement

3. **Arrays and nested for Loops**
   - arrays represent matrices
   - searching a two dimensional array

4. **Summary + Assignments**
Flowchart for Simulations

\[ s = 0; i = 0 \]

\[ i < n? \]

\[ \text{True} \]

\[ \text{sample; } i = i + 1 \]

\[ \text{success?} \]

\[ \text{False} \]

\[ \text{True} \]

\[ s = s + 1 \]

\[ \text{print } s \]
Simulation
Monte Carlo methods
random numbers

Repeat Until
binary expansion
break statement

Arrays and nested for loops
arrays represent matrices
searching a two dimensional array

Summary + Assignments

Flowchart for Simulations

\[
\begin{align*}
\text{s &= 0; i &= 0} \\
\text{i < n?} \\
\text{True} & \rightarrow \text{sample; i &= i + 1} \\
\text{success?} & \\
\text{True} & \rightarrow \text{s &= s + 1} \\
\text{False} & \rightarrow \text{print s}
\end{align*}
\]
Flowchart for Simulations

\[ s = 0; i = 0 \]

\[ i < n? \]

True:
\[ \text{sample}; i = i + 1 \]

\[ \text{success?} \]

False:
\[ \text{print } s \]

True:
\[ s = s + 1 \]
Flowchart for Simulations

- $s = 0; i = 0$
- $i < n$?
  - True: sample; $i = i + 1$
  - False: print $s$
- success?
  - False: $s = s + 1$
  - True: $i < n$?
Running Simulations
repeat until: break

1. Simulation
   Monte Carlo methods
   random numbers

2. Repeat Until
   binary expansion
   break statement

3. Arrays and nested for Loops
   arrays represent matrices
   searching a two dimensional array

4. Summary + Assignments
Random number generators are in the module `random`. Three things we need to know:

1. **import random** loads the module into a session. Afterwards, `help(random)` shows a description of the definitions and functions offered by the module.

2. **random.seed()**
   Giving a fixed number as argument results in the same sequence of random numbers.

3. **r = random.uniform(a, b)**
   `r` is a randomly generated number, drawn from a uniform distribution over the interval `[a, b)`. 

Random Numbers
as available in Python
Random Numbers
as available in Python

Random number generators are in the module `random`.
Three things we need to know:

1. `import random` loads the module into a session. Afterwards, `help(random)` shows a description of the definitions and functions offered by the module.

2. `random.seed()`
   Giving a fixed number as argument results in the same sequence of random numbers.

3. `r = random.uniform(a, b)`
   `r` is a randomly generated number, drawn from a uniform distribution over the interval `[a, b)`. 
Random number generators are in the module `random`. Three things we need to know:

1. `import random` loads the module into a session. Afterwards, `help(random)` shows a description of the definitions and functions offered by the module.

2. `random.seed()`
   Giving a fixed number as argument results in the same sequence of random numbers.

3. `r = random.uniform(a, b)`
   `r` is a randomly generated number, drawn from a uniform distribution over the interval `[a, b)`. 
Using Random Numbers

a sample program

A sample program `randuse.py`:

```python
# L-12 MCS 260 Mon 22 Sep 2008 random numbers
#
import random     # use module random
random.seed(21342342)  # get same sequence
print 'uniformly distributed random numbers'
a = input('give lower bound : ')   
b = input('give upper bound : ')   
r = random.uniform(a,b)       # generate a number
print 'a random number in [%.2f,%.2f] : %.15f' % (a,b,r)
```
Using Random Numbers

a sample program

A sample program `randuse.py`:

```python
# L-12 MCS 260 Mon 22 Sep 2008 random numbers
#
import random  # use module random
random.seed(21342342)  # get same sequence
print 'uniformly distributed random numbers'
a = input('give lower bound : ')
b = input('give upper bound : ')
r = random.uniform(a,b)  # generate a number
print 'a random number in [%.2f,%.2f] : %.15f' % (a,b,r)
```
Using Random Numbers
a sample program

A sample program randuse.py:

```python
# L-12 MCS 260 Mon 22 Sep 2008 random numbers
#
import random       # use module random
random.seed(21342342)  # get same sequence
print 'uniformly distributed random numbers'
a = input('give lower bound : ')  
b = input('give upper bound : ') 
r = random.uniform(a,b)    # generate a number
print 'a random number in [%1.2f,%1.2f] : %1.15f'
    % (a,b,r)
```
Estimating Areas and Volumes

high dimensional integrals

- Expected values are expressed as integrals. When many parameters are involved, the integration is high dimensional and only estimation is possible.

- The area of the unit disk is $\pi$.

Generate random uniformly distributed points with coordinates $(x, y) \in [-1, +1] \times [-1, +1]$. We count a success when $x^2 + y^2 \leq 1$. 
Estimating Areas and Volumes

high dimensional integrals

- Expected values are expressed as integrals. When many parameters are involved, the integration is high dimensional and only estimation is possible.

- The area of the unit disk is $\pi$.

Generate random uniformly distributed points with coordinates $(x, y) \in [-1, +1] \times [-1, +1]$. We count a success when $x^2 + y^2 \leq 1$. 

\[ \text{Area of the unit disk} = \pi \]
Estimating Areas and Volumes

high dimensional integrals

- Expected values are expressed as integrals. When many parameters are involved, the integration is high dimensional and only estimation is possible.

- The area of the unit disk is $\pi$.

![Diagram of a square and a circle]

Generate random uniformly distributed points with coordinates $(x, y) \in [-1, +1] \times [-1, +1]$. We count a success when $x^2 + y^2 \leq 1$. 
Flowchart for Estimating $\pi$

$s = 0; \ i = 0$

$\text{True}$

$\text{False}$

$i < n$?

pick $(x, y) \in [-1, +1]^2; \ i = i + 1$

$x^2 + y^2 \leq 1$?

$s = s + 1$

print $4s/n$
Simulation
Monte Carlo methods
random numbers

Repeat Until
binary expansion
break statement

Arrays and nested for loops
arrays represent matrices
searching a two dimensional array

Summary + Assignments

Flowchart for Estimating $\pi$

$s = 0; i = 0$

$i < n$?

pick $(x, y) \in [-1, +1]^2; i = i + 1$

$x^2 + y^2 \leq 1$?

$s = s + 1$

print $4s/n$
Flowchart for Estimating $\pi$

1. $s = 0; i = 0$
2. $i < n$?
   - True: $x^2 + y^2 \leq 1$?
     - True: $s = s + 1$
     - False: $i = i + 1$
   - False: print $4s/n$

- Simulation
  - Monte Carlo methods
  - random numbers

- Repeat Until
  - binary expansion
  - break statement

- Arrays and nested for Loops
  - arrays represent matrices
  - searching a two dimensional array

- Summary + Assignments
Flowchart for Estimating $\pi$

$s = 0; i = 0$

$i < n?$

- True
  - pick $(x, y) \in [-1, +1]^2; i = i + 1$
    - $x^2 + y^2 \leq 1$?
      - True
        - $s = s + 1$
      - False
      - $i < n$?
        - True
          - pick $(x, y) \in [-1, +1]^2; i = i + 1$
        - False
          - print $4s/n$

- False
  - print $4s/n$
Flowchart for Estimating \( \pi \)

1. **Initialize**:
   - \( s = 0; i = 0 \)

2. **Repeat Until**
   - Repeat until \( i < n \)

3. **Check Condition**
   - If \( i < n \)
     - **Pick** \((x, y) \in [-1, +1]^2; i = i + 1\)
     - **Check** \(x^2 + y^2 \leq 1\)
       - If True
         - \( s = s + 1 \)
       - If False
         - **Stop** and **Print** \( \frac{4s}{n} \)

4. **End**
Estimating $\pi$

The program `mc4pi.py`:

```python
# L-12 MCS 260 Mon 22 Sep 2008 Monte Carlo
#
import random
print 'Monte Carlo simulation for Pi'
n = input('Give number of runs : ')
s = 0
for i in range(0,n):
    x = random.uniform(-1,1)
    y = random.uniform(-1,1)
    if x**2 + y**2 <= 1: s += 1
print 'After %d runs : %f' % (n,4.0*s/n)
```

Why multiply by 4? 4 is the area of $[-1,+1]^2$. 
Estimating $\pi$

The program `mc4pi.py`:

```python
# L-12 MCS 260 Mon 22 Sep 2008 Monte Carlo
#
import random
print 'Monte Carlo simulation for Pi'
n = input('Give number of runs : ')
s = 0
for i in range(0,n):
x = random.uniform(-1,1)
y = random.uniform(-1,1)
    if x**2 + y**2 <= 1: s += 1
print 'After %d runs : %f' % (n,4.0*s/n)
```

Why multiply by 4? 4 is the area of $[-1,+1]^2$. 

Estimating $\pi$

The program `mc4pi.py`:

```python
# L-12 MCS 260 Mon 22 Sep 2008 Monte Carlo
#
import random
print 'Monte Carlo simulation for Pi'

n = input('Give number of runs : ')
s = 0
for i in range(0,n):
    x = random.uniform(-1,1)
    y = random.uniform(-1,1)
    if x**2 + y**2 <= 1: s += 1
print 'After %d runs : %f' % (n,4.0*s/n)
```

Why multiply by 4? 4 is the area of $[-1, +1]^2$. 
Estimating $\pi$

The program `mc4pi.py`:

```python
# L-12 MCS 260 Mon 22 Sep 2008 Monte Carlo
#
import random
print 'Monte Carlo simulation for Pi'
n = input('Give number of runs : ')
s = 0
for i in range(0,n):
    x = random.uniform(-1,1)
    y = random.uniform(-1,1)
    if x**2 + y**2 <= 1: s += 1
print 'After %d runs : %f' % (n,4.0*s/n)
```

**Why multiply by 4?** 4 is the area of $[-1,+1]^2$. 
Running Simulations
repeat until: break

1. Simulation
   Monte Carlo methods
   random numbers

2. Repeat Until
   binary expansion
   break statement

3. Arrays and nested for Loops
   arrays represent matrices
   searching a two dimensional array

4. Summary + Assignments
Binary Expansions
repeat until loops

The bits of a number are the remainders of division by 2. divmod() is an intrinsic operation:

```python
>>> divmod(9, 2)
(4, 1)
```

Use as

```
(n, r) = divmod(n, 2)
```

to obtain remainder \(n \mod 2\) in \(r\) and to replace \(n\) by \(n/2\).

Pseudo code to compute the binary expansion:

```python
n = input()
repeat
    (n, r) = divmod(n, 2)
    print r
until (n == 0).
```
Binary Expansions
repeat until loops

The bits of a number are the remainders of division by 2. 
\texttt{divmod()} is an intrinsic operation:

\begin{verbatim}
>>> divmod(9,2)
(4, 1)
\end{verbatim}

Use as \( (n, r) = \text{divmod}(n, 2) \)
to obtain remainder \( n \% 2 \) in \( r \) and to replace \( n \) by \( n/2 \).

Pseudo code to compute the binary expansion:

\begin{verbatim}
  n = input()
  repeat
    (n,r) = divmod(n,2)
    print r
  until (n == 0).
\end{verbatim}
Binary Expansions
repeat until loops

The bits of a number are the remainders of division by 2. divmod() is an intrinsic operation:

```python
>>> divmod(9, 2)
(4, 1)
```

Use as \((n, r) = \text{divmod}(n, 2)\) to obtain remainder \(n \% 2\) in \(r\) and to replace \(n\) by \(n/2\).

Pseudo code to compute the binary expansion:

```python
n = input()
repeat
    (n, r) = divmod(n, 2)
    print r
until (n == 0).
```
Flowchart of Binary Expansion
picture of repeat until

```
n = input()
(n, r) = divmod(n, 2)
print r
```

```
n == 0?
  True
  False
```

Arrays and nested for loops
arrays represent matrices
searching a two dimensional array
Flowchart of Binary Expansion

picture of repeat until

\[ n = \text{input()} \]

\[(n, r) = \text{divmod}(n, 2)\]

\[\text{print } r\]

\[n == 0? \quad \text{True}\]

\[n == 0? \quad \text{False}\]
Flowchart of Binary Expansion

```
Flowchart of Binary Expansion
picture of repeat until

n = input()

(n,r) = divmod(n,2)
print r

n == 0?

True

False
```
Flowchart of Binary Expansion

```
import math
n = input()
(n, r) = divmod(n, 2)
print(r)
while n != 0:
    n = n // 2
    print(n)
```

Summary + Assignments
A first Python solution

```python
# L-12 MCS 260 Mon 22 Sep 2008 binary expansion
#
# This first version prints the bits in the order as they are computed.
#
print 'computing the binary expansion'
n = input('Give a number : ')
(n,r) = divmod(n,2)
print r
while n > 0:
    (n,r) = divmod(n,2)
    print r
```

avoid duplication of code
# L-12 MCS 260 Mon 22 Sep 2008 binary expansion
#
# This first version prints the bits in the order as they are computed.
#
print 'computing the binary expansion'

n = input('Give a number : ')
(n,r) = divmod(n,2)
print r
while n > 0:
    (n,r) = divmod(n,2)
    print r

avoid duplication of code
Running Simulations
repeat until: break

1. Simulation
   Monte Carlo methods
   random numbers

2. Repeat Until
   binary expansion
   break statement

3. Arrays and nested for Loops
   arrays represent matrices
   searching a two dimensional array

4. Summary + Assignments
The **break** Statement

repeat until as while true break

To exit a loop inside the body of a loop, the statement `break` occurs usually within an `if` statement.

```
repeat
    < body of loop >
until < condition >
```

is realized in Python as

```
while True:
    < body of loop >
    if < condition > : break
```

The `while True` starts an infinite loop, terminated when `< condition >` becomes True.
The break Statement
repeat until as while true break

To exit a loop inside the body of a loop, the statement **break** occurs usually within an **if** statement.

```
repeat
    < body of loop >
until < condition >
```

is realized in Python as

```
while True:
    < body of loop >
    if < condition > : break
```

The **while** True starts an infinite loop, terminated when < condition > becomes True.
The break Statement

repeat until as while true break

To exit a loop inside the body of a loop, the statement **break** occurs usually within an **if** statement.

```
repeat
  < body of loop >
until < condition >
```

is realized in Python as

```
while True:
  < body of loop >
  if < condition > : break
```

The **while True** starts an infinite loop, terminated when **< condition >** becomes True.
Binary Expansions with break
a better solution

The program below avoids the duplication of code:

# L-12 MCS 260 Mon 22 Sep 2008 binary expansion
#
# Use of break for repeat until.
#
print 'computing the binary expansion'
n = input('Give a number : ')
while True:
    (n,r) = divmod(n,2)
    print r
    if n == 0: break

Exercise: how to print bits in correct order?
Binary Expansions with break

a better solution

The program below avoids the duplication of code:

```python
# L-12 MCS 260 Mon 22 Sep 2008 binary expansion
#
# Use of break for repeat until.
#
print 'computing the binary expansion'
n = input('Give a number : ')
while True:
    (n,r) = divmod(n,2)
    print r
    if n == 0: break
```

Exercise: how to print bits in correct order?
Binary Expansions with break
a better solution

The program below avoids the duplication of code:

```python
# L-12 MCS 260 Mon 22 Sep 2008 binary expansion
#
# Use of break for repeat until.
#
print 'computing the binary expansion'
n = input('Give a number : ')
while True:
    (n,r) = divmod(n,2)
    print r
    if n == 0: break
```

Exercise: how to print bits in correct order?
Binary Expansions with break
a better solution

The program below avoids the duplication of code:

```python
# L-12 MCS 260 Mon 22 Sep 2008 binary expansion
#
# Use of break for repeat until.
#
print 'computing the binary expansion'
n = input('Give a number : ')
while True:
    (n,r) = divmod(n,2)
    print r
    if n == 0: break
```

Exercise: how to print bits in correct order?
Two Loops, Two Breaks

As long as the number typed in by the user is nonnegative, the loop continues.

A break only effects the one loop it is in.

```python
# L-12 MCS 260 Mon 22 Sep 2008 binary expansion
#
# A break only effects one loop.
#
print 'computing the binary expansion'
while True:
    n = input('Give a number (< 0 to exit) : ')
    if n < 0: break
    while True:
        (n,r) = divmod(n,2)
        print r
        if n == 0: break
```

A break only effects the one loop it is in.
Two Loops, Two Breaks

As long as the number typed in by the user is nonnegative, the loop continues.

```python
# L-12 MCS 260 Mon 22 Sep 2008 binary expansion
#
# A break only effects one loop.
#
print 'computing the binary expansion'
while True:
    n = input('Give a number (< 0 to exit) : ')
    if n < 0: break
    while True:
        (n,r) = divmod(n,2)
        print r
        if n == 0: break
```

A break only effects the one loop it is in.
As long as the number typed in by the user is nonnegative, the loop continues.

```python
while True:
    n = input('Give a number (< 0 to exit) : ')
    if n < 0: break

while True:
    (n,r) = divmod(n,2)
    print r
    if n == 0: break
```

A break only effects the one loop it is in.
As long as the number typed in by the user is nonnegative, the loop continues.

```python
# L-12 MCS 260 Mon 22 Sep 2008 binary expansion
#
# A break only effects one loop.
#
print 'computing the binary expansion'
while True:
    n = input('Give a number (< 0 to exit) : ')
    if n < 0: break
    while True:
        (n,r) = divmod(n,2)
        print r
        if n == 0: break
```

A break only effects the one loop it is in.
Running Simulations
repeat until: break

1. Simulation
   Monte Carlo methods
   random numbers

2. Repeat Until
   binary expansion
   break statement

3. Arrays and nested for Loops
   arrays represent matrices
   searching a two dimensional array

4. Summary + Assignments
Arrays represent Matrices
using numpy

Python has no array type, we use numpy:

```python
>>> from numpy import *
>>> A = zeros((2,3),int)
>>> A
array([[0, 0, 0],
       [0, 0, 0]])
```

We can define arrays via a list of lists:

```python
>>> B = array([[1,2,3],[4,5,6]])
>>> B
array([[1, 2, 3],
       [4, 5, 6]])
```

Two ways of accessing elements:

```python
>>> B[1][1]
5
>>> B[1,1]
5
```
Arrays represent Matrices

Python has no array type, we use numpy:

```python
>>> from numpy import *
>>> A = zeros((2,3),int)
>>> A
array([[0, 0, 0],
       [0, 0, 0]])
```

We can define arrays via a list of lists:

```python
>>> B = array([[1,2,3],[4,5,6]])
>>> B
array([[1, 2, 3],
       [4, 5, 6]])
```

Two ways of accessing elements:

```python
>>> B[1][1]
5
>>> B[1,1]
5
```
Arrays represent Matrices using numpy

Python has no array type, we use `numpy`:

```python
>>> from numpy import *
>>> A = zeros((2,3),int)
>>> A
array([[0, 0, 0],
       [0, 0, 0]])
```

We can define arrays via a list of lists:

```python
>>> B = array([[1,2,3],[4,5,6]])
>>> B
array([[1, 2, 3],
       [4, 5, 6]])
```

Two ways of accessing elements:

```python
>>> B[1][1]
5
>>> B[1,1]
5
```
Random Arrays

A 2-by-3 array with elements in $[-10, +10]$:

```python
>>> X = random.randint(-10,10,(2,3))
array([[-5, -10,  0],
       [ 7,   1,  -5]])
```

Random arrays of floats:

```python
>>> Y = random.rand(2,3)
array([[ 0.43049771,  0.62879832,  0.54863368],
       [ 0.37764022,  0.32186191,  0.43726834]])
>>> Z = random.randn(2,3)
array([[ 1.05289782, -0.01733416, -0.29531646]])
```
Random Arrays

A 2-by-3 array with elements in $[-10, +10]$:

```python
>>> X = random.randint(-10, 10, (2,3))
>>> X
array([[-5, -10,  0],
       [ 7,   1, -5]])
```

Random arrays of floats:

```python
>>> Y = random.rand(2,3)
>>> Y
array([[0.43049771, 0.62879832, 0.54863368],
       [0.37764022, 0.32186191, 0.43726834]])
>>> Z = random.randn(2,3)
>>> Z
array([[-0.55872826, -0.342225,  0.50209898],
       [1.05289782, -0.01733416, -0.29531646]])
```
Running Simulations
repeat until: break

1. Simulation
   Monte Carlo methods
   random numbers

2. Repeat Until
   binary expansion
   break statement

3. Arrays and nested for Loops
   arrays represent matrices
   searching a two dimensional array

4. Summary + Assignments
Search a 2-dimensional Array

Problem statement:

Input:  
A is an array of $n$ rows and $m$ columns,
$x$ is some number.

Output: if $A[i, j]$ equals $x$, then print $(i, j)$,
else print $x$ does not occur in $A$.

We develop an interactive program:

1. The user provides $n$ and $m$
2. The computer generates $n$-by-$m$ array $A$ of random integer numbers in the interval $[-100, +100]$.
3. The program prompts the user for $x$, and
4. searches $A$ for $x$ and prints search result.
Search a 2-dimensional Array

Problem statement:

Input:  A is an array of $n$ rows and $m$ columns,  
        $x$ is some number.

Output: if $A[i,j]$ equals $x$, then print $(i,j)$,  
        else print $x$ does not occur in $A$.

We develop an interactive program:

1. The user provides $n$ and $m$
2. The computer generates $n$-by-$m$ array $A$ of random integer numbers in the interval $[-100, +100]$.
3. The program prompts the user for $x$, and
4. searches $A$ for $x$ and prints search result.
Search a 2-dimensional Array

Problem statement:

Input: A is an array of \( n \) rows and \( m \) columns, 
\( x \) is some number.

Output: if \( A[i,j] \) equals \( x \), then print \((i,j)\),
else print \( x \) does not occur in \( A \).

We develop an interactive program:

1. The user provides \( n \) and \( m \)
2. The computer generates \( n \)-by-\( m \) array \( A \) of random 
   integer numbers in the interval \([-100, +100]\).
3. The program prompts the user for \( x \), and
4. searches \( A \) for \( x \) and prints search result.
$ python findelem.py

give number of rows : 4

give number of columns : 5

a random 4-by-5 array :

\[
\begin{bmatrix}
-13 & -9 & -35 & -82 & -21 \\
-46 & 99 & 62 & 93 & 37 \\
88 & 47 & 66 & -12 & 36 \\
-59 & 95 & 53 & -97 & -93 \\
\end{bmatrix}
\]

give a number : 47

found 47 at [2,1]

If the given number does not occur, then
‘%d does not occur in array’ is printed.
Running the Code at the command prompt

```python
$ python findelem.py

give number of rows : 4

give number of columns : 5

a random 4-by-5 array :

[[-13  -9  -35  -82  -21]
 [-46   99   62   93   37]
 [ 88   47   66  -12   36]
 [-59   95   53  -97  -93]]

give a number : 47

found 47 at [2,1]

If the given number does not occur, then '
%d does not occur in array' is printed.
Running the Code
at the command prompt

$ python findelem.py
give number of rows : 4
give number of columns : 5
a random 4-by-5 array :

\[
\begin{bmatrix}
-13 & -9 & -35 & -82 & -21 \\
-46 & 99 & 62 & 93 & 37 \\
88 & 47 & 66 & -12 & 36 \\
-59 & 95 & 53 & -97 & -93 \\
\end{bmatrix}
\]
give a number : 47
found 47 at [2,1]

If the given number does not occur, then '
%d does not occur in array' is printed.
Simulation
Monte Carlo methods
random numbers

Repeat Until
binary expansion
break statement

Arrays and nested for
Loops
arrays represent matrices
searching a two dimensional array

Summary + Assignments

Flowchart

\[ i = 0; \ j = 0 \]

\[ i < n? \]

\[ i = i + 1 \]

\[ j = j + 1 \]

\[ A_{i,j} = x? \]

\[ \text{print } (i, j) \]

\[ \text{print 'not found'} \]
Simulation
Monte Carlo methods
random numbers

Repeat Until
binary expansion
break statement

Arrays and nested for Loops
arrays represent matrices
searching a two dimensional array

Summary + Assignments

Flowchart

\( i = 0; \ j = 0 \)

\( i < n? \) False \( \rightarrow \) print 'not found'

\( i < n? \) True

\( j < m? \) False \( \rightarrow \) \( i = i + 1 \)

\( j < m? \) True

\( A_{i,j} = x? \) False \( \rightarrow \) \( j = j + 1 \)

\( A_{i,j} = x? \) True \( \rightarrow \) print \((i, j)\)
Simulation
Monte Carlo methods
random numbers

Repeat Until
binary expansion
break statement

Arrays and nested for loops
arrays represent matrices
searching a two dimensional array

Summary + Assignments
**Flowchart**

```

```

```
```

```
```

```
```

```
```

```
```
Simulation
Monte Carlo methods
random numbers

Repeat Until
binary expansion
break statement

Arrays and nested for Loops
arrays represent matrices
searching a two dimensional array

Summary + Assignments

Flowchart

\[ i = 0; \ j = 0 \]

\[ i < n? \]

\[ j < m? \]

\[ A_{i,j} = x? \]

\[ \text{print } (i, j) \]

\[ \text{print 'not found'} \]

\[ i = i + 1 \]

\[ j = j + 1 \]
# L-12 MCS 260 Mon 22 Sep 2008 : find element

# Illustration of a double for loop to find
# an element in a two dimensional array.

from numpy import *
n = input('give number of rows : ')
m = input('give number of columns : ')
A = random.randint(-100,100,(n,m))
print 'a random %d-by-%d array :
' % (n,m), A
x = input('give a number : ')
Start of findelem.py

# L-12 MCS 260 Mon 22 Sep 2008 : find element

# Illustration of a double for loop to find
# an element in a two dimensional array.

from numpy import *
n = input('give number of rows : ')
m = input('give number of columns : ')
A = random.randint(-100,100,(n,m))
print 'a random %d-by-%d array :
' % (n,m), A
x = input('give a number : ')
the double for loop

Search $n$-by-$m$ array $A$ for $x$:

```python
found = False
for i in range(0,n):
    for j in range(0,m):
        if A[i,j] == x:
            found = True
            break
if found: break
if found:
    print 'found %d at [%d,%d]' % (x,i,j)
else:
    print '%d does not occur in array' % x
```
the double for loop

Search \( n \)-by-\( m \) array \( A \) for \( x \):

```python
found = False
for i in range(0,n):
    for j in range(0,m):
        if A[i,j] == x:
            found = True
            break
if found:
    print 'found %d at [%d,%d]' % (x,i,j)
else:
    print '%d does not occur in array' % x
```
the double for loop

Search \( n \)-by-\( m \) array \( A \) for \( x \):

```python
found = False
for i in range(0,n):
    for j in range(0,m):
        if A[i,j] == x:
            found = True
            break
if found:
    print 'found %d at [%d,%d]' % (x,i,j)
else:
    print '%d does not occur in array' % x
```
Summary + Assignments

We covered more of

- chapter 5 of *Python Power!*
- section 5.4 in *Computer Science, an overview.*

Assignments:

1. Use a stack to store the bits in the binary expansion to print the bits *after* the loop in the correct order.
2. Given a list of numbers between 0 and 100, define the algorithm to assign a letter grade to each number:
   \[ \geq 90: \text{A}, \in [80, 89]: \text{B}, \in [70, 79]: \text{C}, \text{etc.} \]
   Report at the end how many As, Bs, Cs, etc. Write the algorithm in words and draw a flowchart.
3. Implement exercise 2 in Python.
4. Write a Python program that generates \( n \) numbers uniformly distributed in \([0, 1]\) and counts how many numbers are \(< 0.5\).