Outline

Simulation
- Monte Carlo methods
- random numbers

Repeat Until
- binary expansion
- break statement

stats in Maple

Summary + Assignments

MCS 260 Lecture 12
Introduction to Computer Science
Jan Verschelde, 24 September 2007

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
random numbers
Repeat Until
binary expansion
break statement
stats in Maple
Summary + Assignments

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
random numbers
Repeat Until
binary expansion
break statement
stats in Maple
Summary + Assignments

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
random numbers
Repeat Until
binary expansion
break statement
stats in Maple
Summary + Assignments

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
random numbers
Repeat Until
binary expansion
break statement
stats in Maple
Summary + Assignments

Outline

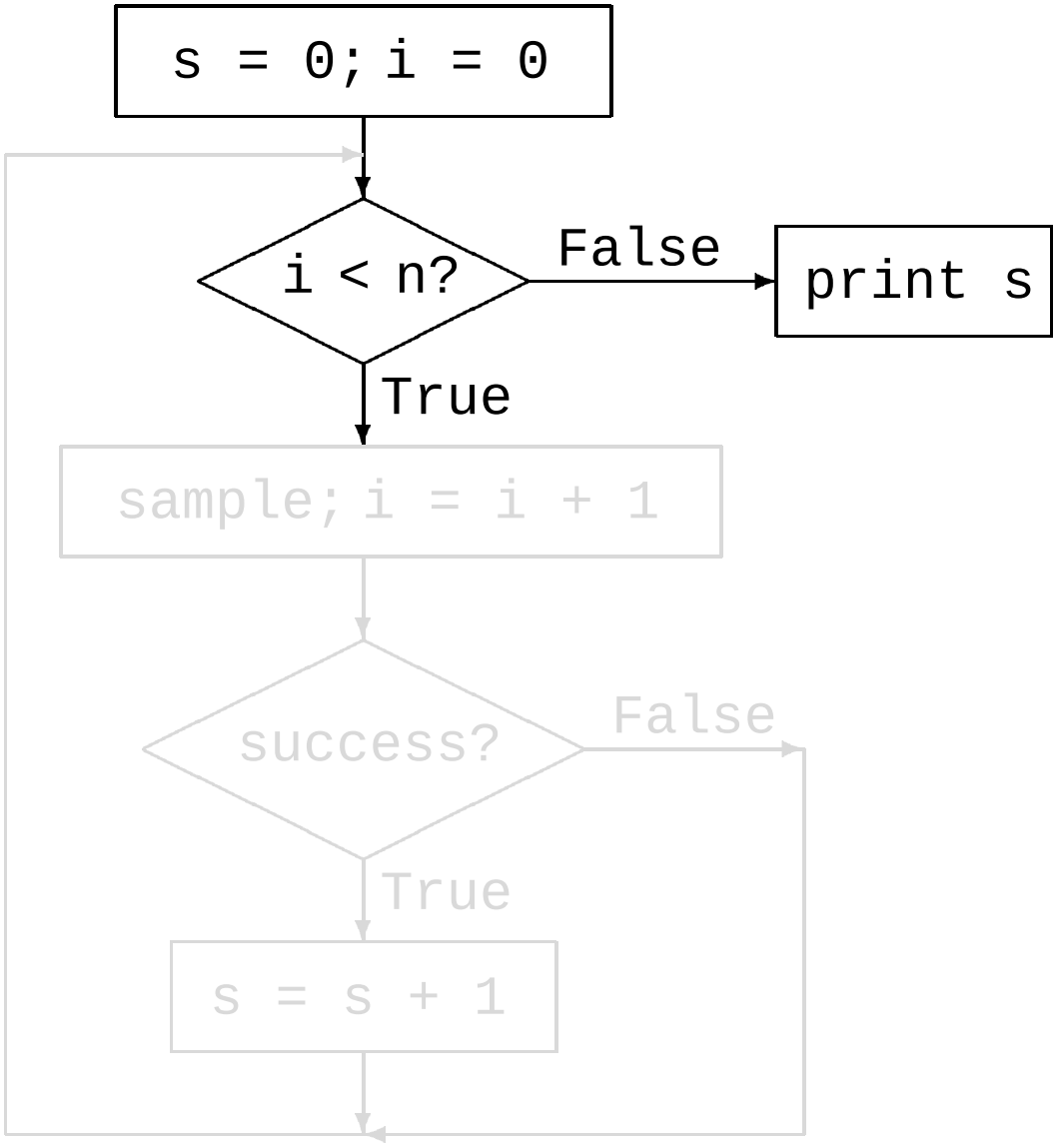
Simulation
- Monte Carlo methods
- random numbers

Repeat Until
- binary expansion
- break statement

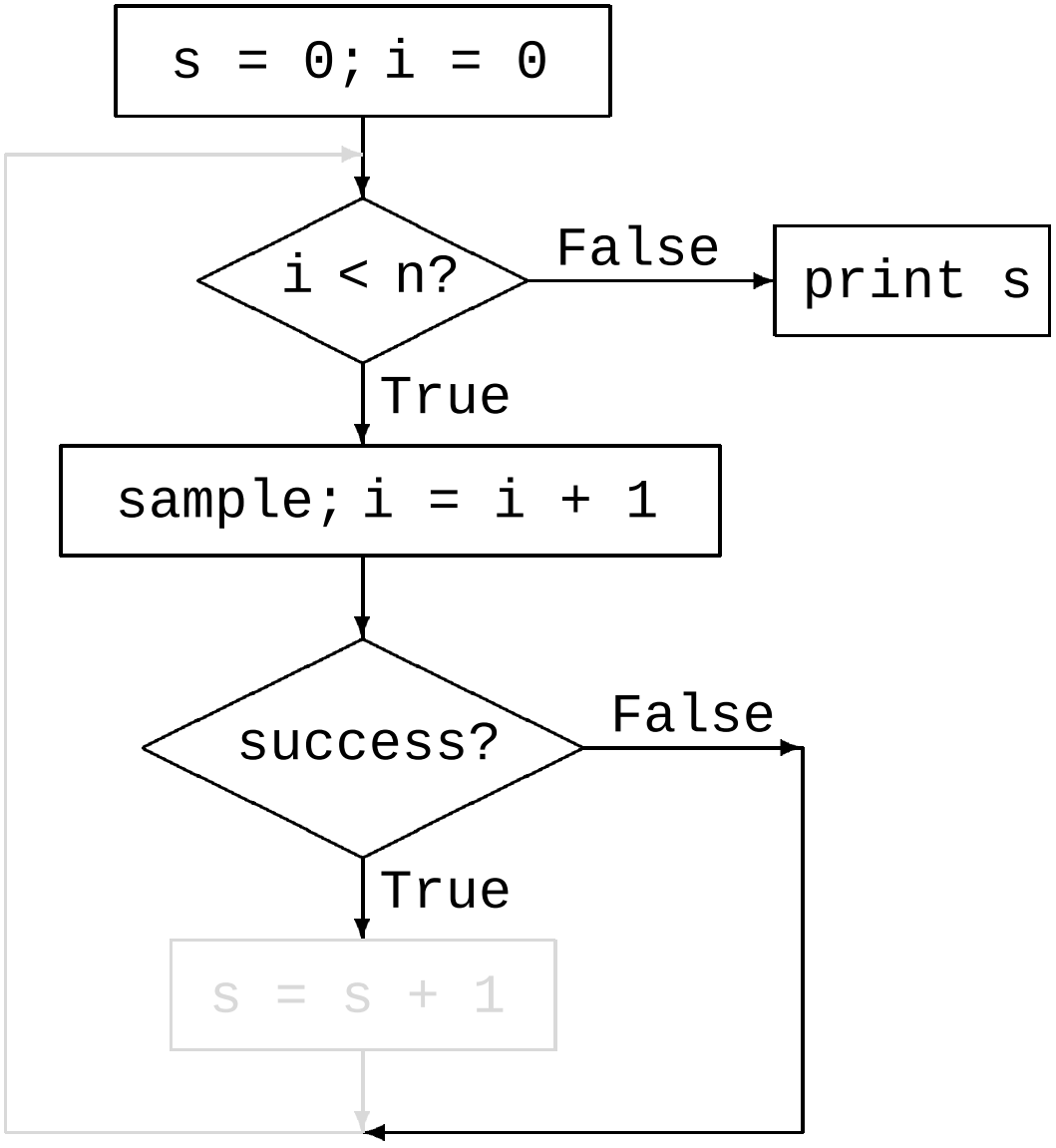
stats in Maple

Summary + Assignments

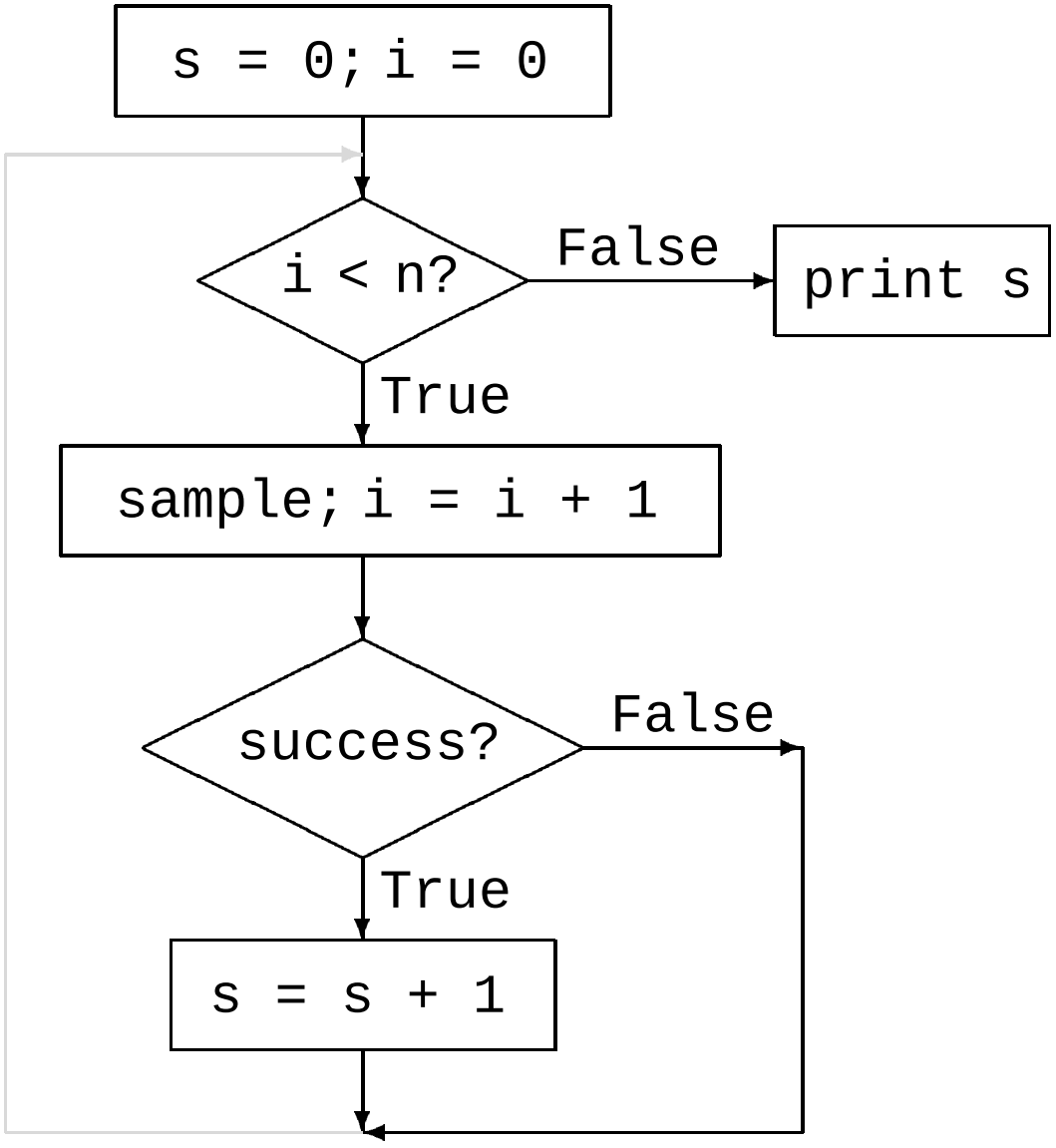
Flowchart for general Simulations



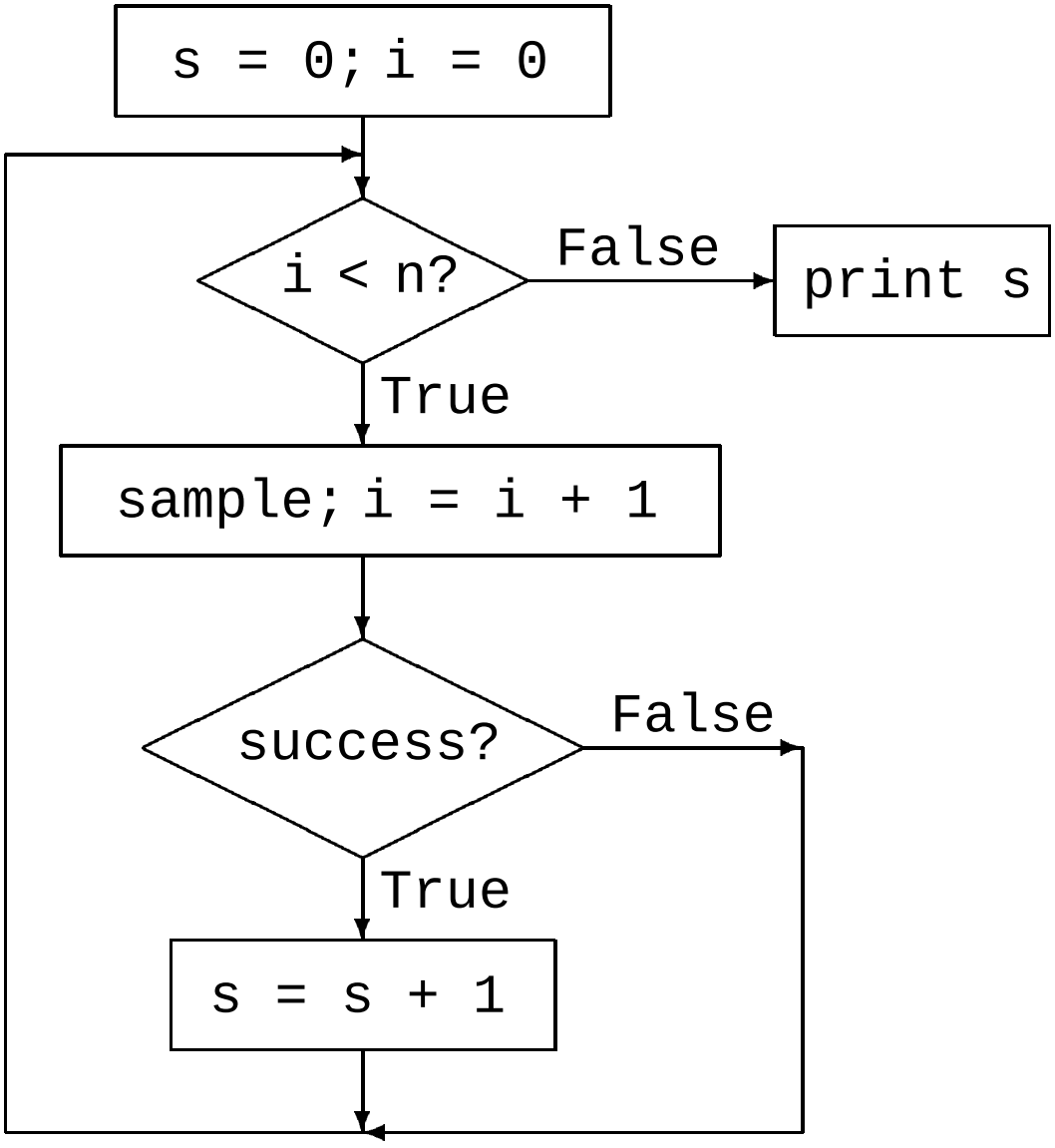
Flowchart for general Simulations



Flowchart for general Simulations



Flowchart for general Simulations



Outline

Simulation
- Monte Carlo methods
- random numbers

Repeat Until
- binary expansion
- break statement

stats in Maple

Summary + Assignments

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)`.

Simulation
Monte Carlo methods
random numbers
Repeat Until
binary expansion
break statement
stats in Maple
Summary + Assignments

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)`.

Simulation
Monte Carlo methods
random numbers
Repeat Until
binary expansion
break statement
stats in Maple
Summary + Assignments

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)`.

Simulation
Monte Carlo methods
random numbers
Repeat Until
binary expansion
break statement
stats in Maple
Summary + Assignments

Using Random Numbers

a sample program

A sample program randuse.py:

```
# L-12 MCS 260 Mon 24 Sep 2007  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:

```
# L-12 MCS 260 Mon 24 Sep 2007  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:

```
# L-12 MCS 260 Mon 24 Sep 2007  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)
```

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 π.

Generate random uniformly distributed points with coordinates $(x, y) \in [-1, +1] \times [-1, +1]$.

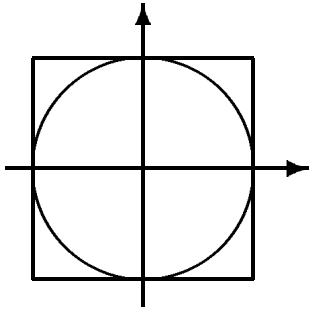
We count a success when $x^2 + y^2 \leq 1$.

Simulation
Monte Carlo methods
random numbers
Repeat Until
binary expansion
break statement
stats in Maple
Summary + Assignments

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 π.



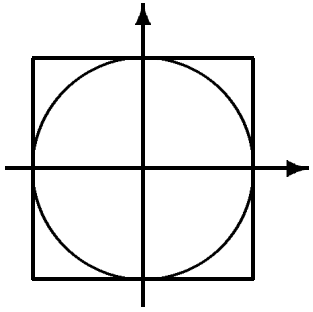
Generate random uniformly distributed points with coordinates $(x, y) \in [-1, +1] \times [-1, +1]$.

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 π.

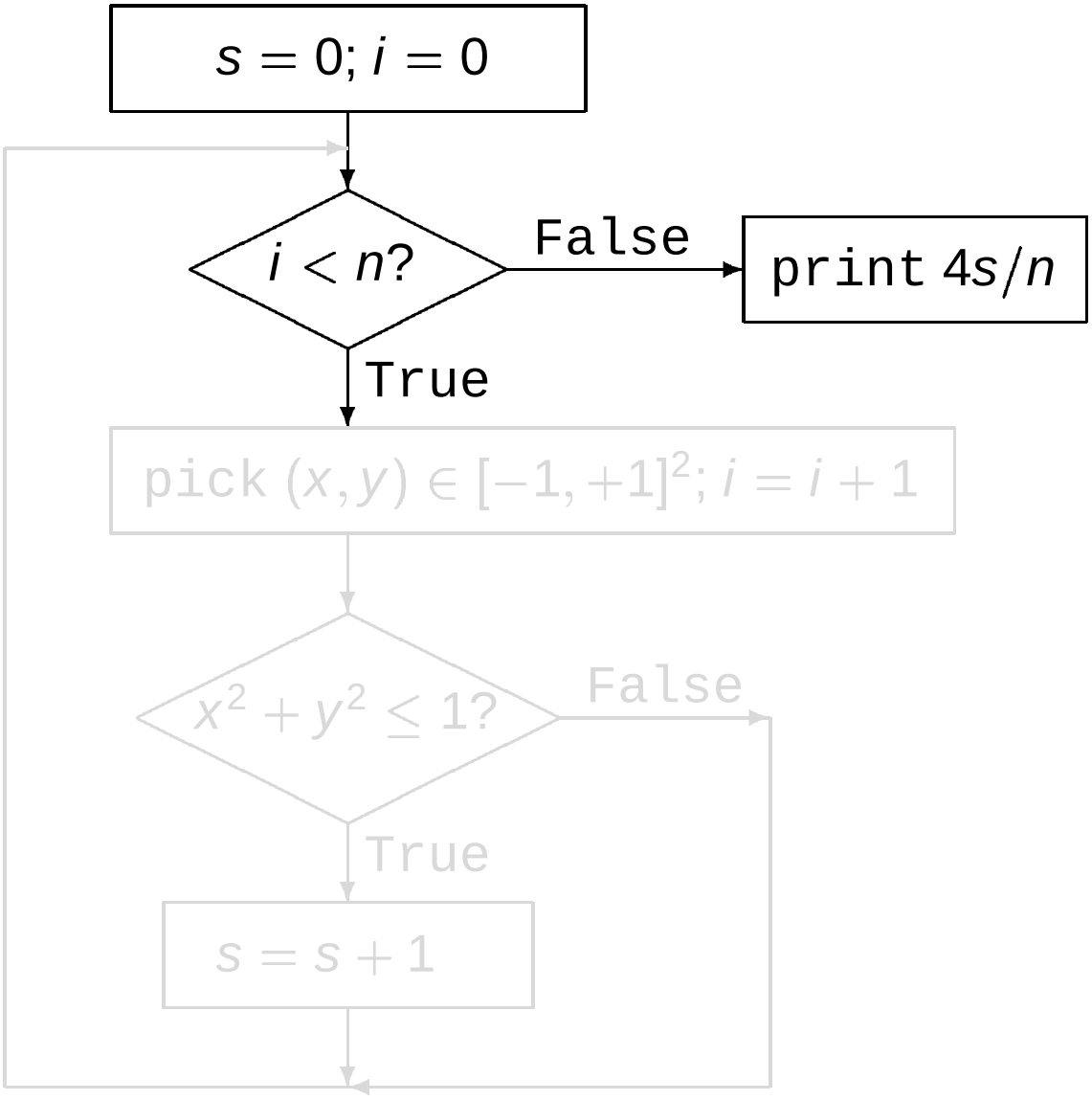


Generate random uniformly distributed points with coordinates $(x, y) \in [-1, +1] \times [-1, +1]$.

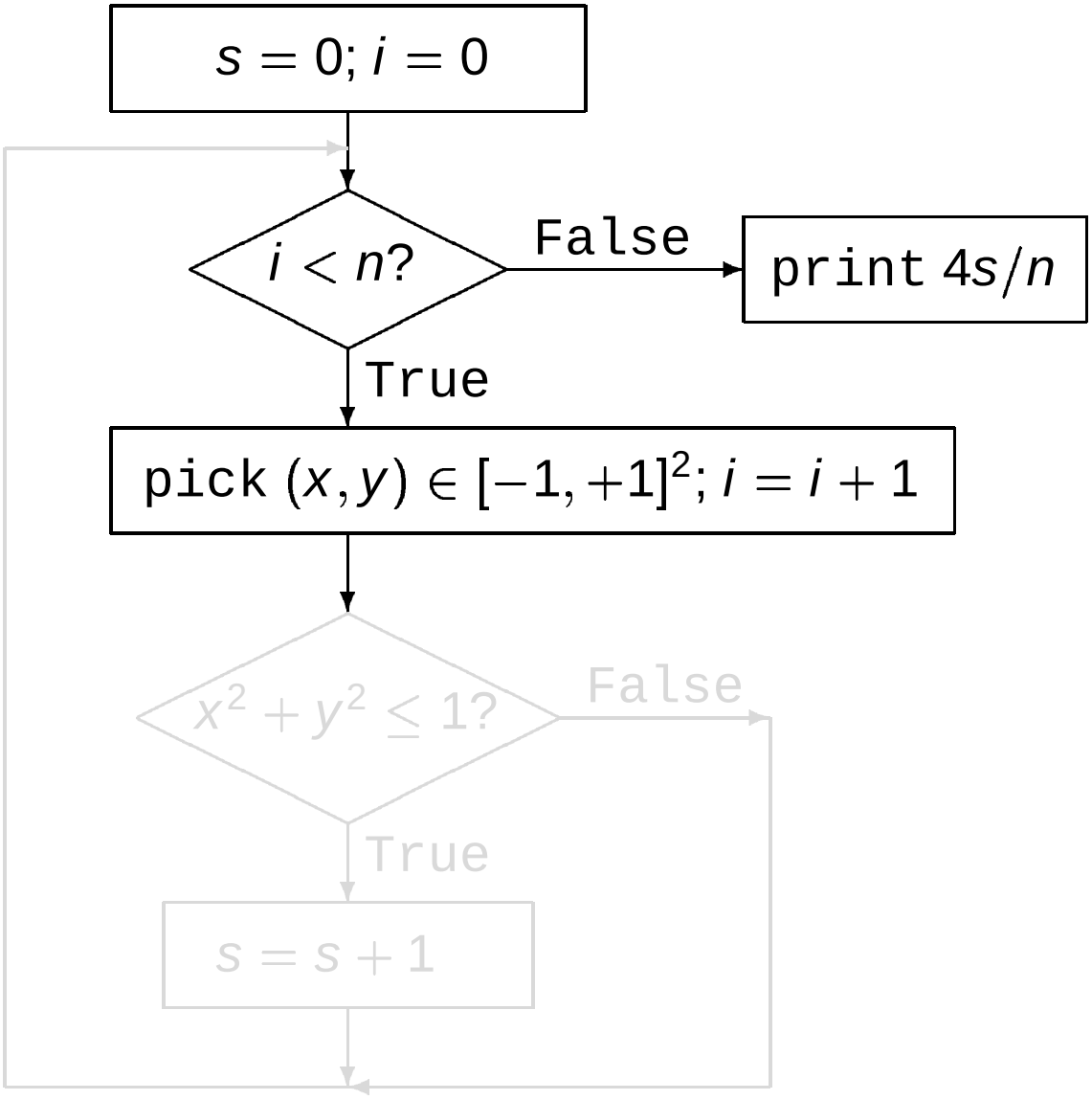
We count a success when $x^2 + y^2 \leq 1$.

Simulation
Monte Carlo methods
random numbers
Repeat Until
binary expansion
break statement
stats in Maple
Summary + Assignments

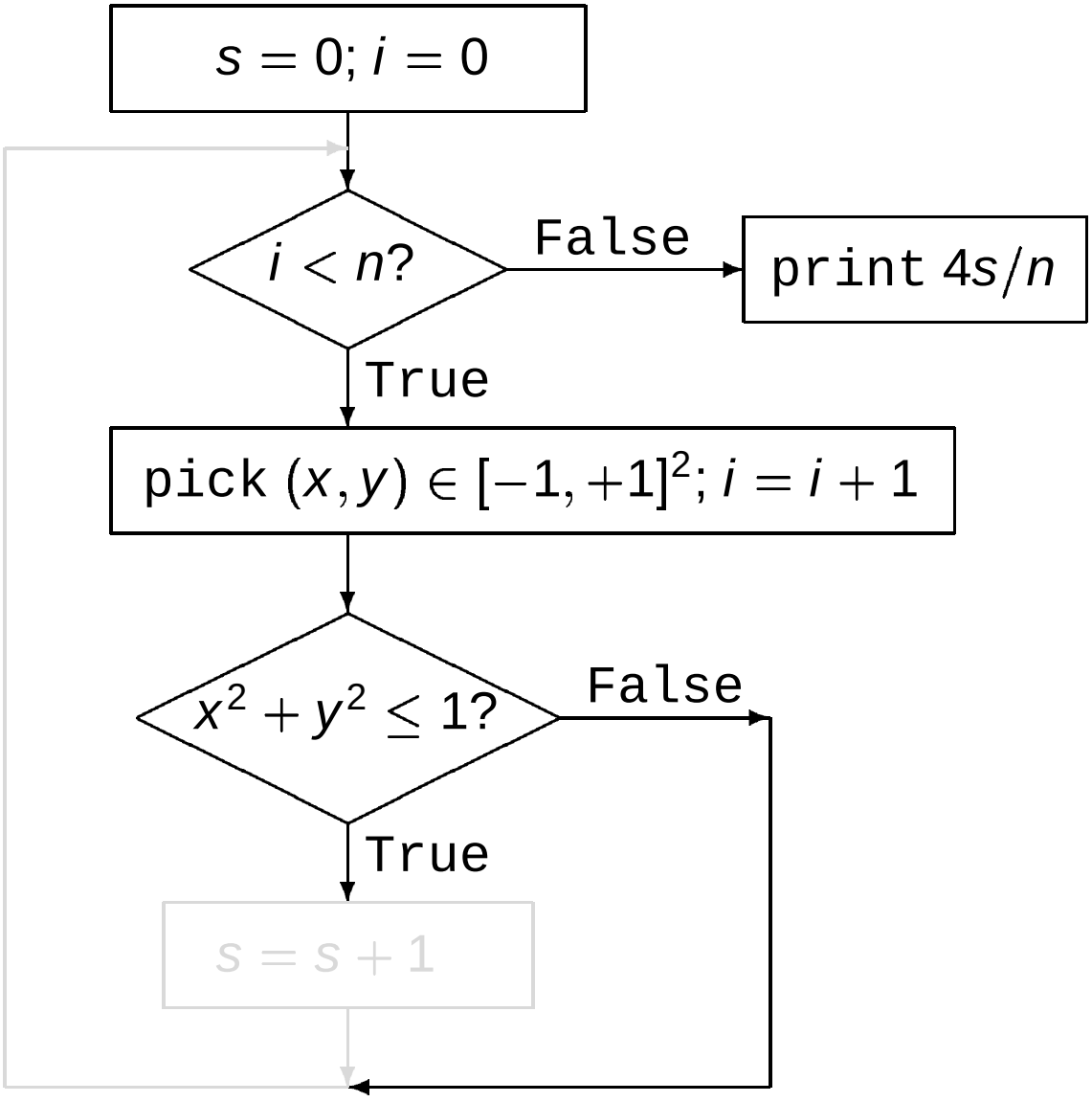
Flowchart for Estimating π



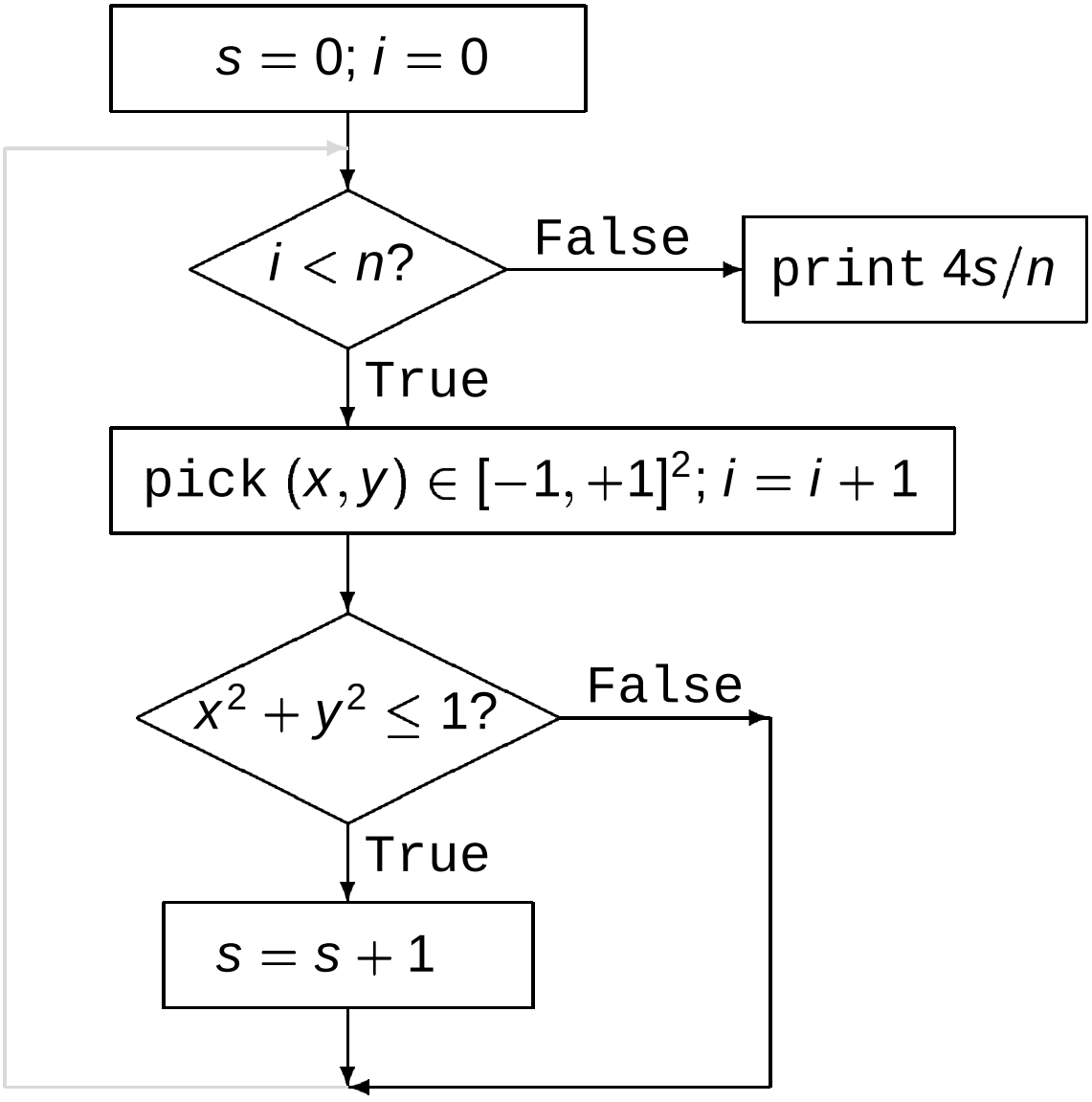
Flowchart for Estimating π



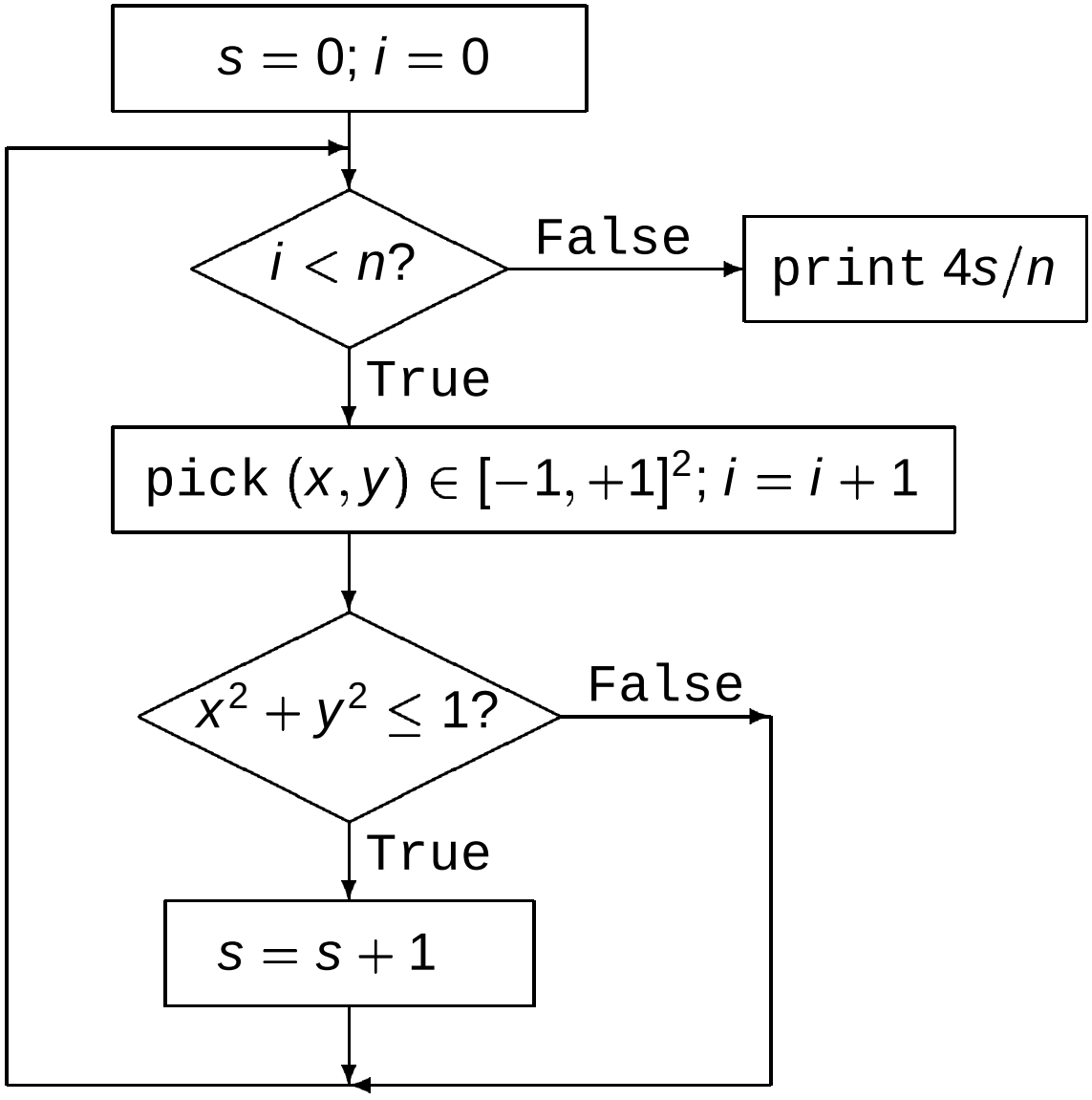
Flowchart for Estimating π



Flowchart for Estimating π



Flowchart for Estimating π



Estimating π

The program mc4pi.py:

```
# L-12 MCS 260 Mon 24 Sep 2007  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 π

The program mc4pi.py:

```
# L-12 MCS 260 Mon 24 Sep 2007  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 π

The program mc4pi.py:

```
# L-12 MCS 260 Mon 24 Sep 2007  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 π

The program mc4pi.py:

```
# L-12 MCS 260 Mon 24 Sep 2007  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 $[-1,+1]^2$.

Outline

Simulation
- Monte Carlo methods
- random numbers

Repeat Until
- binary expansion
- break statement

stats in Maple

Summary + Assignments

Binary Expansions

repeat until loops

The bits of a number are the remainders of division by 2.

divmod() is an intrinsic operation:

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

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

Pseudo code to compute the binary expansion:

```
n = input()
repeat
   (n,r) = divmod(n,2)
   print r
until (n == 0).
```


Simulation
Monte Carlo methods
random numbers
Repeat Until
binary expansion
break statement
stats in Maple
Summary + Assignments

Binary Expansions

repeat until loops

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

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

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

Pseudo code to compute the binary expansion:

```
n = input()
repeat
   (n,r) = divmod(n,2)
   print r
until (n == 0).
```


Simulation
Monte Carlo methods
random numbers
Repeat Until
binary expansion
break statement
stats in Maple
Summary + Assignments

Binary Expansions

repeat until loops

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

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

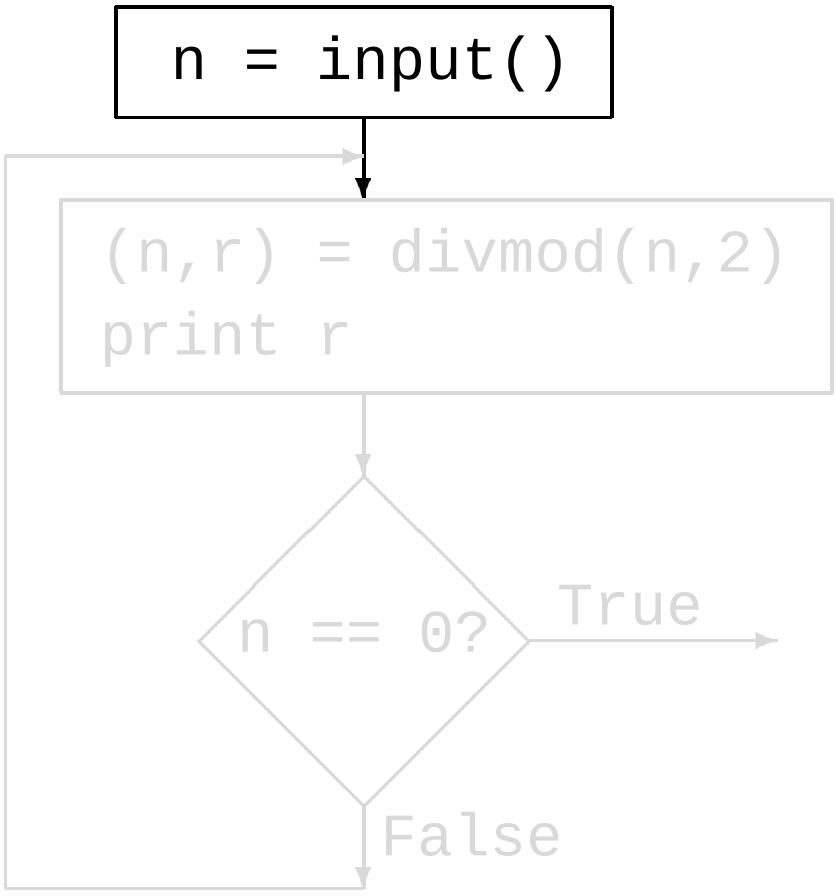
Use as `(n,r) = divmod(n,2)`
to obtain remainder n%2 in `r` and to replace n by n/2.

Pseudo code to compute the binary expansion:

```
n = input()
repeat
   (n,r) = divmod(n,2)
   print r
until (n == 0).
```

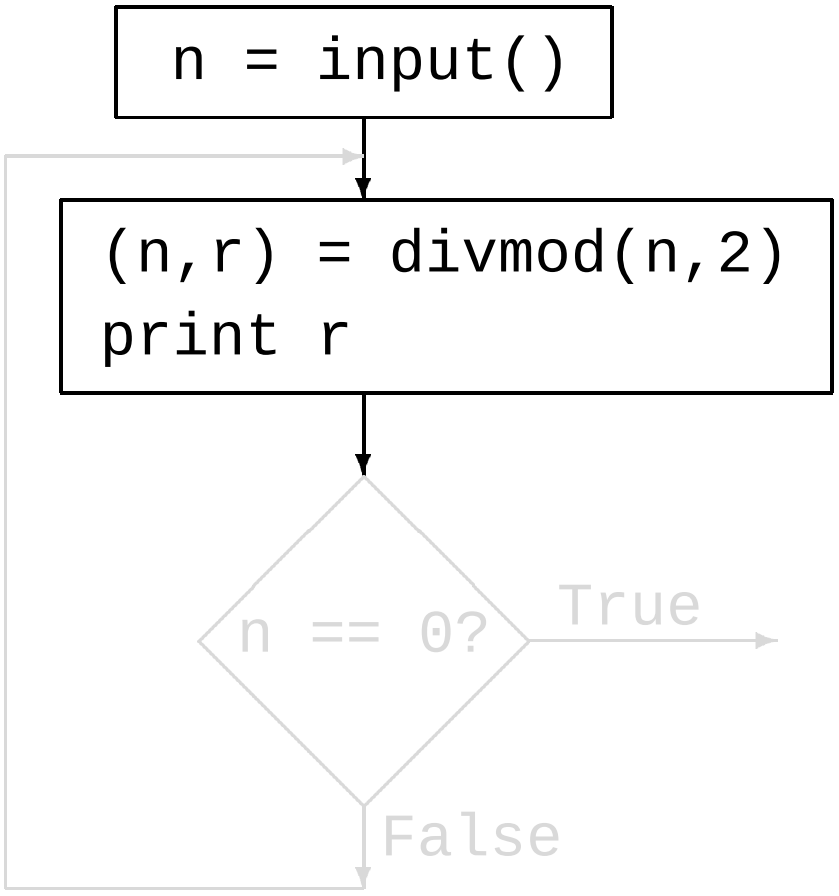
Flowchart of Binary Expansion

picture of repeat until



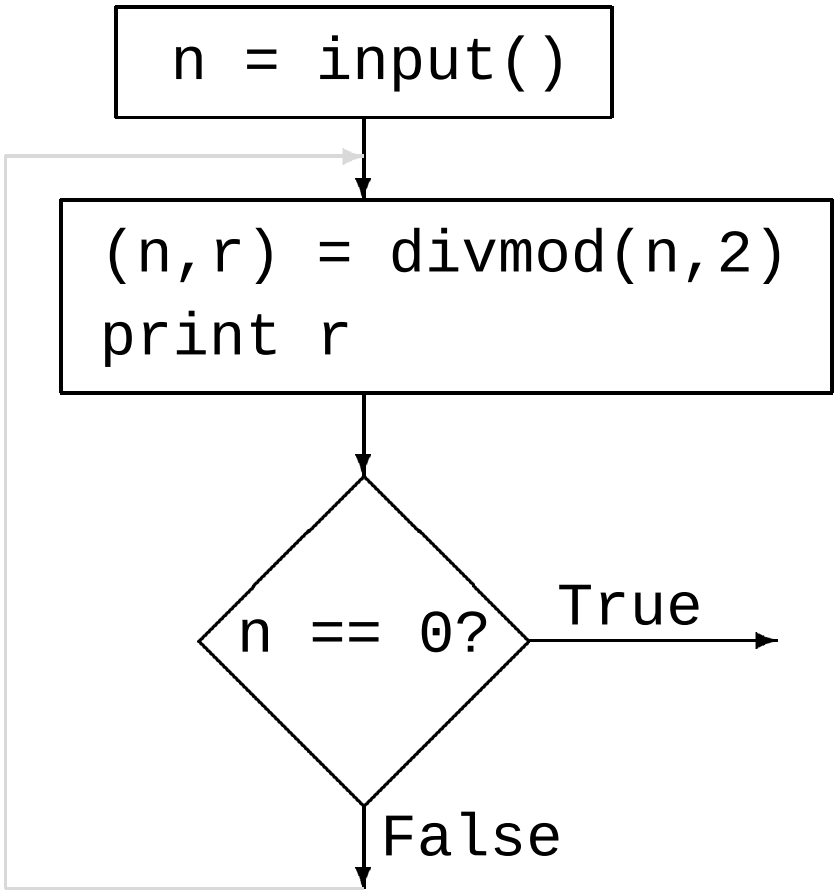
Flowchart of Binary Expansion

picture of repeat until



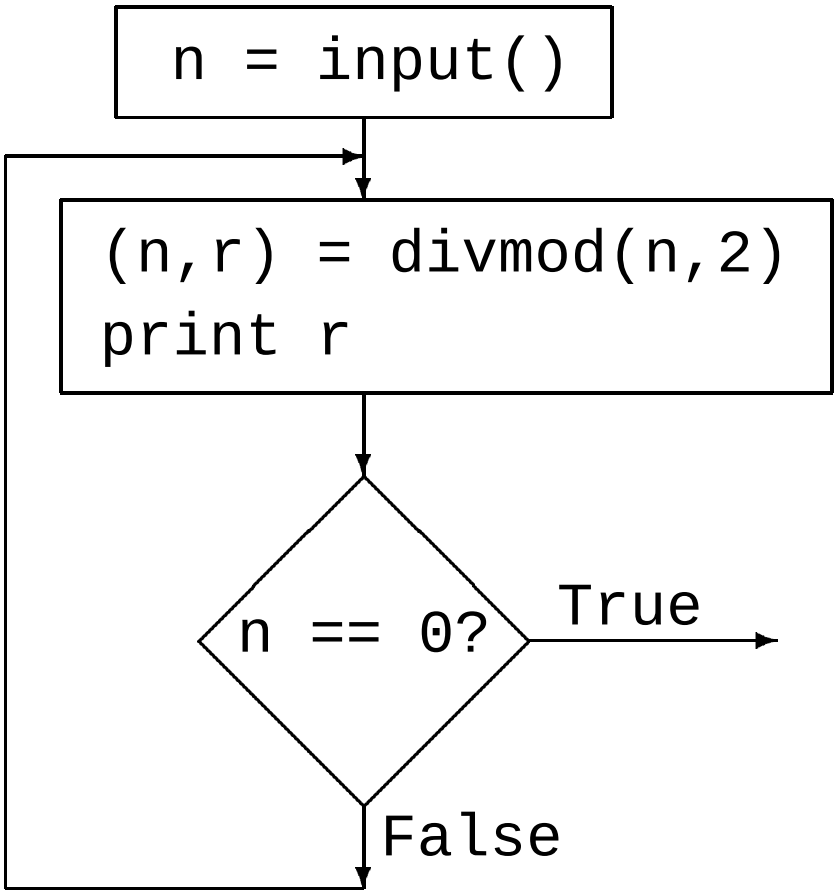
Flowchart of Binary Expansion

picture of repeat until



Flowchart of Binary Expansion

picture of repeat until



A first Python solution

```
# L-12 MCS 260 Mon 24 Sep 2007  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

A first Python solution

```
# L-12 MCS 260 Mon 24 Sep 2007  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

Outline

Simulation
- Monte Carlo methods
- random numbers

Repeat Until
- binary expansion
- break statement

stats in Maple

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.

Simulation
Monte Carlo methods
random numbers
Repeat Until
binary expansion
break statement
stats in Maple
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.

Simulation
Monte Carlo methods
random numbers
Repeat Until
binary expansion
break statement
stats in Maple
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.

Simulation
Monte Carlo methods
random numbers
Repeat Until
binary expansion
break statement
stats in Maple
Summary + Assignments

Binary Expansions with break

a better solution

The program below avoids the duplication of code:

```
# L-12 MCS 260 Mon 24 Sep 2007  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?

Simulation
Monte Carlo methods
random numbers
Repeat Until
binary expansion
break statement
stats in Maple
Summary + Assignments

Binary Expansions with break

a better solution

The program below avoids the duplication of code:

```
# L-12 MCS 260 Mon 24 Sep 2007  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:

```
# L-12 MCS 260 Mon 24 Sep 2007  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:

```
# L-12 MCS 260 Mon 24 Sep 2007  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

scope of a break

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

```
# L-12 MCS 260 Mon 24 Sep 2007  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

scope of a break

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

```
# L-12 MCS 260 Mon 24 Sep 2007  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

scope of a break

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

```
# L-12 MCS 260 Mon 24 Sep 2007  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

scope of a break

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

```
# L-12 MCS 260 Mon 24 Sep 2007  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.

Random Numbers in Maple

Maple offers random numbers generators for various probability distributions:

```
> with(stats[random]);
```

The `uniform` returns uniformly distributed numbers.

Generating 5 random numbers in $[-1, +2]$:

```
> uniform[-1,+2](5);
```

or without loading the package `stats[random]`:

```
> stats[random,uniform[-1,+2]](5);
```


Simulation
Monte Carlo methods
random numbers
Repeat Until
binary expansion
break statement
stats in Maple
Summary + Assignments

Random Numbers in Maple

Maple offers random numbers generators for various probability distributions:

```
> with(stats[random]);
```

The `uniform` returns uniformly distributed numbers.

Generating 5 random numbers in $[-1, +2]$:

```
> uniform[-1,+2](5);
```

or without loading the package `stats[random]`:

```
> stats[random,uniform[-1,+2]](5);
```

Summary + Assignments

We covered in *Making Use of Python*:

- towards the end of chapter 4;
- some of the topics in chapter 6.

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: ≥ 90: A, $\in [80, 89]$: B, $\in [70, 79]$: C, 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.