

# Review I

- 1 Policies & Material
- 2 General Questions
- 3 Python code and Flowcharts
- 4 Circuits and Logic Gates
- 5 Functions
- 6 Summary

MCS 260 Lecture 19  
Introduction to Computer Science  
Jan Verschelde, 24 February 2010

# Policies

furloughed midterm on Friday 26 February

Because of mandated furloughs, our first midterm exam, originally scheduled for this Friday, will not happen.

**The policy on skipping exams is:** If an exam is missed, then greater weight will be placed on the final exam, especially on the material covered on the missing exam.

So by default, you all skip the first midterm exam, and your final counts for 300 out of 700 points.

An extra project 3.5, due Wednesday 10 March, could give you 40 bonus points.

# Material Covered

The material breaks down in three parts:

- 1 computer science concepts: algorithms, computer architecture, functionality of operating systems, mass storage, formal languages, flowcharts and pseudocode, logic gates, transistors and flip flops, adder circuits, functional programming.
- 2 programming in Python: variables and assignments, elementary and composite types, intrinsic operations, if else elif, while and for, queues and stacks, top down design, functions, lambda.
- 3 mathematical CS: computer algebra, binary representations of numbers, precision and accuracy, using Sage, boolean algebra, truth tables, simulation and Monte Carlo, histograms.

# General Questions

- 1 List the characteristics of an algorithm.

**Answer:** precise and detailed enough to be implemented by a computer.

- 2 How many bytes are 15 terabytes?

**Answer:**  $10^{12}$  bytes.

- 3 Explain the difference between  $34/87$  and  $34.0/87$ .

**Answer:**  $34/87$  gives the integer division, whereas  $34.0/87$  returns a float.

- 4 What is an interrupt?

**Answer:** an interrupt causes an executing process to go in waiting or ready state.

# General Questions

- 1 List the characteristics of an algorithm.

**Answer:** precise and detailed enough to be implemented by a computer.

- 2 How many bytes are 15 terabytes?

**Answer:**  $10^{12}$  bytes.

- 3 Explain the difference between  $34/87$  and  $34.0/87$ .

**Answer:**  $34/87$  gives the integer division, whereas  $34.0/87$  returns a float.

- 4 What is an interrupt?

**Answer:** an interrupt causes an executing process to go in waiting or ready state.

# General Questions

- 1 List the characteristics of an algorithm.  
**Answer:** precise and detailed enough to be implemented by a computer.
- 2 How many bytes are 15 terabytes?  
**Answer:**  $10^{12}$  bytes.
- 3 Explain the difference between  $34/87$  and  $34.0/87$ .  
**Answer:**  $34/87$  gives the integer division, whereas  $34.0/87$  returns a float.
- 4 What is an interrupt?  
**Answer:** an interrupt causes an executing process to go in waiting or ready state.

# General Questions

- 1 List the characteristics of an algorithm.  
**Answer:** precise and detailed enough to be implemented by a computer.
- 2 How many bytes are 15 terabytes?  
**Answer:**  $10^{12}$  bytes.
- 3 Explain the difference between  $34/87$  and  $34.0/87$ .  
**Answer:**  $34/87$  gives the integer division, whereas  $34.0/87$  returns a float.
- 4 What is an interrupt?  
**Answer:** an interrupt causes an executing process to go in waiting or ready state.

# General Questions

- 1 List the characteristics of an algorithm.  
**Answer:** precise and detailed enough to be implemented by a computer.
- 2 How many bytes are 15 terabytes?  
**Answer:**  $10^{12}$  bytes.
- 3 Explain the difference between  $34/87$  and  $34.0/87$ .  
**Answer:**  $34/87$  gives the integer division, whereas  $34.0/87$  returns a float.
- 4 What is an interrupt?  
**Answer:** an interrupt causes an executing process to go in waiting or ready state.

## 5. Dispensing Coin Change

remember the first quiz?

Write a Python program that takes on input a nonnegative number less than 100.

The number represents the number of cents.

Compute the minimal number of coins (quarters, nickels, dimes, and pennies) that are needed to obtain the value of the input number.

For example, for 68 cents we need  
2 quarters, 1 dime, 1 nickel, and 3 pennies.

# Coin Change in Python

answer to question 5

```
x = input('give a number : ')
m = x
q = m/25 # number of quarters
m = m%25 # rest of money
d = m/10 # number of dimes
m = m%10 # rest of money
n = m/5 # number of nickels
p = m%5 # number of pennies
print '$0.%d is \
#quarters = %d, #dimes = %d, \
#nickels = %d, #pennies = %d' % (x,q,d,n,p)
```

# Coin Change in Python

answer to question 5

[Policies & Material](#)[General Questions](#)[Python code and Flowcharts](#)[Circuits and Logic Gates](#)[Functions](#)[Summary](#)

```
x = input('give a number : ')
m = x
q = m/25 # number of quarters
m = m%25 # rest of money
d = m/10 # number of dimes
m = m%10 # rest of money
n = m/5 # number of nickels
p = m%5 # number of pennies
print '$0.%d is \
#quarters = %d, #dimes = %d, \
#nickels = %d, #pennies = %d' % (x,q,d,n,p)
```

# Coin Change in Python

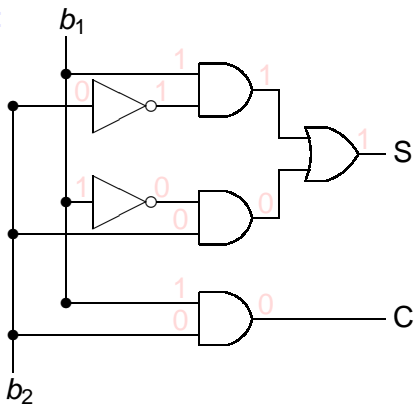
answer to question 5

```
x = input('give a number : ')
m = x
q = m/25 # number of quarters
m = m%25 # rest of money
d = m/10 # number of dimes
m = m%10 # rest of money
n = m/5 # number of nickels
p = m%5 # number of pennies
print '$0.%d is \
#quarters = %d, #dimes = %d, \
#nickels = %d, #pennies = %d' % (x,q,d,n,p)
```

## 6. How Circuits Work

Consider the circuit shown below. For  $b_1 = 1$  and  $b_2 = 0$  as input, indicate on the picture below the input and output for every logical gate.

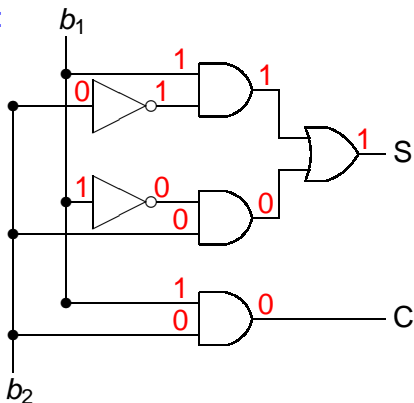
Answer:



## 6. How Circuits Work

Consider the circuit shown below. For  $b_1 = 1$  and  $b_2 = 0$  as input, indicate on the picture below the input and output for every logical gate.

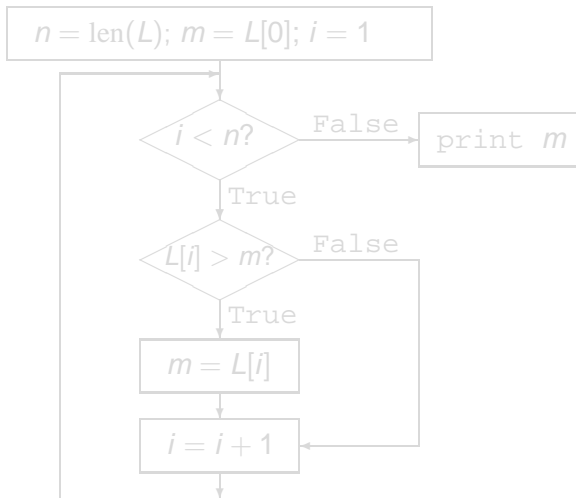
Answer:



## 7. Flowchart of a Search

Draw the flowchart for the algorithm to search for the maximum in a list of unsorted numbers.

Answer:

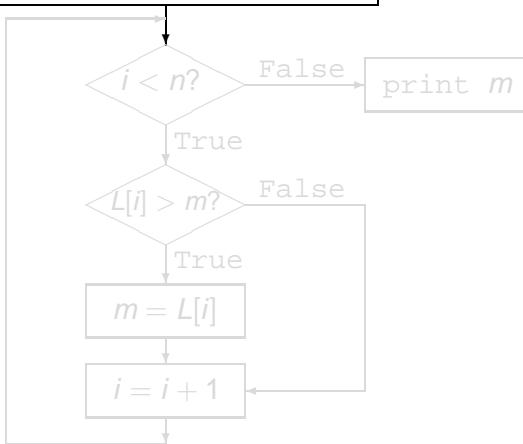


## 7. Flowchart of a Search

Draw the flowchart for the algorithm to search for the maximum in a list of unsorted numbers.

Answer:

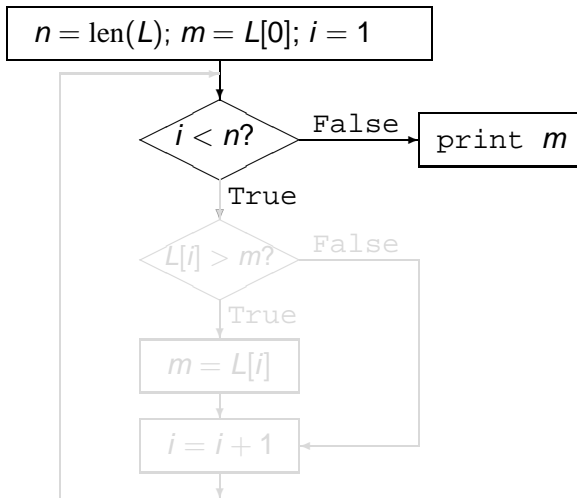
```
n = len(L); m = L[0]; i = 1
```



## 7. Flowchart of a Search

Draw the flowchart for the algorithm to search for the maximum in a list of unsorted numbers.

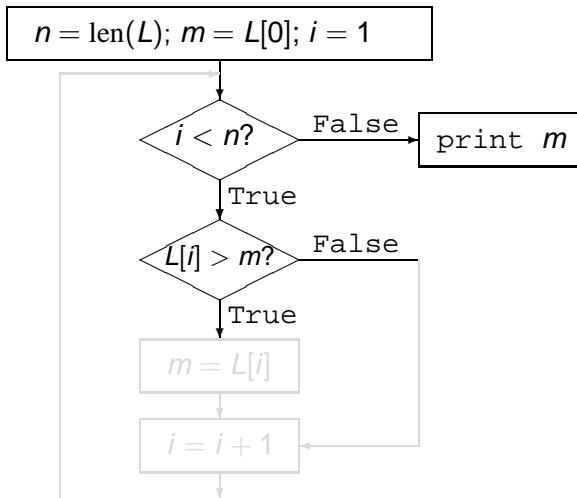
Answer:



## 7. Flowchart of a Search

Draw the flowchart for the algorithm to search for the maximum in a list of unsorted numbers.

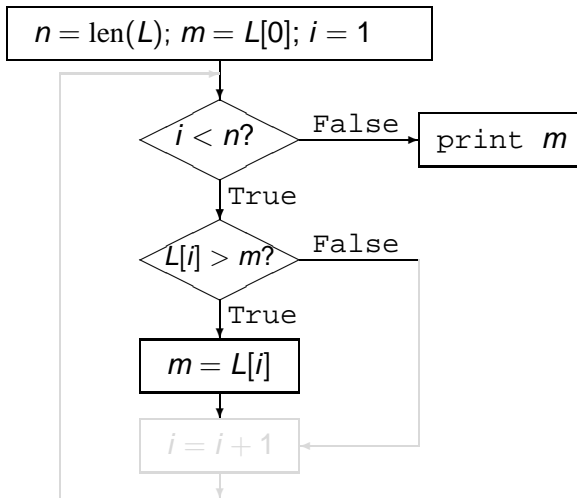
Answer:



## 7. Flowchart of a Search

Draw the flowchart for the algorithm to search for the maximum in a list of unsorted numbers.

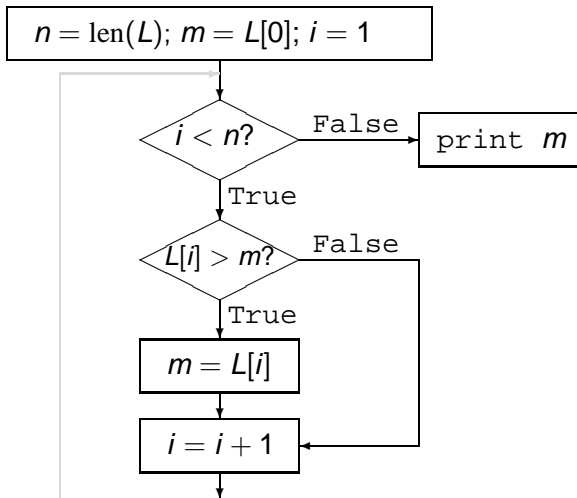
Answer:



## 7. Flowchart of a Search

Draw the flowchart for the algorithm to search for the maximum in a list of unsorted numbers.

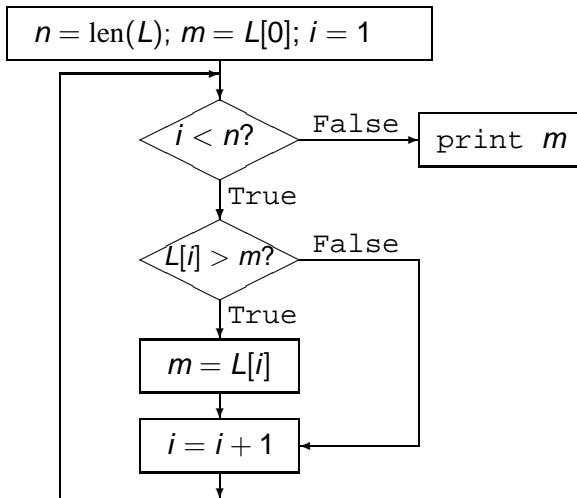
Answer:



## 7. Flowchart of a Search

Draw the flowchart for the algorithm to search for the maximum in a list of unsorted numbers.

Answer:



## 8. The Exclusive OR: XOR

An exclusive or, denoted as XOR returns False only when the inputs are both the same, and True otherwise.

- 1 Give the truth table for the exclusive or.
- 2 Show how the XOR can be realized with NOT, OR, and AND by giving the logical expressions and their corresponding truth tables.
- 3 Draw the circuit for the XOR, using the symbols for the gates NOT, OR, and AND.

# Truth Tables for XOR

Policies &  
Material

General  
Questions

Python code  
and  
Flowcharts

Circuits and  
Logic Gates

Functions

Summary

## 1 Truth table for XOR:

x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0

## 2 Realization of XOR:

x	y	x OR y	x AND y	NOT (x AND y)	b
0	0	0	0	1	0
0	1	1	0	1	1
1	0	1	0	1	1
1	1	1	1	0	0

where  $b = (x \text{ OR } y) \text{ AND } (\text{NOT } (x \text{ AND } y))$

# Truth Tables for XOR

Policies &  
Material

General  
Questions

Python code  
and  
Flowcharts

Circuits and  
Logic Gates

Functions

Summary

## 1 Truth table for XOR:

x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0

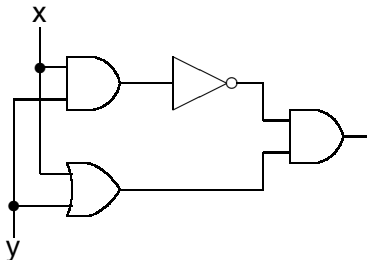
## 2 Realization of XOR:

x	y	x OR y	x AND y	NOT (x AND y)	b
0	0	0	0	1	0
0	1	1	0	1	1
1	0	1	0	1	1
1	1	1	1	0	0

where  $b = (x \text{ OR } y) \text{ AND } (\text{NOT } (x \text{ AND } y))$

### 3. $b = (x \text{ OR } y) \text{ AND } (\text{NOT } (x \text{ AND } y))$

the circuit with logic gates



## 9. Double Loops

Give the Python code to generate a list of all tuples containing all possible input values for a Boolean expression, i.e.:

`[(False,False),(False,True),(True,False),(True,True)].`  
Extend the code so it will work for triplets.

Answer:

```
L = []
for i in (False,True):
    for j in (False,True):
        L.append((i,j))
```

```
L = []
for i in (False,True):
    for j in (False,True):
        for k in (False,True):
            L.append((i,j,k))
```

## 9. Double Loops

Give the Python code to generate a list of all tuples containing all possible input values for a Boolean expression, i.e.:

`[(False,False),(False,True),(True,False),(True,True)].`  
Extend the code so it will work for triplets.

**Answer:**

```
L = []  
for i in (False,True):  
    for j in (False,True):  
        L.append((i,j))
```

```
L = []  
for i in (False,True):  
    for j in (False,True):  
        for k in (False,True):  
            L.append((i,j,k))
```

## 9. Double Loops

Give the Python code to generate a list of all tuples containing all possible input values for a Boolean expression, i.e.:

[(False,False),(False,True),(True,False),(True,True)].  
Extend the code so it will work for triplets.

**Answer:**

```
L = []
for i in (False,True):
    for j in (False,True):
        L.append((i,j))
```

```
L = []
for i in (False,True):
    for j in (False,True):
        for k in (False,True):
            L.append((i,j,k))
```

## 10. Using Dictionaries

Policies &  
Material

General  
Questions

Python code  
and  
Flowcharts

Circuits and  
Logic Gates

Functions

Summary

Write a program to sum the values of coins. The coins are represented by a sequence of characters, like `pnqdpqnppdqddpp`, where `p`, `n`, `d`, `q` represent respectively a penny (\$0.01), nickel (\$0.05), dime (\$0.10), and quarter (\$0.25). Use a dictionary.

Answer:

```
d = {'p':0.01,'n':0.05,'d':0.10,'q':0.25}
s = raw_input('give a sequence : ')
sum = 0
for i in range(0,len(s)):
    sum = sum + d[s[i]]
print 'value = ', sum
```

## 10. Using Dictionaries

Write a program to sum the values of coins. The coins are represented by a sequence of characters, like `pnqdpqnppdqddpp`, where `p`, `n`, `d`, `q` represent respectively a penny (\$0.01), nickel (\$0.05), dime (\$0.10), and quarter (\$0.25). Use a dictionary.

**Answer:**

```
d = {'p':0.01,'n':0.05,'d':0.10,'q':0.25}
s = raw_input('give a sequence : ')
sum = 0
for i in range(0,len(s)):
    sum = sum + d[s[i]]
print 'value = ', sum
```

## 10. Using Dictionaries

Policies &  
Material

General  
Questions

Python code  
and  
Flowcharts

Circuits and  
Logic Gates

Functions

Summary

Write a program to sum the values of coins. The coins are represented by a sequence of characters, like `pnqdpqnppdqddpp`, where `p`, `n`, `d`, `q` represent respectively a penny (\$0.01), nickel (\$0.05), dime (\$0.10), and quarter (\$0.25). Use a dictionary.

**Answer:**

```
d = {'p':0.01,'n':0.05,'d':0.10,'q':0.25}
s = raw_input('give a sequence : ')
sum = 0
for i in range(0,len(s)):
    sum = sum + d[s[i]]
print 'value = ', sum
```

## 10. Using Dictionaries

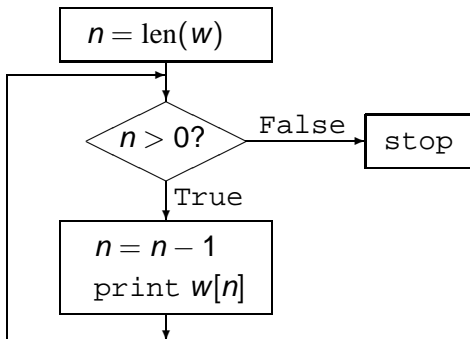
Write a program to sum the values of coins. The coins are represented by a sequence of characters, like `pnqdpqnnppdqddpp`, where `p`, `n`, `d`, `q` represent respectively a penny (\$0.01), nickel (\$0.05), dime (\$0.10), and quarter (\$0.25). Use a dictionary.

**Answer:**

```
d = {'p':0.01,'n':0.05,'d':0.10,'q':0.25}
s = raw_input('give a sequence : ')
sum = 0
for i in range(0,len(s)):
    sum = sum + d[s[i]]
print 'value = ', sum
```

## 11. From Flowchart to Program

Consider the flowchart, that takes on input the word  $w$ :



- 1 Explain what the algorithm in this flowchart does.
- 2 Use a `while` to implement the algorithm in Python.
- 3 Use a `for` to implement the algorithm in Python.
- 4 Modify the code so that printing occurs after the loop.

# Answers to from Flowchart to program

1 Explanation: the algorithm prints  $w$  backwards.

2 Using a while:

```
n = len(w)
while n > 0:
    n = n - 1
    print w[n]
```

3 Using a for:

```
for i in range(0, len(w)):
    print w[n-i-1]
```

4 Printing *after* the loop:

```
r = ""
for i in range(-len(w), 0):
    r = r + w[-i-1]
print r
```

# Answers to from Flowchart to program

1 Explanation: the algorithm prints  $w$  backwards.

2 Using a while:

```
n = len(w)
while n > 0:
    n = n - 1
    print w[n]
```

3 Using a for:

```
for i in range(0, len(w)):
    print w[n-i-1]
```

4 Printing *after* the loop:

```
r = ""
for i in range(-len(w), 0):
    r = r + w[-i-1]
print r
```

# Answers to from Flowchart to program

1 Explanation: the algorithm prints  $w$  backwards.

2 Using a while:

```
n = len(w)
while n > 0:
    n = n - 1
    print w[n]
```

3 Using a for:

```
for i in range(0, len(w)):
    print w[n-i-1]
```

4 Printing *after* the loop:

```
r = ""
for i in range(-len(w), 0):
    r = r + w[-i-1]
print r
```

# Answers to from Flowchart to program

① Explanation: the algorithm prints  $w$  backwards.

② Using a while:

```
n = len(w)
while n > 0:
    n = n - 1
    print w[n]
```

③ Using a for:

```
for i in range(0, len(w)):
    print w[n-i-1]
```

④ Printing *after* the loop:

```
r = ""
for i in range(-len(w), 0):
    r = r + w[-i-1]
print r
```

## 12. Unfair Coins

Explain how to simulate an unfair coin that when flipped returns head with probability 0.6.

Give a Python function for such unfair coin.

### Answer:

For  $x$ , uniformly chosen in  $[0, 1]$ , we decide that  $x$  yields head if  $x < 0.6$  and tail if  $x \geq 0.6$ . Generating numbers  $x$  randomly in  $[0, 1]$ , about 60% will turn up head.

```
def unfair_coin():  
    "returns head with probability 0.6"  
    import random  
    toss = random.uniform(0,1)  
    if toss < 0.6:  
        return 'head'  
    else:  
        return 'tail'
```

## 12. Unfair Coins

Explain how to simulate an unfair coin that when flipped returns head with probability 0.6.

Give a Python function for such unfair coin.

### Answer:

For  $x$ , uniformly chosen in  $[0, 1]$ , we decide that  $x$  yields head if  $x < 0.6$  and tail if  $x \geq 0.6$ . Generating numbers  $x$  randomly in  $[0, 1]$ , about 60% will turn up head.

```
def unfair_coin():  
    "returns head with probability 0.6"  
    import random  
    toss = random.uniform(0,1)  
    if toss < 0.6:  
        return 'head'  
    else:  
        return 'tail'
```

## 12. Unfair Coins

Explain how to simulate an unfair coin that when flipped returns head with probability 0.6.

Give a Python function for such unfair coin.

### Answer:

For  $x$ , uniformly chosen in  $[0, 1]$ , we decide that  $x$  yields head if  $x < 0.6$  and tail if  $x \geq 0.6$ . Generating numbers  $x$  randomly in  $[0, 1]$ , about 60% will turn up head.

```
def unfair_coin():
    "returns head with probability 0.6"
    import random
    toss = random.uniform(0,1)
    if toss < 0.6:
        return 'head'
    else:
        return 'tail'
```

## 13. Roots of a Quadric

Define a Python function that returns as a tuple the roots of a polynomial  $p(x) = ax^2 + bx + c$ , using  $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ .

Write the function so that 1 is the default value for  $a$ .

**Answer:**

```
def roots ( c, b, a=1 ):
    "roots of c + b*x + a*x^2"
    import math
    d = math.sqrt(b**2 - 4*a*c)
    return ((-b+d)/(2*a), (-b-d)/(2*a))
```

*default arguments must come last*

## 13. Roots of a Quadric

Define a Python function that returns as a tuple the roots of a polynomial  $p(x) = ax^2 + bx + c$ , using  $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ .

Write the function so that 1 is the default value for  $a$ .

**Answer:**

```
def roots ( c, b, a=1 ):
    "roots of c + b*x + a*x^2"
    import math
    d = math.sqrt(b**2 - 4*a*c)
    return ((-b+d)/(2*a), (-b-d)/(2*a))
```

*default arguments must come last*

# Summary

Printer friendly versions of the slides are posted at <http://www.math.uic.edu/~jan/mcs260>.

Also take a look at the posted Python code. Solutions to quizzes are posted as well.

Background reading for the first 18 lectures:

- chapters 1 to 5 in *Python Programming in Context* except files (Chapter 6) and modules (Chapter 7).
- chapters 1, 2, 3 in *Computer Science*, also part of Chapter 5 on Algorithms, and programming concepts of Chapter 6.

Organize notes from lectures and lab sessions!