Some Basic (pre) Algebra maths

The Identity Element

• This special element is know as the identity element for the addition operator

$$x + 0 = x$$
$$0 + x = x$$

$$5 + 0 = 5$$

I's are special too

- When we add 0 it does nothing
- The same is true with I for multiplication, this is know as the multiplication identity element

$$1x = x$$

$$1(3) = 3$$

The inverse operation

- The "inverse" of addition is subtraction.
- You can think of subtraction as A + (-1)B

$$(x+y) - y = x$$

$$(3+6) - 6 = 3$$

Associative rule

- The Associative rule says that the order of operation are not important as long as the operands do not change
- Operands are the variables in this case

$$(x + y) + z = x + (y + z)$$

(2 + 3) + 1 = 2 + (3 + 1)
5 + 1 = 2 + 4
6 = 6

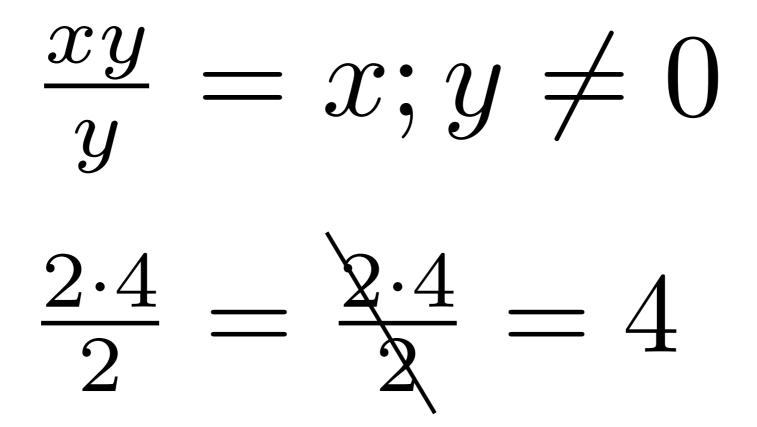
Commutativity

• Commutativity allows us to change the order of operations without changing the end result.

$$xy = yx \qquad = 4 \times 2$$
$$x \times y \equiv xy \qquad = 2 \cdot 4 = 4 \cdot 2$$
$$\equiv 2 \cdot 4 = 4 \cdot 2$$
$$\equiv 2 \cdot 4 = 4 \cdot 2$$
$$\equiv 2 \cdot 4 = 4 \cdot 2$$
$$\otimes 8 = 8$$

Inverse Operation

• The inverse of multiplication is division



Associative Multiplication

• Like addition multiplication is also associative

$$(xy)z = x(yz)$$

$$(3 \cdot 4)2 = 3(4 \cdot 2)$$

 $(12)2 = 3(8)$
 24

Distributive Multiplication

- This property is useful in algebra when we need to factor things
- It is also used in Matrix manipulation and boolean logic

$$(x + y)z = xz + yz$$

(7+3)2 = 7 · 2 + 3 · 2
10 · 2 = 14 + 6
20 = 20

Associative Division

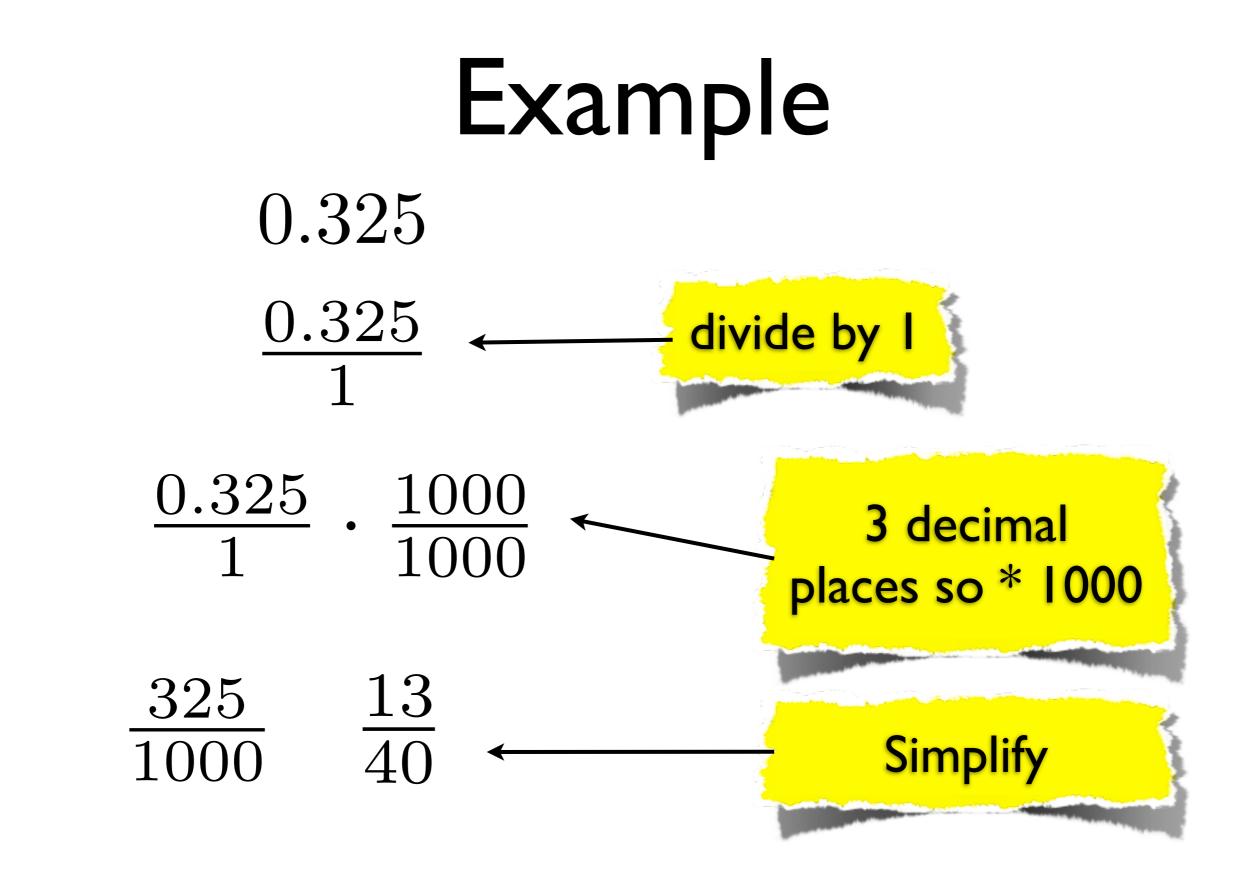
• There is a similar rule for division

$$a\left(\frac{b}{c}\right) = \frac{ab}{c}$$

$$\begin{array}{ccc} 6\left(\frac{5}{2}\right) = \frac{6 \cdot 5}{2} & \equiv & \frac{5}{2} = 2.5 \\ \frac{30}{2} = 15 & & 2.5 \times 6 = 15 \end{array}$$

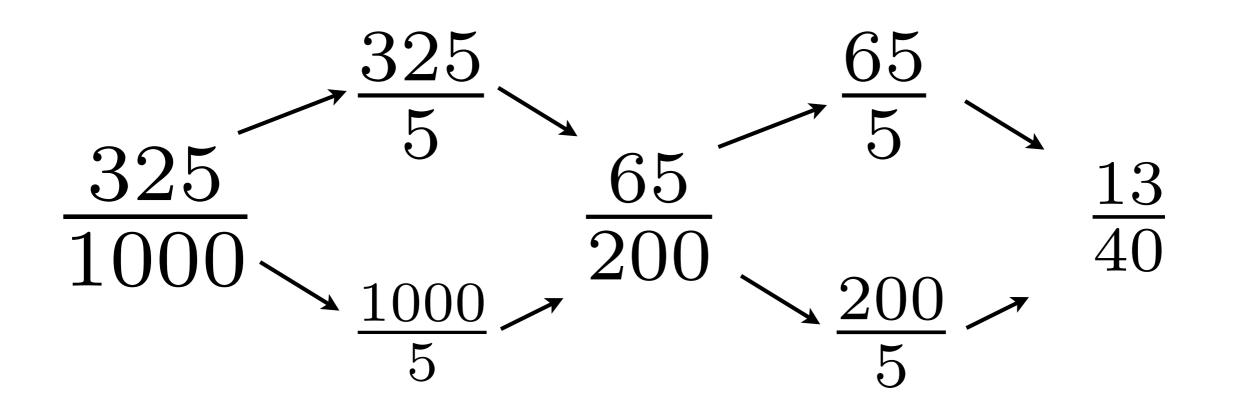
Convert decimals to fractions

- This may seem complex but it's actually fairly simple
 - Write down the decimal and divide it by one (decimal /1)
 - 2. Multiply top and bottom of fraction by 10 for each number after the decimal point
 - 3. Simplify the new fraction



Greatest Common Factor

- AKA Greatest Common Divisor
- In the previous example to simplify we need to find the greatest common factor of the numerator and denominator
- In this case it is fairly intuitive if we know about numbers and especially 5



We could also have noticed that both are multiples of 25 (another number trick)

$$\frac{\frac{325}{25}}{\frac{1000}{25}} = \frac{13}{40}$$

Some python

- Python is a strongly typed language.
- This means that the python interpreter keeps track of all of the data types
- When using maths we have two types
 - integers (number without decimal points)
 - floating point numbers



declare some variables which python to use

#!/usr/bin/python

integer=1
floating=0.25

print integer
print "%0.4f_" %(floating)



reading values in

- Python uses a function called raw_input to read values from the shell
- These values are always character values (even when we press the numbers)
- If we wish to read numbers in we need to convert the text to a numeric value
- This is shown in the next example

```
#!/usr/bin/python
a=int(raw_input("enter_an_int_value_>"))
b=float(raw_input("enter_a_float_value>"))
print a,b
```

- int ([value]) will attempt to convert the value into an integer
- float ([value]) will attempt to convert the value into a float

Arithmetic expressions

- Most programs are algorithmic in nature which means we have to do some maths
- The table below shows the available arithmetic operators

Operator	Meaning	Examples
+	addition	5 + 2 is 7 5.0 + 2.0 is 7.0
-	subtraction	5 - 2 is 3 5.0-2.0 is 3.0
*	multiplication	5*2 is 10 5.0*2.0=10.0
/	division	5/2 is 2 5.0/2.0 is 2.5
%	remainder (modulus)	5%2 is 1

The / Operator

- When applied to two positive integers the division operator computes the integral part of the result dividing its first operand by its second
- For example

```
7.0 / 2.0 is 3.5
7 / 2 is 3
299.0 / 100.0 is 2.99 (float value)
299 / 100 is 2 (integer value)
```

- If the / Operator is used with a negative and positive integer, the results vary from one implementation to another
- For this reason you should avoid division by -ve integers

The % (modulus) Operator

- The remainder operator (%) returns the integer remainder of the result of dividing the first operand with the second
- For example the value of 7 % 2 is 1
- The magnitude of m % n must always be lest than the division n

$$7/2 \qquad 299/100 \\ \downarrow \qquad \downarrow \\ 7 \div 2 = 3 \qquad 299 \div 100 = 2 \\ 3 * 2 = 6 \qquad 2 * 100 = 200 \\ \frac{6}{7-6} \leftarrow 7 \% 2 = 1 \qquad \frac{200}{299-200} = 299 \% 100 = 99$$

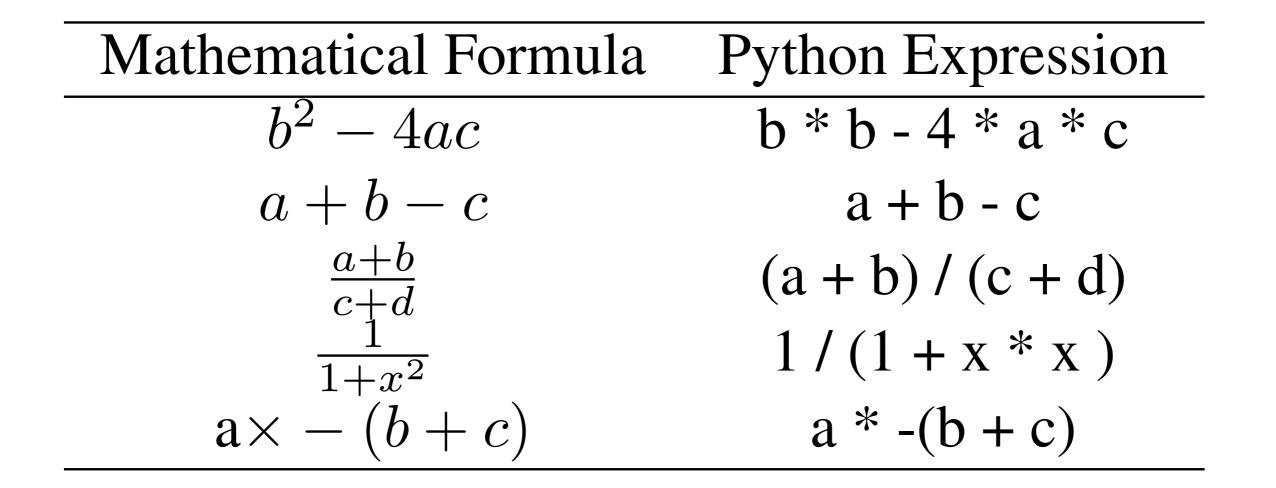
Expressions with Multiple Operators

- There are rules as to how expressions are evaluated
 - Parentheses Rule : All expressions in parentheses must be evaluated separately. Nested parenthesised expressions must be evaluated from the inside out, with the innermost expression evaluated first.
 - Operator precedence rule : Operators in the same expression are evaluated in the following order.

Expressions with Multiple Operators

- Associativity Rule : Unary operators in the same subexpression and at the same precedence levels (such as + and -) are evaluated right to left.
- Binary operators in the same sub-expression and the same precedence level (such as + and -) are evaluated left to right.
- To help avoid problems with the order of evaluation it is best to use parenthesis

Mathematical Formulas as Python expressions



#!/usr/bin/python

```
a=float (raw_input ("enter_a_>"))
b=float (raw_input ("enter_b_>"))
c=float (raw_input ("enter_c_>"))
d=float (raw_input ("enter_d_>"))
x=float (raw_input ("enter_x_>"))
```

```
answer=b * b - 4 * a * c
```

print answer

answer=a+b-c print answer

```
answer=(a+b)/(c+d)
```

```
print answer
```

```
answer=1.0/(1+x*x)
print answer
```

```
answer=a \star - (b+c)
```

```
print answer
```

Law of Indices

• The Law of Indices can be expressed as

$$a^{m} \times a^{n} = a^{m+n}$$
$$a^{m} \div a^{n} = a^{m-n}$$
$$(a^{m})^{n} = a^{mn}$$

• Examples

$$2^3 \times 2^2 = 8 \times 4 = 32 = 2^5$$

 $2^4 \div 2^2 = 16 \div 4 = 4 = 2^2$
 $(2^2)^3 = 64 = 2^6$

The pow function

```
#!/usr/bin/python
a=float(raw_input("enter_an_int_value_>"))
b=float(raw_input("enter_a_float_value>"))
```

print "a^b_=_", pow(a,b)

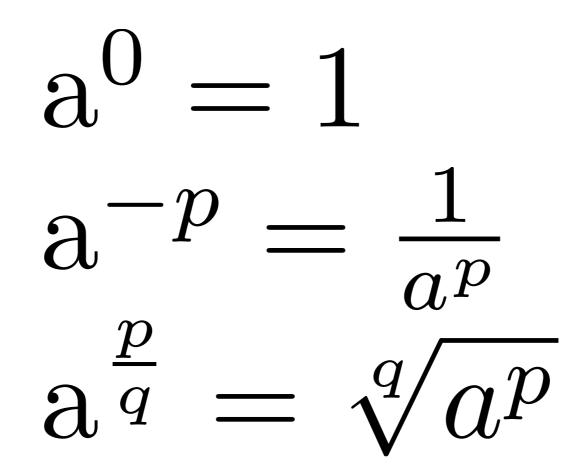
Indices.py

- We can also do powers in python using the ** syntax
- a**b means a^b

```
#!/usr/bin/python
2
 3
   a=int(raw_input("Enter_a_value_for_a_>"))
   m=int(raw_input("Enter_a_value_for_m_>"))
   n=int(raw_input("Enter_a_value_for_n_>"))
   print "for values a=%d and m=%d n=%d" % (a, m, n)
8
9
   print "Multiplication"
   print "a^m_*_a^n_=_", a**m * a**n
10
   print "sum_of_indices_=_",m+n
11
   print "a^ (m+n) = ", a** (m+n)
12
13
   print "Division,"
14
15
   print "a^m_/_a^n_=_", a**m / a**n
16
   print "difference_of_indices_=_",m-n
17
   print "a^ (m+n) = ", a** (m-n)
18
19
20
   print "Powers,"
21
22
   print " (a^m) ^n = ", (a**m) **n
   print "a^, m*n, ==, ", a** (m*n)
23
```

Law of Indices

• From the previous examples, it is evident that



Indices2.py

```
#!/usr/bin/python
2
3
   from math import *
  a=int(raw_input("Enter_a_value_for_a_>,"))
4
   p=int(raw_input("Enter a value for p > "))
5
6
   print "a^0____", a**0
7
8
   print "a^-p_=,", a**-p
9
  print "1/a^p_=_", 1.0 / (a**p)
10
```

[jmacey@neuromancer:Lecture2]\$./Indices2.py Enter a value for a > 2Enter a value for p > 4 $a^0 = 1$ $a^-p = 0.0625$ $1/a^p = 0.0625$

Roots

- Most programming languages have a function to find the square root (usually sqrt)
- However higher roots are no implemented.
- We can use the law of indices shown previously to calculate higher roots

```
#!/usr/bin/python
import math
a=int(raw_input("enter_a_value"))
print math.sqrt(a)
```

Roots.py

```
#!/usr/bin/python
1
2
3
   from math import *
4
   a=int(raw_input("Enter_a_value_for_a_>_"))
5
   # here we loop in the range 1 to 10 as the range
6
7
   # function returns the values range(s, e-1)
8
9
   for n in range(1,11) :
10
     print "the_%d_root_of_%d_=_" %(n,a) ,a**(1.0/n)
```

>>> 2**10
1024
>>> 32**2
1024
>>> 4**5
1024
>>> []

[jmacey@neuromancer:Lecture2]\$./Roots.py				
Enter a value for a >	1024 of %d = 1			
the 1 root of $1024 =$	1024.0			
the 2 root of $1024 =$	32.0			
the 3 root of $1024 =$	10.0793683992			
the 4 root of $1024 =$	5.65685424949			
the 5 root of $1024 =$	4.0			
the 6 root of $1024 =$	3.17480210394			
the 7 root of $1024 =$	2.69180038526			
the 8 root of $1024 =$	2.37841423001			
the 9 root of $1024 =$	2.16011947778			
the 10 root of $1024 = 2.0$				

- Two people are associated with logarithms:
- John Napier (1550-1617) and Joost Bürgi (1552-1632).
- Logarithms exploit the addition and subtraction of indices and are always associated with a base
- For Example, if

$$a^{x} = n$$

 $\log_{a} n = x$
Where a is the base.

$10^2 = 100$ $\log_{10} 100 = 2$

- It can be said "10 has been raised to the power 2 to equal 100"
- The log operation finds the power of the base for a given number

- Multiplication's can be translated into an addition using logs
- We then add the numbers and convert back $36 \times 24 = 864$ $\log_{10} 36 + \log_{10} 24 = \log_{10} 864$ 1.5563025007 + 1.38021124171 = 2.963651374248
- The two bases used in calculators and computer software are 10 and 2.718281846..., the second value is know as the transcendental number e
- Logs to the base 10 are written as log
- Logs to the base e are written as **In**

Friday, 19 October 12

```
Logs.py
   #!/usr/bin/python
   from math import *
   a=int(raw_input("Enter_a_value_for_a_>,"))
   b=int(raw_input("Enter_a_value_for_b_>_"))
   print "Using * the answer is ", a*b
   log10a=log10(a)
11 log10b=log10(b)
  lna=log(a)
12
  lnb=log(b)
   print "log10(a)__+_log10(b)_=_%f_+_%f_=" %(log10a,log10b), log10a+log10b
   print "log(a)__+_log(b)_=_%f_+_%f_=" %(lna,lnb), lna+lnb
18 | print "Anti Logs."
19 c=log10a+log10b
20 | print "log_10"
                                            [jmacey@neuromancer:Lecture2]$./Logs.py
21 | print "%f, = 10^%f, =" %(c,c), 10**c
                                            Enter a value for a > 1234
  c=lna+lnb
23 print "Natural, log, (e)"
                                            Enter a value for b > 4321
24 print "%f_=_exp(%f)_=_" %(c,c), exp(c)
                                            Using * the answer is 5332114
                                            log10(a) + log10(b) = 3.091315 + 3.635584 = 6.72689942601
                                            log(a) + log(b) = 7.118016 + 8.371242 = 15.4892583404
                                            Anti Logs
                                            log 10<sup>sult</sup> is between -pi and pi. The vector in the plane from the origin
                                            6.726899 = 10^6.726899 = 5332114.0 For example, at an (1) and a
                                            Natural log (e)
                                            15.489258 = \exp(15.489258) = 5332114.0
```

1 2 3

4 5

6 7 8

9 10

13 14 15

16 17

22

 $\log(ab) = \log a + \log b$ $\log(\frac{a}{h}) = \log a - \log b$ $\log(a^n) = n \log a$ $\log(\sqrt[n]{a}) = \frac{1}{n} \log a$

References

- Mathref <u>http://happymaau.com/projects/math-ref/</u>
- http://python.org/
- "Essential Mathematics for Computer Graphics fast" John VinceSpringer-Verlag London
- http://en.wikipedia.org/wiki/Johannes_Kepler